UNIVERSITY OF GRONINGEN

MASTER THESIS

Controlling Recurrent Neural Networks by Diagonal Conceptors

Author:

J.P. De Jong

Examiners:

Prof. Dr. H. Jaeger

Assistant Professor C.P. Hirsch

June 14, 2021

Contents

Abstract

The human brain is capable of learning, memorizing, and regenerating a panoply of temporal patterns. A neuro-dynamical mechanism called *conceptors* offers a method for controlling the dynamics of a recurrent neural network by which a variety of temporal patterns can be learned and recalled. However, conceptors are matrices whose size scales quadratically with the number of neurons in the recurrent neural network, hence they quickly become impractical. This report introduces a variation of conceptors, called *diagonal conceptors*, which are diagonal matrices, thus reducing the computational cost drastically. It will be shown that diagonal conceptors achieve the same accuracy as conceptors, but are slightly more unstable. This instability can be improved, but requires further research. Nevertheless, diagonal conceptors show to be a promising practical alternative to conceptors.

1 Introduction

In 1997, the world chess champion Garry Kasparov lost to the IBM supercomputer Deep Blue. This was the first time a world chess champion was beaten by a machine [1]. Ever since, the field of Artificial Intelligence (AI) has expanded and grown immensely. Currently, there are several books which are solely devoted to the discussion about the progress of AI, both the good and bad sides of it [1, 2]. One important open question is how close AI research is to creating AI that can outperform humans in most tasks. At present, AI can transcend human intelligence in some individual tasks, such as playing games like Atari, chess, and Go [3], or arithmetic. Furthermore, the field of neural networks has made an impressive leap forward since its creation, instigated by the perceptron [4]. At present, neural networks are researched extensively and are used for a wide range of tasks. Nevertheless, the human brain remains overall superior to AI. One of the reasons is that the human brain is able to learn a panoply of skills consisting of, for instance language, sport, and social interaction. Specifically, the ability to learn, recognize, recall, and combine a vast amount of *temporal* patterns with little effort. It is thus desirable to research a neural network that is capable of such tasks. One of the difficulties is managing long-term memory, which is the ability to permanently store temporal patterns in its synaptic weights such that it is able to recall the learned patterns whenever asked without changing those weights, also referred to as neural long-term memory. There are several researches devoted to this task, which will be discussed in Section 1.1. However, a neuro-computational mechanism called *conceptors*, not only offers a solution to managing long-term memory for temporal patterns, but also offers insight into a variety of other problems [5].

The conceptors architecture is a neuro-computational mechanism that can be used to control the dynamics of an *Recurrent Neural Network* (RNN), henceforth also referred to as the *reservoir*. Conceptors were first introduced in the technical report *Controlling Recurrent Neural Network by Conceptors* written by H. Jaeager, which will be referred to as the *conceptors report*. In the conceptors report many of the applications of conceptors, such as temporal pattern classification, human motion generation, de-noising and signal separation were demonstrated [5]. Consequentially, conceptors have been successfully used in several other researches [6–13]. It must be noted that conceptors can take different forms, but in this report they will only be in matrix form and will therefore be referred to as *conceptor matrices* or simply *conceptors*.

Intuitively, conceptors act as filters through which an RNN can be controlled. The crucial observation is that when an N-dimensional RNN is driven by different patterns $1, 2, ..., p$, different areas of the neural state space $A^1, A^2, ..., A^p$ are being activated. Conceptors exploit this observation, which allows for retrieving a desired pattern from the neural state space. Differently stated, conceptors attempt to identify the subvolumes in which the respective patterns live. A conceptor $C^j \in \mathbb{R}^{N \times N}$ representing a pattern j constrains the neural dynamics to the volume of state space A^j such that the network will regenerate pattern j. It achieves this by leaving the area of state space associated with pattern j for the most part unchanged, yet suppressing the other areas of state space. Conceptors are robust against small parameter changes and noise. However, conceptors can become computationally inefficient. As the number of neurons in the reservoir N increases, the conceptor matrices grow quadratically in N, which poses problems with storage and computational cost. For example, a matrix with 1000 rows and columns uses approximately 8 MB of storage, so for 10 patterns, 80 MB is needed. However, if the number of neurons in the reservoir increases from $N = 1000$ to $N = 5000$, then the conceptor matrix will have 5000 rows and columns, requiring 200 MB of memory for a single conceptor. Not only storage, but also the computational cost increases drastically. If $C \in \mathbb{R}^{N \times N}$ and $\mathbf{x} \in \mathbb{R}^N$, then the complexity of the multiplication $C\mathbf{x}$ is $O(N^2)$. Therefore, conceptor matrices quickly become unpractical as the reservoir size increases, which is unfortunate, as conceptors carry many great qualities and it would be ideal if they could be used in practice. An alternative architecture was introduced, called the *random feature conceptors* architecture, to overcome this impracticality as well as biological implausibility [5]. The idea of the random feature conceptors architecture is to expand the state of the RNN onto a higher-dimensional space, where the state is manipulated by a diagonal conceptor matrix rather than a full matrix, and finally the manipulated state is projected back to the original space. In this architecture, the diagonal conceptors manipulate the higher-dimensional state element-wise, so it can be written in vector form. This is computationally much more efficient. It was then discovered that the architecture still works if the dimension of the higher-dimensional space is set equal to dimension of the original space and both projection matrices are set to the identity matrix, i.e. the same architecture as conceptors, but with a diagonal conceptor matrix. However, this only works well if a slight adjusted is made in the training scheme, which is a vital adjustment and will be discussed later. This discovery lead to the neuro-computational mechanism that is the subject of this report, called *diagonal conceptors*.

Diagonal conceptors offer a solution to the impracticality of conceptors. As can be deduced from the name, diagonal conceptors are diagonal matrices, meaning that they reduce the computational cost significantly and require much less storage. A diagonal conceptor matrix $D \in \mathbb{R}^{N \times N}$ can be written as $D = \text{diag}(\mathbf{c})$, where $\text{diag}(.)$ denotes a diagonal matrix with zeros everywhere except for the diagonal. The diagonal $\mathbf{c}$ is an N-dimensional vector called the *conception vector* and its elements c_i, for $i = 1, ..., N$, are called the *conception weights*. In practice, the conception vector $\mathbf{c}$ is used, but in this report, for notational simplicity, the matrix notation D will be used. In comparison to a conceptor matrix, a conception vector of size 5000 requires only 40 KB of storage, so for 10 patterns, only 0.4 MB is required to store the conception vectors. Moreover, the complexity of computing the conception vector $\mathbf{c}$ times a vector $\mathbf{x} \in \mathbb{R}^N$ is only $O(N)$, which is a considerable decrease compared to $O(N^2)$.

In addition, it will be shown that the conception weights can be trained individually, which provides two advantages over conceptors. First, diagonal conceptors are not biologically implausible, whereas conceptors are. Conceptors can be online adapted in a variation called *autoconceptors*, but the online adaptation requires non-local computations, making the conceptors biologically implausible. Diagonal conceptors, on the other hand, can also be adapted online, for which only local computations are required. Therefore, it can be said that diagonal conceptors are not biologically implausible, in the sense that all information for the adaptation of a synaptic weight is available as the synapse. Second, the global learning rate for conceptors is constrained by the largest local curvature in the gradient space, resulting in slow convergence in areas where the curvature is small. The learning rate of the online adaptation of diagonal conceptors, on the other hand, can be chosen for each conception weight individually. Therefore, the constraint of the global learning rate for conceptors is lifted for diagonal conceptors.

Diagonal conceptors are trained differently than conceptors, which is not surprising, since the degrees of freedom of a conceptor is much higher than the degrees of freedom of a diagonal conceptor. The main changing in the training scheme is that before the reservoir is driven, a randomly initialized diagonal conceptor is inserted in the update loop. Intuitively, these initial random diagonal conceptors scale the reservoir state to a different area in state space and the diagonal conceptor will be trained on that newly positioned area rather than the original area. This is especially useful when different areas in state space overlap, since, in contrast to conceptors, diagonal conceptors have less degrees of freedom to characterize the nuances of each area. The use of this initial random scaling is an interesting observation and it offers opportunities for more future work, because diagonal conceptors can surely be improved by reducing the randomness of the initial random scaling.

This report introduces the diagonal conceptors by examples that are also used in the conceptors report [5]. The examples consist of *four periodic patterns*, *chaotic attractors*, and *human motion*. Since each example highlights a different quality of conceptors, it will become evident which characteristics translate to diagonal conceptors and which do not. For each example, both conceptors and diagonal conceptors are trained, allowing a direct comparison to be made.

The objective of this report is to introduce diagonal conceptors so they can be used as a practical alternative for conceptors. Furthermore, this report aims to analyse some of the properties of diagonal conceptors and how they relate the properties of conceptors. Lastly, this report is intended as a intuitive guide to applying diagonal conceptors to real-world problems.

1.1 Related Work

To begin this part of the section, it should be mentioned that there are two ways in which diagonal conceptors can be related to other works. The first is by viewing diagonal conceptors as a variation of the conceptors. The scope of the conceptors architecture is much broader than learning, recognizing, and recalling temporal patterns. Conceptors were introduced as a novel perspective on the neuro-symbolic integration problem [5]. The field of neuro-symbolic integration aims to bridge the gap between two fundamentally different paradigms: symbolic systems in AI and neural network systems in AI. In turn, it intents to unite the research of different fields. There are many researches from different fields dedicated to the neuro-symbolic integration problem [14–19]. However, the field of neuro-symbolic integration is outside the scope of this report, so for an easy introduction the interested reader is referred to [20]. As diagonal conceptors are only used in for storing and recalling temporal patterns in this report, the neuro-integration problem will not be discussed. Nonetheless, whether the symbolic properties of conceptors also translate diagonal conceptors is an open and interesting question.

The second way of comparing diagonal conceptors to other works is by viewing the diagonal conceptors architecture as a means to store temporal patterns in long-term memory and retrieve them at exploitation time. As mentioned in the article *Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns* by H. Jaeger, associative memory is the paradigmatic model for neural long-term memory, which was introduced by a variety of researches [21–23], only a few of which are cited. However, with the development of other fields of research, temporal pattern learning and recognition has also been researched by deep neural networks [24–28], RNNs [29–31], and others [32–34].

2 Theory

The first part of this section largely follows the conceptors report [5]. Nevertheless, this section is self-contained and introduces all the required background knowledge for this report. It is organized in four parts. First and second, the mathematical landscape of an RNN is introduced as well as how to store patterns in a reservoir and how to train the output weights. Third, conceptors are introduced and defined, followed by a brief introduction of autoconceptors. Fourth, diagonal conceptors are introduced and defined.

As for mathematical notation, there are a few notations that will be consistent throughout this report. A matrix is denoted by a capital letter. A vector is denoted by a bold letter. The element on the i-th row and j-th column of a matrix A is denoted by a_{ij}. The i-th element of a vector $\mathbf{a}$ is denoted by a_i. The transpose of a matrix A is denoted by A^T. The Frobenius norm of a matrix A is denoted by $||A||_{fro}$ and is defined, for a real matrix, as $||A||_{fro} = \sqrt{\sum_i \sum_j |a_{ij}|^2}$. If the norm has no subscript, i.e. $||A||$, the reader can assume it is the Frobenius norm.

2.1 Mathematical Landscape

Let $\mathbf{P} = \{\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^p\}$ be a set of p discrete-time patterns, where pattern j at time step n is given by $\mathbf{p}^j(n) \in \mathbb{R}^M$. Furthermore, assume M input neurons as well as a bias neuron, a RNN consisting of N simple tanh neurons called *reservoir neurons*, and M' output neurons. In the more general case one can assume $M' \neq M$ output neurons, which would be the case if the driving patterns are used to produce output based on the driving patterns, e.g. classification. However, in this report the reservoir and output neurons serve as a means to self-generate the driving patterns, hence the number of input and output neurons is set equal. The input neurons drive the reservoir with $\mathbf{p}^j(n)$ via an input weights matrix $W^{in} \in \mathbb{R}^{N\times M}$ and a bias vector $\mathbf{b} \in \mathbb{R}^N$. The N neurons in the RNN are recurrently connected via a weights matrix $W^* \in \mathbb{R}^{N\times N}$. When the reservoir is driven by pattern $\mathbf{p}^j(n)$, the reservoir neurons are activated. The state of neuron i at time n is denoted by $x_i^j(n) \in (-1,1)$ and is called the *neuron state*. The state of the reservoir at time n, denoted by $\mathbf{x}^j(n) \in (-1,1)^N$, is given by the vector containing all the neuron states at time n and is called the *reservoir state*. Finally, the output neurons serve to read out the target signal at time step n, denoted by $\mathbf{y}^j(n) \in \mathbb{R}^M$, from the reservoir state via the output weights matrix $W^{out} \in \mathbb{R}^{M\times N}$. A visual representation of the setup can be seen in Figure 1.

Let W^*, W^{in}, and $\mathbf{b}$ be fixed random matrices. They will not be adjusted after initialization. The reservoir state vector, or simply *state vector*, is updated according to the update equation

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

and the output signal is given by

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

where the output weights W^{out} are learned, see Section 2.2. Let the reservoir be driven for L time steps. The state vectors $\mathbf{x}^j(n)$ are collected in the *state collection matrix* X^j. The state collection matrix gives a sample of states that is representative of the volume of state space that

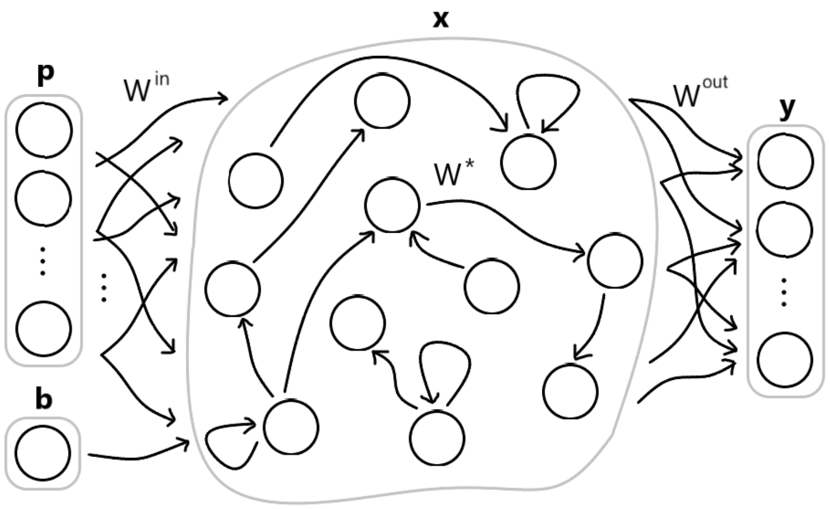

Figure 1: Basic setup of an RNN. The input neurons drive the reservoir with pattern **p** through an input weights matrix W^{in} as well as a bias vector **b**. The reservoir neurons are interconnected via a reservoir weights matrix W^*. The output **y** can be read from the reservoir state **x** via an output weights matrix W^{out}.

is occupied by the driving pattern j and is given by

$$X^j = \begin{bmatrix} x_1^j(1) & x_1^j(2) & \dots & x_1^j(L) \\ x_2^j(1) & x_2^j(2) & \dots & x_2^j(L) \\ \vdots & & \ddots & \vdots \\ x_N^j(1) & x_N^j(2) & \dots & x_N^j(L) \end{bmatrix}. \tag{3}$$

Note that all the information about pattern j is encoded in the X^j.

2.2 Storing the Patterns and Training the Output Weights

As mentioned earlier, one of the goals of this report is to store a variety of patterns in a single reservoir in such a way that they can later be retrieved. Up until this point, the reservoir has been driven by the input patterns. However, in order to store the patterns, the reservoir must be able to update the state vector in the absence of a driving input, which yields the following approximation:

$$\tanh(W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}) \approx \tanh(W\mathbf{x}^j(n) + \mathbf{b}), \tag{4}$$

where W comprises the recomputed reservoir weights. W can be computed by minimizing the mean square error

$$W = \underset{\tilde{W}}{\operatorname{argmin}} \sum_j \sum_n ||W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n+1) - \tilde{W}\mathbf{x}^j(n)||^2. \tag{5}$$

For such linear regression tasks, ridge regression will be used throughout this report.

Another way of removing the driving input from the update equation, which is computationally more efficient, is by leaving $W^*\mathbf{x}^j(n)$ unchanged and finding a matrix $H \in \mathbb{R}^{N\times N}$ such that

$$\tanh(W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}) \approx \tanh(W^*\mathbf{x}^j(n) + H\mathbf{x}^j(n) + \mathbf{b}). \tag{6}$$

The so-called *input simulation matrix H* is then computed by minimizing the mean square error

$$H = \underset{\tilde{H}}{\operatorname{argmin}} \sum_j \sum_n ||W^{in}\mathbf{p}^j(n+1) - \tilde{H}\mathbf{x}^j(n)||^2. \tag{7}$$

The notation here differs from [5], where the input simulation matrix is denoted by D. In this report, the notation D is reserved for the diagonal conceptors.

One could even go one step further by leaving W^* and W^{in} unchanged and finding a matrix $R \in \mathbb{R}^{M\times N}$ such that

$$\tanh(W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}) \approx \tanh(W^*\mathbf{x}^j(n) + W^{in}R\mathbf{x}^j(n) + \mathbf{b}), \tag{8}$$

where the so-called *input recreation weights R* are computed by

$$R = \underset{\tilde{R}}{\operatorname{argmin}} \sum_j \sum_n ||\mathbf{p}^j(n+1) - \tilde{R}\mathbf{x}^j(n)||^2. \tag{9}$$

Intuitively, W should yield better results than H or R, because of a larger number of recomputed weights, leading to higher accuracy. In simulations it was found that there is a slight difference in the results, depending on the complexity of the simulation, but since the computational cost of the simulations in this report was fairly small, the preference was given to computing W rather than H or R.

For computing W^{out}, a similar approach is employed, where the output $\mathbf{y}^j(n) = W^{out}\mathbf{x}^j(n)$ must approximate the input pattern $\mathbf{p}^j(n)$. This gives the approximation

$$\mathbf{p}^j(n) \approx \mathbf{y}^j(n) = W^{out}\mathbf{x}^j(n), \tag{10}$$

where W^{out} is then computed by minimizing the mean square error

$$\sum_j \sum_n ||\mathbf{p}^j(n) - W^{out}\mathbf{x}^j(n)||^2. \tag{11}$$

The process of recomputing the reservoir weights W and computing the output weights W^{out} is referred to as *storing* the patterns in the reservoir. The reservoir is called *loaded* after the patterns have been stored.

2.3 Conceptors

After the patterns are stored in the reservoir, there exists a superposition of patterns in the reservoir. If the network were to simply start updating the state vector by $\mathbf{x}(n+1) = \tanh(W\mathbf{x}(n) + \mathbf{b})$ it would exhibit unpredictable behavior, since the reservoir does not know which pattern to engage in. So, how is a specific pattern retrieved from the reservoir? Note that driving the reservoir with pattern j creates a cloud of points in state space, which is characteristic for pattern j. This point cloud is given by the columns of the state collection matrix X^j. The idea is to project the indecisive state vector $\mathbf{x}(n+1)$ to the point cloud associated with the pattern that needs to be retrieved. Each pattern j will have its own projection matrix, which is called the *conceptor* associated with pattern jq and is denoted by C^j. The conceptor matrix C^j should behave such

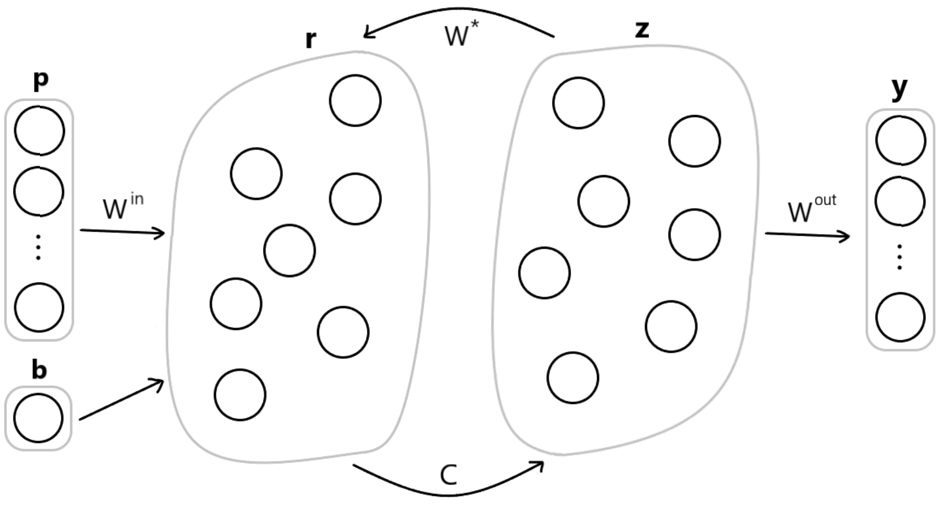

Figure 2: A visual representation of Equations 12 and 13. The reservoir is driven by pattern $\mathbf{p}$ via the input weights W^{in}. The state $\mathbf{r}$ is expanded into a state $\mathbf{z}$ via a conceptor C and $\mathbf{z}$ is inserted back in the loop via W^*. The output is read from the state $\mathbf{z}$ by the output weights W^{out}.

that it leaves states associated with pattern j intact, but suppresses the states of other patterns. This trade-off is mitigated by a parameter called *aperture*[1].

To retrieve the state vector associated with pattern j, a conceptor matrix C^j is projected onto the state vector. This can be viewed as a neural network consisting of two layers, where the first layer is projected to the second layer via C^j and is projected back via W. Therefore, the update equation is written in two parts

$$\begin{aligned}\mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n)+\mathbf{b}\right),\\ \mathbf{z}(n+1) &= C^j\mathbf{r}(n+1),\end{aligned} \tag{12}$$

where the output is given by

$$\mathbf{y}(n) = W^{out}\mathbf{z}(n). \tag{13}$$

A visual representation is depicted in Figure 2.

2.3.1 Computing Conceptors

At this point, the superscript j is omitted for notational simplicity. Conceptor C must leave states $\mathbf{z}(n)$ unchanged, but also suppress components that are not typical for $\mathbf{z}(n)$. This essentially means that C should act as the identity matrix for states $\mathbf{z}(n)$, but as the null matrix for the unwanted components. This leads to the following quadratic loss function:

$$\begin{aligned}\mathcal{L}(C) &= \mathbb{E}[||\mathbf{z}-C\mathbf{z}||^2] + \alpha^{-2}||C||^2\\ &= \mathrm{Tr}[(I-C^T)(I-C)]\mathbb{E}[\mathbf{z}\mathbf{z}^T] + \alpha^{-2}\mathrm{Tr}[C^TC],\end{aligned} \tag{14}$$

where α is the aperture and $||.||$ is the Frobenius norm.

[1]The name aperture has its roots in optics, where it means the diameter of the effective lens opening, hence it negotiates the amount of light energy that reaches the film.

The first component of $\mathcal{L}(C)$ is the time-average of the difference between the projected state vectors $C\mathbf{z}$ and the state vector $\mathbf{z}$. This reflects that C must leave $\mathbf{z}$ unchanged. This component is optimal for $C = I$. The second part of the loss function represents the suppression of the outlying components of the state vector. Note that it is minimal for C equals the null matrix. Therefore, a large aperture results in a conceptor matrix close to the identity matrix. Conversely, a small aperture will shrink the conceptor matrix towards the null matrix.

To find to matrix C for which $\mathcal{L}(C)$ is minimal, the gradient of $\mathcal{L}(C)$ with respect to C is computed. The gradient is given by

$$\frac{\partial}{\partial C}\left(\mathbb{E}[||\mathbf{z}-C\mathbf{z}||^2]+\alpha^{-2}||C||^2\right) = -2(I-C)\mathbb{E}[\mathbf{z}\mathbf{z}^T]+2\alpha^{-2}C. \tag{15}$$

It is then straightforward to see that the minimization of $\mathcal{L}(C)$ leads to the following solution

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

where $R = \mathbb{E}[\mathbf{z}\mathbf{z}^T]$ is the *state correlation matrix*. For $\alpha \in (0,\infty)$, the solution is well-defined, but for $\alpha = 0$ it is not. The cases of $\alpha = 0$ and $\alpha = \infty$ a will not be discussed here, but they are discussed in Section 3.8.1 of the conceptors report [5]. In simulations, the state correlation matrix R is estimated by $\hat{R} = ZZ^T/L$, where $Z = [\mathbf{z}(1)\,\mathbf{z}(2)\,...\,\mathbf{z}(L)] \in \mathbb{R}^{N\times L}$ is the state collection matrix from Section 2.1.

A few properties of C and R can be inferred and are worth mentioning.

1. C and R have the same N eigenvectors, meaning that if $R = U\Sigma U^T$ is the singular value decomposition of R then $C = USU^T$, where Σ and S contain the singular values of R and C respectively.

2. The singular values s_i of C are the normalized singular values σ_i of R and relate by $s_i = \sigma_i/(\sigma_i+\alpha^{-2})$.

3. From the property 2 it can be concluded that $0 \leq s_i \leq 1$.

2.3.2 Morphing

The dynamics of the reservoir can be morphed by conceptors. The term morphing is usually used to describe the smooth transition from one image to another by slow gradual interpolation, but here is it used to describe a smooth transition between reservoir dynamics. Conceptors can be used to morph different patterns that are stored in the reservoir. Let the reservoir be loaded with patterns i and j and let the corresponding conceptors C^i and C^j be given. Then, a mixture of the reservoir dynamics associated with $\mathbf{p}^i$ and $\mathbf{p}^j$ can be obtained by a linear combination of C^i and C^j, given by

$$\begin{aligned}\mathbf{r}(n+1) &= \tanh(W\mathbf{z}(n)+\mathbf{b}),\\ \mathbf{z}(n+1) &= \left((1-\mu)C^i+\mu C^j\right)\mathbf{r}(n+1),\end{aligned} \tag{17}$$

where $\mu \in \mathbb{R}$ is the *mixture parameter*. Notice that the mixture parameter μ is not constrained to $[0,1]$. As will be shown in the simulations in Section 4.4, conceptors are not only able to interpolate, but also extrapolate. In the conceptors report, it was shown that for two sine waves with different periods that conceptors are capable of capturing the period of the sine waves and then not only interpolate between them, but also extrapolate [5]. The periods in the example were ≈ 8.83 and 9.83, but with the extrapolation, the conceptors were able to create sine waves of periods between ≈ 7.5 and ≈ 11.9.

2.3.3 Autoconceptors

Thus far, conceptors are computed with Equation 16, after which they are stored somewhere so the dynamics of the reservoir can be constrained at a later time. This is practical and achievable for machine learning applications, where the conceptors can be written to a file. However, this is not always the case. Specifically, from a neuroscience point of view, it seems unlikely that a filter is created for every new pattern that needs to be learn, where the filter has the same size as the reservoir. This compels for a different architecture. One where the conceptors are created while the reservoir is being driven and the conceptors need not be stored.

The idea is to make a conceptor matrix time-dependent, so it can be adapted while the reservoir is being driven. The update equations in Equation 12 then change to

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n) + \mathbf{b}\right), \\ \mathbf{z}(n+1) &= C(n)\mathbf{r}(n+1), \end{aligned} \tag{18}$$

where the only difference is that C is now time-dependent. The adaptation rule for the online adaptation of $C(n)$ can be read directly from Equation 15, yielding

$$C(n+1) = C(n) + \lambda\Big(\big(I - C(n)\big)\mathbf{z}(n)\mathbf{z}^T(n) - \alpha^{-2}C(n)\Big), \tag{19}$$

where $\lambda > 0$ is the learning rate. Note that λ is constrained by the highest local curvature in the gradient landscape. This may lead to slow convergence in areas where the local curvature is lower.

It was shown that if the autoconceptor $C(n)$ converges, it will possess the same algebraic properties as a conceptor [5]. Autoconceptors comprise a large part of the conceptors report, so for more details the reader is referred there.

2.4 Diagonal Conceptors

There are two main areas in which conceptors fall short. First, conceptors are computationally expensive. As mentioned before, each pattern that needs to be stored in the reservoir, also requires a conceptor matrix, which has the same size as the reservoir. Furthermore, the conceptor matrices quickly increase in size as they scale quadratically with the number of reservoir neurons N. Therefore, conceptors could hardly be used in real-world applications where the dimension of the reservoir is large. Second, conceptors are biologically implausible. The online adaptation of an element of the conceptor matrix, denoted by C_{ij}, requires information that would biologically not all be available at the synapse of C_{ij}.

Here, an architecture is proposed that solves the above shortcomings of conceptors. The idea is to simply substitute the conceptor matrix C^j in Equation 12 with a *diagonal conceptor* matrix D^j. It will be shown that this substitution only works well if an adjustment is made in the

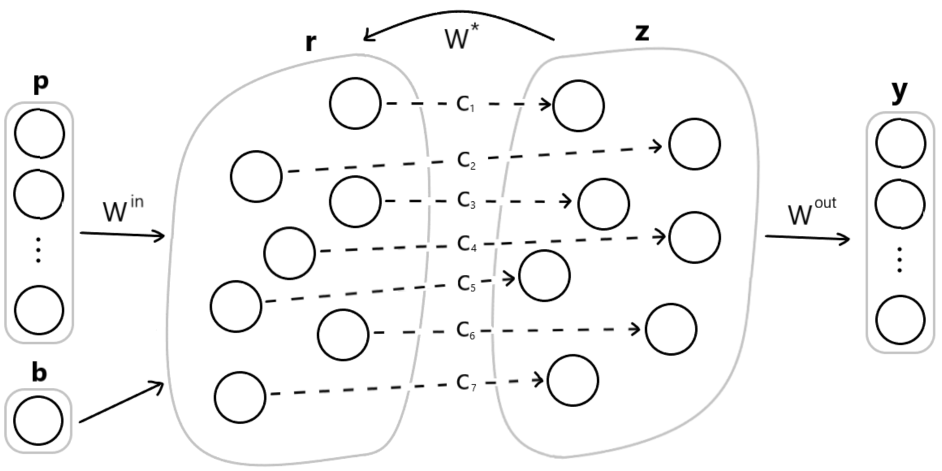

Figure 3: A visual representation of Equations 22 and 23. The reservoir is driven by pattern **p** via the input weights W^{in}. The state **r** is element-wise scaled to the state **z**. This is denoted by the dashed lines, where each dashed line depicts a scaling by a conception weight c_i. The state **z** is inserted back in the loop via W^*. The output is read from the state **z** by the output weights W^{out}.

training algorithm of conceptors, but this will be made clear in Section 3.3. This section only discusses how the diagonal conceptors would theoretically be computed.

A diagonal matrix is a matrix in which all the off-diagonal entries are zero. Let $\mathbf{c} = [c_1\, c_2\, ...\, c_N] \in \mathbb{R}^N$, then a diagonal matrix $D \in \mathbb{R}^{N\times N}$ is often denoted by

$$D = \text{diag}(\mathbf{c}) = \begin{bmatrix} c_1 & 0 & \dots & 0 \\ 0 & c_2 & \dots & 0 \\ \vdots & 0 & \ddots & \vdots \\ 0 & 0 & \dots & c_N \end{bmatrix}. \tag{20}$$

Let $D^j = \text{diag}(\mathbf{c}^j) \in \mathbb{R}^{N\times N}$ denote the diagonal conceptor associated with pattern j, where $\mathbf{c}^j = [c_1^j\, c_2^j\, ...\, c_N^j] \in \mathbb{R}^N$ is called the conception vector and its elements c_i^j, for $i = 1,2,...,N$, are called the conception weights. In Equation 12, the diagonal conceptor D^j is substituted for the conceptor matrix C^j, which yields

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n) + \mathbf{b}\right) \\ \mathbf{z}(n+1) &= D^j\mathbf{r}(n+1) \end{aligned} \tag{21}$$

and the output can be read from the reservoir according to

$$\mathbf{y}(n) = W^{out}\mathbf{z}(n). \tag{22}$$

Note that $\mathbf{z}(n)$ is now a scaling of $\mathbf{r}(n)$, where the scaling parameters are given by $\mathbf{c}^j$. A convenient way to write Equation 21 is by means of the conception weights. This gives

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n) + \mathbf{b}\right) \\ \mathbf{z}(n+1) &= \mathbf{c}^j \circ \mathbf{r}(n+1), \end{aligned} \tag{23}$$

where $\circ$ is the Hadamard, or element-wise, multiplication operator. A visualization of Equation 23 is shown in Figure 3, where a dashed line depicts an simple scaling operation. Simulations make use of Equation 23 rather than Equation 21, because $\mathbf{z}(n)$ can then be computed by vector-vector multiplication rather than matrix-vector multiplication.

Diagonal conceptors are computationally cheaper than conceptors, as they allow for vector storage and vector-vector multiplication. Furthermore, diagonal conceptors are not biologically implausible, which will be discussed further in Section 2.4.3. However, it must be noted that the diagonal conceptors architecture does not imply biological plausibility. It merely suggest that it is not biologically implausible.

2.4.1 Computing Diagonal Conceptors

For notional simplicity the superscript j will be omitted. The derivation of a conceptor matrix can be translated directly to diagonal conceptors. Moreover, instead of deriving an expression for the diagonal conceptor D, an expression for the individual conception weights c_i is derived. Similar to conceptor matrices, the diagonal conceptor associated with pattern j must leave the states associated with pattern j untouched, and while suppressing the states associated with other patterns. Again, this balance is mitigated by the parameter *aperture* denoted by α. This leads to a loss function similar to the loss function for conceptors. The loss function for a diagonal conceptors is given by

$$\mathcal{L}(D) = \mathbb{E}[||\mathbf{z} - D\mathbf{z}||^2] + \alpha^{-2}||D||^2_{fro}, \tag{24}$$

Since D is a diagonal matrix, Equation 24 can be more conveniently written element-wise, which gives

$$\begin{aligned} \mathcal{L}(c_i) &= \mathbb{E}[(z_i - c_i z_i)^2] + \alpha^{-2} c_i^2 \\ &= (1 - c_i)^2 \mathbb{E}[z_i^2] + \alpha^{-2} c_i^2. \end{aligned} \tag{25}$$

An expression for c_i is found by minimizing the loss function $\mathcal{L}(c_i)$, for which the derivative with respect to c_i must be computed. This gives

$$\frac{\partial}{\partial c_i}\left((1-c_i)^2\mathbb{E}[z_i^2] + \alpha^{-2}c_i^2\right) = -2(1-c_i)\mathbb{E}[z_i^2] + 2\alpha^{-2}c_i. \tag{26}$$

Setting this equal to 0 and solving for c_i immediately yields

$$c_i = \frac{\mathbb{E}[z_i^2]}{\mathbb{E}[z_i^2] + \alpha^{-2}}. \tag{27}$$

This expression indicates that $0 < c_i < 1$, where the boundaries 0 and 1 require a bit more attention.

First, lets consider $\mathbb{E}[z_i^2]$. Note that $\mathbb{E}[z_i^2]$ can be written as $c_i^2\mathbb{E}[r_i^2]$, where $0 < \mathbb{E}[r_i^2] < 1$, because of the hyperbolic tangent function. Therefore, $\mathbb{E}[z_i^2]$ will come close to 1, but it will never reach 1. In addition, $\mathbb{E}[z_i^2]$ will only be 0 is $c_i = 0$. Therefore, $0 \le \mathbb{E}[z_i^2] < 1$.

Second, lets consider α. In simulations, it is assumed that $0 < \alpha < \infty$, for which c_i is always well-defined. However, for completeness, one can look at the boundary values $\alpha = 0$ and $\alpha = \infty$

by considering the behavior of c_i in the limits $\alpha \downarrow 0$ and $\alpha \uparrow \infty$. Let $0 \leq \mathbb{E}[z_i^2] < 1$ be constant. Then, c_i in the limiting values of $\alpha \downarrow 0$ and $\alpha \uparrow \infty$ are given by

$$\lim_{\alpha\downarrow 0} c_i = \lim_{\alpha\downarrow 0} \frac{\mathbb{E}[z_i^2]}{\mathbb{E}[z_i^2]+\alpha^{-2}} = 0 \quad \text{and} \quad \lim_{\alpha\uparrow\infty} c_i = \lim_{\alpha\downarrow 0} \frac{\mathbb{E}[z_i^2]}{\mathbb{E}[z_i^2]+\alpha^{-2}} = 1. \tag{28}$$

There is an intuitive relation between α and $\mathbb{E}[z_i^2]$, which is clearer if Equation 27 is written as

$$c_i = \frac{1}{1+\frac{1}{\alpha^2\mathbb{E}[z_i^2]}}. \tag{29}$$

Notice that excited neurons, i.e. $\mathbb{E}[z_i^2]$ close to 1, yield larger values of c_i, hence those neuron states remain mostly untouched. Conversely, less excited neurons, i.e., $\mathbb{E}[z_i^2]$ close to 0, yield smaller values of c_i, hence those neuron states are suppressed. How much the neurons are untouched or suppressed depends on α. Increasing α also increases c_i and, conversely, decreasing α also decreases c_i. However, this behavior should not come as a surprise, as this is exactly how the loss function was designed.

2.4.2 Morphing

Morphing conceptors can be directly translated to diagonal conceptors. Let the reservoir be loaded with patterns i and j and let the associated diagonal conceptors D^i and D^j be given. Then, similar to morphing conceptors, a mixture of the reservoir dynamics associated with patterns i and j is obtained by taking a linear combination of D^i and D^j as follows:

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n)+\mathbf{b}\right), \\ \mathbf{z}(n+1) &= \left((1-\mu)D^i+\mu D^j\right)\mathbf{r}(n+1), \end{aligned} \tag{30}$$

where $\mu \in \mathbb{R}$ is the mixture parameter. As mentioned in Section 2.3.2, the mixture parameter for morphing conceptors could be used to not only interpolate between reservoir states, but also extrapolate. In the simulations in Section 4.4 it will be shown that, in the case of diagonal conceptors, interpolation is possible, but extrapolation is not possible.

2.4.3 Diagonal autoconceptors

The arguments for autoconceptors can be translated directly to diagonal conceptors, creating diagonal autoconceptors. If the diagonal conceptor matrix D is made time-dependent, it yields the following update equations:

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n)+\mathbf{b}\right), \\ \mathbf{z}(n+1) &= D(n)\mathbf{r}(n+1). \end{aligned} \tag{31}$$

The adaptation rule for $D(n) = \operatorname{diag}(\mathbf{c}(n))$, where $\mathbf{c}(n)$ is the time-dependent conception vector, can be written element-wise, and is read directly from Equation 26. This yields

$$c_i(n+1) = c_i(n) + \lambda_i\Big(\big(1-c_i(n)\big)z_i^2(n) - \alpha^{-2}c_i(n)\Big), \tag{32}$$

where $c_i(n)$ and $z_i(n)$ are the i-th elements of $\mathbf{c}(n)$ and $\mathbf{z}(n)$, respectively, and λ_i is the learning rate associated with neuron i. Note that the learning rate can be set individually for each neuron, hence lifting the constraint of a global learning rate for autoconceptors. Furthermore, the biological implausibility of autoconceptors is lifted as well, since each neuron state is updated independently of each other. Therefore, in contrast to conceptors, all the information required for updating a conception weight at a synapse is available at that synapse. This does not directly imply biological plausibility, but it certainly does not deny it. This is argued in more detail in Section 3.15 of the conceptors report and this argumentation can be used for diagonal autoconceptors as well [5], so for more information the reader is referred there.

Furthermore, the dynamics of diagonal autoconceptors are easier to analyse than the dynamics of autoconceptors, because it comprises scalars instead of matrices. A short, intuitive analysis is given here. It will only grasp a small portion of the full dynamics of the reservoir, but it will give some insight nonetheless. Writing the adaptation rule 32 a continuous-time differential equation and letting $z_i = c_i r_i$ gives

$$\dot{c}_i(t) = \big(1 - c_i(t)\big)c_i^2(t)r_i^2(t) - \alpha^{-2}c_i(t). \tag{33}$$

Setting this equal to 0 and solving for $c_i(t)$ gives

$$c_{i,\pm} = \frac{1}{2} \pm \frac{1}{2}\sqrt{1 - 4\alpha^{-2}(\mathbb{E}[r_i^2])^{-1}} \quad \text{and} \quad c_i = 0. \tag{34}$$

Upon closer inspection it turns out that only $c_{i,+}$ and $c_i = 0$ are stable solutions, which is neatly shown in Figure 4.

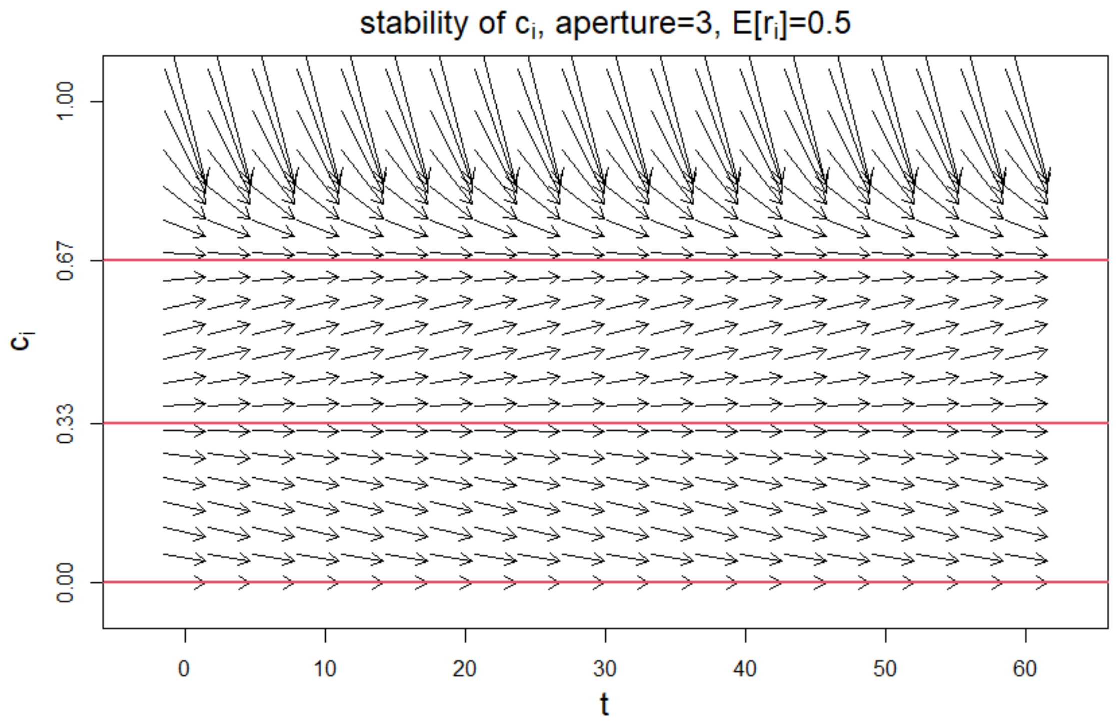

Figure 4: A direction field of Equation 33. The solutions, as computed in Equation 34, are plotted as red lines.

Figure 4 shows that over time the value of c_i will either go towards the upper red line, corresponding to $c_{i,+}$ in Equation 34, or $c_i = 0$. In can also be seen that $c_{i,-}$ is not a stable solution.

Furthermore, the curvature is much larger for values above $c_{i,+}$ than for values $c_{i,+}$, hence the convergence rate in those areas will most likely differ. Therefore, in simulations one could, for example, increase the learning rate for values of c_i smaller than 0.5 to increase the rate of convergence.

At the present moment, diagonal autoconceptors remain mostly unexplored. A more comprehensive analysis is required to fully understand the dynamics of this system.

3 Simulation Methods

Several simulations will be discussed in Section 4 to demonstrate the capabilities of diagonal conceptors compared to conceptors. This section describes how to set up those simulations. In addition, this section describes how to train conceptors and diagonal conceptors. The last part of this section describes how the accuracy of the conceptors and diagonal conceptors is determined.

3.1 Initializing the Parameters

When setting up computer simulations, the RNN must be initialized appropriately. How many connections between reservoir neurons are optimal and how should they be weighted? From what distribution should they be drawn? The same questions can be asked for the input weights and the bias vector. This section introduces some of the rules of thumb for initializing the simulation parameters.

Remember that the system is given by the equations

$$\mathbf{x}^j(n+1) = \tanh\left(W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}\right) \tag{35}$$

where $\mathbf{x}^j(n) \in \mathbb{R}^N$ is the reservoir state associated with pattern j at time step n, $\mathbf{p}^j(n) \in \mathbb{R}^M$ is the n-th time step of pattern j, $\mathbf{b} \in \mathbb{R}^N$ is the bias vector, $W^* \in \mathbb{R}^{N\times N}$ are the reservoir weights, and $W^{in} \in \mathbb{R}^{N\times M}$. The simulations in this report will use a slightly different system consisting of so-called *leaky-integrated* neurons, which changes Equation 35 to

$$\begin{aligned}\tilde{\mathbf{x}}^j(n+1) &= \tanh\left(W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}\right)\\ \mathbf{x}^j(n+1) &= (1-\alpha_l)\mathbf{x}^j(n) + \alpha_l\tilde{\mathbf{x}}^j(n+1)\end{aligned} \tag{36}$$

where $0 \leq \alpha_l \leq 1$ is the leaking rate. Leaky-integrated neurons are an extension of the non-leaky integrated neurons that were described so far, since Equation 35 is recovered for $\alpha_l = 1$ and they offer better fine-tuning of the reservoir dynamics [35]. The leaking rate determines how much of the previous state is still remembered in computing the next state. Adding the leaking rate into the update equation does not change anything that has been discussed up until this point. Intuitively, lowering the leaking rate results in a smoother output pattern, since big deviations are slightly suppressed.

The elements of W^{in}, $\mathbf{b}$ and W^* are all drawn from the standard normal distribution and scaled appropriately. The input weights matrix and bias vector are full matrices, whereas the reservoir weights matrix is a sparse matrix. The sparsity of the reservoir weights matrix is determined by the number of neurons and can be chosen in different manners [36]. For simulations discussed in this report, the sparsity was set to 1 if $N < 20$ and $10/N$ if $N \geq 20$, where N is the number of reservoir neurons.

The scaling of W^{in} and $\mathbf{b}$ mostly determines how much the neurons in the reservoir are excited. For example, lower scalings of W^{in} and $\mathbf{b}$ result in less excited states. Intuitively, if the neurons are too excited their values will be close to -1 or 1, which will make the neurons act more in a binary switching way. This is not ideal for smooth patterns. On the other hand, if the neurons are not excited enough, the their values will be close to 0, which is desired for very linear tasks, but not for non-linear tasks.

Furthermore, the spectral radius of W^* must be small enough to ensure that the RNN possesses the *echo state property* [37]. The echo state property ensures that the RNN ”forgets” the initial state. Often, the initial state of a network is not known and will simply be set to e.g. the all-zero vector. Therefore, the network must be given some time to *washout* the initial state of the system. This is ensured via a washout period in the simulation. However, the initial state of the RNN must also be forgotten and the echo state property makes sure this happens. The length of the washout period, denoted by $n_{washout}$, can be found by driving the reservoir from two initial states, e.g. all-zero vector and the all-ones vector, and then determining how long it takes for the two reservoir states to converge to the same state. Note that $n_{washout}$ depends on the parameters, e.g., a lower leaking rate will require a longer washout period.

Setting the parameters mentioned above is mainly a matter of trial and error. In order to get some intuition into the parameters, it is advised to manually adjust a single parameter and see how it changes the dynamics of the reservoir or the outputs. It is possible to implement parameter optimization algorithms, but when the output requires visual inspection it may be better to try manually adjusting the parameters one-by-one. Much research has been conducted in finding good scaling parameters, which has been concisely summarized in the paper by M. Lukoševičius called *A Practical Guide to Applying Echo State Networks* [36]. For more intuition on setting up the parameters, the reader is referred to that paper.

3.2 Training Conceptors

This section describes how conceptors are trained. Let there be a set of patterns $\mathbf{P} = \{\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^p\}$, where $\mathbf{p}^j \in \mathbb{R}^M$ for all j. Initialize the following:

- number of neurons in the reservoir (N),
- input weights ($W^{in} \in \mathbb{R}^{N \times M}$): Random values drawn from the standard normal distribution and scaled appropriately,
- bias ($\mathbf{b} \in \mathbb{R}^N$): Random values drawn from the standard normal distribution and scaled appropriately,
- initial reservoir weights ($W^* \in \mathbb{R}^{N \times N}$): Sparse matrix with random values drawn from the standard normal distribution and scaled appropriately,
- leaking rate α_l.

Washout
Drive the reservoir with pattern j for $n_{washout}$ time steps according to Equation 36. All information during this period is discarded. The initial state can be initialized as any random vector in $[-1,1]^N$, but usually $\mathbf{x}^j(0) = \mathbf{0}$ for simplicity.

Learning
The reservoir is continued to be driven for another n_{learn} time steps. During the learning phase, the aim is to compute:

- $R^j = X^j(X^j)^T/n_{learn}$, where X^j is the state collection matrix,
- W, such that $W\mathbf{x}^j(n) \approx W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n)$ over all n, j,
- W^{out}, such that $W^{out}\mathbf{x}^j(n) \approx \mathbf{p}^j(n)$ over all n, j,

so the following vectors are collected:

- $\mathbf{x}^j(n)$, to create: X^j, where $(X^j)_{*,n} = \mathbf{x}^j(n_{washout} + n)$,
- $\mathbf{x}^j(n-1)$, to create: $\tilde{X}^j$, where $(\tilde{X}^j)_{*,n} = \mathbf{x}^j(n_{washout} + n - 1)$,
- $\mathbf{p}^j(n)$, to create: P^j, where $(P^j)_{*,n} = \mathbf{p}^j(n_{washout} + n)$
- $W^*\mathbf{x}^j(n) + W^{in}\mathbf{p}^j(n)$, to create: W^j_{target}, where $(W^j_{target})_{*,n} = W^*\mathbf{x}^j(n_{washout}+n) + W^{in}\mathbf{p}^j(n_{washout} + n)$.

In the above, $(.)_{*,n}$ is used to denote the n-th column of a matrix.

Computing Conceptors
The washout and learning phase are repeated for all patterns $j = 1, 2, ..., P$, which yields, during which the state correlation matrices R^j are collected. Furthermore, the aperture α^j is set for each pattern j. The conceptor matrix C^j is computed with

$$C^j = R^j(R^j + (\alpha^j)^{-2}I)^{-1}. \tag{37}$$

The aperture is chosen heuristically, but there exist optimization techniques, see Section 3.8 in [5]. These techniques will not be employed in this report.

Computing W and W^{out}
The collected matrices X^j, $\tilde{X}^j$, P^j, and W^j_{target} are concatenated, which gives $X = [X^1|X^2|...|X^p]$, $\tilde{X} = [\tilde{X}^1|\tilde{X}^2|...|\tilde{X}^p]$, $P = [P^1|P^2|...|P^p]$, and $W_{target} = [W^1_{target}|W^2_{target}|\cdots|W^p_{target}]$. Following Section 2.2, the patterns are stored and the output weights are computed via ridge regression as follows:

$$W = (\tilde{X}\tilde{X}^T + \varrho^W I)^{-1}\tilde{Z}W^T_{target}, \quad W^{out} = (XX^T + \varrho^{W^{out}} I)^{-1}XP^T, \tag{38}$$

where ϱ^W and $\varrho^{W^{out}}$ are the regularization constants for W and W^{out} respectively, and I is the identity matrix. The regularization constants can influence the output of the simulation greatly. The regularization constants determine how much outliers are suppressed. In other words, a small regularization constant may result in overfitting, whereas a large regularization constant may lead to underfitting. The regularization constants are essentially another two parameters in the simulation and they need to be chosen appropriately. Again, this is done heuristically.

Self-generating reservoir
After W, W^{out}, and C^j have been computed, the reservoir is able to self-generate any pattern j by inserting C^j in the update equation, yielding

$$\begin{aligned} \mathbf{r}(n+1) &= (1-\alpha_l)\mathbf{r}(n) + \alpha_l \tanh(W\mathbf{z}(n) + \mathbf{b}) \\ \mathbf{z}(n+1) &= C^j\mathbf{r}(n+1) \end{aligned} \tag{39}$$

and

$$\mathbf{y}(n) = W^{out}\mathbf{z}(n), \tag{40}$$

where $\mathbf{y}$ will behave like pattern j.

3.3 Training Diagonal Conceptors

As mentioned in previous parts of this report, the training scheme of diagonal conceptors differs from the training scheme for conceptors. This section describes how diagonal conceptors are trained and where it differs from the training scheme of conceptors. The training commences with initializing the following:

- number of neurons in reservoir (N),
- input weights ($W^{in} \in \mathbb{R}^{N \times M}$): Random values drawn from the standard normal distribution and scaled appropriately,
- bias ($\mathbf{b} \in \mathbb{R}^{N}$): Random values drawn from the standard normal distribution and scaled appropriately,
- initial reservoir weights ($W^{*} \in \mathbb{R}^{N \times N}$): Sparse matrix with random values drawn from the standard normal distribution and scaled appropriately,
- leaking rate α_l,
- initial diagonal conceptors ($D_0^j = \mathrm{diag}(\mathbf{c_0}^j)$, where $\mathbf{c_0}^j \in [0,1]^N$): Conception weights c_i^j are random values drawn from the uniform distribution on the interval $[0,1]$,

Washout
The reservoir is driven with pattern j for $n_{washout}$ time steps according to the following update equations:

$$\begin{aligned} \mathbf{r}^j(n+1) &= (1-\alpha_l)\mathbf{r}^j(n) + \alpha_l \tanh\left(W^{*}\mathbf{z}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}\right), \\ \mathbf{z}^j(n+1) &= D_0^j \mathbf{r}(n+1), \end{aligned} \tag{41}$$

where the initial state $\mathbf{z}^j(0)$ is, again, often simply set to the all-zero vector. Note that the initial diagonal conceptors are in the loop from the start. It is this vital step in this training algorithm that makes training diagonal conceptors different from training conceptors.

Intuitively, having the initial diagonal conceptors in the loop from the start creates a different network for each pattern j, adding more nonlinearity to train to. The states $\mathbf{r}(n)$ are randomly scaled to a new area in state space and the diagonal conceptors are trained on those new areas. This is necessary as diagonal conceptors have significantly less degrees of freedom than conceptors. Assume that the area in state space occupied by pattern j has a lot of similarity with another area in state space occupied by another pattern $i \neq j$. Then the large number of degrees of freedom of conceptors makes it possible to distinguish between the different areas regardless. Diagonal conceptors have less degrees of freedom, so optimizing the diagonal conceptors will most likely be very difficult. Instead, the similar areas in state space are scaled randomly by which they are ”pulled apart”. This will be shown by means of an example in Section 5.1.1. Having the randomly initialized diagonal conceptors in the loop from the start is a crucial step in this training algorithm.

Adaptation
After the washout period, the reservoir is continued to be driven for n_{adapt} time steps according

to Equation 41, but now the states $\mathbf{z}^j(n)$ are collected in the state collection matrix

$$Z^j = \begin{bmatrix}\mathbf{z}^j(n_{washout}+1) & \mathbf{z}^j(n_{washout}+2) & \dots & \mathbf{z}^j(n_{washout}+n_{adapt})\end{bmatrix} \in \mathbb{R}^{N\times n_{adapt}}.$$

As shown in Section 2.4, the conception weights c_i^j can be computed element-wise by element-wise

$$c_i^j = \frac{\mathbb{E}[(z_i^j)^2]}{\mathbb{E}[(z_i^j)^2]+(\alpha^j)^{-2}}, \tag{42}$$

where z_i^j is the i-th element of $\mathbf{z}^j$ and α^j is the aperture for pattern j. Note that in simulations, $\mathbf{c}^j$ is computed in a vectorized manner, which gives

$$\mathbf{c}^j = \mathbb{E}[\mathbf{z}^j \circ \mathbf{z}^j] \oslash (\mathbb{E}[\mathbf{z}^j \circ \mathbf{z}^j] + (\alpha^j)^{-2}), \tag{43}$$

where $\circ$ is the Hadamard product, or the element-wise multiplication operator, and $\oslash$ is the element-wise division operator. Furthermore, $\mathbb{E}[\mathbf{z}^j \circ \mathbf{z}^j]$ is approximated by

$$\mathbb{E}[\mathbf{z}^j \circ \mathbf{z}^j] \approx \frac{1}{n_{adapt}} \sum_{l}^{n_{adapt}} (Z^j \circ Z^j)_{*,l}, \tag{44}$$

which states that Z^j is multiplied with itself element-wise and then the mean of each row is taken. $(Z^j \circ Z^j)_{*,l}$ denotes the l-th column of $Z^j \circ Z^j$.

Learning
The reservoir is continued to be driven for another n_{learn} time steps, but now with the newly computed diagonal conceptors $D^j = \mathrm{diag}(\mathbf{c}^j)$ in the loop rather than D_0^j. This gives the new, slightly different update equations:

$$\begin{aligned}\mathbf{r}^j(n+1) &= (1-\alpha_l)\mathbf{r}^j(n) + \alpha_l \tanh(W^*\mathbf{z}^j(n) + W^{in}\mathbf{p}^j(n+1) + \mathbf{b}),\\ \mathbf{z}^j(n+1) &= D^j\mathbf{r}(n+1).\end{aligned} \tag{45}$$

During the learning phase, the goal is to compute:

- W, such that $W\mathbf{z}^j(n) \approx W^*\mathbf{z}^j(n) + W^{in}\mathbf{p}^j(n)$ over all n, j,
- W^{out}, such that $W^{out}\mathbf{z}^j(n) \approx \mathbf{p}^j(n)$ over all n, j,

so the following vectors are collected:

- $\mathbf{r}^j(n)$, to create: X^j, where $(X^j)_{*,n} = \mathbf{r}^j(n_0+n)$,
- $\mathbf{z}^j(n-1)$, to create: $\tilde{Z}^j$, where $(\tilde{Z}^j)_{*,n} = \mathbf{z}^j(n_0+n-1)$,
- $\mathbf{p}^j(n)$, to create: P^j, where $(P^j)_{*,n} = \mathbf{p}^j(n_0+n)$,
- $W^*\mathbf{z}^j(n) + W^{in}\mathbf{p}^j(n)$, to create: W^j_{target}, where $(W^j_{target})_{*,n} = W^*\mathbf{z}^j(n_0+n) + W^{in}\mathbf{p}^j(n_0+n)$,

where $n_0 = n_{washout} + n_{adapt}$ for simplicity and $(.)_{*,n}$ denotes the n-th column of a matrix.

Computing W and W^{out}

The washout, adaptation, and learning period are repeated for all patterns $j = 1, 2, ..., p$ and the collected matrices are concatenated. This gives $X = [X^1|X^2|...|X^p]$, $\tilde{Z} = [\tilde{Z}^1|\tilde{Z}^2|...|\tilde{Z}^p]$, $P = [P^1|P^2|...|P^p]$, and $W_{target} = [W^1_{target}|W^2_{target}|...|W^p_{target}]$. Using ridge regression, the new reservoir weights W and the output weights W^{out} are then computed as follows:

$$W = (\tilde{Z}\tilde{Z}^T + \varrho^W I)^{-1}\tilde{Z}W^T_{target}, \quad W^{out} = (XX^T + \varrho^{W^{out}} I)^{-1} X P^T, \tag{46}$$

where ϱ^W and $\varrho^{W^{out}}$ are the regularization constants for W and W^{out} respectively, and I is the identity matrix.

Self-generating reservoir

After computing W and W^{out}, the reservoir can self-generate pattern j by letting the reservoir run according to the update equations

$$\begin{aligned} \mathbf{r}^j(n+1) &= (1-\alpha_l)\mathbf{r}^j(n) + \alpha_l \tanh(W\mathbf{z}^j(n) + \mathbf{b}), \\ \mathbf{z}^j(n+1) &= D^j \mathbf{r}(n+1), \end{aligned} \tag{47}$$

where the output can be read by

$$\mathbf{y}(n+1) = W^{out}\mathbf{r}(n). \tag{48}$$

The output $\mathbf{y}$ will behave like pattern j.

3.4 Pattern Comparison

When comparing a network-generated pattern $\mathbf{y}$ and a target pattern $\mathbf{p}$, the first step is always visual inspection. Especially when choosing the simulation parameters, it is useful to see how the network-generated pattern compares to the target pattern. The second step is quantifying the comparison. This is practical in case the patterns are high-dimensional or many patterns are compared simultaneously. In this report, the accuracy of the match between the network-generated patterns and the target patterns will be compared by the Normalized Root Mean Square Error (NRMSE). Let $\mathbf{y} \in \mathbb{R}^L$ be the observed pattern and let $\mathbf{p} \in \mathbb{R}^L$ be the target pattern. The NRMSE of $\mathbf{y}$ and $\mathbf{p}$ is given by

$$NRMSE(\mathbf{y}, \mathbf{p}) = \sqrt{\frac{\frac{1}{L}\sum_{n=1}^{L}(\mathbf{y}(n) - \mathbf{p}(n))^2}{\mathrm{Var}(\mathbf{p})}}, \tag{49}$$

where $\mathrm{Var}(\mathbf{p})$ is the variance of $\mathbf{p}$. The NRMSE is a normalized version of the Root Mean Square Error (RMSE):

$$RMSE(\mathbf{y}, \mathbf{p}) = \sqrt{\frac{1}{L}\sum_{n=1}^{L}(\mathbf{y}(n) - \mathbf{p}(n))^2}. \tag{50}$$

Note that NRMSE $= 1$ if $\mathbf{y}$ is equal to the mean of $\mathbf{p}$ for all n. This indicates that a reasonable model should achieve a NRMSE of between 0 and 1. In this report, a simulation has achieved good accuracy if the NRMSE between the outputted pattern and the target pattern is 0.1 or lower.

If the observed pattern $\mathbf{y} \in \mathbb{R}^{L \times M}$ and the target pattern $\mathbf{p} \in \mathbb{R}^{L \times M}$, are higher-dimensional, then the NRMSE is taken for each column. This yields an M-dimensional vector of NRMSEs. One could simply take the mean over all values. However, if the observed pattern has a small NRMSE in one column, but a large NRMSE in another column, this information would get lost in the mean. Therefore, when comparing higher-dimensional patterns, it is more informative to not only take the mean, but also show the smallest and largest NRMSE values, and the standard deviation among all NRMSE values. In this case, a simulation has achieved good accuracy if the mean over all NRMSEs is 0.1 or lower and the standard deviation is not too large.

4 Simulations

As mentioned in the previous sections, conceptors were first introduced in the conceptor report [5]. In this report, several examples with different patterns are used to highlight the numerous qualities of conceptors. It is for that reason that the patterns used in this section are taken from those examples. This allows for a direct comparison between conceptors and diagonal conceptors. Therefore, this section comprises three examples. A simulation with conceptors and a simulation with diagonal conceptors was conducted for each example. The findings are reported in the first three parts of this section. In addition, a morphing simulation with conceptors as well as a morphing simulation with diagonal conceptors was conducted. This is reported in the last part of this section. It should be noted that this section does not go into elaborate detail about the results and implications of the simulations, as those will be discussed in Section 5.

The patterns used in the three examples increase in complexity. The first example consists of four periodic patterns, two irrational-periodic sine waves and two 5-periodic random points. This example serves as a introductory demonstration, which also shows how similar patterns can still be distinguished by both conceptors and diagonal conceptors. The second example comprises four patterns sampled from chaotic attractors, the Lorenz, Rössler, Mackey-Glass, and Hénon attractor. Chaotic attractors are intrinsically unpredictable, which makes them a fitting challenge for conceptors as well as diagonal conceptors. In this example, the change in dynamics due to the aperture is highlighted. The last example models a collection of human motions and it is the most difficult out of the three examples. It comprises 15 61-dimensional patterns of lengths between roughly 200 and 900 time steps. The patterns also vary in smoothness and complexity.

All the simulations in this section were implemented in R and the source code can be found here.

4.1 Periodic Patterns

A function f is periodic if $f(x+k) = f(x)$ for some constant k. For discrete-time patterns this means that the pattern is either *integer-periodic* or *irrational-periodic*. If a pattern is integer-periodic, there exists a positive integer k such that $\mathbf{p}(n+k) = \mathbf{p}(n)$. If a pattern is irrational-periodic, the pattern is sampled with a sample interval $d \in \mathbb{Z}_{>0}$ from a continuous-time signal with periodicity $k \in \mathbb{R}_{>0}$ and the ratio d/k is irrational. An integer-periodic pattern with period k will occupy k points in state space, as it will simply re-visit those points after every cycle. An irrational-periodic pattern, on the other hand, will occupy a more complex set of points in state space. For example, a one-dimensional irrational-periodic pattern will occupy a set of points that can be characterized topologically as a one-dimensional cycle that is homeomorphic to the unit cycle in $\mathbb{R}^2$ [5].

Let $\mathbf{p}^1$ and $\mathbf{p}^2$ be sampled from a sine wave with period ≈ 8.83 and ≈ 9.83, respectively. They are irrational-periodic. Let $\mathbf{p}^3$ be a random 5-periodic pattern and let $\mathbf{p}^4$ be a slight variation of $\mathbf{p}^3$. The set of patterns is given by $\mathbf{P} = \{\mathbf{p}^1, \mathbf{p}^2, \mathbf{p}^3, \mathbf{p}^4\}$. The patterns are 1-dimensional and are initialized to a length of $L = 5000$ time steps. An interval of 30 time steps is shown in the left column of Figure 5.

For both the conceptors and diagonal conceptors simulation, the reservoir is initialized with the number of reservoir neurons $N = 100$, leaking rate $\alpha_l = 1$, and regularization constants $\varrho^W = 0.001$ and $\varrho^{W^{out}} = 0$. The reservoir weights W^* and the input weights W^{in} are both scaled by 1 and the bias $\mathbf{b}$ is scaled by 0.2. The washout period is 200 time steps. In the case of conceptors, the apertures was set to $\alpha^j = 100$ and for diagonal conceptors it was set to $\alpha^j = 8$, for all j. The length of the learning period for conceptors was set to $n_{learn} = L - n_{washout} = 4800$ time steps. For the diagonal conceptors, the adaptation period was set to $n_{adapt} = 500$, and the learning period was set to $n_{learn} = L - n_{washout} - n_{adapt} = 4400$ time steps.

The goal of this example is to demonstrate that diagonal conceptors can distinguish similar patterns with just as little effort as conceptors.

4.1.1 Conceptors

Lets start with the conceptors simulation. The reservoir was driven with the four patterns and the conceptors were trained according to the training algorithm described in Section 3.2. The conceptor matrices were stored and used to self-generate the four patterns. The self-generated patterns were phase-aligned with the original patterns and the accuracy between the outputted patterns and the original patterns was measured by NRMSE, as described in Section 3.4.

Both the outputted patterns and the target patterns are plotted on a small interval, which can be seen in the first column of Figure 5, where the target pattern (driver) is the black line, the conceptors-generated pattern ($\mathbf{y}$) is the red dashed line, and the NRMSE is shown in the bottom left corner. In the second column of Figure 5, the neuron state of four randomly chosen neurons, in the same interval as the first column, is plotted. From the similarity between the neuron state and the input pattern as well as the periodicity of the neuron state, it can be inferred that the dynamics of the patterns have been captured by the reservoir.

To illustrate the difference between the integer- and irrational-periodic patterns, the energies of the principal components of R are plotted in the third and fourth columns of Figure 5. The energies of the principal components of R are also known as the singular values of R. Let $R = U\Sigma U^T$ be the singular value decomposition of R, then the singular values of R are given by the diagonal of the diagonal matrix Σ. The third column of Figure 5 neatly shows that for the irrational-periodic patterns, the principal components span all of R^N, whereas the principal components of the 5-periodic patterns are only non-zero in 5 directions, as the reservoir periodically visits the same 5 states. Nevertheless, the fourth column shows that most of the principal components energy is concentrated on a few principal directions for all four patterns. The small energies for the 5-periodic patterns are due to numerical errors.

Adjusting the simulation parameters may result in slightly better or worse performance. Specifically, the conceptors still achieve good accuracy, as defined in Section 3.4, when the parameters are individually modified in the following ranges:

- input weights scaling: $[0.6, 2]$,
- reservoir weights scaling: $[0.8, 1]$,
- bias scaling: $[0.2, 0.6]$,

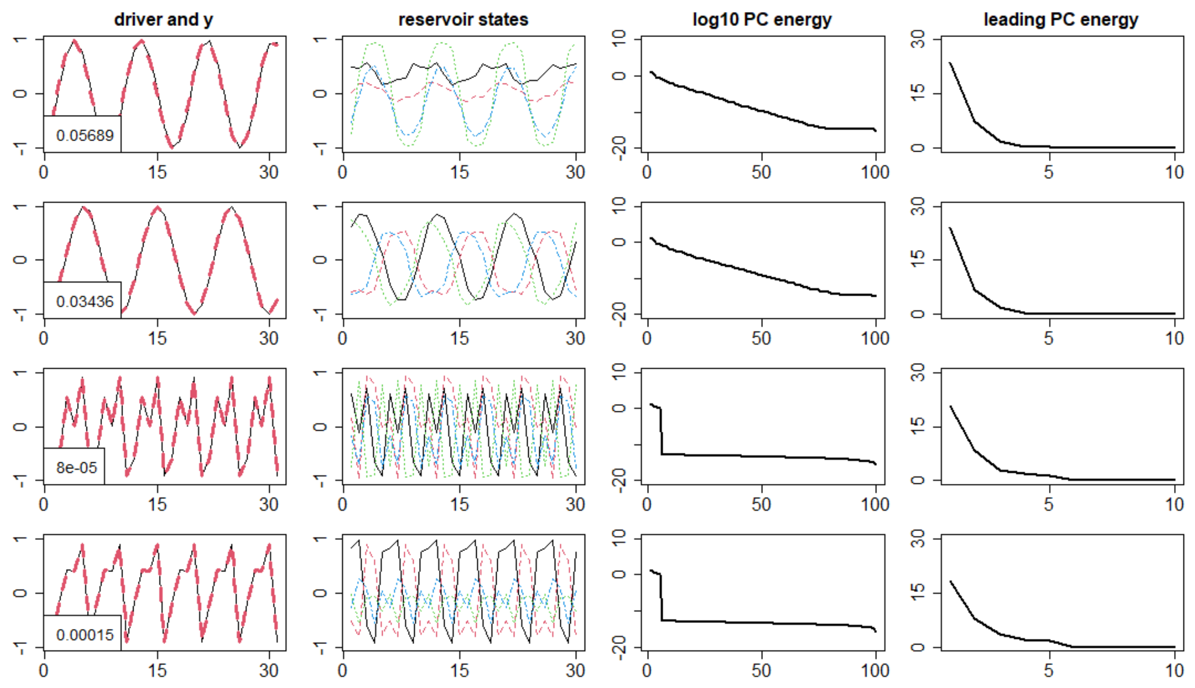

Figure 5: Row j depicts the results for pattern j. The left column shows the first 30 time steps of the original patterns (black lines) and the conceptor-generated patterns (red dashed lines) and the NRMSE between patterns over all time steps. The second column shows the first 30 time steps of four randomly chosen neuron states. The third column shows the $\log_{10}$ principal components energies and the fourth column shows the leading principal components energies of the state correlation matrix R^j.

- leaking rate: $[0.9, 1]$,
- regularization constant W^{out}: $[0, 1000]$,
- regularization constant W: $[0, 0.01]$,
- aperture: $[10, 10000]$.

It must be noted that these ranges indicate that good accuracy is still achieved if a single parameter is adjusted to anywhere in its given range, while the other parameters remain unchanged. Regardless, there are combinations of parameters possible that lie outside of these ranges, e.g., the following parameters also yield good accuracy: W^{in} scaling 0.5, W scaling 1, $\mathbf{b}$ scaling 0.7, $\alpha_l = 1$, $\varrho^{W^{out}} = 0.000001$, regularization constant $\varrho^{W} = 0.001$, $\alpha^j = 100$ for all j.

4.1.2 Diagonal Conceptors

The reservoir is driven with the four patterns. The diagonal conceptors are computed and the patterns are stored in the reservoir. Afterwards, the reservoir is able to self-generate the patterns with the diagonal conceptors. The self-generated patterns are phase-aligned and compared to the original patterns using NRMSE, see Section 3.4. The results are depicted in Figure 6 and are depicted in a similar way as Figure 5, which allows for direct comparison. Since the meaning of every column has already been discussed in Section 4.1.1, the interpretation of Figure 6 will not be described again.

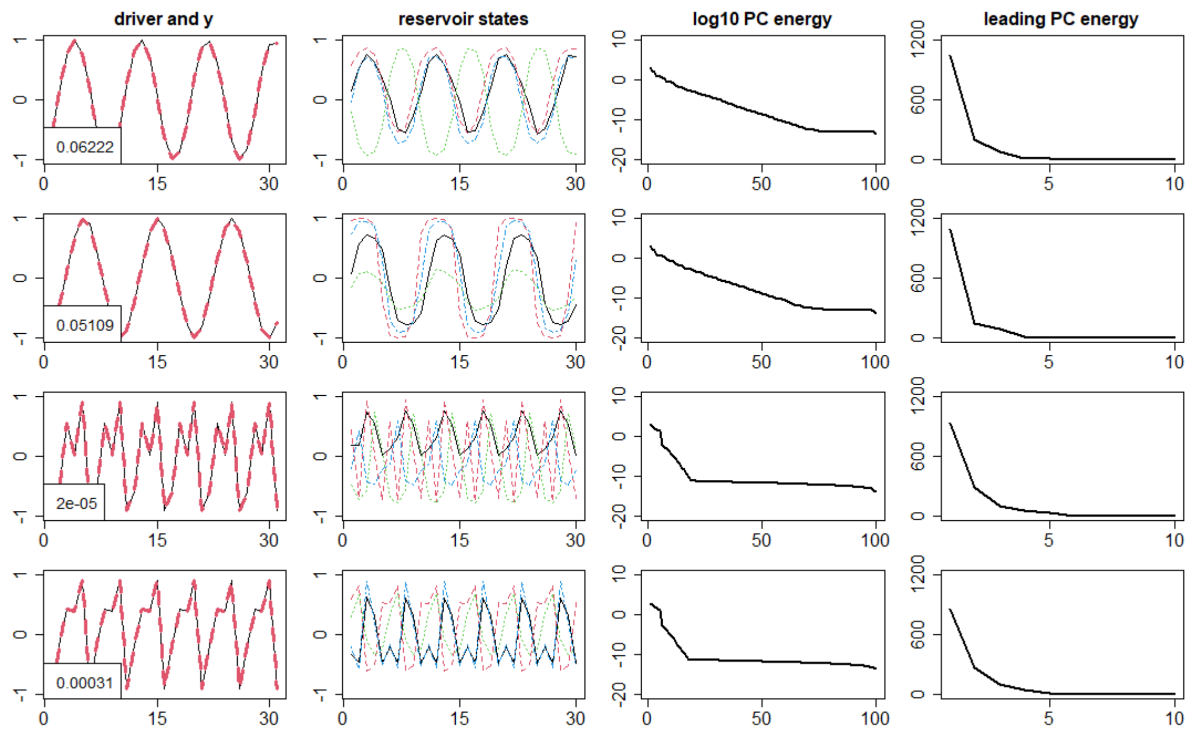

Figure 6: Row j depicts the results for a pattern j. The left column shows the first 30 time steps of the target pattern (black lines) and the conceptor-generated patterns (red dashed lines) and the NRMSE between patterns over all time steps in the bottom left corner. The second column shows the first 30 time steps of four randomly chosen neuron states. The third column shows the $\log_{10}$ principal components energies and the fourth column shows the leading principal components energies of the state correlation matrix R^j.

From the first column of Figure 6 it is evident that diagonal conceptors achieve equally good accuracy as conceptors. The second and third column of Figure 6 are approximately identical to the second and third column of Figure 5. However, the fourth column shows that the leading principal component energies are larger for diagonal conceptors than for conceptors. Upon further inspection, it turns out that the principal components are simply scaled, meaning that the cumulative percentage of the singular values is roughly the same for diagonal conceptors as for conceptors. This is not surprising since the reservoir states are scaled as well.

Similar to the conceptors simulation, individual parameters were adjusted to see how it would affect the accuracy. The diagonal conceptors still achieve desirable results when the parameters are individually modified in the following ranges:

- input weights scaling: $[0.8, 1.05]$ and $[1.15, 1.4]$,
- reservoir weights scaling: $[0.9, 1]$,
- bias scaling: $[0.15, 0.35]$,

- leaking rate: $[0.85, 1]$,
- regularization constant W^{out}: $[0, 10000]$,
- regularization constant W: $(0, 0.01]$,
- aperture: $[6, 9]$.

For the regularization constant for W it should be noted that 0 is not included in the interval due to the singularity in computing the inverse matrix in ridge regression. In addition, it is remarkable that the input weights scaling is split up into two intervals. For values $1.05 - 1.15$, the simulation yielded diagonal conceptors that were unstable.

Again, the above intervals only indicate that the simulation still yields satisfactory results if one parameter at a time is modified. Other parameter combinations that yield good accuracy are possible, e.g., W^{in} scaling 0.6, W scaling 1, $\mathbf{b}$ scaling 0.2, $\alpha_l = 1$, $\varrho^{W^{out}} = 0$, $\varrho^{W} = 0.000001$, $\alpha^j = 25$ for all j.

4.2 Chaotic Attractors

This example comprises four chaotic patterns sampled from the well-known Rössler, Lorenz, Mackey-Glass and Hénon attractors. The patterns are intrinsically chaotic, hence fragile, and it will require careful fine-tuning of the dynamics of the reservoir in combination with the aperture of the conceptors and diagonal conceptors. As this section serves to demonstrate how diagonal conceptors work with more fragile patterns, going into detail about how these chaotic attractors were generated is a bit beyond the purpose of this section. Therefore, for more details, the reader is referred to Section 4.2 of the conceptors report [5]. The four chaotic patterns are shown in Figure 7.

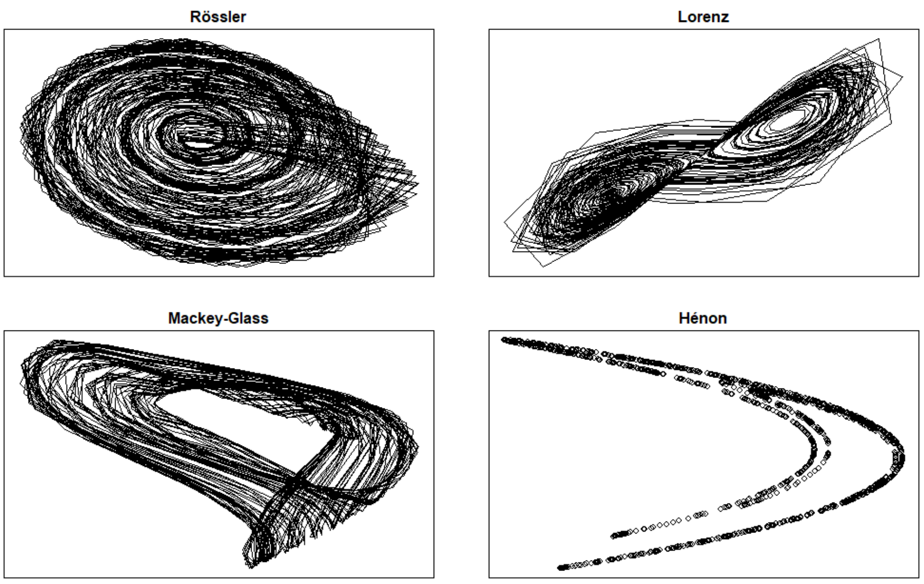

Figure 7: Four chaotic patterns, sampled from the Rössler (top left), Lorenz (top right), Mackey-Glass (bottom left) and Hénon (bottom right) attractors.

Let the set of patterns be denoted by $\mathbf{P} = \{\mathbf{p^1}, \mathbf{p^2}, \mathbf{p^3}, \mathbf{p^4}\}$, where $\mathbf{p^1}, \mathbf{p^2}, \mathbf{p^3}, \mathbf{p^4} \in \mathbb{R}^2$ were sampled from the *Rössler*, *Lorenz*, *Mackey-Glass*, and *Hénon* attractors, respectively. Each pattern has length $L = 1500$. The number of neurons in the reservoir was set to $N = 500$. The input weights W^{in} and the reservoir weights W^* were both scaled by 1.5, the bias $\mathbf{b}$ was scaled by 1. The leaking rate was set to $\alpha_l = 1$. The washout period was set to $n_{washout} = 100$ time steps, and for diagonal conceptors the adaptation period was set to $n_{adapt} = 400$, yielding a learning period of $n_{learn} = L - n_{washout} - n_{adapt} = 1000$ for diagonal conceptors and $n_{learn} = L - n_{washout} = 1400$ for conceptors. For diagonal conceptors, the apertures were set to $\alpha^1 = 10$, $\alpha^2 = 6$, $\alpha^3 = 9$, $\alpha^4 = 5$, where α^j corresponds to pattern j. For conceptors, the apertures were set to $\alpha^1 = 140$, $\alpha^2 = 50$, $\alpha^3 = 140$ and $\alpha^4 = 20$. The regularization constant for computing the output weights was set to $\varrho^{W^{out}} = 0$ for both conceptors and diagonal conceptors and the regularization constant for the recomputing the reservoir weights was set to $\varrho^W = 0.1$ for conceptors and $\varrho^W = 0.01$ for diagonal conceptors.

The chaotic behavior of the patterns requires a carefully chosen aperture. Therefore, the goal of this example is to demonstrate how the role of the aperture for diagonal conceptors differs from the role of the aperture for conceptors. It will become evident that the control that the aperture offers for conceptors is slightly lost for diagonal conceptors. The reason is that the aperture is more entangled in the training procedure of diagonal conceptors, because the computation of the diagonal conceptors is executed before the learning period. Therefore, the patterns are only stored in the reservoir after the aperture has already played its part, so W and W^{out} are computed based on the states that are influenced by the aperture. This leads to a delicate interplay between the regularization constants ϱ^W and $\varrho^{W^{out}}$, and the aperture. Consequently, changing the aperture for diagonal conceptor j may result in a change in diagonal conceptor $i \neq j$. Conceptors do not have this problem, because the conceptor matrices are only computed after the learning period. Therefore, adjusting the aperture for conceptor j will only affect conceptor j, which offers more control with the aperture.

4.2.1 Conceptors

The conceptors were trained according to the procedure described in Section 3.2. The resulting conceptor matrices were then used to self-generate the stored patterns. The self-generated patterns were compared to the original patterns by visual inspection, because the self-generated patterns are not expected to align perfectly with their target patterns. The target patterns are chaotic in nature, so if the self-generated patterns would be able to predict the chaotic patterns it would defy the unpredictability of the chaotic patterns. Nevertheless, the self-generated patterns are expected to capture the characteristic of the target patterns. In Figure 8 it can be seen that the characteristics are indeed captured. The black patterns are the target patterns and the red patterns are the conceptor-generated patterns. Furthermore, they are stable, meaning that the self-generated patterns will continue to behave like this for a long period of time and they are robust to random starting reservoir states.

In the periodic patterns simulation, the individual parameters were slightly adjusted in order to find in which ranges the self-generated patterns would still be satisfactory. However, in the case of chaotic attractors, this is a bit more complicated. The reason is that the aperture must be handled with care, so changing a parameter such as the bias scaling leads to a necessary adjustment in the apertures. For example, adjusting the input weights scaling from 1.5 to 1.4

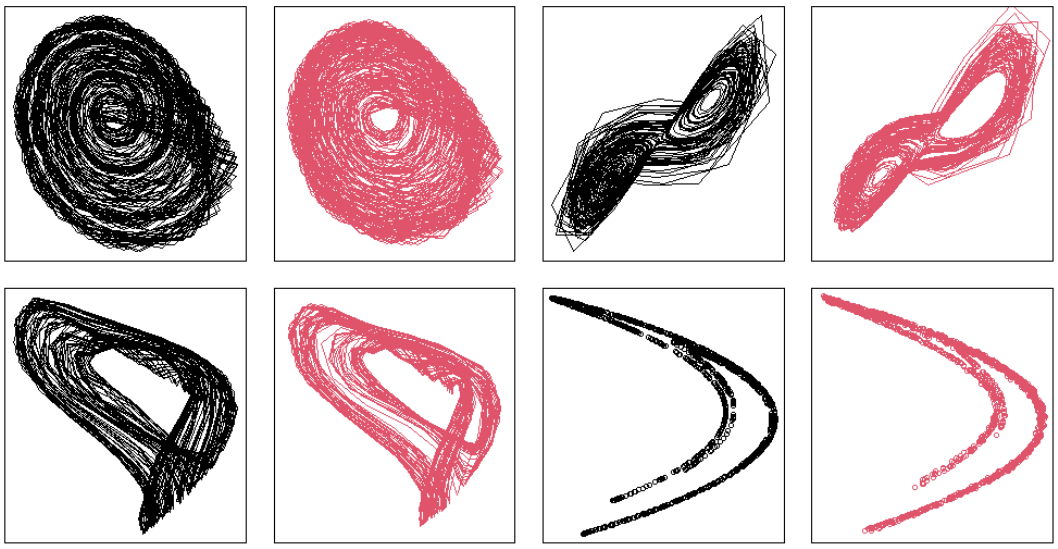

Figure 8: Conceptor-generated patterns (red) plotted next to their target patterns (black) for a period of $L = 1500$ time steps.

yields uncontrolled self-generated patterns, but if the apertures of patterns 3 and 4 are then also adjusted to $\alpha^2 = 35$ and $\alpha^3 = 65$, the self-generated patterns are stable again.

4.2.2 Diagonal Conceptors

The diagonal conceptors were trained according to the scheme described in Section 3.3. The resulting diagonal conceptors were then used to self-generate the patterns. Similar to conceptors, the self-generated patterns were compared to the target patterns by visual inspection, and both the target patterns and the self-generated patterns are shown in Figure 9. The diagonal conceptor-generated patterns resemble the target patterns, just as well as in the conceptors simulation. Furthermore, they are stable, meaning that the self-generated patterns will continue to behave like this for a long period of time and they are robust to random starting reservoir states.

Similar to the parameters in the conceptors simulation, the parameters in the diagonal conceptors simulation could not simply be adjusted individually. Moreover, the aperture adjustment for the chaotic patterns was a bit more difficult than for the periodic patterns, i.e., a slight modification could yield unstable results. The main reason is that adjusting the aperture for j may affect the outcome for pattern $i \neq j$, as mentioned in the introduction of this example. For example, adjusting the aperture of pattern 1 from $\alpha^1 = 10$ to $\alpha^1 = 11$ resulted in instability for pattern 2, whereas pattern 1, 3 and 4 remained stable. For this reason, it was a bit harder to find stable solutions. Once a stable solution has been found for part of the patterns, one cannot simply adjust the aperture of the unstable pattern, because it could disrupt the stability of the other patterns. In contrast to conceptors, one must start by adjusting the apertures to overall acceptable solutions and then find the optimal solution. Unfortunately, in the current training procedure, the diagonal conceptors cannot be adjusted one by one. Furthermore, the stability of the solutions is not only dependent on the aperture, but also on the regularization constants ϱ^W and $\varrho^{W^{out}}$. There is a subtle balance between ϱ^W and the aperture. Intuitively, if the diag-

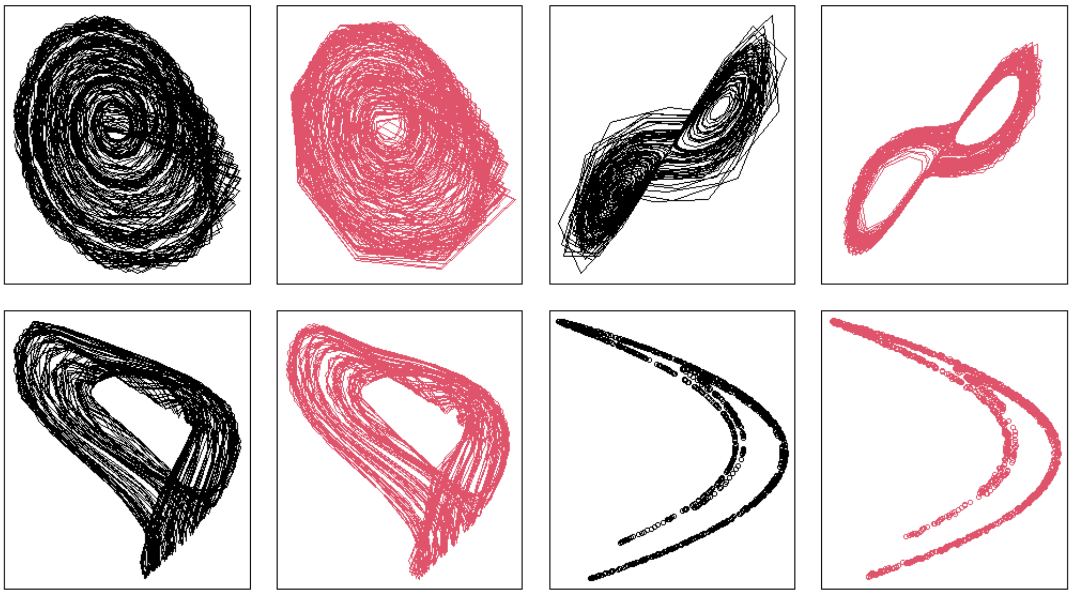

Figure 9: Diagonal conceptor-generated patterns (red) plotted next to their target patterns (black) for a period of $L = 1500$ time steps.

onal conceptor-generated patterns seem to be chaotic, either the aperture must be decreased or the regularization constant must be increased. However, regardless of the increased difficulty of finding the appropriate parameters, there were still many different parameter configurations that yielded stable solutions.

4.3 Human Motion

The patterns used in the simulations in this section are taken from the supplementary code of the paper *Using Conceptors to Manage Long-Term Memory For Temporal Patterns* by H. Jaeger [38]. In this paper, it is shown how conceptors can be used to manage long-term memory for temporal patterns by the use of a few examples. The human motion example is a real-world example and demonstrates the scalability of the conceptors approach to more complex patterns as well as a larger amount of patterns. Therefore, the human motion patterns are ideal for demonstrating the matching scalability of diagonal conceptors. Even though the patterns in this example were taken from the paper, it must be noted that the simulations were also performed with a different set of human motions, which yielded similar results.

The patterns are derived from real human motion, recorded by a motion capture lab consisting of 12 infrared camera capable of recording 120 Hz with images of 4 megapixel resolution. The data is taken from the open source Carnegie Mellon University Motion Capture Database (MoCap) [39]. A test subject wears a black jumpsuit and has 41 markers taped to the suit. The cameras pick up the markers and the different images are then used to triangulate the positions of the markers to output 3D data.

The data can be used in two ways. Either the data is given in the form of a .c3d file, which contains the marker positions, or the data is given by a pair of .asf and .amc files, where the .asf file describes the skeleton and its joints and the .amc file describes the movement data. Since the data is in a specific format, it is not so straightforward to extract the human motion from the data. Fortunately, code has been written in different programming languages to accommodate for this difficulty. The patterns in the supplementary code of [38] are already processed for the most part, but in the case of data taken directly from the MoCap database an R package called *mocap* (`https://github.com/gsimchoni/mocap`) was used for parsing the .asf and .amc files. For a more detailed explanation about the structure of the data, the reader is referred to the MoCap website (`http://mocap.cs.cmu.edu/`). Since the data processing is rather involved, it will not be discussed here, but in the case of the patterns that are used in this section, the reader can read more about it in the appendix of [38].

Let $\mathbf{P} = \{\mathbf{p}^1, \mathbf{p}^2, ..., \mathbf{p}^{15}\}$ be the set of patterns where pattern j at time step n is given by $\mathbf{p}^j(n) \in \mathbb{R}^{61}$, for all j. The patterns are diverse in complexity and their lengths L range between roughly 200 and 900 time steps. Some are transient (sitting down), some are periodic (walking, running), and some are irregular stochastic (boxing). The columns of each pattern are scaled such that they lie in the range $[-1, 1]$. In addition, the patterns were smoothed to remove some of the rough edges of the data. The first dimension of four patterns is shown in Figure 10.

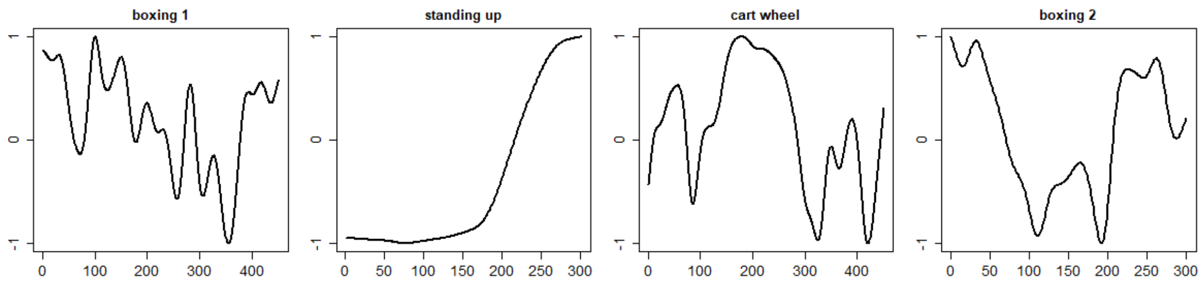

Figure 10: The first of 61 dimensions is plotted for four different human motion patterns: boxing 1, standing up, cart wheel, and boxing 2.

The goal of this example is to demonstrate the ability of diagonal conceptors for a larger, more complex set of patterns. The fact that this is a real-world example makes it a fitting challenge for diagonal conceptors.

4.3.1 Conceptors

The number of neurons in the reservoir was set to $N = 600$. The input weights W^{in} were scaled by 0.2, the bias $\mathbf{b}$ was scaled by 0.8 and the reservoir weights W^* were scaled by 1. The leaking rate was set to $\alpha_l = 0.3$. The washout period was set to $n_{washout} = 50$ time steps, yielding a learning period of $n_{learn} = L - 50$, which was different for each patterns. The apertures were set to $\alpha^1 = 25$, $\alpha^2 = 55$, $\alpha^3 = 17$, $\alpha^4 = 49$, $\alpha^5 = 89$, $\alpha^6 = 49$, $\alpha^7 = 9$, $\alpha^8 = 65$, $\alpha^9 = 49$, $\alpha^{10} = 9$, $\alpha^{11} = 25$, $\alpha^{12} = 25$, $\alpha^{13} = 17$, $\alpha^{14} = 9$, and $\alpha^{15} = 7$, where each α^j corresponds to the aperture for pattern j. The apertures were found with grid search where the grid was set from 1 to 100 with intervals of 9. The regularization constant for computing the output weights was set to $\varrho^{W^{out}} = 0$ and the regularization constant for the re-computing the reservoir weights was set to $\varrho^{W} = 0.001$.

The patterns were stored in the reservoir and the conceptor matrices were computed. With the conceptor matrices, the reservoir was able to self-generate the patterns from a given starting reservoir state. This is different from the periodic patterns and chaotic attractors simulations, where the reservoir could start from any reservoir state. The conceptor-generated patterns would sometimes show chaotic behavior if the reservoir would start from a random initial reservoir state. Therefore, the starting state of the learning period was saved and used as a starting state of the reservoir for the self-generation of the patterns using conceptors. The reservoir was run for a period 4 times longer than the target pattern and the first dimensions of the outputted patterns are shown in Figure 11.

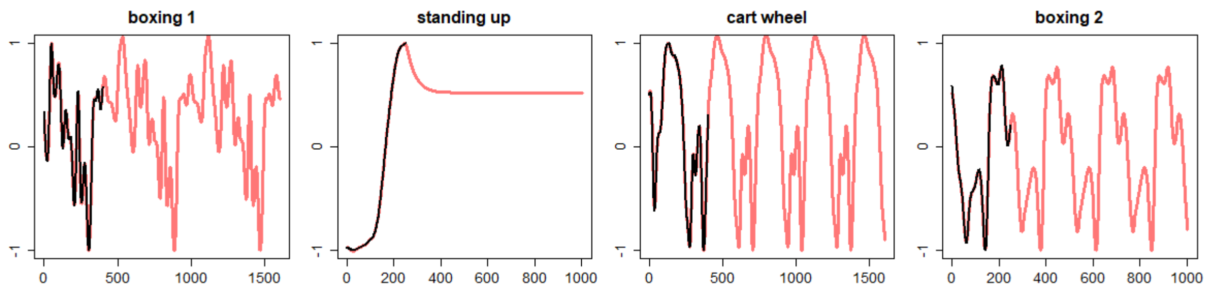

Figure 11: The first of 61 dimensions is plotted for the conceptor-generated output (red) as well as the target pattern (black), where the conceptor-generated patterns are run for a period that is 4 times longer than the target pattern. The human motion associated with the four different patterns are: boxing 1, standing up, cart wheel, and boxing 2.

For each target pattern and conceptor-generated pattern, the NRMSE is computed along each dimension for the overlapping time period. This yields a vector of 61 NRMSEs per pattern, of which the minimum, maximum, mean and standard deviation are shown in Table 1. The table shows that the mean NRMSE of each pattern, except for pattern 15, is below 0.1, which is desired. Furthermore, the standard deviation row shows that the NRMSEs across all 61 dimensions stay roughly around the mean NRMSE, i.e. the mean is representative of the overall NRMSE. The parameters were less stable than the parameters in previous simulations. For example, changing the input weights scaling from 0.2 to 0.3 disrupts the conceptors associated with pattern 2 and 15. However, the conceptors associated with the disrupted patterns can often be recomputed with different apertures, without changing the conceptors of the other patterns. This way, if the balance is disrupted, it can be restored by adjusting the apertures accordingly. Furthermore, the conceptors are moderately robust to unknown reservoir states, which is why it has no trouble repeating the same pattern periodically as is seen Figure 11. However, the conceptors are not robust to any unknown state, which is seen in the second plot from the left in Figure 11. After the target pattern has been generated, the reservoir arrives in ”unknown territory” and it does not know what to do, so it stagnates.

4.3.2 Diagonal Conceptors

The number of neurons in the reservoir was set to $N = 1000$. The input weights W^{in} were scaled by 0.1, the bias $\mathbf{b}$ was scaled by 0.1 and the reservoir weights W^* were scaled by 1. The leaking rate was set to $\alpha_l = 0.15$. The washout period was set to $n_{washout} = 50$ time steps and the adaptation period n_{adapt} was set to the ceiling of one-third of the total length of the patterns, except for pattern 15, where a slightly larger period of $n_{adapt} = 400$ was required. This resulted

Pattern	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Min	0.058	0.061	0.030	0.021	0.041	0.039	0.012	0.009	0.070	0.031	0.041	0.023	0.008	0.058	0.123
Max	0.123	0.183	0.099	0.074	0.141	0.108	0.077	0.096	0.202	0.237	0.107	0.117	0.161	0.289	0.466
Mean	**0.076**	**0.092**	**0.047**	**0.036**	**0.057**	**0.058**	**0.030**	**0.023**	**0.083**	**0.060**	**0.056**	**0.039**	**0.025**	**0.084**	**0.234**
Std	0.017	0.025	0.016	0.012	0.023	0.018	0.017	0.016	0.030	0.039	0.017	0.022	0.023	0.042	0.084

Table 1: The NRMSE of the target patterns and the conceptor-generated patterns. The NRMSE is computed per dimension, which yields a vector of 61 NRMSEs. The Mean row shows the average NRMSE. The Min row shows the smallest NRMSE. The Max row shows the largest NRMSE. The Std row shows the standard deviation of the NRMSEs vector.

in a learning period of $n_{learn} = L - n_{adapt} - n_{washout}$, which was different for every pattern. The apertures were set to $\alpha^1 = 8$, $\alpha^2 = 8$, $\alpha^3 = 20$, $\alpha^4 = 8$, $\alpha^5 = 35$, $\alpha^6 = 35$, $\alpha^7 = 35$, $\alpha^8 = 8$, $\alpha^9 = 35$, $\alpha^{10} = 35$, $\alpha^{11} = 35$, $\alpha^{12} = 35$, $\alpha^{13} = 30$, $\alpha^{14} = 35$, $\alpha^{15} = 8$, where each α^j corresponds to the aperture for pattern j. The regularization constant for computing the output weights was set to $\varrho^{W^{out}} = 0.1$ and the regularization constant for the re-computing the reservoir weights was set to $\varrho^{W} = 0.05$.

After the patterns were stored in the reservoir and the diagonal conceptors were computed and saved, the reservoir was run freely with diagonal conceptors in the loop, where, again, the starting reservoir state was given. The diagonal conceptor-generated patterns are run for a longer period, similar to the conceptors simulation, and the first dimension of a few of the outputted patterns are shown in Figure 12. The black patterns show the training patterns and the red patterns are the diagonal conceptor-generated patterns.

First of all, notice that the adaptation period shortens the training patterns. For the periodic patterns and the chaotic attractors, this was not a problem, because the data was synthetically generated and an adaptation period could be simply be generated to fit the needs of the simulation. However, for these real-world patterns this becomes a bit of an issue, because an adaptation period is required to train the diagonal conceptors. Ideally, the characteristics of a pattern are fully captured in the adaptation period and then repeated in the learning period. Unfortunately, this is sometimes not the case for real-world examples. One could overcome this problem by putting in more work in processing the data to fit the training procedure better. Logically, one would conclude that diagonal conceptors are better suited for periodic patterns. This way, the adaptation period captures the same characteristics of the pattern as the learning period and no information will get lost.

Second of all, from Figure 12 it seems that, after the target pattern has been matched, the diago-

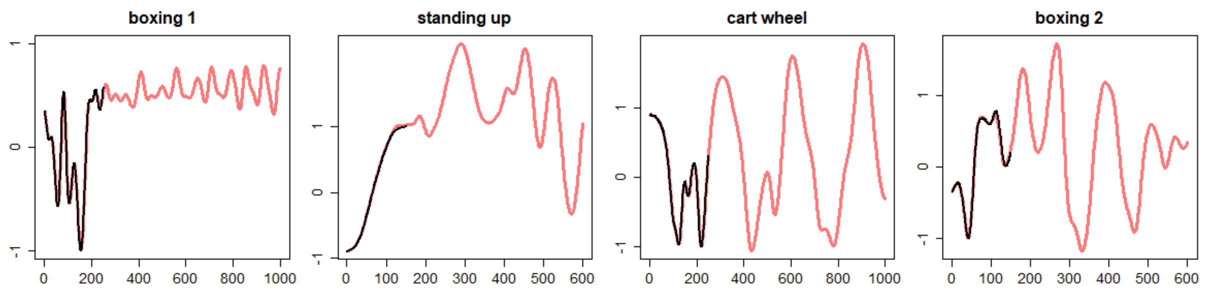

Figure 12: The first of 61 dimensions is plotted for the conceptor-generated output (red) as well as the target pattern (black), where the conceptor-generated patterns are run for a period that is 4 times longer than the target pattern. The human motion associated with the four different patterns are: boxing 1, standing up, cart wheel, and boxing 2.

nal conceptors-generated patterns are not fully behaving like they are expected to. The diagonal conceptors are trained on n_{learn} reservoir states, so starting from a given reservoir state the diagonal conceptors can constrain the dynamics reservoir as they were taught for n_{learn} time steps. However, after those n_{learn} time steps, the reservoir state is unknown to the diagonal conceptors. At that point, the diagonal conceptors should force the reservoir state back to the reservoir state it knows. In the conceptors simulation it was seen that the conceptors would sometimes show the same behavior, but they were more resistant to unknown reservoir states. The diagonal conceptors are more vulnerable to unknown reservoir states than conceptors, which is why the diagonal conceptors-generated patterns start to show unpredictable behavior after the reservoir has been run for n_{learn} time steps. In a sense, diagonal conceptors are less powerful than conceptors to guide the reservoir back from unknown states to states that it knows.

The patterns were compared for the first n_{learn} steps corresponding to the black line in Figure 12, and the NRMSEs are shown in Table 2. The results are similar to the results for conceptors. The mean of the NRMSE of each pattern is smaller than 0.1, which is desired. Compared to Table 1, there are a few more outliers, as can be seen in the *Max* row. Especially pattern 8, which has one dimension in which the NRMSE is incredibly large. However, upon closer inspection, this concerns an individual outlier, which only diverges slightly towards the end. Overall, the diagonal conceptors achieve good accuracy.

The stability of the parameters is similar to, but slightly more fragile than, the stability of the parameters of the conceptors simulation. For example, changing the input weights scaling from 0.1 to 0.2 disrupts the diagonal conceptors associated with patterns 2 and 15, similar to conceptors. Again, this disruption can then be countered by adjusting the apertures α^2 and α^{15}. However, unlike in the case of conceptors, the aperture adjustment of the diagonal conceptors is a bit more delicate, as described in the chaotic attractors simulation. Therefore, adjusting apertures α^2 and α^{15} may lead to disruption in other diagonal conceptors as well. It is for this reason that it can be difficult to find the right parameters for diagonal conceptors when it involves more complex patterns. Regardless, other parameter combinations were found that achieved good accuracy.

Since the starting state of the reservoir was given in the simulation, a simple implementation for a useful implementation of the diagonal conceptors for this particular example is described here.

Pattern	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Min	0.013	0.009	0.020	0.005	0.005	0.023	0.009	0.015	0.004	0.010	0.011	0.008	0.019	0.045	0.017
Max	0.058	0.064	0.254	0.037	0.136	0.473	0.364	0.534	0.051	0.339	0.146	0.342	1.308	0.175	0.107
Mean	**0.025**	**0.020**	**0.060**	**0.011**	**0.013**	**0.090**	**0.026**	**0.061**	**0.010**	**0.036**	**0.043**	**0.012**	**0.091**	**0.086**	**0.054**
Std	0.011	0.009	0.044	0.005	0.022	0.067	0.052	0.127	0.011	0.075	0.032	0.061	0.184	0.033	0.021

Table 2: The NRMSE of the target patterns and the diagonal conceptor-generated patterns. The NRMSE is computed per dimension, which yields a vector of 61 NRMSEs. The Mean row shows the average NRMSE. The Min row shows the smallest NRMSE. The Max row shows the largest NRMSE. The Std row shows the standard deviation of the NRMSEs vector.

The diagonal conceptors require a bit of help finding a known reservoir state after n_{learn} time steps. Therefore, after $n_{learn}+1$ time steps, the reservoir state was set to the starting reservoir state. Consequently, the diagonal conceptors will simply repeat the learnt pattern periodically. Instead of the patterns in Figure 12, which are unpredictable after n_{learn} time steps, the resulting patterns are now periodic as can be seen in Figure 13. Note that this is a rather naive and sturdy workaround, but it already suffices for the purpose of this simulation.

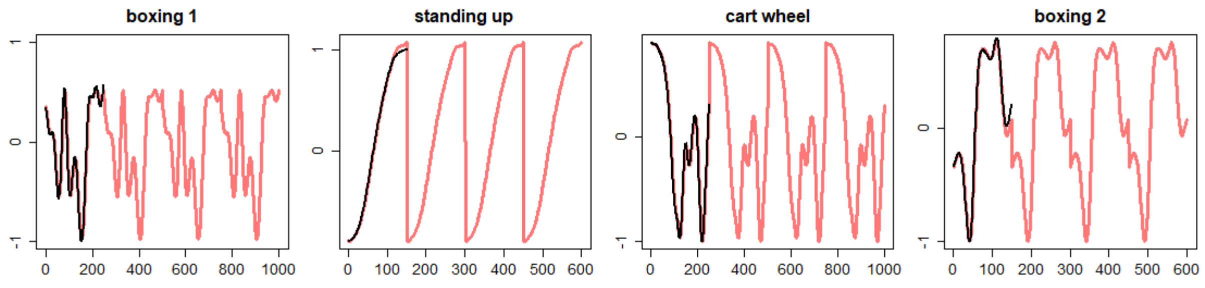

Figure 13: The first of 61 dimensions is plotted for the diagonal conceptor-generated output (red) as well as the target pattern (black), where the conceptor-generated patterns are run for a period that is 4 times longer than the target pattern. The human motion associated with the four different patterns are: boxing 1, standing up, cart wheel, and boxing 2. After each period of n_{learn} time steps, the starting state of the reservoir is inserted in the dynamics to ensure stability.

4.4 Morphing

In this section, the conceptors and diagonal conceptors that were computed in the previous examples were used to morph between different patterns. All examples of morphing in this section are first carried out with conceptors and then with diagonal conceptors to allow for direct comparison.

The first example (periodic patterns) is also demonstrated in the conceptors report and it shows three types of morphing: frequency, shape, and heterogeneous [5]. The first type morphs two sine waves with different periods, the second type morphs two 5-periodic patterns and the third type morphs a 5-periodic pattern into a sine wave. The second example (chaotic attractors) shows a morph between two chaotic patterns, specifically, the Rössler and Mackey-Glass attractors. The third example demonstrates a morph between two human motions.

As discussed in Section 2.3.2, a morph between two or more patterns is governed by the mixture parameter $\mu \in \mathbb{R}$, which is typically between 0 and 1. In this example, a morph will be only between two patterns, however this is necessary in general. Let the patterns be denoted by j_1 and j_2 with their associated conceptors C^{j_1} and C^{j_2}, or diagonal conceptors D^{j_1} and D^{j_2}. Pattern j_1 can be linearly morphed into pattern j_2 by conceptor matrices C^{j_1} and C^{j_2} according to

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n) + \mathbf{b}\right), \\ \mathbf{z}(n+1) &= \left((1-\mu)C^{j_1} + \mu C^{j_2}\right)\mathbf{r}(n+1), \end{aligned} \tag{51}$$

or by diagonal conceptor matrices D^{j_1} and D^{j_2} according to

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W\mathbf{z}(n) + \mathbf{b}\right), \\ \mathbf{z}(n+1) &= \left((1-\mu)D^{j_1} + \mu D^{j_2}\right)\mathbf{r}(n+1), \end{aligned} \tag{52}$$

where the leaking rate is omitted from the equations for notational ease. Note that $0 \leq \mu \leq 1$ indicates interpolation and $\mu < 0$ and $\mu > 1$ indicates extrapolation. The simulations in this section start by driving the reservoir with a constant $\mu = \mu_{min}$ for a period of time. Then the reservoir is continued to be driven while μ is gradually increased for a period of n_{morph} time steps until $\mu = \mu_{max}$. Usually, $\mu_{min} = 0$ and $\mu_{max} = 1$, which means simple interpolation between pattern j_1 and j_2.

4.4.1 Periodic Patterns

In the case of conceptors, the boundaries of the mixture parameter were set to $\mu_{min} = -2$ and $\mu_{max} = 3$ in the case of for the frequency morphing and $\mu_{min} = 0$ and $\mu_{max} = 1$ for the other two types. In the case of diagonal conceptors, the boundaries were set to $\mu_{min} = 0$ and $\mu_{max} = 1$ for all three types. The reservoir was run according to Equation 51 for conceptors and Equation 52 for diagonal conceptors, where μ was initialized as $\mu = \mu_{min}$. The first $n_{washout} = 100$ steps were discarded. Then, the reservoir was continued to be driven for another 50 steps. Afterwards, μ was linearly increased until μ_{max} for n_{morph} time steps. The morphing length was set to $n_{morph} = 200$ for the frequency morphing and $n_{morph} = 50$ for the others. Afterwards, the reservoir was run for another 50 steps with $\mu = \mu_{max}$. The resulting morphed patterns for conceptors and diagonal conceptors are shown in Figure 14 and 15, respectively.

The results for conceptors are similar to the results in the conceptors report, so for more details, the reader is referred to Section 3.7 of the conceptors report [5]. In short, in the case of the two sine waves, the conceptors are able to not only interpolate between $\mu = 0$ and $\mu = 1$, but also extrapolate for values $\mu < 0$ and $\mu > 1$. Remarkably, the information about the patterns that is extrapolated is the period. Therefore, for $\mu = -2$, the period of the sine wave is approximately

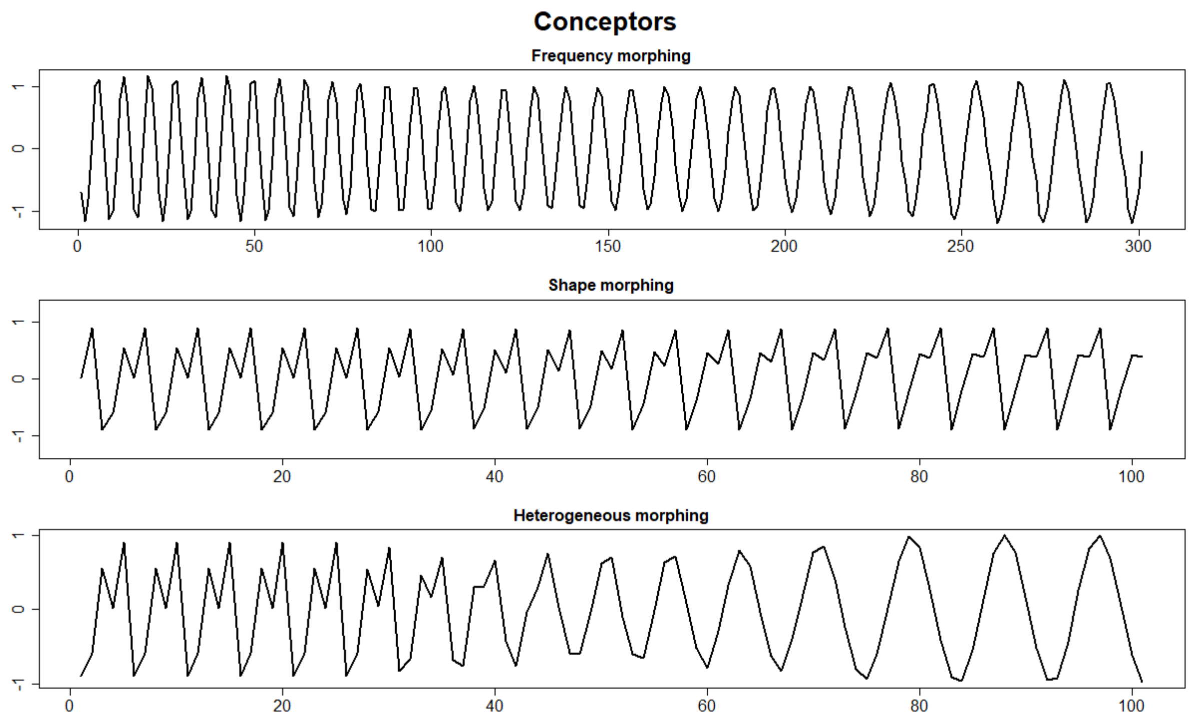

Figure 14: The upper plot shows a morph between two sine waves with periods of approximately 8.83 and 9.93, where $-2 \leq \mu \leq 3$. The middle plot shows a morph between two similar 5-periodic patterns, where $0 \leq \mu \leq 1$. The bottom plot shows a morph between a sine wave and a 5-periodic patterns, where $0 \leq \mu \leq 1$.

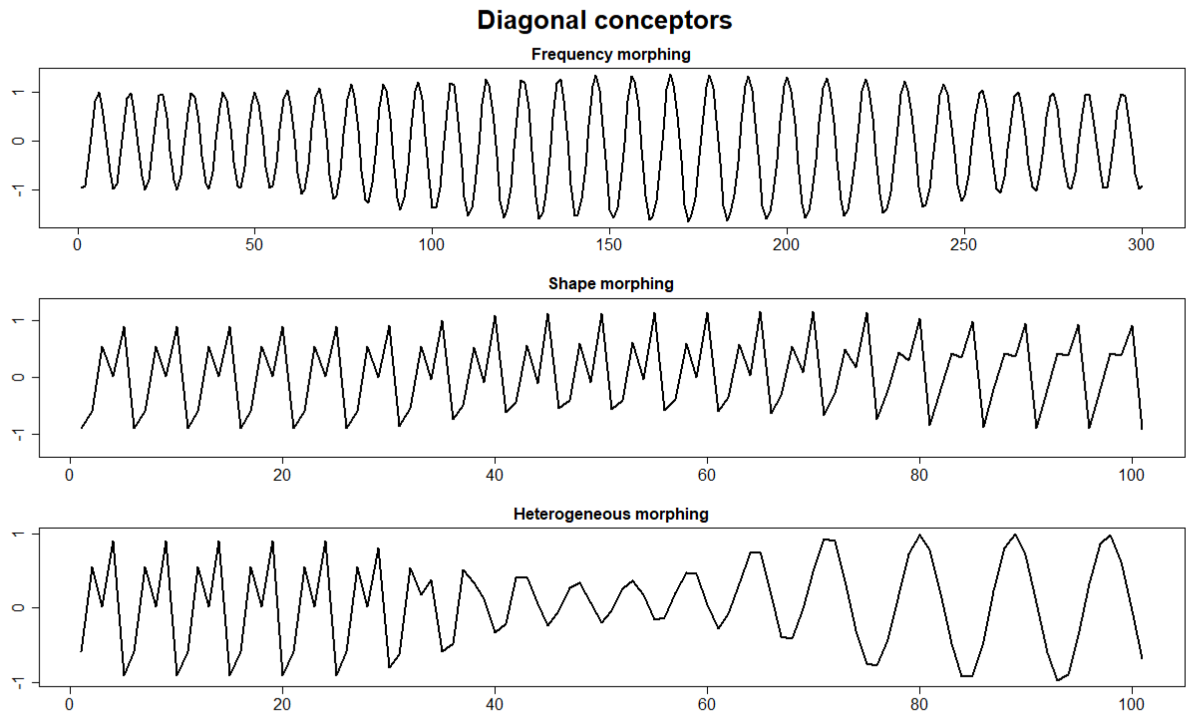

Figure 15: The upper plot shows a morph between two sine waves with periods of approximately 8.83 and 9.93, where $0 \leq \mu \leq 1$. The middle plot shows a morph between two similar 5-periodic patterns, where $0 \leq \mu \leq 1$. The bottom plot shows a morph between a sine wave and a 5-periodic patterns, where $0 \leq \mu \leq 1$.

7.34 and for $\mu = 3$, the period is approximately 12.6, whereas the periods of the original patterns are approximately 8.83 and 9.83. This extrapolation feature does not translate to diagonal conceptors. The morphed pattern shows chaotic behavior for $\mu_{min} < 0$ and $\mu_{max} > 1$. Furthermore, morphing the two 5-periodic patterns, as seen in the middle plot of Figures 14 and 15, shows that diagonal conceptors are just as good as conceptors to morph between similar shapes. Lastly, as seen in the bottom plot of Figures 14 and 14, diagonal conceptors can smoothly transition between dissimilar shapes.

Overall, diagonal conceptors allow for morphing between these periodic patterns almost as well as conceptors, yet slightly less smooth.

4.4.2 Chaotic Attractors

The reservoir was run according to Equation 51 for conceptors and Equation 52 for diagonal conceptors, where μ was initialized as $\mu = 0$. The first $n_{washout} = 100$ steps were discarded. Then, because $\mu = 0$, the reservoir was essentially run with only the (diagonal) conceptor associated with the Rössler pattern, for 500 time steps. Afterwards, μ was linearly increased until $\mu = 1$ over a period of $n_{morph} = 500$ steps. Finally, the reservoir was continued to be run for another 500 time steps with $\mu = 1$. Since the patterns are 2-dimensional, 9 plots were created, where each plot shows a snapshot of 200 data points, taken from the entire period. The resulting plots for conceptors and diagonal conceptors are shown in Figures 16 and 17, respectively.

The figure must be read from the top left to the bottom right plot. The three plots on the top row correspond to $\mu = 0$, so the Rössler pattern. The middle row of three plots corresponds to the morphing period. The three plots on the bottom row correspond to $\mu = 1$, so the Mackey-Glass pattern. The pattern exhibits a much smoother behavior for conceptors than for diagonal conceptors. The diagonal conceptors seem to output chaotic behavior during the morphing period, which stabilizes again when $\mu = 1$.

4.4.3 Human Motion

In the case of conceptors, the reservoir was run according to Equation 51, where C^{j_1} corresponds to a boxing pattern and C^{j_2} corresponds to a cart wheel pattern. Similarly, for diagonal conceptors, the reservoir was run according to Equation 52, where D^{j_1} corresponds to a boxing pattern and D^{j_2} corresponds to a cart wheel pattern. The starting reservoir was given, so no washout period was needed. The boundaries of the mixture parameter were given by $\mu_{min} = 0$ and $\mu_{max} = 1$. The reservoir was run with $\mu = 0$ for the length of the training pattern j_1. Then the pattern was morphed for $n_{morph} = 150$ time steps. In reality, this would amount to a little more than 1 second, which is a fair amount of time needed between boxing jabs and doing a cart wheel. Remember that the human motion diagonal conceptors were unstable to unknown reservoir states. Therefore, in the case of diagonal conceptors, during the morphing period, the reservoir state was slightly nudged to the correct starting state. This was not necessary for conceptors, but there was a washout period needed after the morphing period in the case of conceptors. Afterwards, the reservoir was run for another period with the length equal to that of the training pattern j_2. The first three dimensions of the 61-dimensional output were plotted and are shown in Figures 18 and 19 for conceptors and diagonal conceptors, respectively.

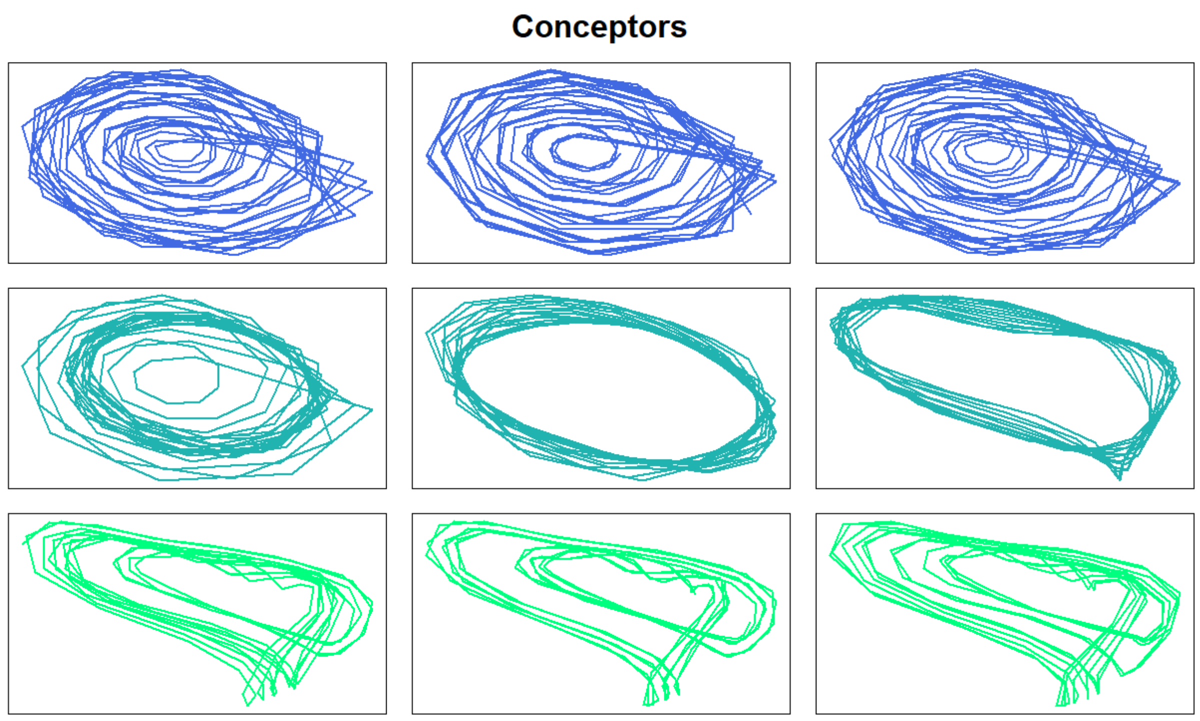

Figure 16: Snapshots of 200 data points of morphing between the Rössler pattern and the Mackey-Glass pattern. The top row shows the Rössler pattern. The middle row shows the morphing period. The bottom row shows the Mackey-Glass pattern.

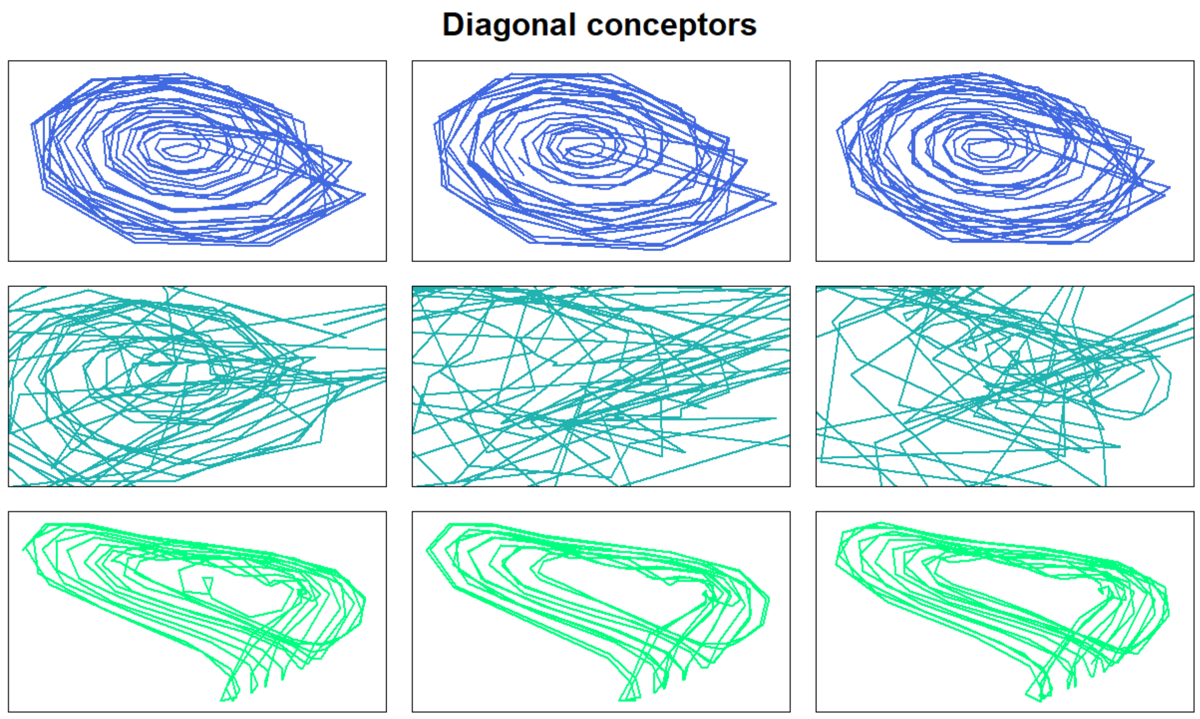

Figure 17: For explanation see Figure 16.

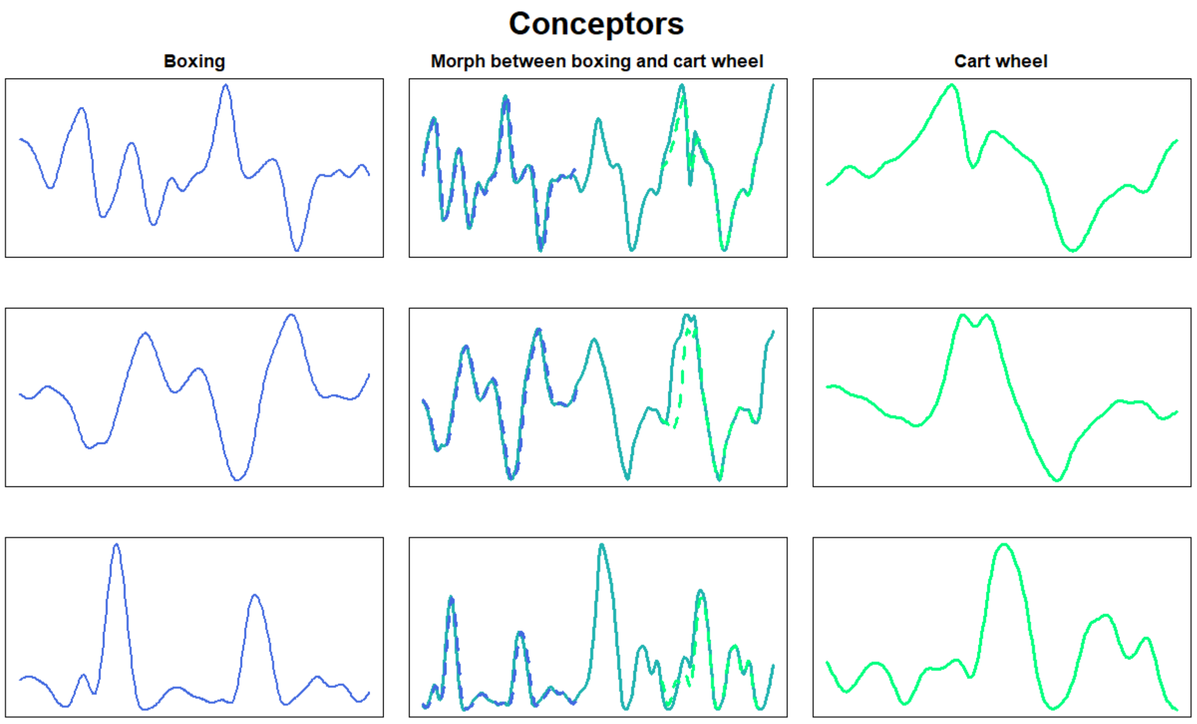

Figure 18: Morphing between two human motions, a boxing motion and a cart wheel motion. Since the patterns are 61-dimensional, only the first 3 dimensions are shown, each corresponding to a row in the figure. The left column shows the boxing patterns and the right column shows the cart wheel pattern. The middle column shows how the boxing patterns is linearly morphed into the cartwheel pattern and the dashed lines show the training patterns and are added to see the accuracy.

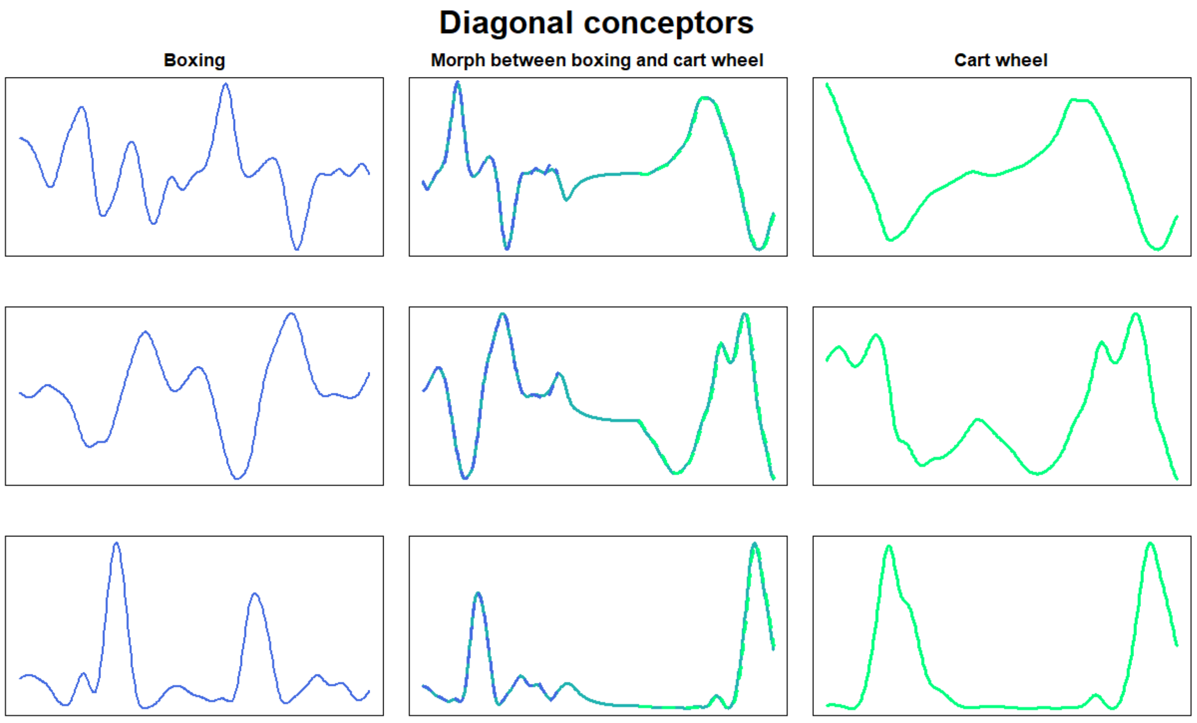

Figure 19: For explanation see Figure 18.

It must be noted that conceptors do not need to be nudged to the starting reservoir state of pattern j_2 in the morphing period, whereas diagonal conceptors do. For this reason, the morphing period is relatively flat for diagonal conceptors compared to conceptors. Nevertheless, the diagonal conceptors are able to morph from one human motion into another.

Overall, diagonal conceptors fall short in morphing patterns compared to conceptors. Intuitively, conceptors take part of the information of one pattern and part of the other and combine it to create a combined new pattern that exhibits information from patterns. Diagonal conceptors do not have that quality, mainly because they carry less information. Consequently, diagonal conceptors have more trouble combining the information they possess, especially in regard to more complex patterns. Diagonal conceptors performed well in the case of the periodic patterns, where the morphing period was smooth. However, where conceptors remained smooth for the chaotic attractors and human motion, diagonal conceptors did not. They either behaved unpredictably in the case of chaotic attractors or they required more help during the morphing period in the case of human motions.

5 Discussion

Lets start this section by stating that diagonal conceptors remain mostly unexplored at the present moment. However, this section dives a bit deeper into some of the features.

From the simulations in the previous section, it was clear that, for the most part, diagonal conceptors work just as well as conceptors. Consequently, diagonal conceptors offer an alternative, computationally efficient architecture for controlling a recurrent neural network. Aside from their advantages, diagonal conceptors also have their disadvantages. Both will be discussed in this section.

This section will attempt to analyse the dynamics of diagonal conceptors and how they relate to, and are different than, conceptors. Some of the steps in the training scheme will be analyzed, specifically, the randomly initialized diagonal conceptors. In addition, an alternative training scheme, similar to autoconceptors, will be discussed. Furthermore, the advantages and disadvantages will be considered as well as some guidelines on how to manually train diagonal conceptors. This is an important section, as it discusses where and how diagonal conceptors thrive.

5.1 Training diagonal conceptors

This section comprises several characteristics of diagonal conceptors, all related to the training algorithm. First, the use of randomly initialized diagonal conceptors is analyzed. Second, the influence of the aperture and the regularization constants of storing the patterns in the reservoir and computing the output weights is explored. Lastly, a few variations of the training algorithm are discussed.

5.1.1 Initial Diagonal Conceptors

Let the reservoir comprise N neurons. For simplicity, it is assumed that the neurons are non-leaky, i.e., the leaking rate is set to $\alpha_l = 1$. The training of diagonal conceptors starts with initializing a number of parameters, specifically, and of interest, the initial diagonal conceptors $D_0^j = \text{diag}(\mathbf{c_0}^j)$, where $\mathbf{c_0}^j = [c_1^j \; c_2^j \; ... \; c_N^j]$ is the conception vector and c_i^j is the conception weight, for all $i = 1, 2, ..., N$. In contrast to the conceptors training scheme, randomly initialized diagonal conceptors are inserted in the update equations from the start, yielding

$$\begin{aligned} \mathbf{r}(n+1) &= \tanh\left(W^*\mathbf{z}(n) + W^{in}\mathbf{p}(n+1) + \mathbf{b}\right), \\ \mathbf{z}(n+1) &= D_0\mathbf{r}(n+1). \end{aligned} \tag{53}$$

The reservoir is run according to Equation 53 for an initial washout period of $n_{washout}$ time steps and then for another n_{adapt} time steps. To understand the necessity of the initial diagonal conceptors in the loop, it is helpful to recap the idea behind conceptors.

If a reservoir is driven by pattern j, it creates a point cloud of reservoir states. The geometry of this point cloud is captured by the conceptor associated with pattern j. The conceptor can then be used to constrain the dynamics of the reservoir by projecting unknown states onto the point cloud. A conceptor can distinguish similar patterns well, because their high degrees of freedom allows for minor adjustments in the projection onto the point cloud. However,

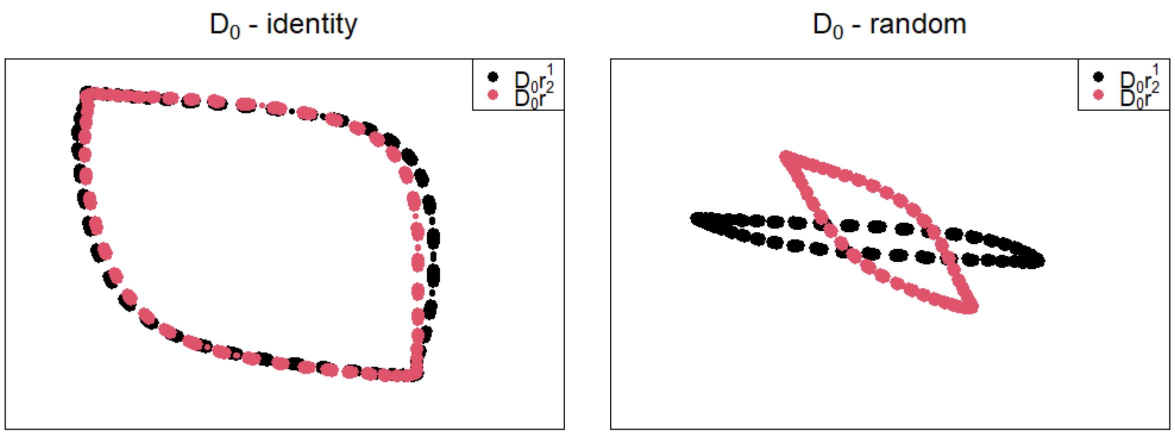

Figure 20: The neuron states during the adaptation period of two neurons is depicted. In the left plot, the initial diagonal conceptors were initialized as identity. In the right plot, the initial diagonal conceptors were initialized randomly. The black dots show states representing pattern 1. The red dots show states associated with pattern 2.

diagonal conceptors have significantly fewer degrees of freedom and will have more trouble distinguishing similar patterns. This is where the initial diagonal conceptors come in. The initial diagonal conceptors create a random scaling of the reservoir state **r**. Intuitively, the initial diagonal conceptor shrinks the area in state space associated with pattern j differently for each pattern j. Therefore, the reservoir states, $\mathbf{z} = D_0\mathbf{r}$, that are created are still characteristic for each pattern. Similar patterns, like the two sine waves in Section 4.1, are ”pulled apart”, making it easier for the diagonal conceptors to determine which state to follow.

To make this clearer, the two sine waves from the periodic patterns example from Section 4.1 are taken as an example. The sine wave with period ≈ 8.83 is called pattern 1 and the sine wave with period ≈ 9.83 is called pattern 2. The simulation is conducted twice. The first time with the diagonal conceptors initialized to the identity matrix and the second time with randomly initialized diagonal conceptors. Regarding terminology it should be noted that initializing the diagonal conceptors to the identity matrix or the conception vector to the all-ones vector or initializing no diagonal conceptors all mean the same thing. In the first simulation, $c_i = 1$ and in the second run c_i is drawn from the uniform distribution on $[0,1]$, for all i. In both simulations, the reservoir states $\mathbf{z}^j = D^0\mathbf{r}^j$, for $j = 1,2$, are collected during the adaptation period of length $n_{adapt} = 500$. Remember that the states $\mathbf{z}^j$ during the adaptation period are used to compute the diagonal conceptors. The results show the neuron state of two selected neurons for both pattern 1 and pattern 2 in both cases. Figure 20 depicts the findings.

It is evident from Figure 20 that the neuron states for the randomly initialized diagonal conceptors are more distinguishable than the neuron states for the identity initialized diagonal conceptors. In the left plot it is seen that the neuron states of the sine waves are very similar, hence difficult for the diagonal conceptors to tell apart. In the right plot, the scaled neuron states have been shrunk randomly and are now much easier to tell apart by the diagonal conceptors.

A few remarks can be made here. First, the randomness of the initialized diagonal conceptors is of a big influence on the training scheme. In general it is good practice to set a seed for

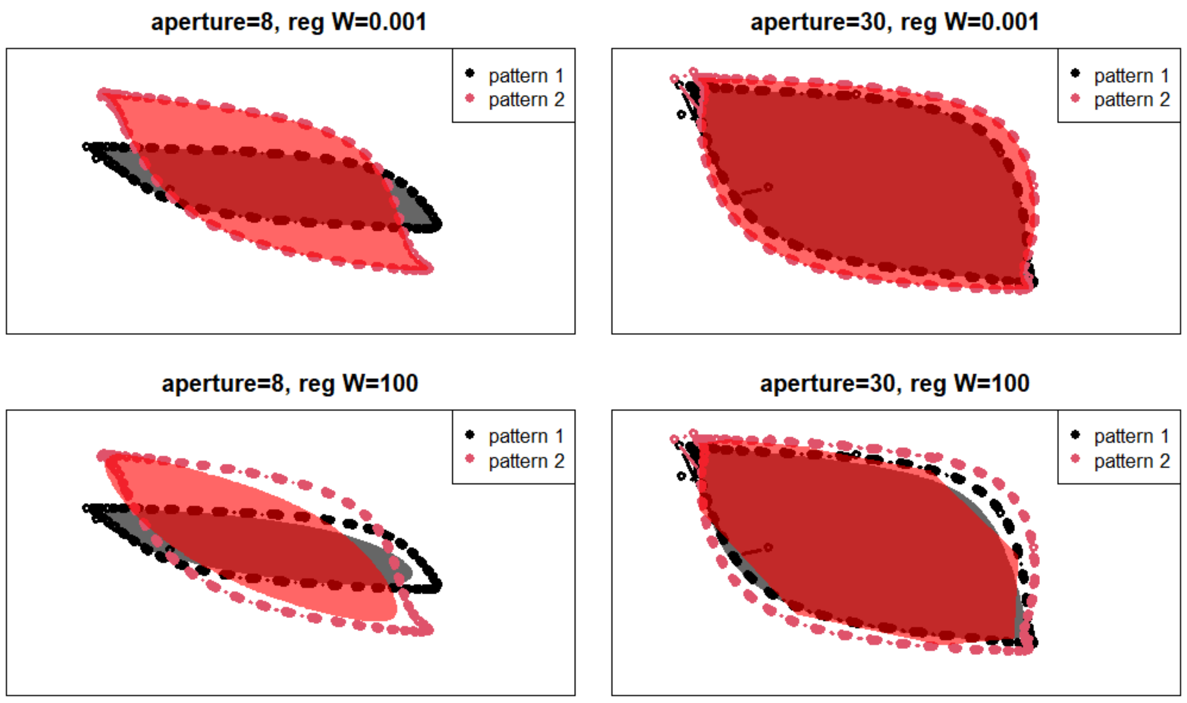

Figure 21

computer simulations for reproducibility. However, in the case of diagonal conceptors this is vital, because underlying the training of diagonal conceptors lies the random initialization of the initial diagonal conceptors. Therefore, changing the seed of the simulation will most likely break the simulation. Second, there probably exists an optimal initialization. In this report, the conception weights were drawn from a uniform distribution on $[0,1]$, but this does not have to be optimal. One could experiment with other distributions and other ranges. A few experiments were conducted where the conception weights were drawn from the uniform distribution on $[-1,1]$ and the standard normal distribution. Both yielded unsatisfactory results. In the case of the former, this could be attributed to the fact that negative conception weights change the sign of negative neuron states, which could disrupt the reservoir dynamics. In the case of the latter, it is not guaranteed that a neuron state stays in the range $[-1,1]$, which can also disrupt the reservoir dynamics. However, neither cases were analyzed thoroughly.

5.1.2 Aperture and Regularization

Besides the randomly initialized diagonal conceptors, the aperture α and regularization constant for recomputing the reservoir weights ϱ^W play a big role in training the diagonal conceptors. Regarding the regularization constant for computing the output weights, $\varrho^{W^{out}}$, it was found that in most cases it was of little influence on the output. Therefore, it will not be discussed here. To demonstrate how α and ϱ^W are of influence on the reservoir dynamics, the example from the previous section is continued. Four simulations were conducted where α and ϱ^W were changed. The results are shown in Figure 21 and require a bit of explaining.

The aperture was either set to $\alpha^j = 8$ or $\alpha^j = 30$ and ϱ^W was either to $\varrho^W = 0.001$ or $\varrho^W = 100$. The reservoir was run with randomly initialized diagonal conceptors for $n_{adapt} = 500$ time steps, after which the diagonal conceptors D^j were computed. This computation depends heavily on α^j. Afterwards, the reservoir was continued to be driven for $n_{learn} = 500$ time steps, during which the states $\mathbf{z}^j = D^j\mathbf{r}^j$ were collected, for $j = 1,2$. Then, the patterns were stored in the reservoir, which is regulated by ϱ^W. Finally, the reservoir was run for $n_{run} = 500$ time steps with the newly computed reservoir weights. During this period, the states $\mathbf{z}^j = D^j\mathbf{r}^j$ were again collected.

Similar to Figure 20, two neuron states where shown, yielding a two-dimensional plot for each simulation. The black color represents the neuron states associated with pattern 1 and the red color represent the neuron states associated with pattern 2. The dotted patterns are the states collected during the learning period and the filled-in areas represent the states collected during the self-generation period, where the states lie on the boundary of the area. The area within this boundary is filled in for aesthetic ease.

From Figure 21 a few observations can be made. First, increasing the aperture decreases the differences between the states. Consequently, for a large aperture, the original point cloud is recovered. The states in the left column of Figure 21 seem to be a combination of the right and left plot of Figure 20. By adjusting the aperture it was found that the shape of the states, as seen in Figure 21, tends towards the left and right plot in Figure 20 for large and small apertures, respectively. Therefore, increasing the aperture appears to remove the advantage of the randomly initialized diagonal conceptors. This was also seen in the chosen apertures for the diagonal conceptors in the simulations in Section 4, which were consistently smaller than the apertures for the conceptors. Second, the regularization constant ϱ^W behaves like a regularization constant is expected to behave. In ridge regression, the regularization constant determines how much the solution is influenced by outliers. Therefore, a large ϱ^W will compress the shape of the states, making the shape smoother. What this means in practice is that if the aperture is increased it will be likely that ϱ^W should be decreased. The increase in aperture will lead to the outliers playing a more significant role, hence they need to be captured by the regularization constant. This phenomenon can be seen most clearly in the right column of Figure 21. As the aperture is increased, the states will become more similar. However, if ϱ^W is small, it will capture the outliers that are distinct for each pattern. If ϱ^W is large, the outliers are suppressed, leading to indistinguishable state patterns. This is seen clearly in the bottom right plot of Figure 21, where the gray and red areas almost completely overlap.

5.1.3 Adaptation Period

Diagonal conceptors require an adaptation period, during which the diagonal conceptors attempt to lock into the dynamics of the randomly scaled dynamics of the reservoir. The states in the adaptation period can not be used for storing the patterns in the reservoir, which means that all information in the adaptation period is not taken into account in the learning period. This information is lost, like in the washout period. Therefore, the learned pattern will be shorter than for conceptors. This was seen in the human motion simulation. The recalled patterns were shorter for diagonal conceptors compared to conceptors. In the case of the periodic patterns or the chaotic attractors, this was not a problem, because the patterns were generated synthetically.

In the case of real-world patterns, the data should be processed such that it takes an adaptation period into account. This is usually done for the washout period already, so taking the adaptation

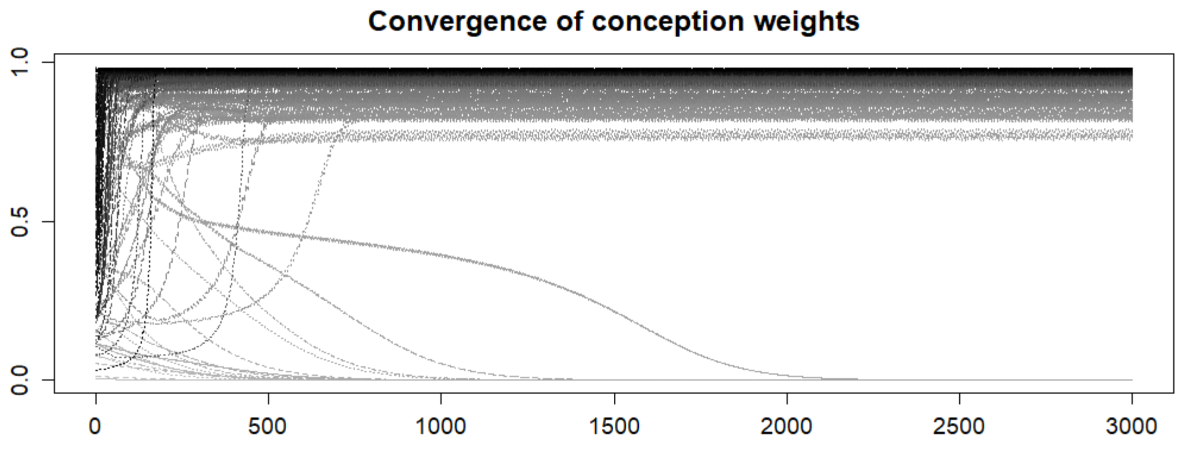

Figure 22: Convergence of the conception weights c_i over the span of $n = 3000$ time steps.

period into account as well should not be difficult. However, unlike the washout period, the pattern during the adaptation period should already be characteristic of the rest of the pattern. Therefore, it is deduced that periodic patterns work best for diagonal conceptors, where as little as a single period could be used for the adaption period.

If the data processing has not taken an adaptation period into account, a solution could be to first run the adaptation period to compute the diagonal conceptors. Then, the pattern from the adaptation period would be reused in the learning period, yielding a much longer learning period. This has been attempted very briefly, but with no success.

5.2 Online adaptation

As mentioned in Section 2.4.3, diagonal conceptors can be trained by online adaptation, similar to autoconceptors. The section dedicated to autoconceptors in the conceptors report comprises various examples and insights into the workings of these autoconceptors [5]. Diagonal autoconceptors were briefly introduced in Section 2.4.3 for the sake of completeness, but they are not analysed to the same extent as autoconceptors in the conceptors report. Nevertheless, this section demonstrates how diagonal conceptors could be learned using the online adaptation rule from Equation 32 with the use of an easy example.

The patterns in this example are, again, the same as in Section 4.1, initialized with all the corresponding parameters. The simulation is conducted almost identically, except that during the adaptation period, the conception weights are adapted every time step according to the adaptation rule

$$c_i = c_i + \lambda_i\big((1-c_i)z_i^2 - \alpha^{-2}c_i\big), \tag{54}$$

where λ_i is the learning rate associated with conception weight i. The learning rate was set to $\lambda = 0.5$ for all i and the adaptation was $n_{adapt} = 3000$ time steps to ensure convergence. Note that true convergence will not be achieved for a learning rate of 0.5, as it is too large. However, for the purpose of this simulation, true convergence is not necessary. The conception weights were collected over the whole adaptation period and plotted as a function of time, which is shown in Figure 22. It can be seen that the conception weights converge to either 0 or a value in the range $[0.5, 1]$, which confirms what was derived in Section 2.4.3.

The simulation was also conducted without the adaptation rule, i.e., the diagonal conceptors were computed at the end of the adaptation period according to Equation 29, to allow for comparison. Computing the diagonal conceptors with an adaptation rule will be referred to as computing the diagonal conceptors *iteratively* and without adaptation rule will be referred to as *explicitly*. At this point it should be noted that both the iterative and explicit method yielded good accuracy for this example. The conception weights after the adaptation period for both the iterative and explicit method simulation are shown in Figure 23.

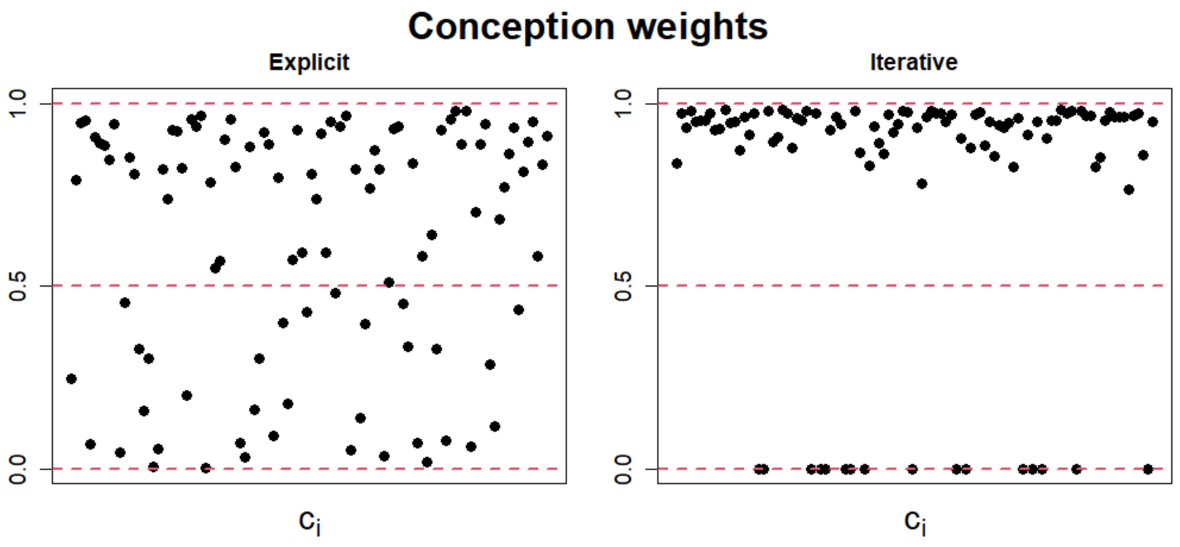

Figure 23: The conception weights after the adaptation period for both the iterative and explicit simulation.

The explicitly computed conception weights are distributed over the range $[0, 1]$, where it appears slightly denser towards 1. Increasing the aperture will shift more conception weights towards 1. The iteratively computed conception weights are either 0 or in the range $[0.5, 1]$, as discussed above. Upon closer inspection of the reservoir states, it turns out that the reservoir states associated with the iteratively computed conception weights during the learning period are similar to the reservoir states depicted in the left plot of Figure 20. Furthermore, some conception weights are 0, which means that some neurons will be completely suppressed. For this reason, it can be hypothesized that the iterative method fails for more complex problems. However, this remains unexplored.

5.3 Stability

Lets start this section with the following observation. For a reservoir comprised on N neurons, a conceptor matrix C has $(N+1)N/2$ degrees of freedom[2]. A diagonal conceptor D has only N degrees of freedom. Therefore, as the number of reservoir neurons increases, one does not expect diagonal conceptors to capture the same amount of characteristics of a pattern as conceptors. Diagonal conceptors simply cannot capture the same amount. However, the simulations in

[2]A conceptor matrix is symmetric, so it has $N + (N-1) + (N-2) + ... + 1 = N(N+1)/2$ degrees of freedom.

Section 4 have shown that diagonal conceptors work much better than one would expect. This is mainly due to their training algorithm and their slightly different dynamics imposed by the randomly initialized diagonal conceptors. However, diagonal conceptors show slightly more unstable behavior than conceptors.

In the periodic patterns and chaotic attractors simulations, the diagonal conceptors were robust unknown reservoir states. Self-generation with diagonal conceptors was usually initiated from the all-zeros vector, but other starting states were tried. It was found that any starting state in $\mathbb{R}^N$ yielded stable patterns, e.g., states drawn from a uniform distribution on $[-10^6, 10^6]^N$. This stable behavior was found for both the periodic patterns and the chaotic attractors. Regardless of the starting state, the correct pattern was always engaged in.

In addition, the parameters of the periodic patterns simulation were quite robust. Large variations in the parameter value still preserved stability. For the chaotic attractors simulation, the parameters were less robust, as a variation in the scaling of the weights could disrupt the stability of the self-generated patterns. Nevertheless, the parameters could still be varied to a certain extend, while stability of the self-generated patterns was retained. In the case of the human motion simulation, setting the parameters required a bit more patience. It was shown that the found parameters could not be varied much, as it would disrupt the stability of the most complex patterns. Overall, the stability of diagonal conceptors depends on the complexity of the task. They are slightly less stable than conceptors, but quite robust nonetheless.

One of the consequences of diagonal conceptors is that the reservoir is less robust to unknown states. As conceptors project the reservoir states to the appropriate area of state space, diagonal conceptors only scale the reservoir state. A projection is less constrained than a scaling, so a projection will require less effort in guiding an unknown state back to a known state compared to a scaling. This was most clearly seen in the human motion example. A slight addition of noise could disrupt the reservoir dynamics of the diagonal conceptors, whereas conceptors were more robust. Making diagonal conceptors more robust to noise is an unexplored area of research at present.

6 Conclusion

Throughout this report, diagonal conceptors have been shown to be a practical alternative to conceptors. Surprisingly, what is gained in computation efficiency is not immediately lost in stability. Nevertheless, diagonal conceptors are more unstable than conceptors for more complex tasks.

In the simulation of the periodic patterns, in Section 4.1, the only comprise of diagonal conceptors was the ability to extrapolate the periods of two sine waves in the morphing. However, besides that, diagonal conceptors perform just as well as conceptors. The fact that diagonal conceptors can distinguish between similar sine wave patterns is mainly due to the randomly initialized diagonal conceptors. They are inserted in the update equations from the beginning to randomly scale the reservoir states. This training step is different from conceptors, but proved to be vital for the performance of diagonal conceptors.

In the simulation of the chaotic attractors, in Section 4.2, diagonal conceptors show that they are also capable of capturing the dynamics of more complicated systems. Furthermore, they proved to be stable and robust to unknown reservoir states. Consequently, the self-generated patterns were stable, regardless of the initial reservoir state. Furthermore, the patterns could be morphed, but the morphed pattern was more chaotic than in the case of conceptors. Nevertheless, after the morphing period, the diagonal conceptors were able to stabilize the self-generated pattern to the desired pattern. Furthermore, in this simulation it was shown that the aperture parameter for diagonal conceptors does not offer the same amount of control as it does for conceptors. An adjustment of the aperture of a single pattern can disrupt the stability of other patterns. This is due to the diagonal conceptors being computed before the learning period, which create a delicate balance between the aperture and the learning period. Specifically, the aperture and regularization constant for recomputing the reservoir weights. Conceptors, on the other hand, are computed after the learning period, hence the aperture offers much more control over the stability of the self-generated patterns.

In the simulation of the human motions, in Section 4.3, diagonal conceptors were able to recall the learned patterns, after which they showed unstable behavior. They required the starting state to be given, because they were not ”strong” enough to guide the unknown state back to a known state. A workaround, where a starting state was inserted in the loop after every self-generation period, was proposed and showed good performance. However, this is not ideal.

6.1 Future Work

The research of diagonal conceptors is still in the early stages. Some features were already discussed in Section 5, but there is much potential for future work, for example:

- Optimizing the randomly initialized diagonal conceptors.
- Researching diagonal autoconceptors, similar to autoconceptors, Section 3.14 of the conceptors report [5].
- Accommodate the adaptation period of the training scheme, such that no information is lost.

- Researching how and if the other features of conceptors translate to diagonal conceptors. They features include, but are not limited to: Boolean operations, neural memory management, recognizing patterns.

The last point is of the most importance, as the comprehensive list of capabilities of conceptors is long and a practical implementation of those features could open up further research in many fields.

Bibliography

[1] M. Tegmark, *Life 3.0: Being Human in the Age of Artificial Intelligence*. Knopf Publishing Group, 2017.

[2] N. Bostrom, *Superintelligence Paths, Dangers, Strategies*. Oxford, UK: Oxford University Press, 2014.

[3] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering Atari, Go, chess and shogi by planning with a learned model," *Nature*, vol. 588, 12 2020.

[4] F. Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65 6, pp. 386–408, 1958.

[5] H. Jaeger, "Controlling Recurrent Neural Networks by Conceptors," *ArXiv*, vol. abs/1403.3369, 2014.

[6] C. Kiefer, "Sample-level sound synthesis with recurrent neural networks and conceptors," *PeerJ Comput. Sci.*, vol. 5, p. e205, 2019.

[7] X. He and H. Jaeger, "Overcoming Catastrophic Interference by Conceptors," *ArXiv*, vol. abs/1707.04853, 2017.

[8] T. Liu, L. Ungar, and J. Sedoc, "Continual Learning for Sentence Representations Using Conceptors," *ArXiv*, vol. abs/1904.09187, 2019.

[9] S. Krause, O. Otto, and F. Stolzenburg, "Fast Classification Learning with Neural Networks and Conceptors for Speech Recognition and Car Driving Maneuvers," *ArXiv*, vol. abs/2102.05588, 2021.

[10] T. Liu, J. Sedoc, and L. H. Ungar, "Correcting the Common Discourse Bias in Linear Representation of Sentences using Conceptors," *ArXiv*, vol. abs/1811.11002, 2018.

[11] A. Strock, N. Rougier, and X. Hinaut, "Transfer between long-term and short-term memory using Conceptors," *ArXiv*, vol. abs/2003.11640, 2020.

[12] R. Gast, P. Faion, K. Standvoss, A. Suckro, B. Lewis, and G. Pipa, "Encoding and Decoding Dynamic Sensory Signals with Recurrent Neural Networks: An Application of Conceptors to Birdsongs," *bioRxiv*, 2017.

[13] F. Meyer zu Driehausen, R. Busche, J. Leugering, and G. Pipa, "Bistable Perception in Conceptor Networks," in *ICANN*, 2019.

[14] P. Hitzler, F. Bianchi, M. Ebrahimi, and M. K. Sarker, "Neural-symbolic integration and the Semantic Web," *Semantic Web*, vol. 11, pp. 3–11, 2020.

[15] E. Tsamoura and L. Michael, "Neural-Symbolic Integration: A Compositional Perspective," in *AAAI*, 2021.

[16] B. Wagner and A. Garcez, "Neural-Symbolic Integration for Fairness in AI," in *AAAI Spring Symposium: Combining Machine Learning with Knowledge Engineering*, 2021.

[17] S. Tran, “Unsupervised Neural-Symbolic Integration,” *ArXiv*, vol. abs/1706.01991, 2017.

[18] B. Hammer and P. Hitzler, “Perspectives of Neural-Symbolic Integration,” in *Studies in Computational Intelligence*, 2007.

[19] S. Bader and P. Hitzler, “Dimensions of Neural-symbolic Integration - A Structured Survey,” *ArXiv*, vol. abs/cs/0511042, 2005.

[20] P. Hitzler and B. Hammer, *Perspectives of Neural-Symbolic Integration*, vol. 77. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007.

[21] D. Willshaw, O. Buneman, and H. Longuet-Higgins, “Non-Holographic Associative Memory,” *Nature*, vol. 222, pp. 960–962, 1969.

[22] L. Cooper, “A Possible Organization of Animal Memory and Learning,” in *Collective Properties of Physical Systems*, Elsevier, 1973.

[23] J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities.,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79 8, pp. 2554–8, 1982.

[24] M. Sundermeyer, R. Schlüter, and H. Ney, “LSTM Neural Networks for Language Modeling,” in *INTERSPEECH*, 2012.

[25] A. Graves, A.-r. Mohamed, and G. E. Hinton, “Speech recognition with deep recurrent neural networks,” *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 6645–6649, 2013.

[26] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, “A Novel Connectionist System for Unconstrained Handwriting Recognition,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, pp. 855–868, 2009.

[27] A. Graves and J. Schmidhuber, “Offline Handwriting Recognition with Multidimensional Recurrent Neural Networks,” in *NIPS*, 2008.

[28] G. Lai, W.-C. Chang, Y. Yang, and H. Liu, “Modeling Long- and Short-Term Temporal Patterns with Deep Neural Networks,” *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, 2018.

[29] X. Hinaut and P. F. Dominey, “A three-layered model of primate prefrontal cortex encodes identity and abstract categorical structure of behavioral sequences,” *Journal of Physiology-Paris*, vol. 105, pp. 16–24, 2011.

[30] J. Kolen and J. Pollack, “Multiassociative Memory,” in *The Proceedings of the Thirteenth Annual Conference of the Cognitive Science Society*, 1991.

[31] A. Krause, V. Dürr, B. Bläsing, and T. Schack, “Evolutionary Optimization of Echo State Networks: Multiple Motor Pattern Learning,” in *ANNIIP*, 2010.

[32] B. K. Iwana, V. Frinken, and S. Uchida, “A Robust Dissimilarity-Based Neural Network for Temporal Pattern Recognition,” *2016 15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 265–270, 2016.

[33] A. Waibel, T. Hanazawa, G. E. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Trans. Acoust. Speech Signal Process.*, vol. 37, pp. 328–339, 1989.

[34] X. Jiang, V. Gripon, C. Berrou, and M. Rabbat, "Storing Sequences in Binary Tournament-Based Neural Networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 27, pp. 913–925, 2016.

[35] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky- integrator neurons," *Neural Networks*, vol. 20, 4 2007.

[36] M. Lukoševičius, "A Practical Guide to Applying Echo State Networks," in *Neural Networks: Tricks of the Trade: Second Edition* (M. Grégoire and G. B. M. K.-R. Orr, eds.), pp. 659–686, Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[37] I. B. Yildiz, H. Jaeger, and S. Kiebel, "Re-visiting the echo state property," *Neural networks : the official journal of the International Neural Network Society*, vol. 35, pp. 1–9, 2012.

[38] H. Jaeger, "Using Conceptors to Manage Neural Long-Term Memories for Temporal Patterns," *Journal of Machine Learning Research*, vol. 18, pp. 1–43, 2017.

[39] Carnegie Mellon Graphics Lab, "CMU Graphics Lab Motion Capture Database."