Reservoir Computing Based Cryptography and Exploration of the Limits of Multifunctionality in NG-RC

Master's Thesis at the Faculty of Physics
of the
Ludwig-Maximilian-University Munich

submitted by
Daniel Köglmayr
born in Starnberg on the 11th February, 1996

Munich, Germany, 1st August, 2022

Supervisor:
PD Dr. Christoph Räth

Reservoir Computing basierte Kryptography und Exploration der Grenzen von Multifunktionalität in NG-RC

Masterarbeit an der Fakultät für Physik
der
Ludwig-Maximilians-Universität München

vorgelegt von
Daniel Köglmayr
geboren in Starnberg am 11. Februar 1996

München, den 1. August 2022

Betreuer:
PD Dr. Christoph Räth

Abstract

Reservoir computing has become the state-of-the-art machine learning algorithm for predicting nonlinear and chaotic dynamics. It features excellent speed and less required training data compared to other deep learning methods. The first part of this thesis makes use of the algorithm's speed aspect. A new encryption algorithm is developed, which outperforms a previous reservoir computing based encryption algorithm by a factor of $\sim 10^3$ in terms of encryption speed. Reservoir computing was also successfully applied to simulate biological neural functions. One of these functions is learning multiple tasks with the identical network structure simultaneously, i.e. the ability to be multifunctional. In reservoir computing, the intrinsic network structure is not changed during multifunctional processing, resembling its biological counterpart. The next generation of reservoir computing (NG-RC) was recently introduced, featuring improved performance. Therefore, the functioning of the reservoir network is replaced by polynomial multiplications of time-shifted input variables. The second part of this thesis explores the limits of multifunctionality in NG-RC. The architecture of the algorithm creates high interpretability of multifunctional behavior. This opens a new perspective on multifunctionality and allows such behavior to be analyzed by learned governing equations.

Contents

Chapter 1

Introduction

The whole is greater than the sum of its parts. This famous phrase introduces a powerful concept of nature, emergence. A system interacting with itself in a non-equilibrium environment may acquire new and unexpected behaviors through self-organization. These systems are called complex systems. One defining feature of complex systems is their ability to encode, store, process, and employ functional information [1], which thrives emergence behaviors and creates the world around us. Famous and poorly understood examples are the emergence of consciousness from the interaction and information processing of neurons in the brain [2], [3] or the emergence of life itself from gene regulatory networks, protein-protein interaction networks, and metabolic interaction networks [4],[5]. Another defining feature of many complex systems is their unpredictability in the long term. However, making the unpredictable predictable for a longer time is of great interest in many scientific fields. Only recently, significant efforts have been made to predict the dynamics of the Covid-19 pandemic [6], [7]. As another example, the prediction of the effects of nonlinear feedback mechanisms of tipping points in the Earth's climate system can be brought up [8]. The problem with predicting these nonlinear or even chaotic dynamics is the inaccessibility of the governing equations of these systems. In 1981 Takens [9] showed that essential properties of nonlinear systems could be reconstructed from pure empirical observations [10]. In recent years, machine learning techniques have substantially leveraged this approach, reaching state-of-the-art performances. In 2016 Brunton et al. [11] introduced an efficient algorithm for reconstructing the governing equations by analyzing data. This essentially reveals algorithmically physical principles from pure observation. The currently benchmarking machine learning algorithm for predicting dynamical systems is reservoir computing [12]. Due to its inherently efficient architecture, the algorithm is faster to train [13] and needs less training data [14] than deep learning methods that are usually hard to train and data hungry. The architecture expands the input through its network

structure and activation function into a high-dimensional nonlinear representation. Only this representation is trained with ridge regression onto the desired output, explaining the remarkable speed of the algorithm. The outstanding prediction performance is based on invertible generalized synchronization [15],[16]. In essence, the reservoir network is a high-dimensional dynamical system driven by the input. The reservoir functionality is given if the reservoir network can generate a high-dimensional embedding of the input through synchronization. However, since not every network structure can generate an embedding of any input, the hyperparameter tuning of reservoir computing is generally comprehensive. If it is successful, it can benchmark even the hardest task, like predicting multiple chaotic attractors [17][18], complex spatio-temporal behavior [19],[20] or controlling nonlinear dynamical systems into arbitrary states [21]. Reservoir computing has also been studied at the intersection of neuroscience, complex systems, and machine learning as it can perform tasks usually associated with biological neural functions, such as multifunctionality with one network structure [16]. Only recently, the next generation reservoir computing (NG-RC) architecture was published, highlighting its lack of randomness, the fewer tuneable parameters, and its performance gain in speed compared to the traditional approach [22]. Essentially, it exchanges the random initialized networks of the traditional approach by creating the high-dimensional representation of the input through unique polynomials of time-shifted input variables. This way, astonishing performance gains are achieved. High-accuracy predictions of chaotic attractors were of a factor $\sim 10^6$ less costly, and spatio-temporal chaos predictions of a factor $10^3 - 10^4$ faster than the traditional approach [23].

Chapter 2 develops separation-based reservoir computing as a high-speed encryption algorithm. Security and performance aspects are discussed, and patches towards real-world applicability are made, which leads to encryption speeds $\sim 10^3$ faster than a previous reservoir computing based encryption algorithm.

Chapter 3 takes up the ideas from [16] and investigates to what extent NG-RC [22] can reproduce multifunctionality and the associated biological neuronal functions without an intrinsic network structure.

Chapter 2

Reservoir Computing Based Cryptography

Due to the rapid development of digital communication and digital information transmission, encryption methods play a fundamental role in ensuring secure and usable digital technologies. Intensive research is being carried out on innovative encryption methods to enable efficient and secure encoding of the increasing amount of information such as audio [24] or images [25],[26],[27],[28]. One extensively investigated field of research is chaos-based cryptography. Due to intrinsic properties like pseudorandomness, complexity, and sensitivity to parameter changes, chaos-based cryptography is a promising candidate for innovative and effective encryption methods and a subject of high research interest. In 2019 alone, approximately 160 papers were published related to chaos-based cryptographic algorithms [29]. In [30], Mohammad et al. compared classic image encryption methods like DES, AES, Vigenère, and RC4 with chaotic encryption techniques and showed that hyper-chaotic maps are the most efficient approach among them. Further research combines the chaotic map approach with machine learning techniques of long short term memory structures (LSTMS), resulting in higher security and efficiency than previous schemes [25]. LSTMS are state-of-the-art recurrent neural network (RNN) architectures. Current research shows that echo state networks (ESN), also called reservoir computers, can level the performance of LSTMS but with far less training time, making reservoir computing an exciting candidate for the intersecting research of machine learning and cryptography [31]. In 2017 Ramamurthy et al. [27] introduced echo state networks for symmetric cryptography. They used the echo state network as a prediction algorithm and reached an encryption and decryption speed of around 3 kilobytes per second on a 2.7 GHz Intel Core i5. Here, the echo state network is introduced as a separation algorithm. This way, the echo state network reached with the performance-enhancing method, an encryption and decryption speed of up to 7.6 megabytes per second

on a 2,9 GHz Dual-Core Intel Core i5. The theory of reservoir computing-based cryptography is explained in section 2.1, followed by a discussion of the security and performance aspects of the proposed algorithm and some patching methods for real-world applicability.

2.1 Reservoir Computing

Reservoir computing is a computationally cheap and efficient recurrent neural network architecture. It reaches state-of-the-art performances in predicting chaotic time series, and it can learn the climate of a chaotic time series even with a small set of training data [15]. The idea behind the architecture is to have a nonlinear dimensionality expansion of the input data and only train the expanded states via ridge regression onto the desired output. Following [32], the dimensionality expansion is reached by applying every data point $\mathbf{u}(t) \in \mathbb{R}^M$ of the input data $\mathbf{X} = [\mathbf{u}(0), \mathbf{u}(1), ..., \mathbf{u}(\tau)]$, where $\mathbf{X} \in \mathbb{R}^{M \times \tau}$, onto an input layer $\mathbf{W}_{in} \in \mathbb{R}^{N \times M}$. The entries of this layer are drawn randomly from a uniform distribution between the range $[-k, k]$, where k acts as a tuneable hyperparameter. Every data point $\mathbf{u}(t)$ is applied via matrix multiplication on the input layer, resulting in a N dimensional vector. This vector is fed at its time t into the reservoir function, defining the reservoir state $\mathbf{r}(t + \Delta t)$ at later time $t + \Delta t$.

$$\mathbf{r}(t + \Delta t) = \tanh(\mathbf{W}_{in}\mathbf{u}(t) + \mathbf{A}\mathbf{r}(t)) \tag{2.1}$$

A primary property of reservoir computers is their intrinsic network, defined by the adjacency matrix A of dimension $\mathbb{R}^{N \times N}$. The current reservoir state vector is iteratively applied onto the adjacency matrix and added with the currently expanded input $\mathbf{W}_{in}\mathbf{u}(t)$. The sum becomes the argument of the activation function $tanh()$, defining the reservoir state at later time $t + \Delta t$. This way the input $\mathbf{X} = [\mathbf{u}(0), \mathbf{u}(1), ..., \mathbf{u}(\tau)]$ is nonlinear expanded to the reservoir state matrix $\mathbf{R} = [\mathbf{r}(1), \mathbf{r}(2), ..., \mathbf{r}(\tau + 1)]$.

2.1.1 Prediction Mode

In prediction mode, the reservoir has to learn to map the current input $\mathbf{u}(t)$ onto the following one $\mathbf{u}(t + \Delta t)$. For this task, the reservoir computer has an output layer $\mathbf{W}_{out}$, which needs to be trained accordingly. The goal is to learn a mapping of the current reservoir state $\mathbf{r}(t + \Delta t)$ of $\mathbf{u}(t)$ onto $\mathbf{u}(t + \Delta t)$ simply by matrix multiplication, so that

$$\mathbf{u}(t + \Delta t) \approx \mathbf{W}_{out}\mathbf{r}(t + \Delta t). \tag{2.2}$$

Therefore, the target matrix of the reservoir state matrix $\mathbf{R} = [\mathbf{r}(1), \mathbf{r}(2), ..., \mathbf{r}(\tau)]$ becomes $\mathbf{Y} = [\mathbf{u}(1), \mathbf{u}(2), ..., \mathbf{u}(\tau)]$. The optimization of $\mathbf{W}_{out}$ is done by ridge regression,

$$\mathbf{W}_{out} = argmin(\sum_{t=1}^{\tau} ||\mathbf{W}_{out}^{T}\mathbf{r}(t) - \mathbf{u}(t)||^2 + \beta||\mathbf{W}_{out}||^2) \tag{2.3}$$

or differently formulated by optimizing,

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{R}^{T}(\mathbf{R}\mathbf{R}^{T} + \beta\mathbf{I})^{-1}. \tag{2.4}$$

The hyperparameter $\beta > 0$ is the regression parameter or Tikhonov regularisation parameter. It can balance the optimization between overfitting and underfitting the data. The $\mathbf{W}_{out}$ is the only matrix that is trained in the reservoir computing regime, and ridge regression is a computational cheap optimization algorithm. Both lead to a significant performance in speed. Applying $\mathbf{r}(\tau + 1)$ onto the trained $\mathbf{W}_{out}$ in Eq. 2.2 leads to the prediction $\mathbf{u}(t + 1)$, which can be fed into the reservoir to receive $\mathbf{r}(\tau + 2)$ and so on.

2.1.2 Prediction Mode for Cryptography

In 2017 Ramamurthy et al. [27] introduced echo state networks for symmetric cryptography. Both Alice, the sender of an image, and Bob, the receiver of the image, share the identical echo state network and the identical secret key. The secret key $\mathbf{k}$ is a random vector with the same dimension as a column vector $\mathbf{i}(t) \in \mathbb{R}^M$ of the image to be encrypted. If Alice wants to encrypt the image $\mathbf{I} = [\mathbf{i}(0), \ldots, \mathbf{i}(t)]$, she appends it with the secret key such that $\mathbf{I}' = [\mathbf{k}, \mathbf{i}(0), \ldots, \mathbf{i}(t-1)]$ and transforms it with Eq. 2.1 to the reservoir state matrix $\mathbf{R}$. She then trains the reservoir states via Eq. 2.5 onto the desired image, where $\mathbf{Y} = \mathbf{I}$. Hence, she can send the trained part of the echo state network, the $\mathbf{W}_{out}$, as the cipher to Bob. With this, Bob can plug the received part into his echo state network and reproduce Alice's message. Therefore, he uses his secret key, and transforms it to the reservoir state with Eq. 2.1, applying the reservoir state onto $\mathbf{W}_{out}$ to get $\mathbf{i}(0)$. Applying this procedure for $\mathbf{i}(0)$ instead of the secret key gives $\mathbf{i}(1)$. And iteratively applying it, decrypts the cipher such that the image $\mathbf{I} = [\mathbf{i}(0), \ldots, \mathbf{i}(t)]$ is reproduced. As the trained part of Alice echo state network does not contain any original pixel of the image, the trained part acts as the cipher image. In contrast, the remaining parts of the echo state network become the key for encrypting the image and decrypting the cipher image. Further, Ramamurthy et al. experimentally showed that the cipher $\mathbf{W}_{out}$ contains the two fundamental cryptography properties of diffusion and confusion.

2.1.3 Separation Mode

Krishnagopal et al. introduced reservoir computing to chaotic signal separation [33]. She showed that separating a chaotic signal from superimposed data works in the reservoir computing regime. Following the structure of section 2.1.1, the superimposed data acts as the input data $\mathbf{X} = [\mathbf{u}(0), \ldots, \mathbf{u}(\tau)]$. And the training target $\mathbf{Y} = [\mathbf{s}(0), \ldots, \mathbf{s}(\tau)]$ must be the chaotic signal, for which the reservoir should learn the mapping to. Note that there is no direct relation between the input data and the training target. After transforming the training data of $\mathbf{X}$ to the reservoir state matrix $\mathbf{R}$, the training is identically completed with the ridge regression in Eq. 2.5.

2.1.4 Separation Mode for Cryptography

In the following, the idea of section 2.1.3 will be applied to cryptography. One major difference to section 2.1.2 is that this time the input data $\mathbf{X} = [\mathbf{u}(0), \ldots, \mathbf{u}(\tau)]$ becomes the secret key of Alice and Bob. Both additionally share the identical reservoir computer and are hence able to reproduce the same reservoir state matrix $\mathbf{R}$. The data to encrypt becomes in this context $\mathbf{Y} = [\mathbf{s}(0), \ldots, \mathbf{s}(\tau)]$. If Alice wants to encrypt a message now, she can train her reservoir state matrix onto $\mathbf{Y} = [\mathbf{s}(0), \ldots, \mathbf{s}(\tau)]$ with Eq. 2.5 and send the trained $\mathbf{W}_{out}$ as a cipher of $\mathbf{Y}$ to Bob. Bob can decrypt the message by matrix multiplication of his reservoir state matrix onto $\mathbf{W}_{out}$. Experiments have shown that ridge regression is extremely powerful in reproducing its training data, which in our case is the massage $\mathbf{Y}$. It works so well that the information about $\mathbf{Y}$ does not need to be in $\mathbf{X}$, since this was the case in the separation mode of section 2.1.3. This allows the free choice of the secret key, which is necessary for any encryption algorithm. In the next section, the encryption algorithm is developed. Then, its performance in terms of security and speed is evaluated and discussed.

2.2 Encryption Algorithm

The algorithm presented here is based on symmetric cryptography. This means that the sender and receiver of an encrypted message are both in possession of the same key, which is used to encode and decode the message. For illustration, the sender and receiver share an identical image as the secret key $\mathbf{X} \in \mathbb{R}^{M\times\tau}$ and the sender wants to securely transmit an image . Each column vector $\mathbf{u}(i) \in \mathbb{R}^M$ of the secret key is transformed to the reservoir state vector $\mathbf{r}(i) \in \mathbb{R}^N$ by the likewise identical reservoir computer, where $N > M$. This vector is trained by means of ridge regression on the corresponding column vector of the image to be encrypted $\mathbf{s}(i) \in \mathbb{R}^J$, where $N > J$ in general. The ridge regression optimizes the $\mathbf{W}_{out}$ for every transformed column vector of the secret key, resulting in $\mathbf{W}_{out} \in \mathbb{R}^{N\times J}$. This matrix acts as the cipher image and can be sent as an encrypted representation of $\mathbf{Y}$. Note that this matrix does not contain any original pixel, nor information about the column dimension τ of the image $\mathbf{Y}$. The encryption algorithm is schematically illustrated in Fig. 2.1.

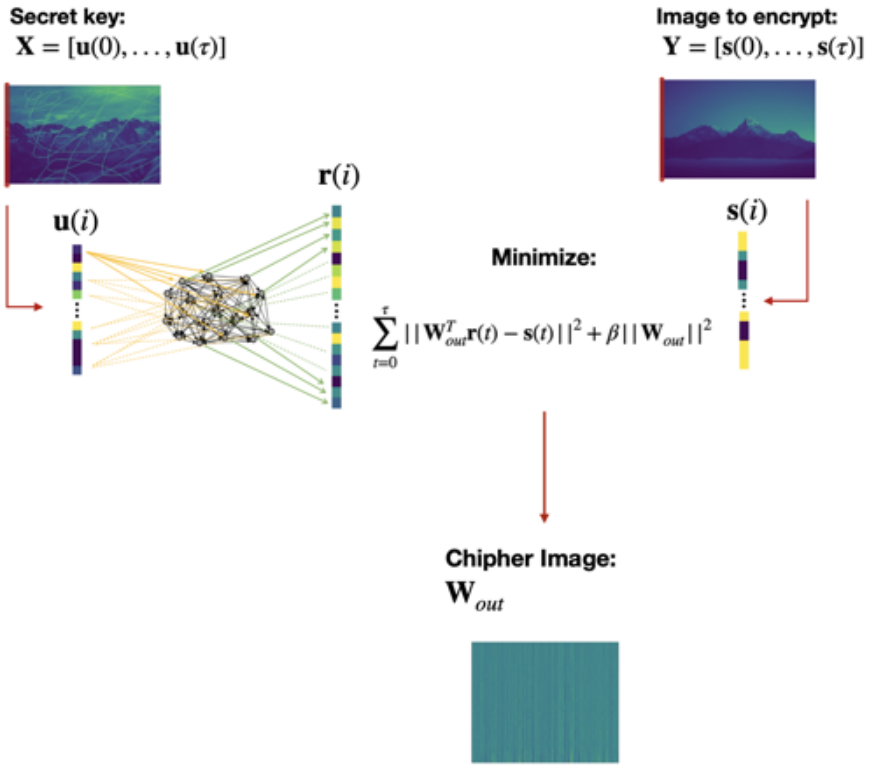


Figure 2.1: Schematic illustration of the encryption algorithm. Every column vector of the secret key $\mathbf{X}$ is transformed to its reservoir state $\mathbf{r}_i$ which is trained via ridge regression onto the corresponding target vector $\mathbf{s}_i$ of the image to be encrypted $\mathbf{Y}$. Applying this procedure to the whole image $\mathbf{Y}$ results in the cipher representation $\mathbf{W}_{out}$.[34],[35].

Decryption

The decryption of the cipher image is straightforward. The receiver creates the reservoir states matrix $\mathbf{R} \in \mathbb{R}^{N \times \tau}$ with his secret key $\mathbf{X}$ and his reservoir computer. The receiver can then reproduce the original image by matrix multiplication.

$$\mathbf{Y} = \mathbf{W}_{out}^{T}\mathbf{R} \tag{2.5}$$

The decryption in section 2.1.2 required iterations to decrypt the image. In this method, the matrix multiplication for decryption leads to a great speedup.

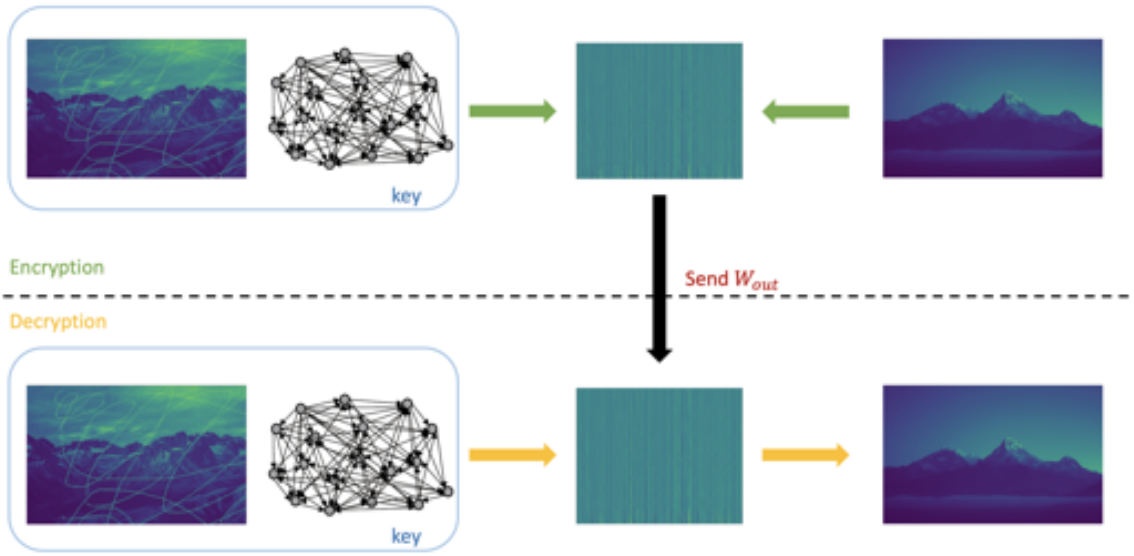


Figure 2.2: Encryption and decryption procedure.

Examples

In Fig. 2.3, three examples are shown. The cipher does not contain any pixel of the original image. As the cipher results from an optimization algorithm, the decrypted image can not match the original image to one hundred percent, but high-resolution decrypted images can be achieved. In the following sections, the security and performance aspects of the proposed algorithm are discussed.

Figure 2.3: Left: Boat image 512×512. Middle: Peper image 128×128. Right: Baboon image 128×128.

2.2.1 Security Aspects

We perform a series of tests on the encryption and decryption results to evaluate the security of the proposed encryption algorithm. Nevertheless, as the proposed algorithm differs in its core architecture from previously published image encryption algorithms, there is no mathematical proof that confirms absolute certainty in security. However, Ramamurthy et al. [27] have shown experimentally that their $\mathbf{W}_{out}$ contains the two fundamental cryptographic properties of diffusion and confusion. The architecture here can be seen as a basic construction kit, where additional security-enhancing methods can easily be integrated, but more to that in this section.

Key Sensitivity

The architecture contains two keys. One is the secret key $\mathbf{X}$ and the other is the reservoir computer. Therefore, the sender and receiver need to share the identical $\mathbf{W}_{in}$ and the identical adjacency matrix $\mathbf{A}$. Further, both need to commit to the

same regression parameter. If those are set, both are able to communicate securely. Note that the large key space makes it impossible to brute force the cipher, as the matrix $\mathbf{R} \in \mathbb{R}^{N \times \tau}$ needs to be guessed, with no information on τ. If an eavesdropper gets his hand on the reservoir computer, he additionally needs to crack the secret key $\mathbf{X}$. The reservoir computer non-linearly transforms every column of the secret key with a dependency on previous transformations into the reservoir state matrix $\mathbf{R}$. This way, high key sensitivity is achieved, which is a necessary property of an encryption algorithm. In Fig. 2.4 this sensitivity is illustrated.

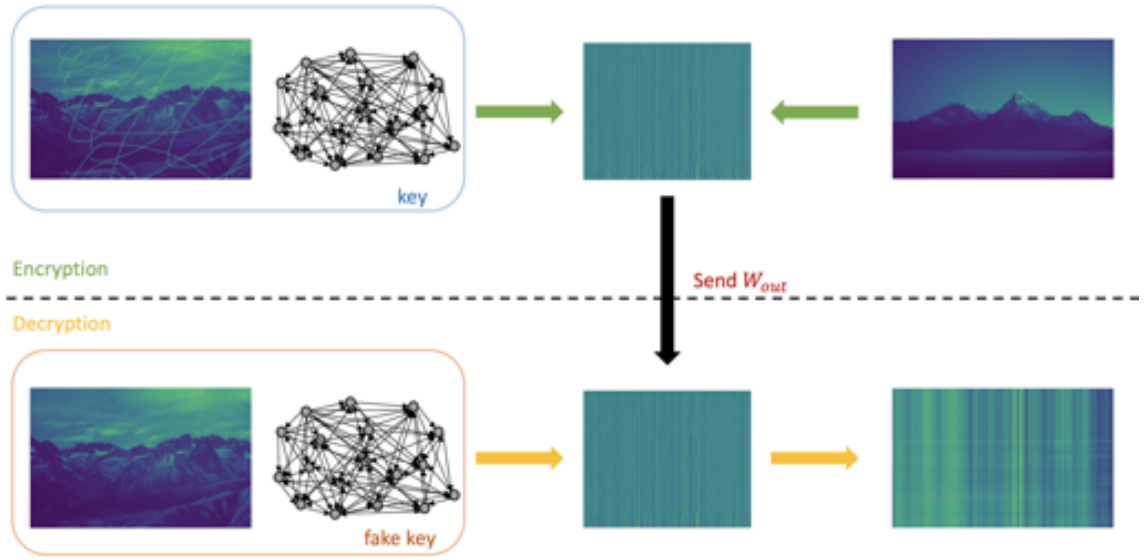


Figure 2.4: Decryption with a slightly different secret key.

Robustness Analysis

A good encryption algorithm needs some range of resilience towards noise and robustness against data loss in the cipher. The decryption results of two examples of data loss are shown in Fig. 2.5. To decrypt $\mathbf{W}_{out}$, it is transposed and then multiplied by a matrix. If some areas in the cipher are missing, matrix multiplication on rows of $\mathbf{W}_{out}$ with missing data will yield no results. However, the rest of the decrypted image is not affected by the missing data, which is the case with chaos-based cryptography, making it vulnerable to data loss attacks. Further, the decryption shows resilience against noise. The pixels of the encrypted image were normalized before training, and gaussian noise $A\mathcal{G}(0, 1)$ with zero mean and a variance of one was multiplied with a percentage factor A and then added on the $\mathbf{W}_{out}$ before decryption. The results for different noise amplitudes can be seen in Fig. 2.6.

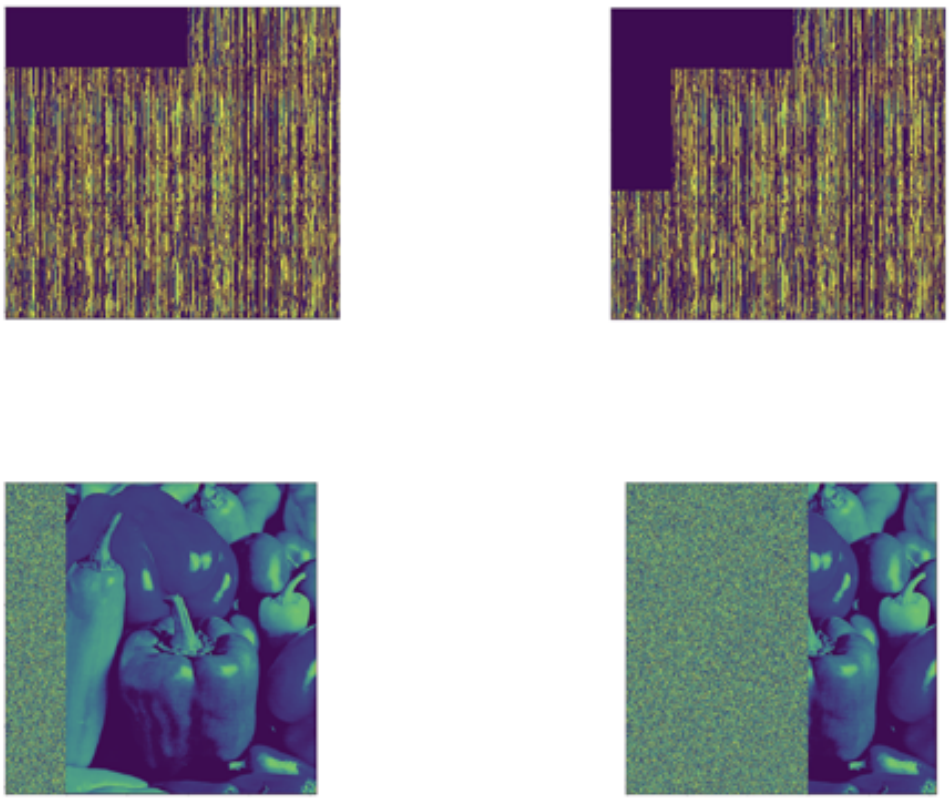


Figure 2.5: Data loss in the cipher and the resulting decrypted images.

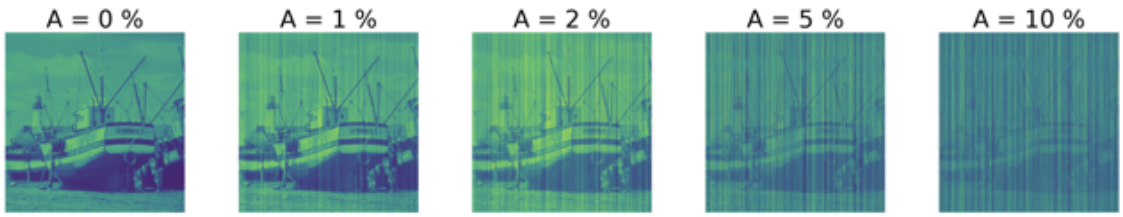


Figure 2.6: Decrypted images after noise attack on the cipher image with gaussian noise. The noise has amplitude A of the maximal value of the image to be encrypted. Even for $A = 5\%$, the images can be recognized in the decrypted image.

Manipulating $\mathbf{W}_{out}$

If the encryption method presented here should have security gaps in its base form, additional methods can improve the security. To be able to extract information from the $\mathbf{W}_{out}$, it must have the correct arrangement. If this is not the case, the arrangement must be restored first, for which there are no logical steps. This allows

the use of a variety of security-enhancing techniques. The cipher $\mathbf{W}_{out}$ can be further encrypted using a different image encryption algorithm. Or the $\mathbf{W}_{out}$ can act as $\mathbf{Y}$ in a second encryption loop. Or one simply multiplies $\mathbf{W}_{out}$ by the already existing adjacency matrix $\mathbf{A}$ of the reservoir computer.

2.2.2 Performance Aspects

The limiting parameter that determines the speed of encryption and the quality of decryption is the network size. In the following, we will determine the minimum network size required to decrypt the Boat image with 128x128, 256x256, and 512x125 pixels with sufficient quality. Therefore, the encryption and decryption are applied to images with different network sizes, and the mean absolute error is taken.

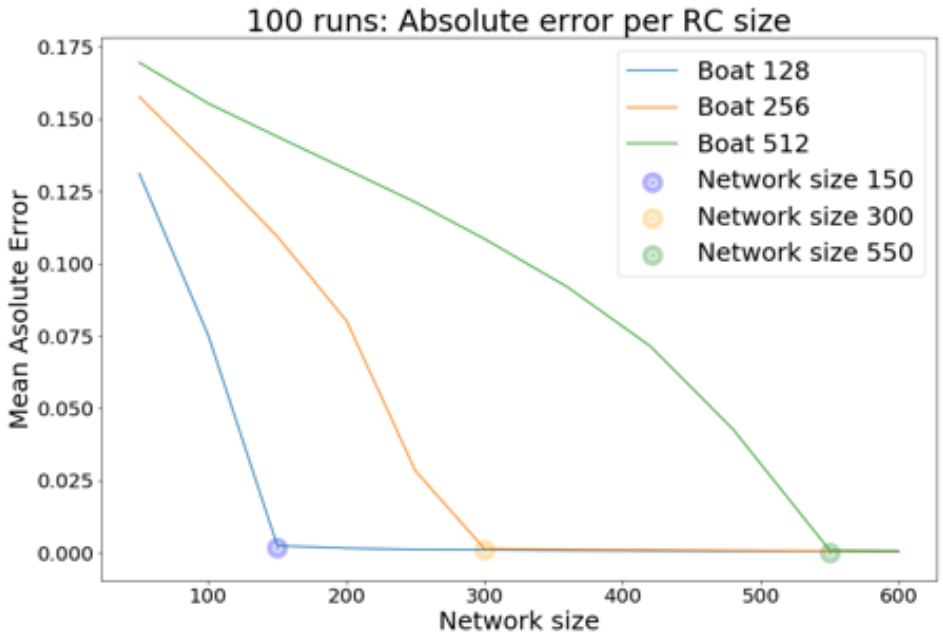


Figure 2.7: Mean absolute error of the decrypted image for different network sizes.

The plot shows that the network size N of the reservoir needs to be larger than the dimension J of the column vectors of the image to be encrypted to achieve sufficient decryption quality. The next question is how long it takes to encrypt and decrypt the Boat images in sufficient quality. Therefore, the upper experiment is repeated 100 times per network size, and the mean encryption and decryption time is taken and plotted.

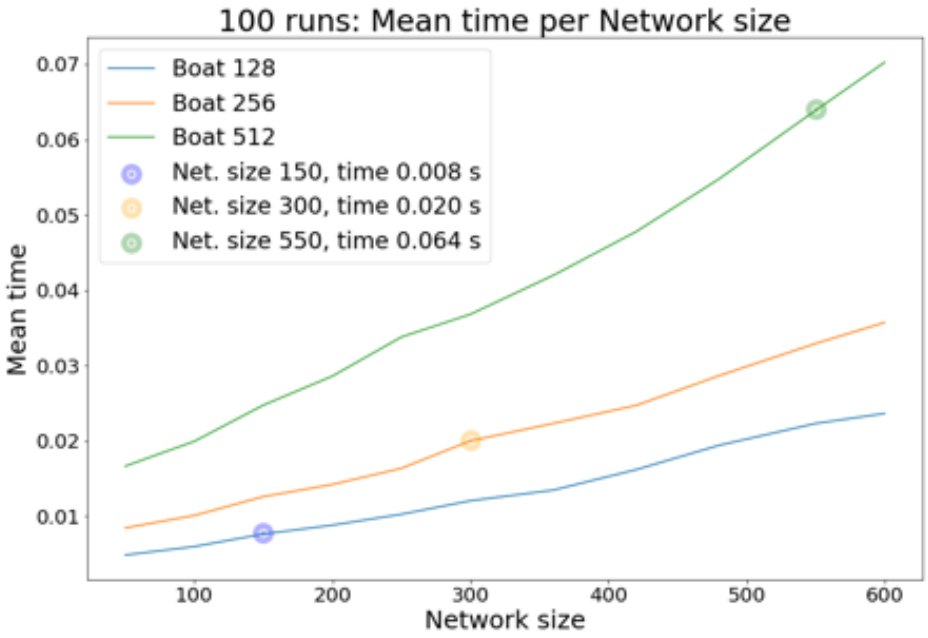


Figure 2.8: Mean encryption and decryption time for different network sizes.

Image size	128×128	256×256	512×512
En- Decryption time in $[s]$	0.008	0.020	0.064

Table 2.1: Minimal mean encryption and decryption time for small error decryption of the Boat image.

In this plot, the highlighted network sizes of Fig 2.7 are likewise marked in Fig. 2.8 and the mean encryption and decryption time is read off and merged in table 2.1. Comparing this performance with those of chaos-based algorithms recently published, the speed of the algorithm becomes apparent, see table 2.2.

To evaluate the algorithm's speed in terms of how many megapixels per second it can process, we apply it with different network sizes onto a set of images ranging from 0.1 up to 24 megapixels, and the corresponding results can be seen in Fig. 2.9. For a fixed network size, the algorithms scale linearly with the megapixel of an image. However, with increasing network sizes, the performance of the algorithm decreases significantly. The result of Fig. 2.7 suggests a larger network size than the dimension of the column vectors of the image to be encrypted to achieve high decryption performance. Hence, for large-scale or real-world images, this algorithm will, in this form, end up in high encryption and decryption times.

Image size	Ref[36]	Ref[37]	Ref[38]	Ref[39]	here proposed
128 ×128	1.93	1.68	1.90	1.28	0.008
256 ×256	7.72	6.72	7.59	5.14	0.020
512 ×512	31.58	26.88	30.35	20.56	0.064

Table 2.2: Comparison of encryption and decryption speed in s with that of chaos-based encryption algorithms [39].

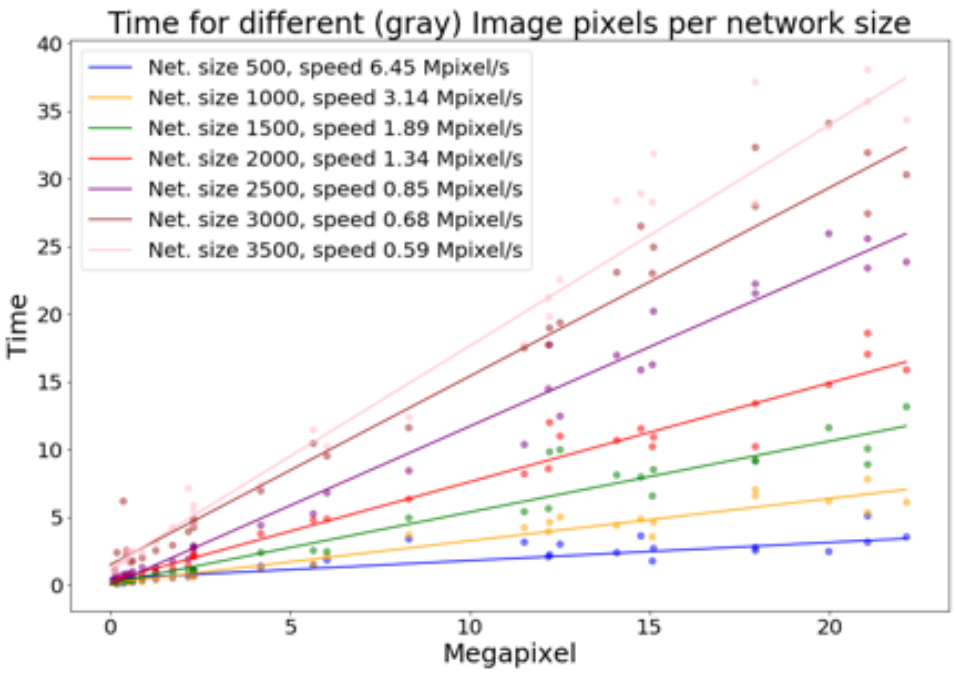


Figure 2.9: Evaluation of encryption and decryption speed in $MPixel/s$ for different network sizes.

In the following subsection, the algorithm will be patched towards a user-friendly encryption algorithm applicable to real-world images.

2.2.3 Patches Towards Real-World Application

RGB Images

So far, the algorithm has been applied to grayscale images. To extend the encryption to RGB images, the algorithm can be readily applied to any color channel of an RGB image, as illustrated in Fig. 2.10. For this purpose, the secret key must also be extended accordingly by the number of channels. The resulting $\mathbf{W}_{out}$ can then be converted into the form of an RGB image and sent to the receiver.

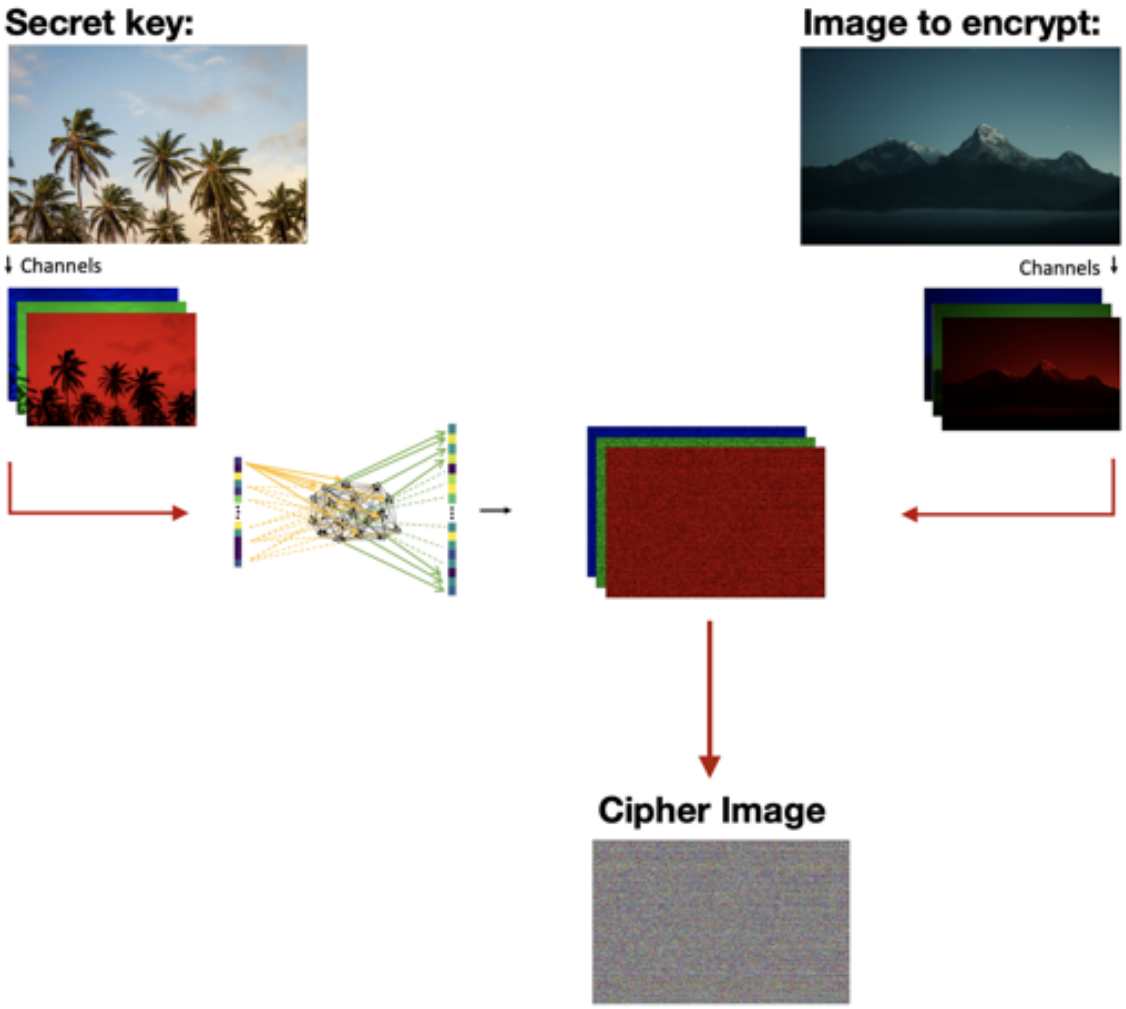


Figure 2.10: Schematic illustration of the encryption algorithm for RGB images [40].

Slicing

However, for RGB images, the number of data points to be processed triples compared to grayscale images using the method described above, further reducing the encryption and decryption speed. This is especially important for large images, which already have a non-user-friendly encoding and decoding time as grayscale images due to the slow processing of the large network sizes required for good decryption performance. The question arises whether one can use the speed advantages of small network sizes to encrypt larger images.

In fact, this is possible if the sender and receiver agree on a specific slicing width for images. For example, a slicing width of $w_s = 100$ could be selected, dividing the image and the secret key into different sections. These can then be concatenated into an image with a column vector dimension of 100 again and a row vector dimension of $L = \frac{M}{w_s}\tau$, illustrated in Fig. 2.11.

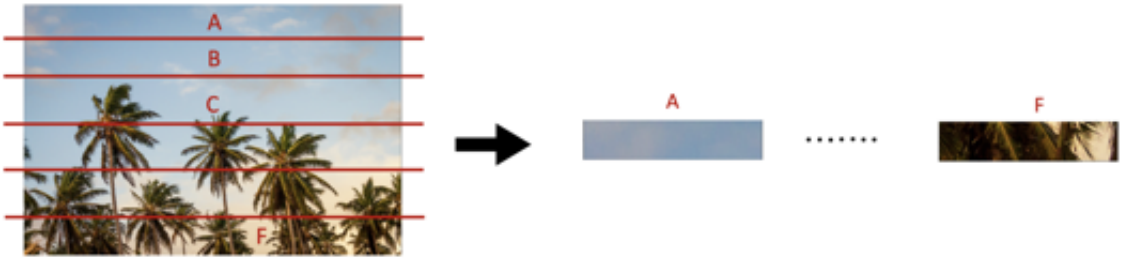


Figure 2.11: Slicing procedure of images for fast encryption by smaller network sizes.

In this way, the speed advantages of small network sizes of, say, $N = 150$ can be exploited, resulting in a $\mathbf{W}_{out}$ of dimension $3 \times L \times N$. The $\mathbf{W}_{out}$ can be reshaped to a format similar to the RGB image and then sent to the receiver. This additional procedure is shown in Fig. 2.12. The receiver can rearrange the $\mathbf{W}_{out}$ with the agreed slicing width and apply his sliced secret key via matrix multiplication to encrypt the cipher. The safety enhancements of the methods described in section 2.2.1 can be further improved by the slicing procedure since it is compatible.

The extra step of slicing increases the performance significantly for large-scale images. Results are shown in table 2.3 for raw image data and the mean encryption and decryption time of 30 runs. This way, encryption and decryption speeds of up to 7 Mbytes were reached on a 2,9 GHz Dual-Core Intel Corei5.

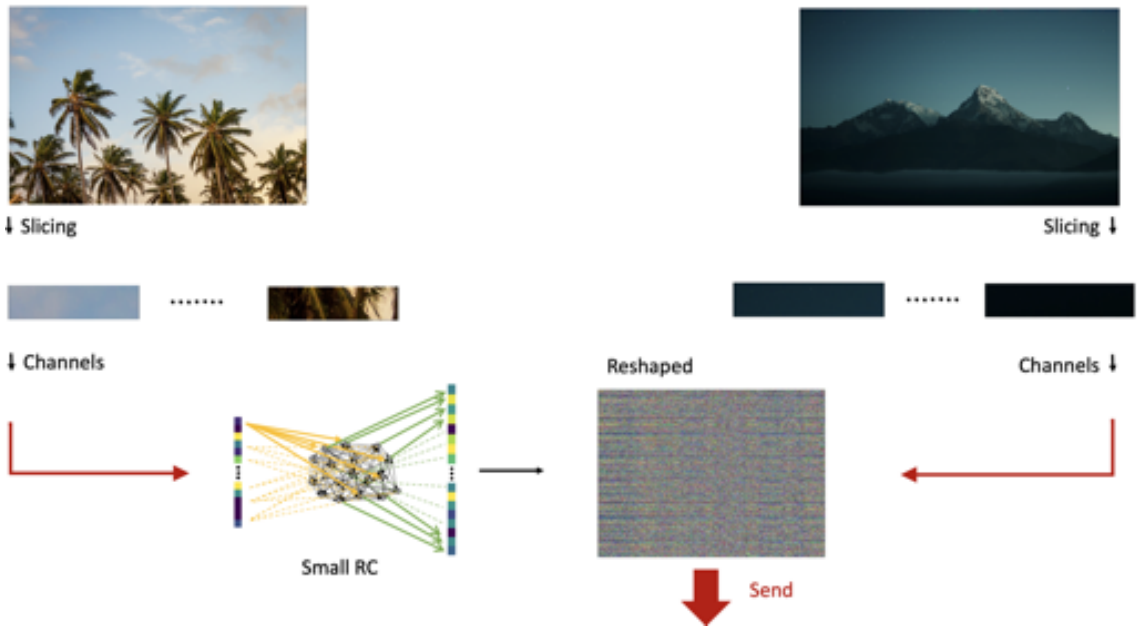


Figure 2.12: Schematic illustration of the encryption algorithm for RGB images with the slicing procedure.

Image size	Mbytes (raw)	mean time in [s] (30 runs)	Mbytes/s
5182×4356 × 3	18.5	4.59	4.03
3456×5183 × 3	33.1	4.34	7.63
5472 ×3468 × 3	25.1	5.6	4.48
5616 ×3744 × 3	19.4	5.6	3.46
5616 ×3744 × 3	33.8	5.6	6.03

Table 2.3: Speed performance of the patched algorithm for large-scale RGB images.

2.3 Conclusion

In this chapter, the theory of reservoir computing for cryptography was introduced and extended by a new method. The realization that the trained weights of the reservoir network are able to reproduce their training data in high accuracy opens the door for cryptographic considerations of this machine learning algorithm. The method presented here is based on a separation approach, where the reservoir computer is trained to learn a mapping between the secret key matrix and a matrix to be encrypted. Since the ridge regression as the core encryption algorithm learns this mapping independently of the relation between the secret key matrix and the matrix to be encrypted, a free choice of the key matrix is possible. This is a fundamental

property of using separation-based reservoir computing as an encryption method. In its basic form, this method has interesting security aspects, such as exceptionally large key length or that the cipher does not contain any information about the matrix's arrangement or row vector dimension to be decrypted. Furthermore, this method can be seen as a construction kit, where further modifications or encryptions of the cipher information can be easily integrated as an additional enhancement of security. Performance-wise, this method has high-speed encryption and decryption speeds for small network sizes. However, due to the quadratic scaling of the network matrix, the speed decreases rapidly for larger networks. Since larger networks are needed to encrypt larger matrices, this effect is counterproductive for real-world applicability. Therefore, a method was introduced to reshape the column vector dimension of the matrix to be encrypted to allow the application of smaller and faster networks. This has enabled encryption and decryption speeds of up to 7 $MBytes/s$, which are over 1000 times faster than the prediction-based reservoir computer encryption algorithm. Moreover, this method was introduced as an image encryption algorithm, but since it works with arrays and matrices, it applies to a wide range of data formats such as audio and video.

Chapter 3

Exploring the Limits of Multifunctionality in NG-RC

Biological neural networks can learn or imitate dynamical systems through pure experience [41], while artificial neural networks do so by learning from training data. Both share a similar approach that attempts to approximate the governing equations of a dynamical system using a high-dimensional representation of the input data. However, how these high-dimensional representations are optimally designed in the artificial environment is far from understood. It is also not known in great detail how learning emerges in biological neural networks [16]. Advances in the field of machine learning often arise from a two-way street between neuroscientific observation and mathematical representation [42]. Current research is concerned with imitating biological neuronal functioning with artificial systems. The goal is to gain a deeper understanding of the involved biological mechanisms, simulate them, and also let artificial intelligence learn from the brain. One characteristic biological function is the ability to learn multiple tasks simultaneously with the same neural network structure [43], [44], [45]. Switching the neural activity pattern based on the processed information to fulfill a specific different task without changing any synapses is a biological phenomenon that is not well understood theoretically. Recent publications are trying to simulate this behavior with artificial neural networks [16], [42], [18], [17], [46]. A successful machine learning algorithm for this task seems to be reservoir computing (RC). The architecture expands the input data through its fixed networks and nonlinear activation function into a high-dimensional nonlinear representation, which is used to learn a particular task. The network structure is not changed during the training or testing process, resembling its biological counterpart. Herteux et al. [17] showed that reservoir computers could simultaneously learn multiple disjointed chaotic attractors with the same readout matrix, introducing artificial neural networks to multifunctionality. Fylnn et al. [18] quantified when multifunctionality occurs

based on its hyperparameters. In [16], Lu et al. draw inferences about how biological neural networks learn, relying on the successful learning of multifunctionality through reservoir computing. They argued that due to the success of reservoir computing, which is based on invertible generalized synchronization (IGS) [15], the functioning of biological neural networks must also be based on IGS.

Lately, the next generation reservoir computer (NG-RC) was introduced [22]. Bollt et al. [47] showed the mathematical equivalence of RC and a nonlinear vector autoregression (NVAR) machine. Gauthier et al. additionally introduced time-shifted coordinates into the process of the NVAR, calling that method NG-RC. A remarkable feature of this algorithm is that it eliminates the need for a network structure. The necessary dimensionality expansion is achieved by concatenating unique polynomials of the input variables. Due to its mathematical equivalence to reservoir computing, the question arises of whether biological neural functions can be imitated without the fundamentally expected network structure.

The following chapter explores the limits of multifunctionality in NG-RC. In the first section, the NG-RC architecture is explained. Then, section 2 applies it to a primary example of learning two oppositely rotating circular trajectories. Section 3 evaluates the multifunctional performance of NG-RC in learning two chaotic attractors. In each of these two sections, there is a conclusion that summarizes the performance of NG-RC from a technical machine learning perspective. Section 4 presents a proof-of-principle of the switching behavior between two learned chaotic attractors. And in Section 5, the results are bundled and discussed from a computational neuroscience perspective.

3.1 NG-RC Architecture

In July 2021, Gauthier et al. published the next generation reservoir computing architecture (NG-RC), highlighting its lack of randomness, the fewer parameters, and its performance gain in speed compared to the traditional approach [22]. Standard reservoir computing uses randomly initialized reservoir matrices as a network structure and a linear readout. The NG-RC uses a set of unique polynomials to achieve a nonlinear dimensionality expansion at its core. The resulting state space of the input data is then consistently trained via ridge regression onto the desired output target.

Loosely following the notation of [22], the d-dimensional data points $\mathbf{x} \in \mathbb{R}^d$ of the input data $\mathbf{X} = [\mathbf{x}_0, ..., \mathbf{x}_n]$ are transformed with a polynomial multiplication dictionary $\mathbf{P}$ into a higher dimensional state space. The unique polynomials of certain orders O, included in $\mathbf{P}^{[O]}$, are labeled as an index.

For illustration purposes, we consider a two-dimensional input data point $\mathbf{x_i} = (x_{i,1}, x_{i,2})^T$ and transform it with the unique polynomials of order 1 and 2,

$$\mathbf{P}^{[\mathbf{1},\mathbf{2}]}(\mathbf{x}_i) = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ x_{i,1}^2 \\ x_{i,2}^2 \\ x_{i,1}x_{i,2} \end{pmatrix} \tag{3.1}$$

Further, Gauthier introduced a time shift expansion $\mathbf{L}_k^s$ of the input data, distinguishing the NG-RC from classic nonlinear vector autoregression (NVAR) algorithms. The k value indicates the number of past data points with which the current data point is concatenated, and the s value indicates how far these points are separated in time. Applying this expansion to the input data defines the linear reservoir layer of the NG-RC. Following the previous example of two-dimensional input data points

$$\mathbf{P}^{[\mathbf{1},\mathbf{2}]}(\mathbf{L}_2^1(\mathbf{x}_i)) = \mathbf{P}^{[\mathbf{1},\mathbf{2}]}(\begin{pmatrix} x_{i,1} \\ x_{i,2} \\ x_{i-1,1} \\ x_{i-1,2} \end{pmatrix}) = \begin{pmatrix} x_{i,1} \\ x_{i,2} \\ x_{i-1,1} \\ \vdots \\ x_{i,1}x_{i-1,2} \\ x_{i,2}x_{i-1,2} \end{pmatrix} = \mathbf{r}_{i+1} \in \mathbb{R}^{14}, \tag{3.2}$$

where $\mathbf{r}_{i+1}$ defines the state vector which is mapped with a readout matrix $\mathbf{W}_{out}$ onto the desired output target $\mathbf{y}_i$. The mapping is learned consistently to the traditional reservoir computing approach via ridge regression. In the training phase of the NG-RC the input training data $\mathbf{X}_T$ of length T is transformed to the state matrix $\mathbf{R} = \mathbf{P}^{[\mathbf{O}]}(\mathbf{L}_k^s(\mathbf{X}_T))$ accordingly. Note that due the k and s value a warm-up time of $\delta t = ks$ is needed, where entries of the state matrix at time $t < \delta$ are not defined. Consequently, the output target matrix $\mathbf{Y}$ needs to be adjusted. The readout matrix $\mathbf{W}_{out}$ is learned via optimizing,

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{R}^T(\mathbf{R}\mathbf{R}^T + \beta\mathbf{I})^{-1}. \tag{3.3}$$

Matrix $\mathbf{I}$ is an identity matrix, and β is the regression parameter.
The output target matrix $\mathbf{Y}$ can be defined in two different ways to learn the prediction of trajectories with ridge regression.

NG-RC as a One-step-ahead Integrator

In this setup, the NG-RC as a one-step-ahead integrator evolves according to,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{W}_{out}\mathbf{r}_{i+1}, \tag{3.4}$$

The reservoir state vector, $\mathbf{r}_{i+1}$, is projected using the readout matrix $\mathbf{W}_{out}$ to resemble the output target $\Delta\mathbf{x}_{i+1} = \mathbf{x}_{i+1} - \mathbf{x}_i$.

Therefore, the output target matrix needs to be defined as follows,

$$\mathbf{Y} = \sum_{i=\delta t}^{T} \mathbf{X}_i - \mathbf{X}_{i-1} = [\Delta\mathbf{x}_{\delta t+1} \ \dots \ \Delta\mathbf{x}_T] \tag{3.5}$$

After training, the trajectory is driven by the NG-RC according to,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{W}_{out}\mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{x}_i)) \tag{3.6}$$

$$\tag{3.7}$$

NG-RC as an Autonomous Dynamical System

In this setup, the NG-RC drives the trajectory directly as,

$$\mathbf{x}_{i+1} = \mathbf{W}_{out}\mathbf{r}_{i+1}, \tag{3.8}$$

As the NG-RC needs to predict the following coordinate, the output target matrix is the input data matrix shifted to one step into the future, such that

$$\mathbf{Y} = [\mathbf{x}_{\delta t+1} \ \dots \ \mathbf{x}_T]. \tag{3.9}$$

This way, the NG-RC drives the trajectory as follows,

$$\mathbf{x}_{i+1} = \mathbf{W}_{out}\mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{x}_i)) \tag{3.10}$$

$$\tag{3.11}$$

Overall, both architectures reduce the tunable hyperparameters to the orders O, the k and s value and the regression parameter β.

3.1.1 Nonlinear Vector Autoregression

In practice, the NG-RC architecture reduces to a nonlinear vector autoregression algorithm in the case where no time shifts are included, so for $\mathbf{P}^{[\mathbf{O}]}(\mathbf{L}_1^1())$ when

$k = 1$ [47] . The ridge regression optimizes the parameters of the $\mathbf{W}_{out}$ such that the polynomial library $\mathbf{P}^{[\mathbf{O}]}$ is fitted to the training data. In combination, $\mathbf{W}_{out}\mathbf{P}^{[\mathbf{O}]}$ provides the fitted governing equation of the training data. The following illustrates this using the Lorenz equations as an example.

Lorenz

To evaluate the performance of reconstructing governing equations simply by processing data, we look at the chaotic Lorenz attractor as an example. From its true equations the training data $\mathbf{X}_T$ of size $T_L = 10.000$ with time step $\Delta t = 0.001$ is generated. The NG-RC is trained as an on-step-ahead integrator in the first application to reconstruct the governing dynamical equations. In the second application, it is used to reconstruct the Lorenz system's integrated equations directly. The true Lorenz equations for this example are defined as follows, and the training data is plotted in Fig. 3.1,

$$\begin{aligned} \dot{x} &= 10(y - x), \\ \dot{y} &= x(28 - z) - y, \\ \dot{z} &= xy - \frac{8}{3}z, \end{aligned} \tag{3.12}$$

Reconstructing the Governing Equations

Following Eq. 3.4 and Eq. 3.5, the training of $\mathbf{R} = \mathbf{P}^{[\mathbf{1,2}]}(\mathbf{L}_1^1(\mathbf{X}_T))$ onto $\mathbf{Y}$, yields to

$$\begin{pmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{pmatrix} = \frac{\mathbf{W_{out}}}{\Delta t}\mathbf{P}^{[\mathbf{1,2}]}$$

$$\approx \begin{pmatrix} -9.96 & 9.99 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 27.9 & -0.94 & 0 & 0 & 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -2.67 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x \\ y \\ z \\ x^2 \\ y^2 \\ z^2 \\ xy \\ xz \\ yz \end{pmatrix}$$

,

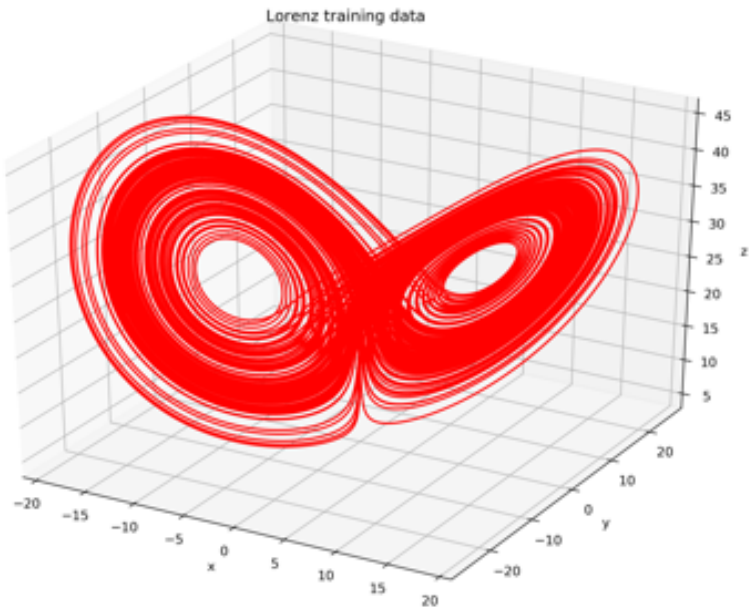


Figure 3.1: Lorenz training data

where $\mathbf{W}_{out}$ values smaller than 0.02 were neglected. This way, the NVAR has learned the following equations,

$$\begin{aligned} \dot{x} &= 9.99y - 9.96x, \\ \dot{y} &= 27.9x - 0.94y - xz, \\ \dot{z} &= -2.67z + xy, \end{aligned} \tag{3.13}$$

which have only a small deviation in their parameterizations compared to those of the true Lorenz equations.

Reconstructing the Integrated Governing Equations

Following Eq.3.8 and Eq. 3.9 , training $\mathbf{R} = \mathbf{P}^{[\mathbf{1},\mathbf{2}]}(\mathbf{L}_1^1(\mathbf{X}_T))$ onto $\mathbf{Y}$, yields

$$\begin{pmatrix} x_{t+1} \\ y_{t+1} \\ z_{t+1} \end{pmatrix} = \mathbf{W_{out}}\mathbf{P}^{[\mathbf{1},\mathbf{2}]}$$

$$\approx \begin{pmatrix} 0.981 & 0.02 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0.055 & 0.999 & 0 & 0 & 0 & -0.002 & 0 & 0 & 0 \\ 0 & 0 & 0.955 & 0 & 0.002 & 0 & 0 & 0 & 0 \end{pmatrix} \begin{pmatrix} x_t \\ y_t \\ z_t \\ x_t^2 \\ y_t^2 \\ z_t^2 \\ x_t y_t \\ x_t z_t \\ y_t z_t \end{pmatrix}$$

,

where $\mathbf{W}_{out}$ values smaller than 0.002 are neglected. The following equations were learned.

$$\begin{aligned} x_{t+1} &= 9.98x_t + 0.02y_t, \\ y_{t+1} &= 0.055x_t + 0.999y_t - 0.002x_t z_t, \\ z_{t+1} &= 0.955z_t + 0.002x_t y_t. \end{aligned} \tag{3.14}$$

In general, both approaches provide an easily accessible solution to the given problem. The basic governing equations can be read from the architecture and the proposed parameterization is good enough to evaluate the characteristics of the Lorenz system, all by pure observation of the training data.

3.2 The 'Seeing Double' Task

The 'Seeing Double' (SD) problem is a paradigmatic example of training a reservoir computer to achieve multifunctionality [18]. In the following, the next generation reservoir computing approach is applied to three different setup-ups of the 'Seeing Double' Task, the non-overlapping case (I), the partly overlapping case (II), and the completely overlapping case (III) of two contrarily rotating circles, see Fig. 3.2-3.4. Due to its experimental simplicity, it allows examining the conditions for multifunctionality on a base level. For the classic RC, this setup showcased the role of the spectral radius in the occurrence of multifunctionality. In next generation reservoir computing, however, there are far fewer parameters to optimize and no randomness involved in the architecture. Therefore an extensive parameter scan can be made to examine the performance of NG-RC on a primary multifunctionality task and evaluate the role of its hyperparameters.

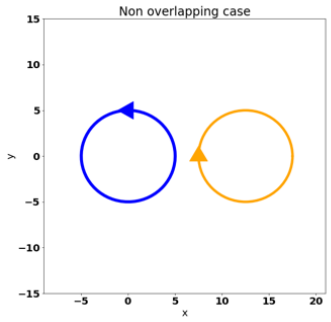


Figure 3.2: Case I

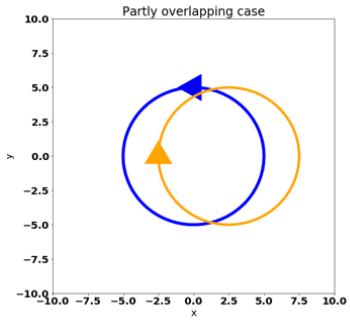


Figure 3.3: Case II

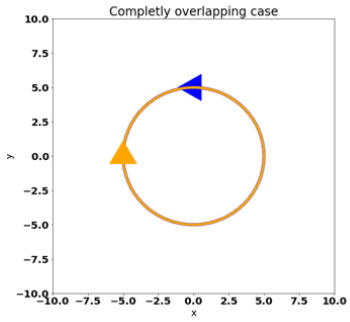


Figure 3.4: Case III

3.2.1 Completely Overlapping Case

Applying the NG-RC as $\mathbf{P}^{[1,2]}(\mathbf{L}_2^1())$ and as a one-step-ahead integrator on the seeing double task showcases some inner workings of the algorithm. After training, the NG-RC could drive the trajectories on the circles in both directions, see Fig. 3.6.

Both predicted trajectories in Fig.3.6, the orange and the blue circle, rotate in opposite directions, indicated by the order of the red and green markers. The direction is defined by the order of two concatenated data points. In this case, the NG-RC starts predicting with the last elements of the corresponding circle training data, capturing its direction of rotation. Noticeable is its behavior when scaling down the starting prediction points, which results in the green and the yellow circle. With the downscaling, the NG-RC seems to scale down the radius of the learned trajectory intrinsically.

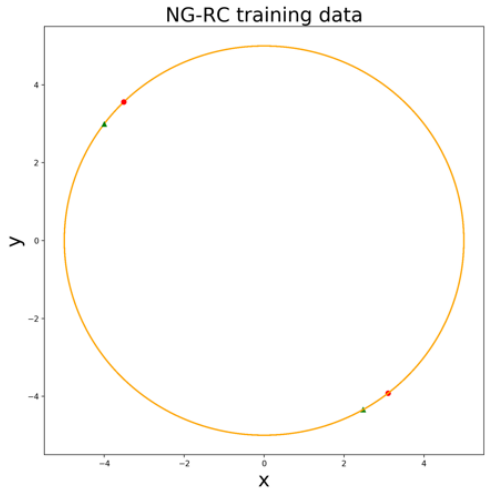


Figure 3.5: Training data of overlapping circles Green/red markers indicate the direction of rotation.

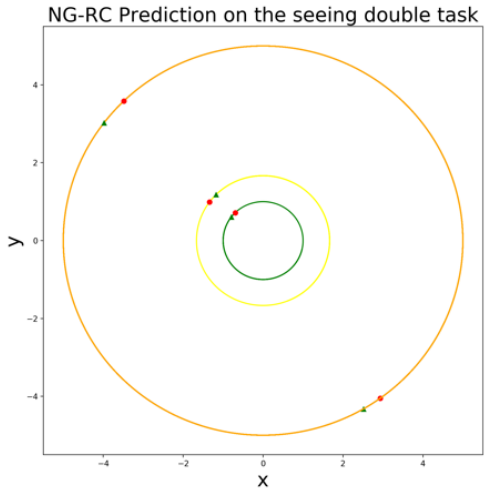


Figure 3.6: Overlapping circles prediction of the $\mathbf{P}^{[1,2]}(\mathbf{L}_2^1())$ NG-RC. Green/red markers indicate the direction of rotation. Green/yellow circles are untrained and predicted trajectories.

Compared to other machine learning algorithms, a significant advantage of this algorithm is its interpretability of what it learned. The architecture of the algorithm allows to read off the learned equations. This time the polynomial multiplication dictionary additionally includes time-shifted coordinates, and the $\mathbf{W}_{out}$ provides the parameters accordingly,

$$\begin{aligned}\begin{pmatrix}\Delta x_t\\ \Delta y_t\end{pmatrix} &= \mathbf{W}_{out}\mathbf{P}^{[1,2]}\\ &\approx \begin{pmatrix}0.99989995 & 0 & -0.99999995 & 0 & 0 & \dots & 0\\ 0 & 0.99990005 & 0 & -1.00000005 & 0 & \dots & 0\end{pmatrix}\mathbf{P}^{[1,2]}\\ &= \begin{pmatrix}0.99989995x_t - 0.99999995x_{t-1}\\ 0.99990005y_t - 1.00000005y_{t-1}\end{pmatrix}\end{aligned}$$

with neglecting entries of $\mathbf{W}_{out}$ with an absolute value smaller than 0.001.

The learned equations are

$$\begin{aligned}x_{t+1} &= x_t + \Delta x_t = 1.99989995x_t - 0.99999995x_{t-1}\\ y_{t+1} &= y_t + \Delta y_t = 1.99990005y_t - 1.00000005y_{t-1}\end{aligned}$$

In this setup, the NG-RC learned an uncoupled linear system. Looking at the learned parameters, the regression seems to approach the parameters to 1 or -1. For x_{t-1} and y_{t-1}, the parameters differs on the eighth digit from -1. However, rounding these values to 1 and -1 results in a $\mathbf{W}_{out}$ that drives the trajectory to infinity. To investigate, how the circular prediction changes towards infinity, n predictions are plotted in Fig. 3.7 with slightly different parameters $c = 0.000005n$ for y_{t-1} respectively, so that

$$x_{t+1} = x_t + \Delta x_t = 1.99989995x_t - 0.99999995x_{t-1}$$
$$y_{t+1} = y_t + \Delta y_t = 1.99990005y_t - (1.00000005 + c)y_{t-1}$$

.

The interesting point here is that the NG-RC learned a linear feedback representation of the Lissajous Curves to describe a circular motion, which is highly sensitive to its parameters. The equations for Lissajous Curves are,

$$x = A\sin(at + \delta)$$
$$y = B\sin(bt)$$

.

where ratios $\frac{a}{b}$ and $\frac{A}{B}$ and phase difference δ defines the shape of the curve.

As the regression parameter β penalizes large $\mathbf{W}_{out}$ values during the optimization, it directly influences the learned parameters of the governing equations. Scanning over β has shown that it determines whether the NG-RC drives the trajectory to infinity, on a circle, or to a fixed point. The evolution of $\mathbf{W}_{out}$ entries with respect to β gives rise to the different behaviors of the NG-RC.

In Fig.3.8, the evolution of the x associated weights in the first-row vector of $\mathbf{W}_{out}$ with respect to β are plotted. Respectively, the y associated weights are plotted in Fig. 3.9. Scanning over β and evaluating the prediction performance showed that for regression parameters smaller than $\beta < 4 \cdot 10^{-8}$ the predicted trajectory diverges. This threshold is highlighted as the yellow dashed line in both plots. For a β slightly smaller than this threshold, the predicted trajectory is plotted in Fig.3.10.

Further, for $\beta > 1 \cdot 10^{-5}$ the predicted trajectory starts circling inwards towards a fixed point, see Fig.3.11. This threshold is indicated by the black dashed line in both plots.

The different behaviors of the NG-RC predictions need to be explainable with the evolution of the $\mathbf{W}_{out}$ weights. The weights associated with the linear states $x_t, y_t, x_{t-1}, y_{t-1}$ are highlighted in Fig. 3.8 and Fig. 3.9. They differ qualitatively

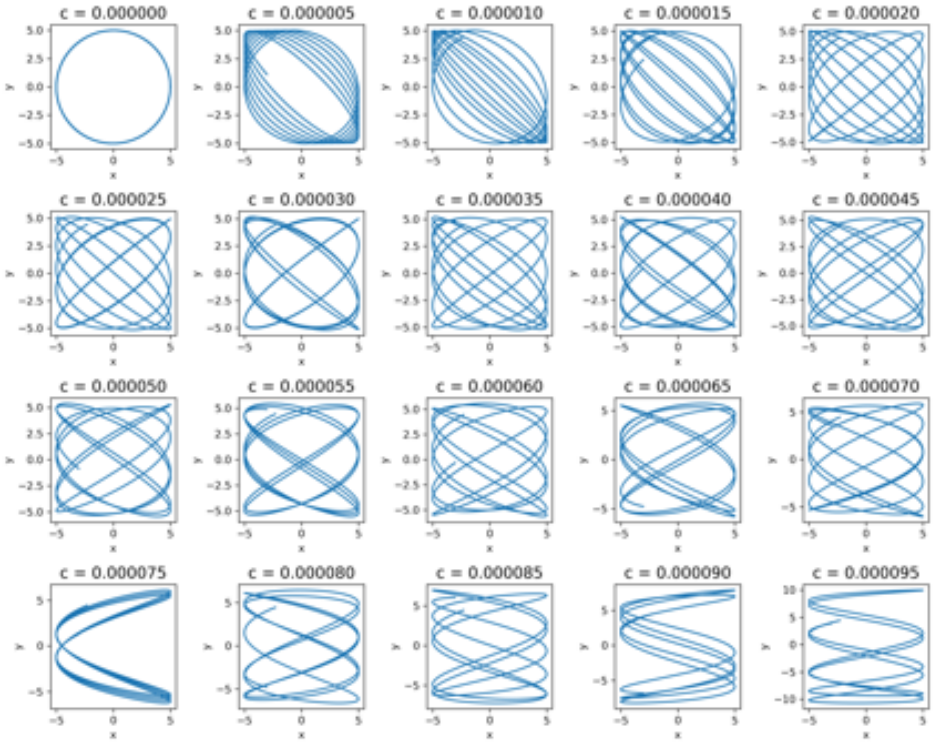


Figure 3.7: Modifying the learned linear system of the NG-RC for the prediction of circle trajectories by changing a parameter on the sixth digit after the comma leads to Lissajous curves.

from those associated with higher orders. The higher-order weights are dominant for small β values. Increasing β showcases how these weights are penalized by the regression parameter. The evolution of the higher-order weights around the threshold $\beta = 4 \cdot 10^{-8}$ is shown in Fig. 3.12 and Fig. 3.13. Essentially, they are suppressed by larger regression parameters such that their effect on the prediction becomes negligible, and the linear system becomes dominant. In Fig. 3.14-3.17 the evolution of the weights associated with the linear states is plotted. Noticeable, is that they stay constant close to 1 or -1 in range $\beta \in [1 \cdot 10^{-9}, 1 \cdot 10^{-5}]$, whereas the higher order weights are penalized. This results in a learned linear system that can predict circular trajectories even in the completely overlapping case. For $\beta > 1 \cdot 10^{-5}$, the regression starts to penalize the linear weights, resulting in a decreasing amplitude of the trajectory per prediction step, which ends up in a fixed point.

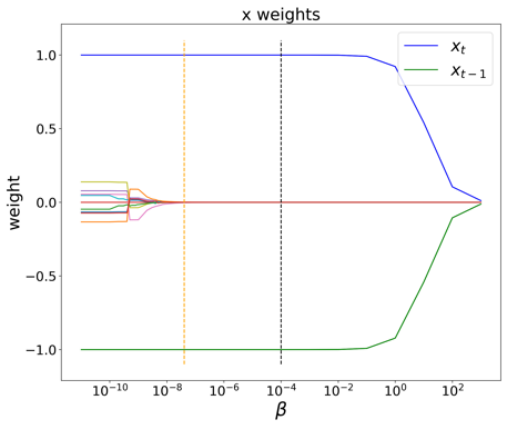


Figure 3.8: Evolution of the x associated weights of the $\mathbf{W}_{out}$ with respect to the regression parameter β.

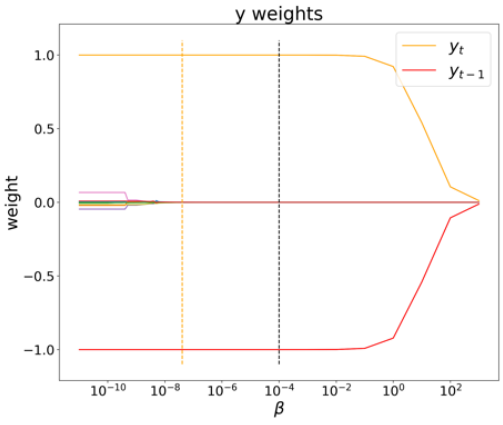


Figure 3.9: Evolution of the y associated weights of the $\mathbf{W}_{out}$ with respect to the regression parameter β.

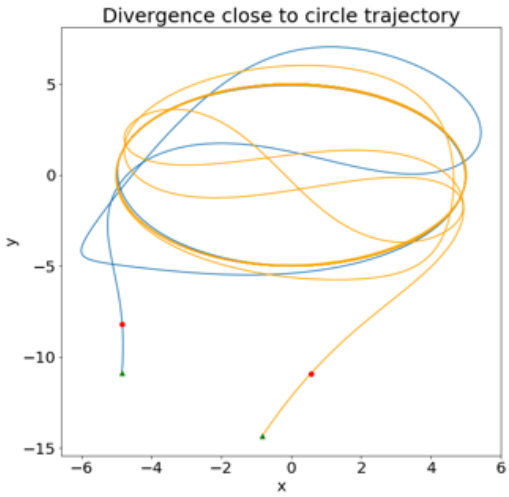


Figure 3.10: Diverging circle predictions for β close to circular predictions (yellow dashed line in Fig. 3.8 and 3.9).

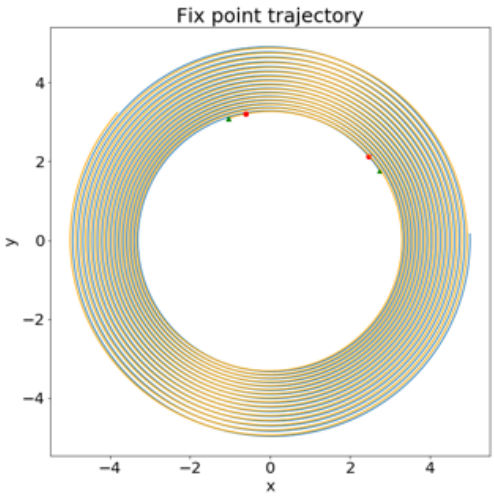


Figure 3.11: Fix point reaching predictions for β larger than those for circular predictions (black dashed line in Fig. 3.8 and 3.9).

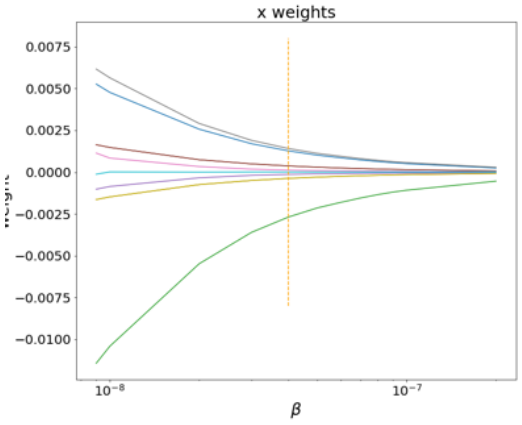


Figure 3.12: Zoomed in plot of the yellow dashed line in Fig. 3.8. Nonlinear x associated weights are getting suppressed by a growing regression parameter.).

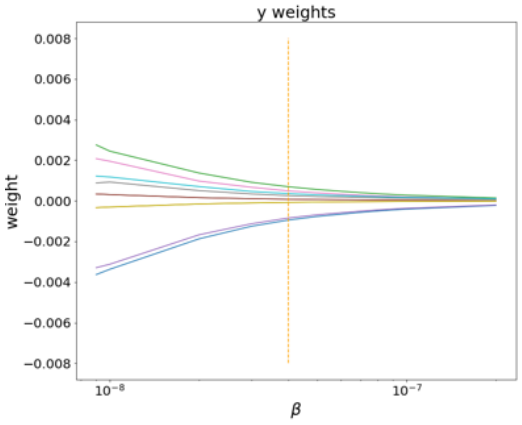


Figure 3.13: Zoomed in plot of the yellow dashed line in Fig. 3.9. Nonlinear y associated weights are getting suppressed by a growing regression parameter.

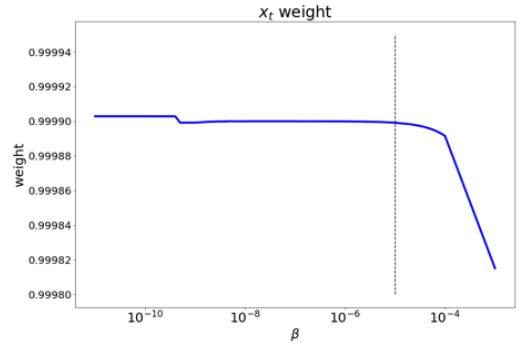


Figure 3.14: Zoomed in plot of the black dashed line in Fig. 3.8. Linear x associated weight is getting suppressed by a growing regression parameter.

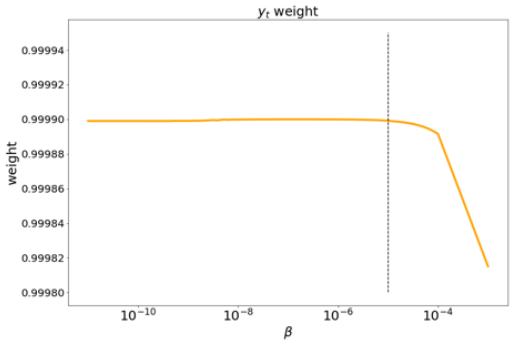


Figure 3.15: Zoomed in plot of the black dashed line in Fig. 3.9. Linear y associated weight is getting suppressed by a growing regression parameter.

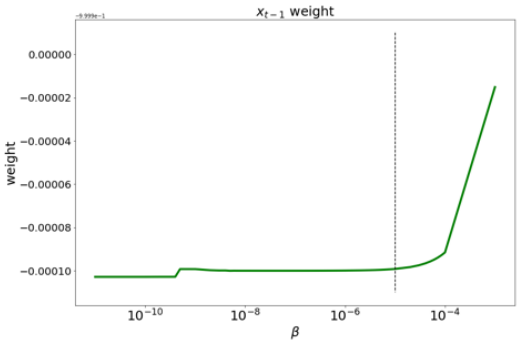


Figure 3.16: Zoomed in plot of the black dashed line in Fig. 3.8. Linear x associated weight is getting suppressed by a growing regression parameter.

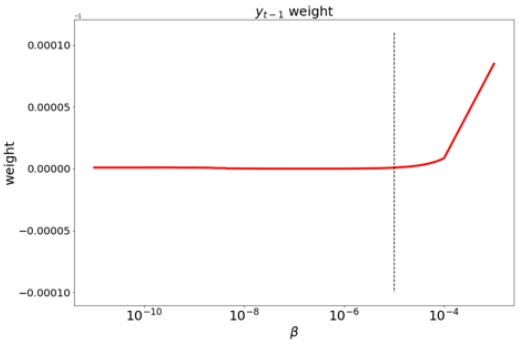


Figure 3.17: Zoomed in plot of the black dashed line in Fig. 3.9. Linear y associated weight is getting suppressed by a growing regression parameter.

3.2.2 Non and Partly Overlapping Case

To examine the multifunctional performance of NG-RC on non and partly overlapping contrarily rotating circles trajectories, we let the circles collide. Therefore, we set the center of the first circle on the x-axis to $x_{C_1,center} \in [-10, 10]$ with a step size of $\Delta x_{C1,center} = 0.5$ and the center of the second circle reflected on the y-axis to $x_{C_2,center} \in [10, -10]$ correspondingly. For every step, the NG-RC prediction of both circles is evaluated with an error metric called roundness [48]. It determines whether a trajectory has a periodic behavior and indicates its direction of rotation. If both properties fit the corresponding circle, the roundness $\delta(C)$ of it is calculated as the difference between the radius of the largest and smallest circle required to enclose and inscribe the predicted cycle. If the relative roundness $\delta(C)_{rel} = \frac{\delta(C)}{r}) < 0.15$, where r is the radius of the circle, then the trajectory is well predicted. And if this holds true for both circle predictions, the NG-RC achieved multifunctionality. In the following, we scan over different regression parameters β for the colliding case and evaluate the NG-RC performance with this metric. The list of regression parameters β is defined as $[9, 8, 7, 6, 5, 4, 3, 2, 1] \cdot 10^{-j}$ for $j \in [1, 2, 3, 4, 5, 6, 7, 8, 9]$.

As a first example, we look at the NG-RC setup used in the completely overlapping case, $\mathbf{P}^{[\mathbf{1,2}]}(\mathbf{L}_2^1())$. In Fig. 3.18, the yellow regions display that both predicted circles have a relative roundness of less than 0.15. This indicates regions of multifunctionality. The β in this figure shows the index of the above-defined regression parameter list.

For $x_{C_1,center} = 0$, it measures more accurately the results of Fig. 3.8 and Fig. 3.9, where the NG-RC shows in this case multifunctional behavior in range $\beta[26] = 0.0001$ and $\beta[86] = 3 \cdot 10^{-8}$. The figure also shows regions of multifunctionality in the partly overlapping case for $5 \geq |x_{C_1,center}| > 0$. This result raises the question of how this area of multifunctionality is affected by changing the k and the s value of the NG-RC.

In Fig. 3.19, this scan is applied on $k \in [2, 3, 4]$ and for $s \in [1, 2, 3, 4, 7]$ for $\mathbf{P}^{[\mathbf{1,2}]}(\mathbf{L}_k^s())$. For $k > 2$ similar multifunctionality behavior emerges. Regarding the behavior of the completely and the partly overlapping case, increasing s results in a less wide but therefore longer area. This means that the ability of NG-RC to predict the partly overlapping trajectories decreases when the time step size between the processed data points increases. This trades off to a larger β range for multifunctionality in the completely overlapping case.

If the enclosing length of the data processed by the NG-RC defined by ks ranges from 4-10, areas of multifunctionality emerge for the non-overlapping case. Interesting to note is that this behavior is similar across different k values. This basically shows that the performance and the functionality of a NG-RC is not inevitably better when it has more features. A deeper understanding of how k and s and the enclosed length of the data ks can be related to the behavior of the training data could be

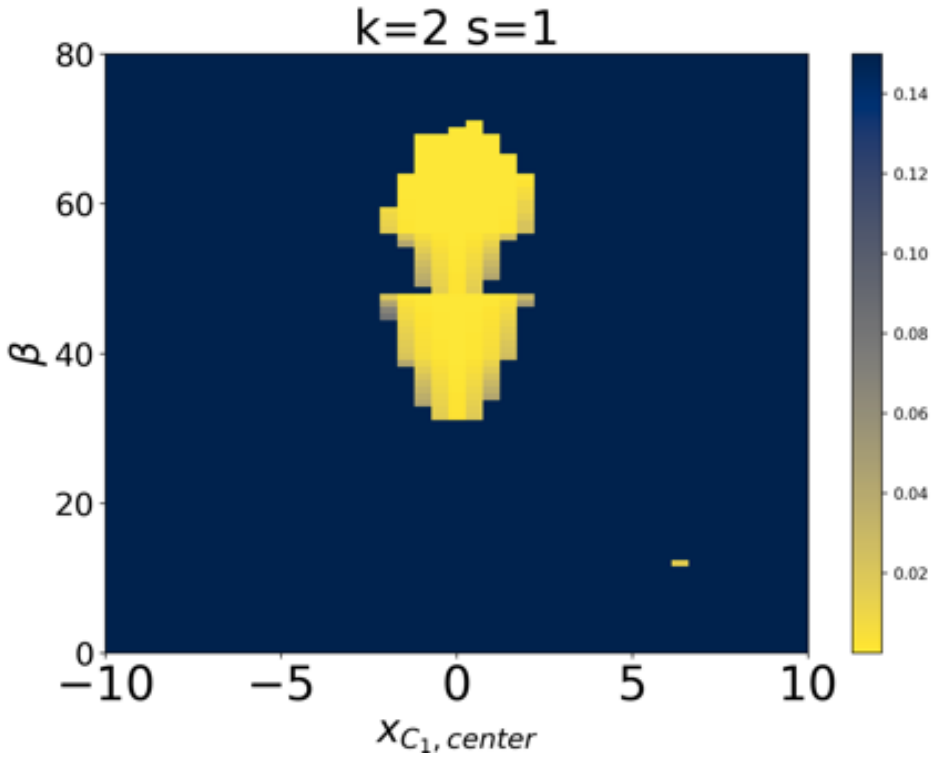


Figure 3.18: Colliding circles case scanned over different regression parameters β for $\mathbf{P}^{[1,2]}(\mathbf{L}_2^1())$. For $|x_{C_{1,center}}| > 5$ non overlapping case, for $5 \geq |x_{C_{1,center}}| > 0$ partly overlapping case and for $x_{C_{1,center}} = 0$ completely overlapping case. Yellow regions relate to the successful multifunctional behavior of the NG-RC. The smaller the value of the pixel, the better the circles were predicted.

worthwhile for setting up multifunctional NG-RCs.

Another question that arises is how the NG-RC performance changes when changing the orders in the polynomial multiplication dictionary. So far, we have stayed with $O = [1, 2]$. In Fig. 3.20, $k = 2$ is kept constant, and per row, the orders are changed or increased.

For $O = [1, 2, 3]$ in the second row, the NG-RC could not provide a stable, multifunctional prediction of the circles

One line below, $O = [1, 2, 4]$ again yields multifunctional predictions. They are dominant, especially in the non-overlapping case. The range of multifunctional predictions tends to increase with increasing s, implying that more regression parameters lead to periodic circular orbits. For $s > 2$, the NG-RC also shows multifunctionality in the partly overlapping case for one specific shift value.

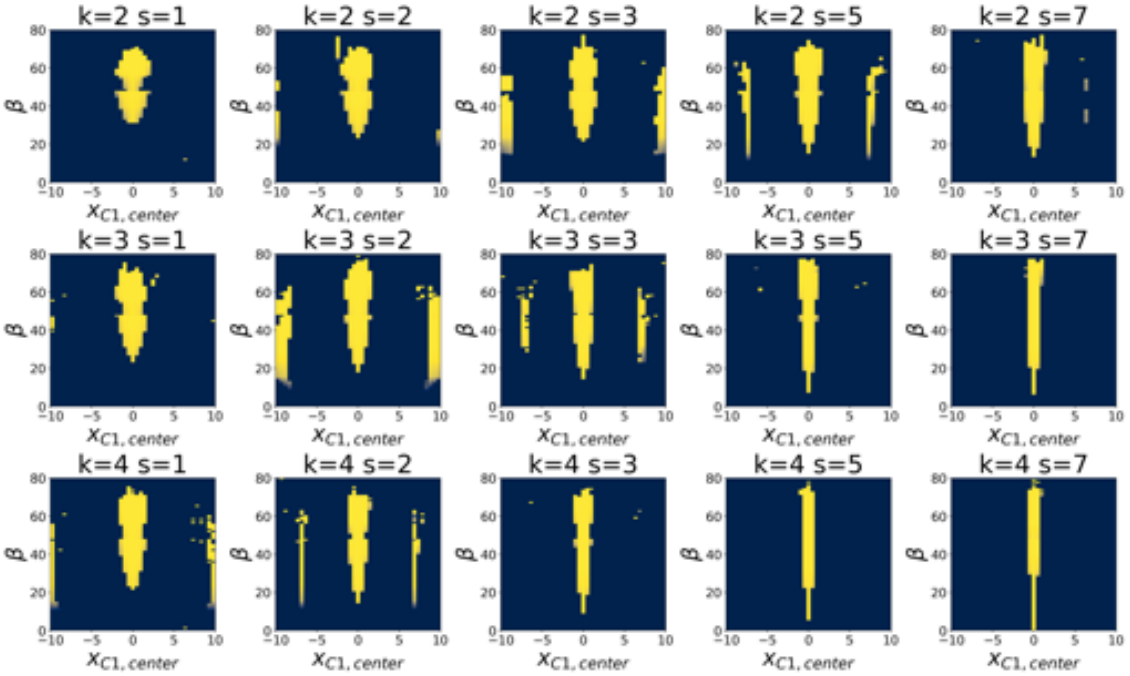


Figure 3.19: Colliding circles case scanned over different regression parameters β for $\mathbf{P}^{[1,2]}(\mathbf{L}_k^s())$.

The orders $O = [1, 2, 3, 4]$ provide multifunctional predictions only in the non-overlapping case. The range of multifunctional predictions decreases with increasing s until for $s = 7$, multifunctionality largely disappears.

These results indicate that the choice of orders plays a crucial role in the performance of the NG-RC. It seems that the performance is best when the selected orders match those of the training data, which is obviously $O = [1, 2]$. The inclusion of order 3 significantly degrades the performance in both cases. Whereas the inclusion of order 4 worsens the result in the completely and partly overlapping case, the NG-RC can still produce multifunctional predictions in the non-overlapping case. It is noteworthy that an increasing feature space does not necessarily lead to better or more functional NG-RC performances.

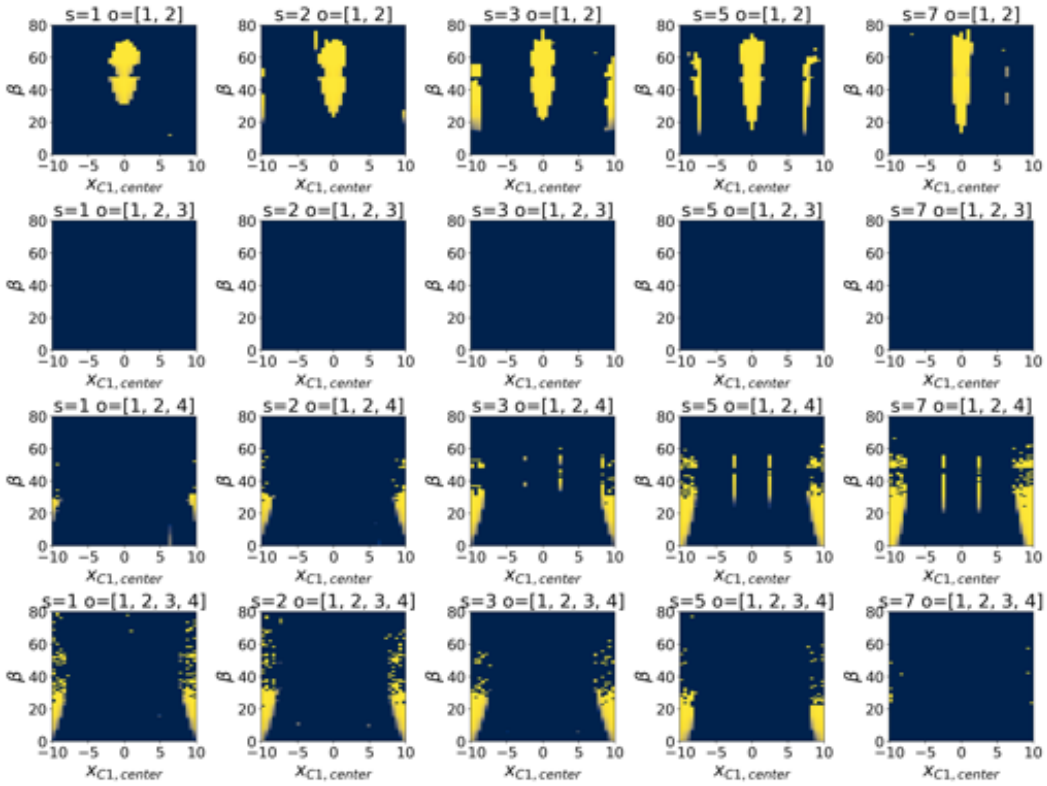


Figure 3.20: Colliding circles case scanned over different regression parameters β for $\mathbf{P}^{[\mathbf{O}]}(\mathbf{L}^{s}_{k=2}())$.

3.2.3 Conclusion

The NG-RC architecture is generally capable of solving the Seeing Double task. Some architectures even solved all three cases with the same parameterization. Despite choosing a NG-RC with polynomial terms, the trained NG-RC results in the completely overlapping case in an uncoupled set of linear equations, which can generate circular trajectories when initialized with two consecutive on any given circle. On the other hand, setting up a traditional RC on that task results in a learned limit cycle [18]. These qualitative differences could not have been resolved by a parameter scan in higher orders or different k and s values.

Considering the non-overlapping case, multifunctionality was achieved in most cases when the enclosing length ks of the data processed by the NG-RC was in the range $[4, 10]$. This experimentally found limitation might be data specific with respect to the time step size of the training data. A more fundamental understanding of this observation could be worthwhile for constructing multifunctional NG-RCs or RCs.

3.3 Lorenz and Halverson System

In neuroscience, the functional nature of chaotic attractors is of interest. The fact that new distinct trajectories keep emerging through one learned system could be the theoretical basis of creativity [49]. This concept is of interest, for example, in the neurological functioning of composing music or speaking language according to grammatical rules [50], [51], [52]. The following section expands the multifunctionality analysis of the NG-RC architecture to a more complex task. Two different chaotic attractors will be placed in phase space, from where the NG-RC should learn the dynamics of both attractors. The setup strictly follows the setup of [17], where the Lorenz attractor and the Halverson attractor are located diagonally opposite. A broad hyperparameter scan is made to test whether the NG-RC can learn both chaotic attractors such as its traditional counterpart.

Lorenz and Halverson Data

The Lorenz system in Eq. 3.15 and the Halverson system in Eq. 3.16 have a symmetry under the transformation $(x, y, z) \to (-x, -y, z)$. Hence, both attractors are suited as a more complex multifunctionality test case.

$$\begin{aligned} \dot{x} &= \sigma_L(y - x), \\ \dot{y} &= x(\rho_L - z) - y, \\ \dot{z} &= xy - \beta_L z \end{aligned} \tag{3.15}$$

$$\begin{aligned} \dot{x} &= -\sigma_H x - 4y - 4z - y^2, \\ \dot{y} &= -\sigma_H y - 4z - 4x - z^2, \\ \dot{z} &= -\sigma_H z - 4x - 4y - x^2 \end{aligned} \tag{3.16}$$

For both equations, the standard parametrization of $\sigma_L = 10$, $\rho_L = 28$, $\beta_L = \frac{8}{3}$ and $\sigma_H = 1.3$ is chosen. The training and testing data is generated by integrating Eq. 3.15 and Eq. 3.16 with the Runge-Kutta 4 method and a time step size of $\Delta t = 0.02$. In both scenarios, a training length of $T_L = 20.000$ for every attractor is used. Considering the warm-up time for each attractor data, the data is concatenated for the training phase. Furthermore, the data is normalized individually. In this setup, the center of the Halverson is shifted into the volume $(-1, -1, -1)^T$ and the center of the Lorenz onto the diagonal opposite $(1, 1, 1)^T$. Both attractors are then normalized, see Fig. 3.21.

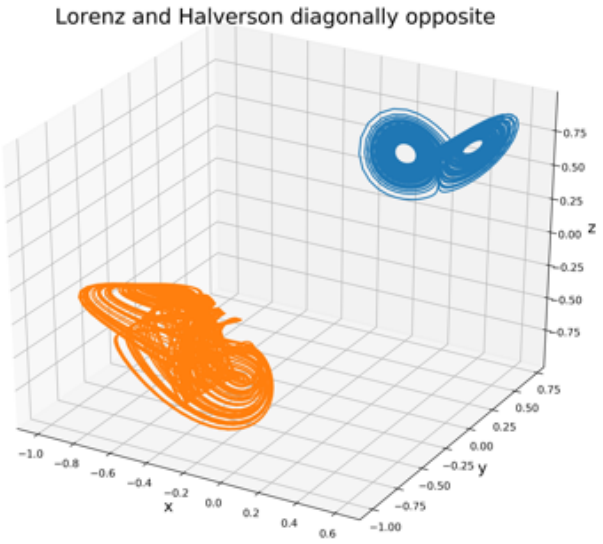


Figure 3.21: Fixed attractor setup as in [17]. Multifunctional performance of the NG-RC is evaluated on this setup up.

3.3.1 Comparing NG-RC and NVAR for Multifunctionality

In [53], Gauthier et. al postulated that the next generation reservoir computer has an implicit traditional reservoir computer. It will work for any task a Reservoir Computer has been applied to, but with much less work to optimize and deploy. He enumerates where NG-RC has been successfully applied, for example, on spatio-temporal systems, image classification, control, and radio-frequency signal identification. The multifunctionality task is an exceptional case of application, where the performance of traditional reservoir computers is outstanding. At this time, results on the performance of NG-RC on the multifunctionality task have not been published. In the following, we take the identical setup [17] and evaluate and compare the performance of NG-RC and NVAR on this task.

Feature Space

The quote that the NG-RC architecture will take far less work to deploy comes from the reduced feature space of the state vector compared to the traditional approach,

which is defined as the number of network nodes. In the paper, Gauthier et al. introduced the NG-RC as $\mathbf{P}^{[1,2]}(\mathbf{L}_k^s())$ and $\mathbf{P}^{[1,3]}(\mathbf{L}_k^s())$ basically, where for a three dimensional input data $\mathbf{x}_i$ the resulting feature spaces for different k values are of size,

Orders	k=1	k=2	k=3	k=4	k=5	k=6
$dim(\mathbf{P}^{[1,2]}(\mathbf{L}_k^s(\mathbf{x}_i))) =$	9	27	54	90	135	189
$dim(\mathbf{P}^{[1,3]}(\mathbf{L}_k^s(\mathbf{x}_i))) =$	13	62	174	376	695	>1000

Herteux et al. showed that a traditional RC with 500 nodes could successfully predict the Lorentz and Halverson task [17]. In section3.2, we saw that the NG-RC performs best if the orders of the polynomial multiplication dictionary relate to those of the true system. Hence, including higher orders to apply the NG-RC on a more complex task could be worthwhile. For the following parameter scan, all combinations of orders up to order eight and different k values are included, such that the resulting feature space is smaller than N = 1000. Further, for $k > 1$ we set $s = 1, 2, 3, 4$ and 5. Consequently, we test 407 different NG-RC architectures, including the 127 NVARs for $k = 1$, on the Lorenz and Halverson system.

Kernel Trick

Herteux et al. investigated the influence of a Lu readout [54] on the multifunctionality performance, which expands the current reservoir state vector $\mathbf{r}_i$ to $\tilde{\mathbf{r}}_i = (\mathbf{r}_i, \mathbf{r}_i^2) = (r_1, r_2, \ldots, r_1^2, r_2^2, \ldots)$. He showed that it breaks the symmetry of the activation function of the reservoir and overall increases the prediction performance. There is no symmetric activation function in NG-RC, but applying the Lu readout expands the current state vector with higher-order features. Gauthier et al. explained in the peer review [55] that going to higher-order polynomials with the NG-RC architecture is computationally expensive and suggests applying the kernel trick to obtain higher-order polynomial features in a computationally cheap way. Essentially, this follows the idea of the Lu readout but also includes higher orders of the state vector. Therefore, the performance of each of the 407 NG-RC architectures will be additionally evaluated for the following readout expansions.

$$\begin{aligned}
\mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{x}_i))^{[1]} &= \mathbf{r}_i \\
\mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{x}_i))^{[2]} &= (\mathbf{r}_i, \mathbf{r}_i^2) \\
&\vdots \qquad (3.17) \\
\mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{x}_i))^{[8]} &= (\mathbf{r}_i, \mathbf{r}_i^2, \mathbf{r}_i^3, \mathbf{r}_i^4, \mathbf{r}_i^5, \mathbf{r}_i^6, \mathbf{r}_i^7, \mathbf{r}_i^8) \qquad (3.18)
\end{aligned}$$

Training

To have a direct comparison to the traditional reservoir computing setup [17], we train the NG-RC to become an autonomous dynamical system, according to Eq. 3.8.

Performance Measures

The performance of a trained NG-RC is evaluated by comparing the correlation dimension and the largest Lyapunov exponent of the predicted data to those of the training data [17]. If these measures match, the NG-RC was able to learn the climate or long-term behavior of the corresponding attractor. Hence, the NG-RC succeeds in the multifunctionality task if the prediction data of the Lorenz system and the Halverson system can reproduce the measures of the corresponding training data.

Regression Parameter

One hyperparameter which is worthwhile optimizing is the regression parameter. The prediction of a trained NG-RC was highly sensitive to the regression parameter in first simulations. Therefore, we scan over $\beta \in [0.9, 0.8, 0.7, 0.6, 0.5, 0.4, 0.3, 0.2, 0.1] \cdot 10^{-i}$ for $i \in [1, 2, 3, 4, 5, 6, 7, 8, 9]$.

Starting Values

Further, a not yet mentioned hyperparameter is the starting time series of length $\delta t = ks$. In the first simulations, the performance of the NG-RC was sensitive to the starting series. Therefore, we expand the parameter scan over β for an additional dimension of 50 different starting time series per attractor. Starting from the last training data point to 500-time steps into the future with a step size of 10. The idea behind this additional scan is that a well-learned multifunctional system should be able to predict the corresponding attractor without dependence on the initial conditions when they are on the attractor. This needs to be fulfilled to call the

trained NG-RC a stable multifunctional system. On the other hand, traditional reservoir computers have a synchronization phase before they start predicting. Due to this phase, the dependence on the initial conditions convergence to zero, uncoupling it from the necessity of scanning over starting values.

3.3.2 Results

To illustrate the approach, the performance of $\mathbf{P}^{[1,2,3,4,5,6,7]}(\mathbf{L}_1^1())$ on the multifunctionality task can be seen for different kernel tricks in Fig. 3.22. In the images on the left, the regression parameter index on the x axis and the starting value shift index on the y axis is used to book multifunctional behaviors of the predictions. If both trajectories did not diverge or reach a fixed point after 20.000 prediction steps, the corresponding β and the starting value shift are marked as a yellow pixel in the image. The Lyapunov exponent and the correlation dimension of each of the predicted data are evaluated and scattered in corresponding plots on the right if the predicted data did not diverge or reach a fixed point correspondingly. An orange dot indicates that the Lorenz and the Halverson prediction run both for 20.000 prediction steps, whereas a blue one indicates a monofunctional. The red circles show the Lyapunov exponents' target region and the training data's correlation dimensions. The Lorenz and Halverson prediction measures should match these regions to have a good-performing multifunctional system. Each row differs by the kernel trick used to expand the state vector. Noticeably, in the first row, no prediction could run 20.000 steps without diverging. Simply concatenating each state vector with its squared entries, let regions of multifunctionality emerge. Kernel trick three further increases the multifunctional performance, and higher orders let it decrease. Hence, scanning for this method could be worthwhile for the performance of NG-RC or NVAR.

Scanning over 81 regression parameters and 51 starting value shifts and applying eight kernel tricks on 407 different NG-RC architectures results in 13.450.536 simulations. From this, the role of hyperparameters in this multifunctional task can be deduced. First, we separate the NVAR's for $k = 1$ and the NG-RC's for $k > 1$. Then we look at the prediction performance on single attractors for both algorithms as well as at the multifunctional predictions, given the feature space dimension. Only the prediction results are considered, whose Lyapunov exponent and correlation dimension have a maximal deviation of 5 % from those of the training data.

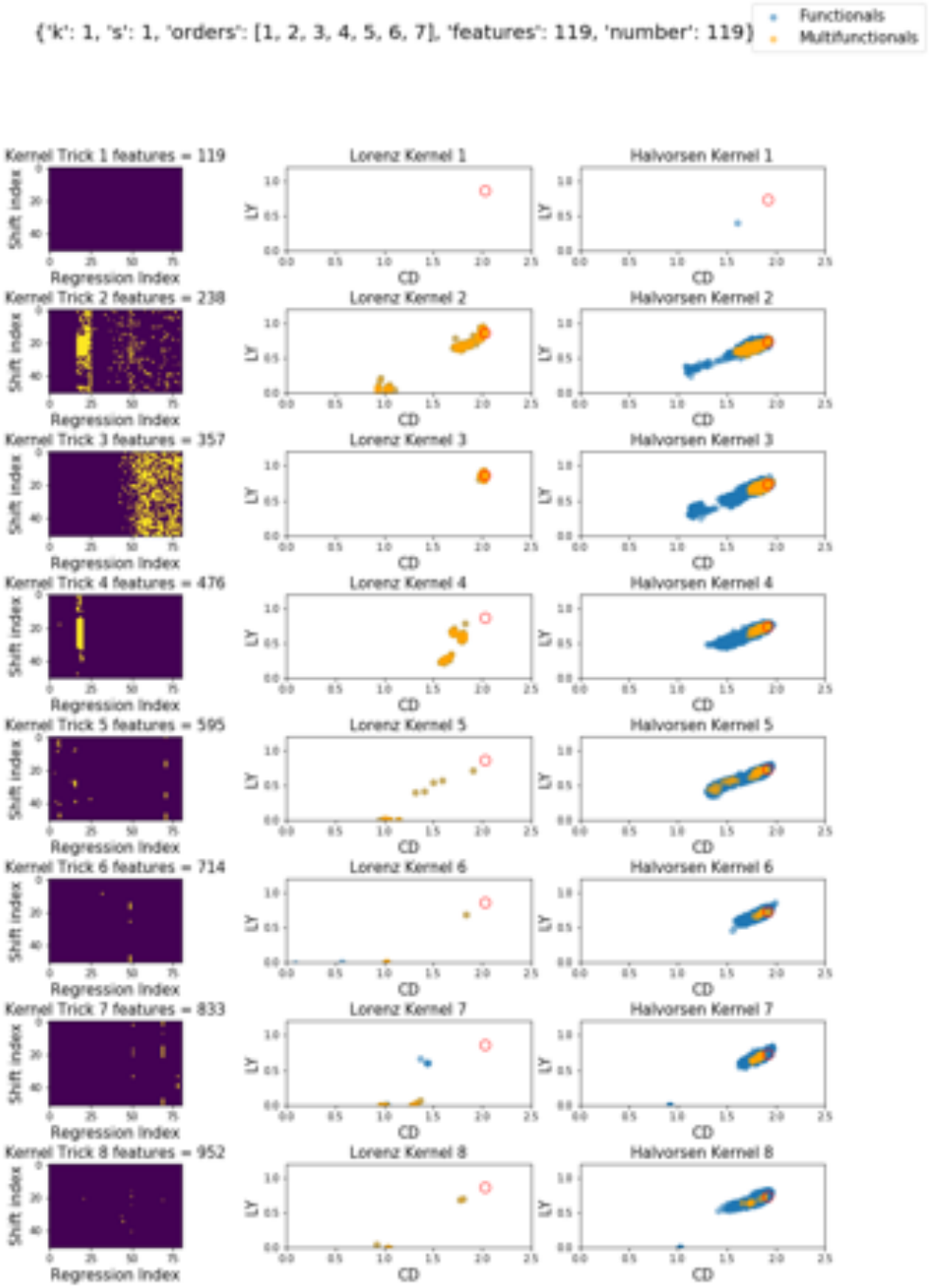


Figure 3.22: Parameter scan results of one out of 407 NG-RC architectures, including the 127 NVARs for $k = 1$. Left images bookkeep for which regression parameter and shift index the prediction of both attractors did not diverge or reach a fixed point, indicated by a yellow pixel. The right plots evaluate the correlation dimension (CD) and the Lyapunov exponent (LY) of the corresponding attractor if the prediction run for 20.000 prediction steps without diverging or reaching a fixed point. Orange dots indicate multifunctional behavior for 20.000 prediction steps. The LY and CD of the training data are highlighted as the red circles. Multifunctional predictions are good if the orange dots matches these circles. Blue dots indicate monofunctionals.

Feature Space

The amount of good performing single attractor predictions given the feature space dimension for a prediction length of 20.000 are scattered in Fig. 3.23 for the NVAR and in Fig. 3.24 for the NG-RC. Correspondingly, the distributions of the multifunctional predictions are shown in Fig. 3.25 and Fig. 3.26. There are a few interesting points to make here. First, the number of occurrences discloses a qualitative difference between the NVAR and the NG-RC for this task.

	Total Simulations	# good Pred.:	Lorenz	Halverson	Multifunctionals
NVAR	4197096		20083	415416	1325
NG-RC	9241047		154	4480	37

Table 3.1: Statistic of good predictions (max. deviation of 5% in the correlation dimension and the Lyapunov exponent compared to those of the training data and prediction length of 20.000 steps without diverging or reaching a fixed point.)

Further, the shapes of the distributions showcase that both performances of the algorithms do not increase with a larger feature space, which is thought to be the case for traditional reservoir computers. Instead, the distributions show maxima between 100 and 500 features. That indicates that the polynomial features represented in this area and the accordingly learned parameterization fit the mathematical description needed to reconstruct the attractor with respect to the training data of the other. Both algorithms could predict the Halverson system 20-30 times more often than the Lorenz system. Comparatively to traditional reservoir computing, only a small fraction of the trained algorithms showed multifunctionality.
One interesting question for further research here would be how the shape, position, and density of these distributions change with the distance of both attractors in real space. And especially if there is a relation between larger feature spaces and the minimized chance of multifunctionality.

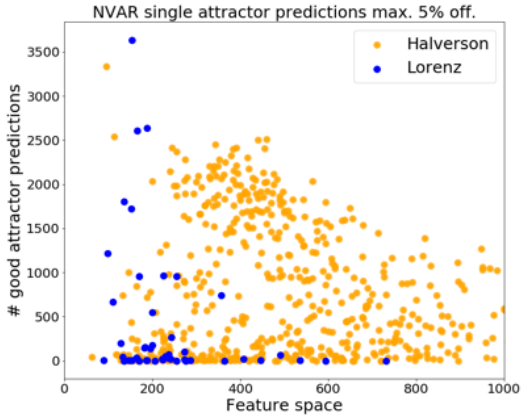


Figure 3.23: The number of good monofunctional NVAR predictions given the feature space dimension.

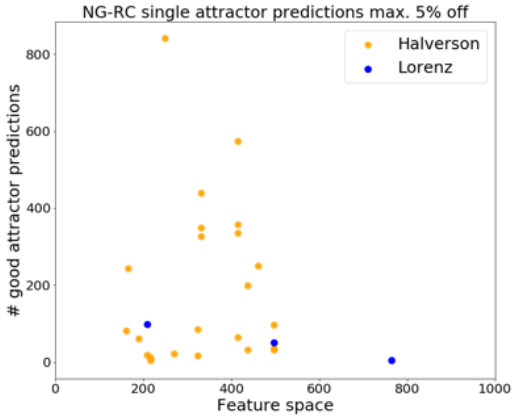


Figure 3.24: The number of good monofunctional NG-RC predictions given the feature space dimension.

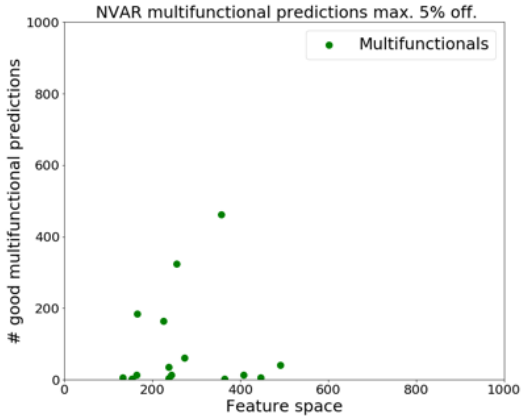


Figure 3.25: The number of good multifunctional NVAR predictions given the feature space dimension.

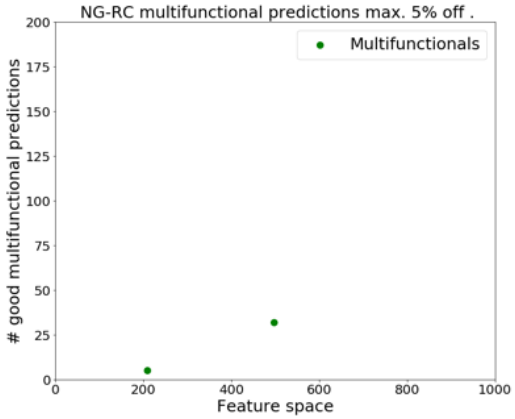


Figure 3.26: The number of good multifunctional NG-RC predictions given the feature space dimension.

Best Performing Prediction

The best performing multifunctional prediction was made by the NVAR $\mathbf{P}^{[1,2,3,4,5,6,7]}(\mathbf{L}_1^1())^{[3]}$ and $\beta = 7e-09$ for 20.000 prediction steps.

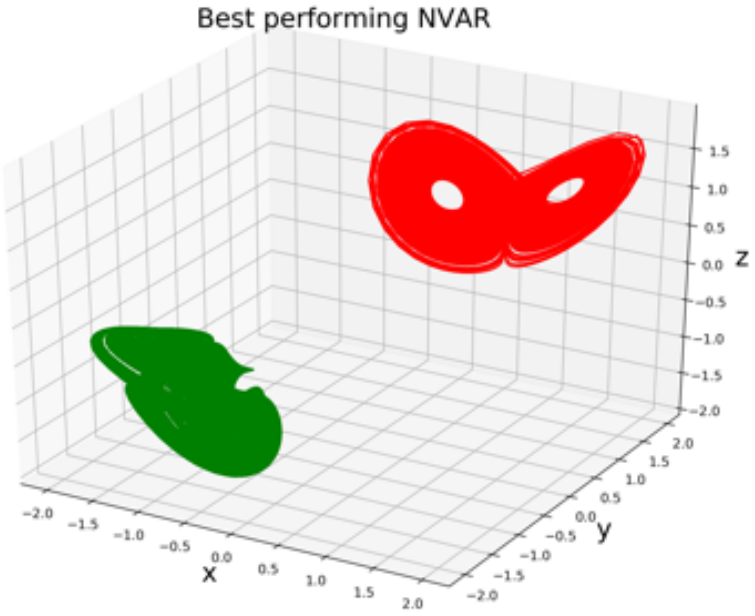


Figure 3.27: Prediction plots of table 3.2

	Training:	Lorenz	Halverson	Prediction:	Lorenz	Halverson
Lyapunov		0.857	0.727		0.859	0.726
Correlations Dim		2.033	1.924		2.019	1.914

Table 3.2: Best multifunctional prediction in this setup for $\mathbf{P}^{[1,2,3,4,5,6,7]}(\mathbf{L}_1^1())^{[3]}$ and $\beta = 7e-09$ and 20.000 prediction steps.

3.3.3 Conclusion

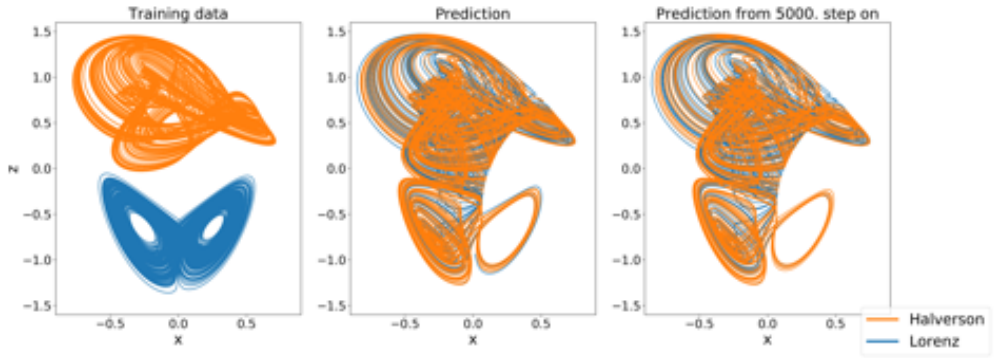
The NG-RC and the NVAR are, in general, capable of learning multistable chaotic dynamical systems with reasonable accuracy in climate prediction. However, the frequency of stable trajectories with 20.000 prediction steps is strikingly small, especially for the NG-RC architecture. This result may expose weaknesses in the NG-RC architecture in comparison to the multifunctional performance of traditional RC. While most of the NG-RC simulations diverge, traditional RC, with its activation function, includes a natural control mechanism that prevents the prediction from divergence. Furthermore, the sensitivity to the starting values of the prediction is high. Small changes often lead from stable trajectories to diverging ones or vice versa. In traditional RC, this behavior is generally not the case since the synchronization phase removes the dependence on the initial conditions. This makes the starting values a new parameter in the NG-RC architecture, which can be worth keeping in mind while searching for multistable systems. Besides this, applying the kernel trick and with it including higher order terms was generally beneficial. However, it is interesting to note that the principle of "the more, the better" does not apply here to the dimension of the feature space. Looking at Fig. 3.23-3.26, we see a decreasing frequency of successful predictions for larger feature spaces. A frequency maximum is reached at about a feature space dimension of N = 400. How the shape of this distribution changes for different multifunctional setups or how it changes for the two attractors with different positions in phase space are worthwhile questions for further research, especially while monitoring the evolution of the involved terms correspondingly. Moreover, as part of this work, the parameter scan was performed to create an autonomous dynamical system according to Eq. 3.8 to directly compare with the traditional RC based results [17]. Consequently, broadening the analysis for the one-step-ahead integrator approach or including a bias term may lead to different results.

3.4 Switching Between Attractors

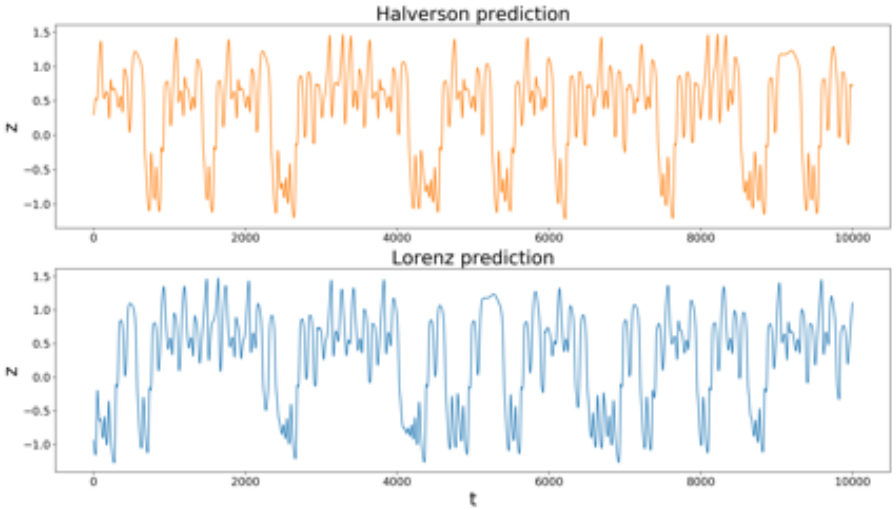
When the Lorenz and Halverson attractors are closer together in phase space, the NG-RC and NVAR architectures may allow new interesting dynamics to emerge, reminiscent of a human cognitive function, the ability to spontaneously switch between mimicking different dynamical patterns [56]. The following three examples are shown as a proof-of-principle. The setup is the same as in 3.3, with the difference that the positions of the attractors are closer together, and training is performed as a one-step integrator according to Eq. 3.4.

Chaotic Switching

Both attractors merge while retaining their structural features. Switching between the attractors appears to occur spontaneously. The results were generated with $\mathbf{P}^{[1,2,3,4,5,6,7]}(\mathbf{L}_1^1())^{[2]}$ and $\beta = 1$.



(a)

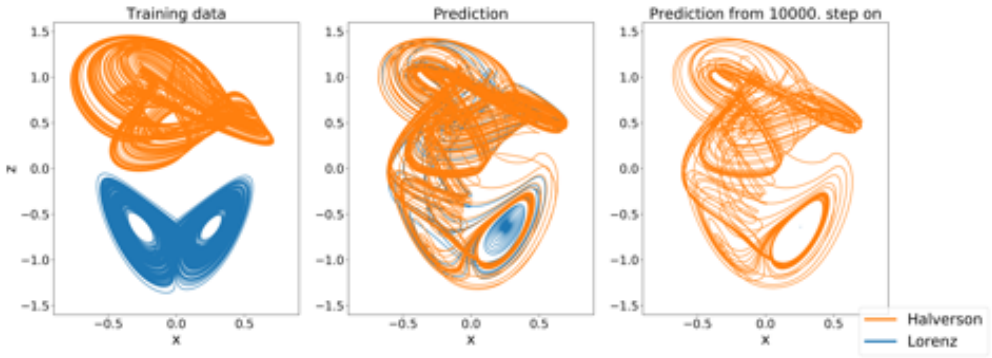


(b)

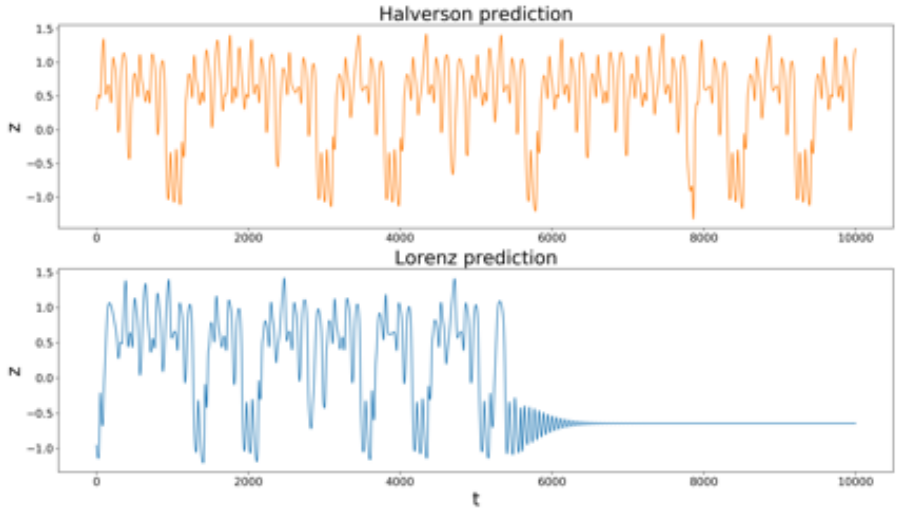
Figure 3.28: (a) Chaotic switching between the learned Halverson and Lorenz trajectories, (b) z-axis trajectories over time of the predictions.

Chaotic Switching to a Fixed Point

Both attractors merge while retaining some of their structural features. Switching between the attractors appears to occur spontaneously, and an additional fixed point is created. The results were generated with $\mathbf{P}^{[1,2,3]}(\mathbf{L}_2^2())^{[2]}$ and $\beta = 2$.



(a)

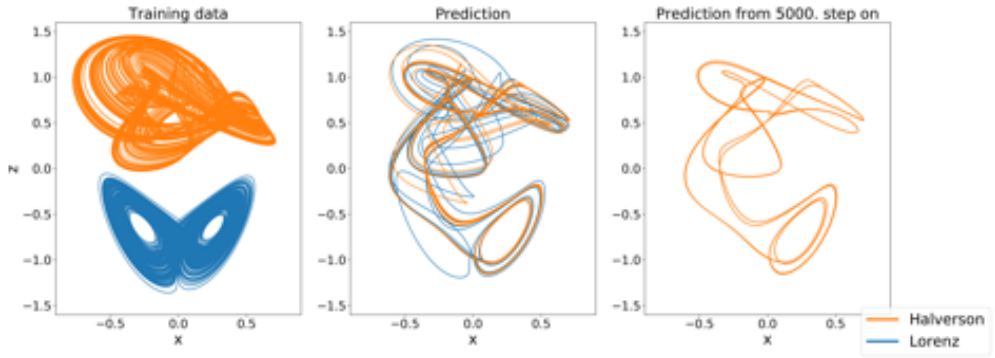


(b)

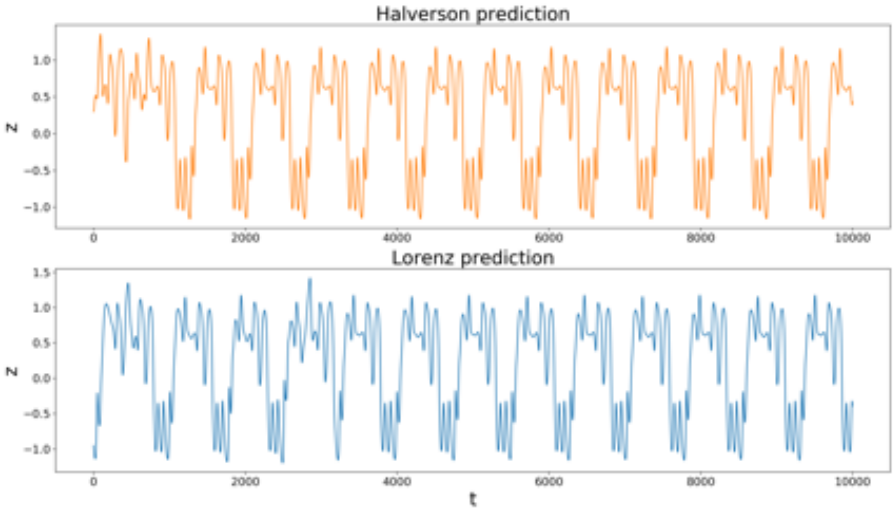
Figure 3.29: (a) Chaotic switching between the learned Halverson and Lorenz trajectories and emergence of additional fixed point, (b) z-axis trajectories over time of the predictions.

Synchronization to a Periodic Orbit

Both attractors synchronize into a new periodic orbit while keeping a few basic structural characteristics. Switching between the old attractor centers is periodic. The results were generated with $\mathbf{P}^{[1,2,3]}(\mathbf{L}_2^2())^{[2]}$ and $\beta = 2.1$. Note that this conceptual change is due to a minor change in the regression parameters compared to the previous example.



(a)



(b)

Figure 3.30: (a) Synchronization to a periodic orbit, (b) z-axis trajectories over time of the predictions.

3.5 Conclusion

By exploring the limits of multifunctionality with next generation reservoir computing (NG-RC), biological neural functions, like learning multiple chaotic attractors or spontaneous switching between them, could be imitated. Remarkably, this was possible without using a network structure. The necessary high dimensional representation of the input data is achieved by expanding it with time-shifted coordinates and polynomial multiplications. This way, high interpretability is created. Learning two completely overlapping oppositely rotating circular trajectories with the NG-RC architecture resulted in a linear feedback representation of the Lissajous curves. The governing equations could be read from the architecture, allowing traceability of how functionality is obtained or changed. This provides an exciting opportunity to study biological neural functions from a different perspective and through the analysis of governing equations. So far, reservoir computing has been used to explore multifunctionality in neural networks. It was shown that the surprising success of reservoir computing relies on invertible generalized synchronization. Furthermore, the mathematical equivalence of nonlinear autoregression architectures and traditional reservoir computing was shown. A logical and presumably worthwhile question for further research is whether invertible generalization synchronization is also the driving mechanism for the success of next generation reservoir computing. Moreover, the interpretability of the architecture allows for a simplified synchronization analysis of the system or even the individual terms. This provides an exciting starting point for further research at the interface of complex systems and machine learning.

Bibliography

[1] *Complex Intelligence: Natural, Artificial, and Collective, Santa Fe Institute*. https://www.santafe.edu/research/themes/complex-intelligence-natural-artificial-and-collec. Accessed: 2022-07-15.

[2] Christof Koch et al. "Neural correlates of consciousness: progress and problems". In: *Nature Reviews Neuroscience* 17.5 (2016), pp. 307–321.

[3] Todd E Feinberg and Jon Mallatt. "Phenomenal consciousness and emergence: Eliminating the explanatory gap". In: *Frontiers in Psychology* 11 (2020), p. 1041.

[4] Oriol Artime and Manlio De Domenico. *From the origin of life to pandemics: emergent phenomena in complex systems*. 2022.

[5] Stephen Mann. "Systems of creation: the emergence of life from nonliving matter". In: *Accounts of chemical research* 45.12 (2012), pp. 2131–2141.

[6] Felix Günther et al. "Nowcasting the COVID-19 pandemic in Bavaria". In: *Biometrical Journal* 63.3 (2021), pp. 490–502.

[7] Omar Sharif, Md Zobaer Hasan, and Azizur Rahman. "Determining an effective short term COVID-19 prediction model in ASEAN countries". In: *Scientific Reports* 12.1 (2022), pp. 1–11.

[8] Jingfang Fan et al. "Statistical physics approaches to the complex Earth system". In: *Physics reports* 896 (2021), pp. 1–84.

[9] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical systems and turbulence, Warwick 1980*. Springer, 1981, pp. 366–381.

[10] Simone Cenci et al. "Assessing the predictability of nonlinear dynamics under smooth parameter changes". In: *Journal of the Royal Society Interface* 17.162 (2020), p. 20190627.

[11] Steven L Brunton, Joshua L Proctor, and J Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: *Proceedings of the national academy of sciences* 113.15 (2016), pp. 3932–3937.

[12] Herbert Jaeger and Harald Haas. "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication". In: *science* 304.5667 (2004), pp. 78–80.

[13] Pantelis R. Vlachas et al. "Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics". In: *Neural networks : the official journal of the International Neural Network Society* 126 (2020), pp. 191–217.

[14] S Bompas, Bertrand Georgeot, and David Guéry-Odelin. "Accuracy of neural networks for the simulation of chaotic dynamics: precision of training data vs precision of the algorithm". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.11 (2020), p. 113118.

[15] Zhixin Lu, Brian R Hunt, and Edward Ott. "Attractor reconstruction by machine learning". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6 (2018), p. 061104.

[16] Zhixin Lu and Danielle S Bassett. "Invertible generalized synchronization: A putative mechanism for implicit learning in neural systems". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.6 (2020), p. 063133.

[17] Joschka Herteux and Christoph Räth. "Breaking symmetries of the reservoir equations in echo state networks". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.12 (2020), p. 123142.

[18] Andrew Flynn, Vassilios A Tsachouridis, and Andreas Amann. "Multifunctionality in a reservoir computer". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (2021), p. 013125.

[19] Jaideep Pathak et al. "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach". In: *Physical review letters* 120.2 (2018), p. 024102.

[20] Sebastian Baur and Christoph Räth. "Predicting high-dimensional heterogeneous time series employing generalized local states". In: *Physical Review Research* 3.2 (2021), p. 023215.

[21] Alexander Haluszczynski and Christoph Räth. "Controlling nonlinear dynamical systems into arbitrary states using machine learning". In: *Scientific reports* 11.1 (2021), pp. 1–8.

[22] Daniel J Gauthier et al. "Next generation reservoir computing". In: *Nature communications* 12.1 (2021), pp. 1–8.

[23] Wendson AS Barbosa and Daniel J Gauthier. "Learning Spatiotemporal Chaos Using Next-Generation Reservoir Computing". In: *arXiv preprint arXiv:2203.13294* (2022).

[24] Roayat Ismail Abdelfatah. "Audio encryption scheme using self-adaptive bit scrambling and two multi chaotic-based dynamic DNA computations". In: *IEEE Access* 8 (2020), pp. 69894–69907.

[25] Yi He et al. "A new image encryption algorithm based on the OF-LSTMS and chaotic sequences". In: *Scientific Reports* 11.1 (2021), pp. 1–22.

[26] Akhil Kaushik and Vikas Thada. "VG4 Cipher: Digital Image Encryption Standard". In: *International Journal of Advanced Computer Science and Applications* 12.4 (2021), pp. 127–133.

[27] Rajkumar Ramamurthy et al. "Using echo state networks for cryptography". In: *International Conference on Artificial Neural Networks*. Springer. 2017, pp. 663–671.

[28] Mohamed Zakariya Talhaoui, Xingyuan Wang, and Mohamed Amine Midoun. "Fast image encryption algorithm with high security level using the Bülban chaotic map". In: *Journal of Real-Time Image Processing* 18.1 (2021), pp. 85–98.

[29] Je Sen Teh, Moatsum Alawida, and You Cheng Sii. "Implementation and practical problems of chaos-based cryptography revisited". In: *Journal of Information Security and Applications* 50 (2020), p. 102421.

[30] Omar Farook Mohammad et al. "A survey and analysis of the image encryption methods". In: *International Journal of Applied Engineering Research* 12.23 (2017), pp. 13265–13280.

[31] Doreen Jirak et al. "Echo state networks and long short-term memory for continuous gesture recognition: A comparative study". In: *Cognitive Computation* (2020), pp. 1–13.

[32] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), p. 13.

[33] Sanjukta Krishnagopal et al. "Separation of chaotic signals by reservoir computing". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.2 (2020), p. 023123.

[34] John Towner. *Ancient Bristlecone Pine Forest, United States*. [Unsplash license; Online; accessed July 25, 2022]. October 25, 2016. URL: https://unsplash.com/photos/JgOeRuGD_Y4.

[35] Daniel Leone. *Poon Hill, Ghode Pani, Nepal*. [Unsplash license; Online; accessed July 25, 2022]. January 6, 2017. URL: https://unsplash.com/photos/g30P1zcOzXo.

[36] Shuliang Sun. "A novel hyperchaotic image encryption scheme based on DNA encoding, pixel-level scrambling and bit-level scrambling". In: *IEEE Photonics Journal* 10.2 (2018), pp. 1–14.

[37] Xiuli Chai, Kang Yang, and Zhihua Gan. "A new chaos-based image encryption algorithm with dynamic key selection mechanisms". In: *Multimedia Tools and Applications* 76.7 (2017), pp. 9907–9927.

[38] Xiuli Chai et al. "A novel image encryption algorithm based on the chaotic system and DNA computing". In: *International Journal of Modern Physics C* 28.05 (2017), p. 1750069.

[39] Dejian Fang and Shuliang Sun. "A new secure image encryption algorithm based on a 5D hyperchaotic map". In: *Plos one* 15.11 (2020), e0242110.

[40] Thomas Lefebvre. *Punta Cana, Dominican Republic*. [Unsplash license; Online; accessed July 25, 2022]. January 15, 2015. URL: https://unsplash.com/photos/V63oM8OPJSo.

[41] Axel Cleeremans, Arnaud Destrebecqz, and Maud Boyer. "Implicit learning: News from the front". In: *Trends in cognitive sciences* 2.10 (1998), pp. 406–416.

[42] Andrew Flynn et al. "Exploring the limits of multifunctionality across different reservoir computers". In: *arXiv preprint arXiv:2205.11375* (2022).

[43] Peter A Getting. "Emerging principles governing the operation of neural networks". In: *Annual review of neuroscience* 12.1 (1989), pp. 185–204.

[44] Patsy S Dickinson. "Interactions among neural networks for behavior". In: *Current opinion in neurobiology* 5.6 (1995), pp. 792–798.

[45] Kevin L Briggman and William B Kristan. "Imaging dedicated and multifunctional neural circuits generating distinct behaviors". In: *Journal of Neuroscience* 26.42 (2006), pp. 10925–10933.

[46] Katsuma Inoue, Kohei Nakajima, and Yasuo Kuniyoshi. "Designing spontaneous behavioral switching via chaotic itinerancy". In: *Science advances* 6.46 (2020), eabb3989.

[47] Erik Bollt. "On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (2021), p. 013108.

[48] Andrew Flynn, Vassilios A Tsachouridis, and Andreas Amann. "Seeing double with a multifunctional reservoir computer". In: *Not yet published* (2022).

[49] Nancy C Andreasen. "A journey into chaos: Creativity and the unconscious". In: *Mens sana monographs* 9.1 (2011), p. 42.

[50] Gottfried Mayer-Kress, Robin Bargar, and Insook Choi. *Musical structures in data from chaotic attractors*. University of Illinois at Urbana-Champaign, 1992.

[51] Rick Bidlack. "Chaotic systems as simple (but complex) compositional algorithms". In: *Computer Music Journal* 16.3 (1992), pp. 33–47.

[52] Jeffrey L Elman. "Language as a dynamical system". In: *Mind as motion: Explorations in the dynamics of cognition* (1995), pp. 195–223.

[53] Dan Gauthier. "Next generation reservoir computer - Dan Gauthier". In: *https://www.youtube.com/watch?v=HGsQrD6pPGM* ().

[54] Zhixin Lu et al. "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.4 (2017), p. 041102.

[55] Daniel J Gauthier et al. "Peer Review File: Next Generation Reservoir Computing". In: *nature portfolio* (2021).

[56] Yoav Kessler, Yael Shencar, and Nachshon Meiran. "Choosing to switch: Spontaneous task switching despite associated behavioral costs". In: *Acta psychologica* 131.2 (2009), pp. 120–128.

Acknowledgements

I would like to thank Christoph for the great supervision of my thesis. I enjoyed being able to follow my ideas while his expertise guided them in the right direction. I'd also like to thank Andrew and Oliver for the great discussions about multifunctionality and experiencing how overly fast things can get done. ;)
I would also like to thank Sebastian for the great support in obtaining sufficient or sometimes excessive computing power.
Looking forward to it!

Erklärung zur Masterarbeit

Hiermit erkläre ich, die vorliegende Arbeit selbständig verfasst zu haben und keine anderen als die in der Arbeit angegebenen Quellen und Hilfsmittel benutzt zu haben.

Master's thesis statement of originality

I hereby confirm that I have written the accompanying thesis by myself, without contributions from any sources other than those cited in the text and acknowledgments.

.....	.....
Date and location	Signature