Multifunktionale Echo State Netzwerke: Auswirkungen von Topologie und Gedächtnis auf die Rekonstruktion von chaotischen Attraktoren

Multifunctional Echo State Networks: Effects of Topology and Memory on the Reconstruction of Chaotic Attractors

Masterarbeit an der Fakultät für Phyisk der
Ludwig-Maximilans-Universität München

vorgelegt von
Oliver Heilmann

betreut von
Dr. Christoph Räth

March 19, 2023

Contents

Chapter 1

Introduction

"Forecasting is very difficult especially about the future" is a famous quote attributed to the physicist Niels Bohr. Although most people would intuitivly agree with this statement it never stopped humanity from predicting or at least trying to predict all kinds of events. In the past, the methods used for prediction were more than just questionable, usually involving some kind of religious or spiritual ritual. The best known example might be the Oracle of Delphi. Of course any serious prediction should be based on facts and be independent of any subjective bias. The informational advantge gained by a reliable prediction cannot be underestimated. In Ancient Egypt for instance, priests developed a secret device, called Nilometer, to measure the water level of the Nile. Based on recorded historic data, they predicted future water levels. Since the economoy of Egypt was mostly farming, they could also predict the output of the economy [1]. Data-based approaches are still popular today. It is argued that the past values of a system contain information about its future development. Still, making predictions just on past data is not easy and leaves one with two options: Either based on the past observations a simple set of equations which describes the time evolution of the system is found or a universal model powerful enough to fit any function is used. The first option is what Newton did when he wrote down the formula of gravitational force to calculate trajectories of arbitrary objects in gravitational fields. Here the difficulty is qualitative. To come up with the correct equations requires a deep understanding of the problem and often also sheer luck. And while Newton was without doubt a genius, in the end it was an apple falling on his head guiding him towards the correct solution. In the second option a lot of data needs to be stored and processed. This requires powerful computer processors and huge amounts of memory. With the computational power of modern processors and the highly available memory this approach is now feasible. The only question left to answer is how such an universal model can be built. It might be surprising but the answer is literally inside us. It is the human brain. The brain is a powerful system allowing us humans to solve complex tasks and to plan and predict future events. Inspired by the human brain, Maass, Natschläger and Markram proposed Liquid State Machines (LSM) as a model for artificial neural networks [2]. Related to LSMs are Echo State Networks (ESN). Instead of using 'integrate-and-fire' neurons such as LSMs, neurons in ESNs are nodes within a graph propagating their values by a simple matrix multiplication. This leads to a straightforward implementation on computers.

Historically ESNs did not derive as better computeable versions of LSMs. Initially they were proposed by Jaeger as a type of a Recurrent Neural Network (RNN) with a single trainable layer [3]. Later on Verstraeten et al. compared LSMs and ESNs and unified both approaches under the umbrella term of "reservoir computers" (RC) [4]. A RC is a general concept utilizing the computational power of a high-dimensional non-linear space. Still the close connection between ESNs and LSMs and as such to Computational Neuroscience and Neuroscience in general is a key factor in this thesis.

The mathematical framework describing ESNs is based on the formulation of dynamical systems. Accordingly, it is a natural step to benchmark ESNs with chaotic attractors. While the time evolution of chaotic attractors, given by a set of ordinary differential equations, is deterministic, reliable forecasting is an untractable problem. As Lorenz noticed in his groundbreaking paper about nondeterministic periodic flow, introducing the Lorenz attractor as the first example of a chaotic attractor: "[The results] indicate that prediction of the sufficiently distant future is impossible by any method, unless the present conditions are known exactly" [5].

Recently, it was shown that a single ESN can learn more than just a single attractor [6] [7]. This phenomenon is now known as multifunctionality. Multifunctionality is another bridge between ESNs and neuroscience. Initially the term multifunctionality was introduced to describe the ability of biological neural circuits to switch between different tasks [8]. Arguably multifunctionality is one of the most exciting recent developments. Consequently, the research in this thesis focuses on multifunctional ESNs with the goal to understand the influence of certain properties of ESNs on multifunctionality. After introducing ESNs in chapter 2, chapter 3 focuses on the topology. The close connection between neuroscience and ESNs is the main driver in this chapter. Random, scale-free and small-world topologies are used to investigate ESNs with different kind of degree distributions. Several node properties are measured with respect to the node input degree. Connecting the results with known results from neuroscience helps to understand if multifunctionality in ESNs is similar to multifunctionality in biological neural circuits. The following two chapters bring multifunctionality to its limits by training ESNs on overlapping attractors. Overlapping attractors are special since trajectories of dynamical systems cannot overlap. It is argued that memory must be used to separate the overlapping attractors within the reservoir state space. The concept of memory is defined in chapter 4. It turns out that there is no unique definition of memory. This issue is addressed by using two different memory measures. The first memory measure was formulated by Jaeger in "Short term memory in Echo State Networks" [9] and measures the ability of an ESN to remember past states. The second memory measure introduced by Inubushi and Yoshimura in "Reservoir computing beyond memory-nonlinearity trade-off" [10] is based on the ability of an ESN to distinguish between two nearby states from the past. Although both definitions are reasonable, it is shown that they capture different properties. Under the variation of certain hyperparameters it is possible to decrease one of the memory measures while increasing the other. Chapter 5 creates multiple ESNs with a large spectrum of memory properties and trains them on overlapping attractors. The main goal of this chapter is to figure out if memory has an influence on the accuracy of short-term predictions. Additionally the measured memory capacities are related to two more quantities from the training process.

Chapter 2

Concepts and Methods

2.1 Dynamical Systems and Chaotic Attractors

A continous dynamical system is defined by an autonoumous ordinary differential equation[1]

$$\frac{ds_1}{dt} = f_1(s_1, s_2, \cdots, s_n) \tag{2.1}$$

$$\frac{ds_2}{dt} = f_2(s_1, s_2, \cdots, s_n) \tag{2.2}$$

$$\vdots \tag{2.3}$$

$$\frac{ds_n}{dt} = f_n(s_1, s_2, \cdots, s_n) \tag{2.4}$$

$$\tag{2.5}$$

In vector notation this can be written as $\dot{\vec{s}} = \vec{f}(\vec{s})$. For a solution $\vec{s}(t)$, $f(\vec{s}(t))$ defines tangent vectors on each point of this trajectory. These vectors are interpreted as the rate of change of the trajectory. As a consequence, two trajectories can never cross each other since this would require two different rates of change at the crossing point. Considering the long-term behaviour of the trajectory there are only a few possible outcomes:

1. The trajectory diverges to infinity: $\lim_{t\to\infty} \|\vec{s}(t)\| \to \infty$.
2. The trajectory converges towards a fix point: $\lim_{t\to\inf} \vec{s}(t) = \vec{s}^\star$
3. The trajectory ends up in a periodic state: $\vec{s}(t+T) = \vec{s}(t)$. In more than one dimension it is also possible to end up in a quasiperiodic state where periodicity holds only for each component: $\vec{s}_i(t+T_i) = \vec{s}_i(t)$.
4. The trajectory stays confined in a bounded space without ending up in a fix point or in a (quasi-) periodic state. Such states are called strange attractors. A strange attractor is called chaotic if two arbitrary close points diverge exponentially fast during time evolution.

The most prominent example for a chaotic attractor is the Lorenz attractor given by the three differential equations;

$$\frac{dx}{dt} = \sigma \cdot (y - x) \tag{2.6}$$

$$\frac{dy}{dt} = x \cdot (\rho - z) - y \tag{2.7}$$

$$\frac{dz}{dt} = x \cdot y - \beta \cdot z \tag{2.8}$$

The values of the parameters are chosen to be $\sigma = 10$, $\rho = 28$ and $\beta = \frac{8}{3}$. In general, it is not possible obtain an analytic solution for a dynamical system. Nonetheless, starting with an initial

[1]This is not the most general definition for a continous dynamical system, but it is sufficient for all problems considered in this thesis

condition $\vec{s}(0)$ a solution can be approximated numerically. This turns the continous dynamical system into a discrete dynamical system where the time evolution is given by a map. All numerical discretizations of chaotic attractors in this thesis, except when mentioned otherwise, use a fourth-order Runge-Kutta method with a timestep of $\Delta t = 0.02$.

$$\vec{s}[t+1] = M[\vec{f}](\vec{s}[t]) = RK4[\vec{f}, \Delta t = 0.02](\vec{s}[t]) \tag{2.9}$$

$\vec{s}[t]$ is then called a (discrete) time series with $\vec{s}[0] = \vec{s}(0)$. It should be noted that in the discrete formulation $t \in \mathbb{N}$ is the time step. It relates to to the time by $t \cdot \Delta t$.

Previously it was shown that for continuous dynamical systems two different trajectories can not cross each other. A similar result holds for discrete time series. Since $M[f]$ is a function on the state space of the dynamical system, every point has only one successor. Therefore, if two points on two different time series are not equal, all preceding points must also be unequal.

2.2 Echo State Networks

In the introduction, Echo State Networks (ESNs) are introduced once as better computeable models of the human brain derived from Liquid State Machines and once as simpler versions of Recurrent Neural Networks. Here, a third way of deriving ENSs is used which leads to their mathematical formulation and their training process.

According to Mallat "Supervised machine learning is a high-dimensional interpolation problem" [11]. Following this approach and applying it to discrete time series prediction tasks is almost enough to understand ESNs and reservoir computers. In the first step time series data is embedded into a high-dimensional space. The second step, and this is missing in Mallat's quote, is a non-linear transformation. In reservoir computers the non-linear transformation plays the role of a time evolution. The third and last step projects states from the high-dimensional space back to the low-dimensional space of the the time series. As stated by Mallat this can be done by an interpolation which in the simplest case is just a linear mapping. The idea behind reservoir computers is to treat the high-dimensional system as a black box and to train only the interpolation/linear map.

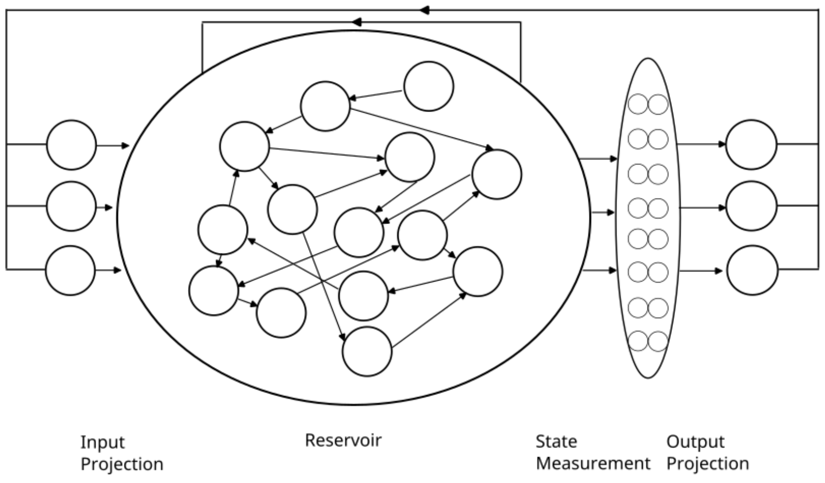

Figure 2.1: Architecture of an Echo State Network. Input data is projected into the high-dimensional reservoir space given by a graph with a non-linear transformation. The reservoir state is measured, for instance by the extended-Lu state measurement, and then projected back into the time series state space by the readout. The result can then be used for the next step in the time evolution.

An ESN is a certain kind of reservoir computer where the reservoir, the high-dimensional non-linear space, is given by a directed weighted Graph with a non-linear transformation. A directed weighted graph G can be described by a set of $N \in \mathbb{N}^+$ nodes $V = \{v_0, v_1, v_2, ...v_{N-1}\}$ where each node v_i holds a real value. Edges $E = \{(v_i, v_j, w), ...\}$ describe the directed connections between the nodes. A tuple (v_i, v_j, w) is part of the set E if and only if there is a connection from node $v_n \in V$ to node $v_m \in V$ with a value $w \in \mathbb{R}$. If there exists a tuple $(v_n, v_m, w_1) \in E$ then there cannot exists another tuple $(v_n, v_m, w_2) \in E$. Edges connecting a node with itself are allowed. With these restrictions a graph $G = (V, E)$ can be represented by an adjacency matrix $M \in \mathbb{R}^{N \times N}$ where $M_{nm} = w$ if and only if $(v_n, v_m, w) \in E$. Otherwise $M_{nm} = 0$. Most of the time it is enough to refer to a node only by its index. For instance, if the n-th node is mentioned, then the node with the index n refering to $v_n \in V$ is meant. A vector $\vec{r} \in \mathbb{R}^N$ contains the values which are assigned to the nodes in the graph. $\vec{r}_n \in \mathbb{R}$ is the value assigned to node n. The vector $\vec{r}$ is called the reservoir state.

A common choice for the non-linear function propagating the reservoir state in time is the tangens hyperbolicus [6] [12] [13]. Apart from the reservoir, an ESN consists of an input matrix $W_{in} \in \mathbb{R}^{N \times D}$ mapping a D-dimensional time series data point into the N-dimensional reservoir space. The interpolation, mapping a N-dimensional reservoir state to a D dimensional time series state is done by the readout matrix $W_{out} \in \mathbb{R}^{D \times N}$. The time evolution of the ESN is then given by the discrete map

$$\vec{r}[t+1] = \tanh(M\vec{r}[t] + \sigma W_{in} W_{out} r[t]) \tag{2.10}$$

Comparing equations 2.10 with 2.9 shows that in fact, an ESN is just a discrete dynamical system. Equation 2.10 also introduces the input-strength $\sigma \in \mathbb{R}$ which scales the input signal. A common extension to ESNs is the introduction of an additional step before the readout. Such an extension can be imagined as a measurement $q(\cdot)$ of the reservoir state. Then the readout matrix W_{out} projects not the reservoir state anymore but the measured reservoir state $q(\vec{r})$. The time evolution of the reservoir needs to be adjusted accordingly to

$$\vec{r}[t+1] = \tanh(M\vec{r}[t] + \sigma W_{in} W_{out} q(\vec{r}[t])) \tag{2.11}$$

Since the exact reservoir state is known the reservoir state measurement q is not as limited as a physical measurement process would be. Instead it can be an arbitrary transformation of the reservoir state. For certain tasks such an additional readout is even required. Herteux and Räth showed that the extended Lu readout, defined by

$$q(\vec{r}[t]) = (\vec{r}_0[t], ..., \vec{r}_{N-1}[t], (\vec{r}_0[t])^2, ..., (\vec{r}_{N-1}[t])^2)^T \tag{2.12}$$

breaks a problematic mirror-symmetry in the reservoir boosting the performance in certain multifunctional tasks [6]. The complete setup of an ESN with its data-flow as used in this thesis is shown in 2.1.

2.3 Training Echo State Networks

The goal of the training is to find a readout matrix W_{out}, such that a measured reservoir state at a given time is mapped to a time series state close to the true state at this time. This is achieved by a supervised learning mechanism. A series of reservoir states is generated by driving the ESN with a known signal. As already mentioned in the introduction, a typical benchmark problem for ESNs is to predict chaotic attractors. Hence a time series consisting of points on the attractor is needed. Generating such a time series is possible by choosing a starting point in the basin of the attractor. The basin of an attractor is the set of all points which eventually converge towards the attractor. Discarding a sufficient numbers of data points from the beginning of the generated time series ensures that all the remaining points are indeed on the attractor.

The input matrix and the adjacency matrix are created before the training procedure starts. By convention, the input matrix is a sparse and real $N \times D$ random-matrix. Every row in W_{in} has only one random non-zero entry between minus and plus one. For a better computational performance the input matrix can directly be multiplied by the input-strength σ. This avoids additonal multiplications in every time evolution step during training and prediction.

The adjacency matrix is created as the adjacency matrix of a random graph with a given average degree and weigths distributed between plus and minus one. The adjacency matrix is then rescaled such that the norm of the largest eigenvalue, the so called spectral radius ρ, is equal to a number chosen a priori. Last but not least an arbitrary initial reservoir state must be created. For all simulations this will be $\vec{0}$. Since the initial state is arbitrary it must not have any influence on the predictions. Synchronizing the reservoir state with the signal, by driving the reservoir with the signal, can eliminate any dependencies between the training process and the chosen initial state. During synchronization the reservoir state is propagated by the following time evolution

$$\vec{r}[t+1] = \tanh(M\vec{r}[t] + \sigma W_{in} \vec{s}[t]) \tag{2.13}$$

An ESN is said to hold the so called Echo state property if a reservoir state generated by driving the reservoir with an infinite long time series is independent of the initially chosen reservoir state. Jaeger showed in "The echo state approach to analysing and training recurrent neural networks" that any ESN with $\rho < 1$ has the Echo state property [3]. Nonetheless the Echo state property might also hold for some ESNs with $\rho > 1$. In practice the reservoir is driven for $T_{sync} \in \mathbb{N}$ steps.[2] While there is no rule what value T_{sync} should be exactly, it should be at least greater than N. But selecting longer synchronization times is never harmful.

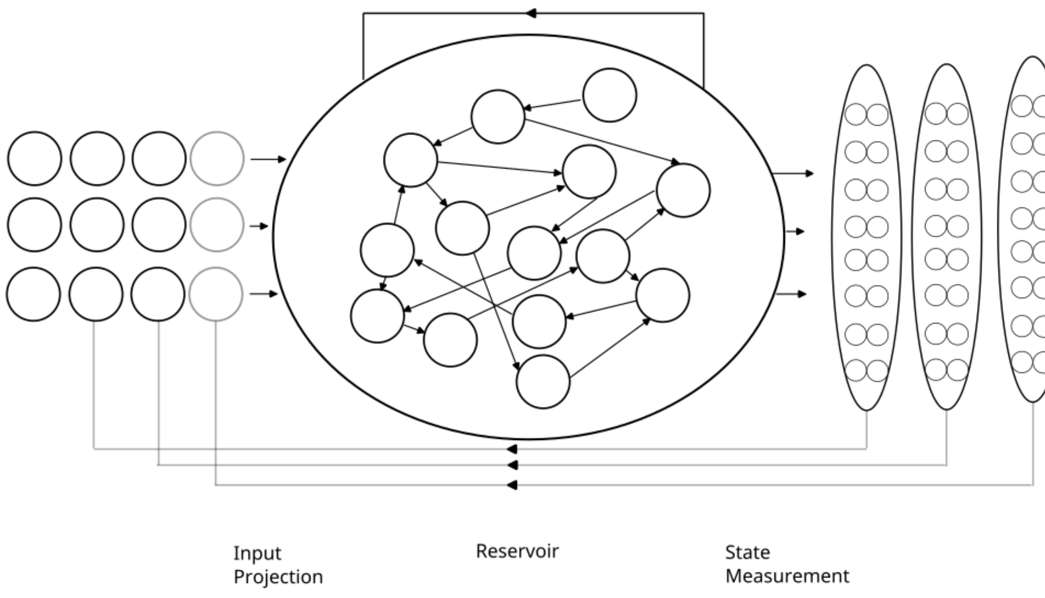

Figure 2.2: Architecture of an untrained Echo State network. The first three input-states are used to generate three measured reservoir states. W_{out} is learned by fitting the three measured reservoir states to the second, third and fourth imput states.

After synchronization the reservoir is further driven by the remaining training signal. This time, the states of the F-signal and the states of the reservoir are recorded. Without loss of generality, the index of the first input signal state during training is set to $t = 1$. Input state $\vec{s}[1]$ is then used to generate reservoir state $\vec{r}[2]$ via equation 2.13 and so on. When a total of T_{train} time series points with time indices ranging from 1 to T_{train} are provided to the training, then the first $T_{train} - 1$ data points of the time series are used to generate $T_{train} - 1$ reservoir states. From equation 2.13 it can be derived that the generated reservoir states have time indices ranging from 2 to T_{train}. W_{out} is then the result of the following optimization problem

$$W_{out} = \arg\min_{W_{out}} \sum_{t=2}^{T_{train}} \|W_{out} q(\vec{r}[t]) - \vec{s}[t]\|^2 + \beta \|W_{out}\|^2 \tag{2.14}$$

While $\vec{s}[T_{train}]$ is not used to generate a reservoir state it appears in the minimization problem. This kind of minimzation problem is called ridge regression. It differs from a standard linear least-square optimization by penalizing large values in W_{out} preventing overfitting. $\beta \in \mathbb{R}^+$ is another hyperparameter called the ridge regression parameter determining the strength of this penalty. After rearranging the recorded states into matrices $R = (\vec{r}[2], \vec{r}[3], ..., \vec{r}[T_{train}])$ and $S = (\vec{s}[2], \vec{s}[3], ..., \vec{s}[T_{train}])$, an analytical solution for equation 2.14 can be written as

$$W_{out} = SR^T(RR^T + \beta\mathbb{I})^{-1} \tag{2.15}$$

To start the prediction procedure a single element of the input time series is required. This element is given by $\vec{s}[T_{train}]$ which until now has not been used to generate a reservoir state. Afterwards it is possible to close the loop and transform the ESN into an autonomous system with the time evolution given by equation 2.11.

[2]Since the mathematical formulation of the time evolution of the ESN is discrete, training times, etc. will be given as a number of steps.

Associated with an ESN are several hyperparameters. Usually, the hyperparameters refer to the spectral-radius ρ , the input-strength σ, the ridge regression β. Sometimes the average degree of the network is also included. While usually a broad range of hyperparameters work for a specific problem, it is a common strategy to search an optimal hyperparameter configuration via a grid search [14] [6].

Despite being deterministic systems, chaotic attractors are an excellent choice as a benchmark problem for any time series prediction test. As Lorenz mentioned in his paper about non-periodic flow, a perfect prediction is only possible by perfect knowledge of the initial conditions. At first this seems counterintuitive, since the prediction phase of the ESN is started with the exactly known state $\vec{s}[T_{train}]$. But since the ESN does not know the exact equations of motion, any prediction will eventually diverge from the true trajectory.

The method by which ESNs learn is strongly connected to a phenomenon from complex systems called synchronization. Synchronization is the mirroring of one system coupled to an identical twin system. In a more general concept called generalized synchronization, these systems do not need to be identical. Instead, a driving system couples to a different response system. Depending on the coupling strength, the time evolution of the response system might collapse onto a manifold which can be mapped from the states of the driving system. This was first investigated in more detail by Rulkov, Sushchik and Tsimring [15]. For ESNs, generalized synchronization is not enough. While during training the reservoir state is a function of the signal state, the relation must be the other way round. ESNs can only make correct predictions if the signal state is a function of the reservoir state. This function is given by W_{out}. The connection between generalized synchronization and the learning process of ESNs was first mentioned by Lu, Hunt and Ott [16]. And because ESNs relay on the inverse direction of generalized synchronization this has also been named invertible generalized synchronization by Lu and Bassett [17]. Lu, Hunt and Ott list four conditions for an ESN to successfully learn an attractor:

1. During synchronization the reservoir reaches a state where its state depends only on the state of the signal $\vec{r}[t] = \phi(\vec{s}[t])$
2. ϕ is an injective function.
3. The training is successful which means $W_{out}q(\vec{r}[t]) \approx \vec{s}[t]$. This can be measured by the training error introduced in section 2.5.
4. The learned attractor is stable. This means long-term predictions as defined later in the sections 2.8 and 2.7 are possible.

The first three points refer to the training and invertible generalized synchronization. The last point referes to the prediction phase and is hard to guarantee. In simpler terms it means that the predicted trajectory visually indistinguishable from the true trajectory.

2.4 Multifunctional Echo State Networks

Originally multifunctionality is a term from neuroscience describing the ability of biological neural networks to perform more than just a single task at different times [18]. In the context of ESNs, multifunctionality describes the ability of a network to learn and predict more than just one chaotic attractor without changing any of its internal connections or its readout matrix. Multifunctionality requires minor adaptions to the training and prediction stage. For every attractor the system is first synchronized with the signal and then trained. Fitting the readout matrix is done in a single step over all attractors.

$$W_{out} = \arg\min_{W_{out}} \sum_{attractors} \sum_{i=2}^{T_{train}} \|W_{out}q(\vec{r}[t]) - \vec{s}[t]\|^2 + \beta\|W_{out}\|^2 \tag{2.16}$$

The solution is again given by 2.15 where the matrices R and S consist now of the concatenated states. Also the prediction phase of the attractors need some adjustments. Before predicting an attractor the ESN must be synchronized with the attractor.

Multifunctional ESNs were first described by Herteux and Räth in "Breaking symmetries of the reservoir equations in echo state networks" [6]. Multifunctionality requires a second attractor.

	Deviation X	Deviation Y	Deviation Z
Lorenz	5.77	7.96	6.86
Halvorsen	2.95	2.95	2.95

Table 2.1: Allowed deviations for short-time predictions. The number of steps until the distance between the predicted trajectory and the true trajectory is greater than the allowed deviation, is called the forecast horizon.

In all further simulations this will be the Halvorsen attractor

$$\frac{dx}{dt} = -\sigma x - 4y - 4z - y^2 \tag{2.17}$$

$$\frac{dy}{dt} = -\sigma y - 4z - 4x - z^2 \tag{2.18}$$

$$\frac{dz}{dt} = -\sigma z - 4x - 4y - x^2 \tag{2.19}$$

Where σ is fixed to 1.3.

2.5 Training Error

The training error is a quantity measured within the context of the training process. In this thesis the training error is defined as the sum over all absolute deviations between the reservoir states mapped by W_{out} encountered during training and their true values.

$$\sum_{attractors} \sum_{t=2}^{T_{train}} ||W_{out}q(\vec{r}[t]) - \vec{s}[t]|| \tag{2.20}$$

It should be noted that in the literature there is no standard definition for the training error. Based on the idea of having a measure that relates the states encountered during training to their true states, several measures have been proposed. For instance, Krishnagopal et al. use in "Separation of chaotic signals by reservoir computing" the square of the deviations which closely resembles the minimization condition given by equation 2.14 [19]. In "Path length statistics in reservoir computing", Carrol defines a quantity he also calls "training error" in terms of the standard deviation of the true and learned signals [20]. Dambre et al. define in "Information processing capacity of dynamical systems" the "computational capacity" of a dynamical system [21]. This quantity is closely related to the training error. The bigger the training error the smaller the computational capacity. In all three papers the training error/computational capacity is normalized with respect to the training signal. This is not necessary for the simulations done in this thesis since all simulations where the training error is measured use the exact same input signal.

2.6 Forecast Horizon

A simple tool to distinguish good from bad predictions is the forecast horizon. The true trajectory and the predicted trajectory of the ESN are compared. The number of time steps the predicted trajectory stays nearby the true trajectory is called the forecast horizon. Nearby refers to a predefined threshold. In the following chapters the choice for the threshold is set to 15% of the attractor extension in each dimension which is consistent with other literature [6]. The allowed deviations per coordinate and attractor are listed in table 2.1. If the attractors are scaled, the thresholds are scaled accordingly. In case of multiple attractors the forecast horizon is extended to the combined forecast horizon given by the geometric mean over all individual forecast horizons. Since the predicted attractors are chaotic, the forecast horizon is always finite. It is therefore a good tool to capture the accuracy of short-term predictions but cannot be used when classifying the quality of long-term predictions.

	ϵ	Minimum time distance	Initial time-offset	Ending time-offset	Standard deviation Lyapunov-Exponent
Lorenz	0.1	0.76	0.5	3.5	0.081966
Halvorsen	0.1	1.5	0.5	3.5	0.049551

Table 2.2: Parameters used for the the Rosenstein-Kantz Algorithm.

2.7 Lyapunov-Exponents and the Rosenstein-Kantz Algorithm

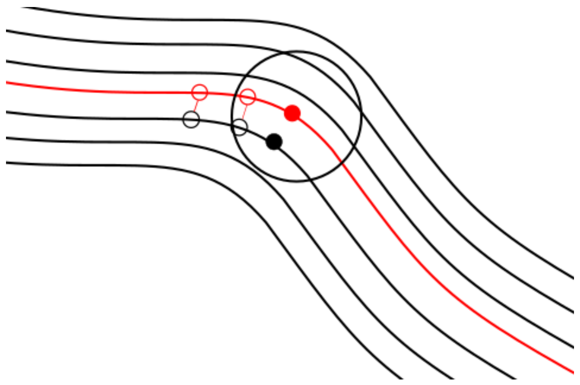

Figure 2.3: Within an ϵ neighbourhood points of nearby trajectories are selected. Then the distance between these two trajectories between an initial time-offset and an ending time-offset is tracked. The result of all measured distances is fitted to the Lyapunov-exponent.

Although chaotic attractors are deterministic, it is still hard to predict the exact time evolution. The reason is the exponentially fast divergence of arbitrary close initial conditions. This observation is mathematically captured by the largest Lyapunov-exponent λ. Given two intial conditions $\vec{s}_1[t = 0]$ and $\vec{s}_2[t = 0] = \vec{s}_1[t = 0] + \vec{\delta}$ where δ is small, the divergence between $\vec{s}_1$ and $\vec{s}_2$ is related to the largest Lyapunov-exponent by $e^{\lambda t} \propto \|\vec{\delta[t]}\| = \|\vec{s}_1[t] - \vec{s}_2[t]\|$. To measure the largest Lyapunov-exponent of a time series, the Rosenstein-Kantz Algorithm, proposed by Kantz in "A robust method to estimate the maximal Lyapunov-exponent of a time series" is used [22]. The basic schema of this algorithm is sketched in figure 2.3. For every point of the discrete time series the algorithm finds all its neighbouring points within an ϵ neighbourhood. The neighbouring data points are first filtered based on a minimum time distance criterium. The remaining points can be considered to be on neighbouring trajectories. Then the distance to every remaining trajectory is measured at an inital time-offset and an ending time-offset. Finally the exponential function $e^{\lambda t}$ is fitted against all measured distances to obtain the Lyapunov-exponent λ. The parameters required to compute the Lyapunov-exponent via the Rosenstein-Kantz Algorithm are the ϵ distance, the minimum time distance between the selected data-point and all found neighbouring data points and the inital and the ending time offsets. The used parameters can be looked up in table 2.2.

2.8 Correlation Dimension and the Grassberger-Procaccia Algorithm

Given a discrete time series. The average number of neighbours within a radius r is given by

$$C(r) = \lim_{N \to \inf} \frac{1}{N^2} \sum_{i,j=1}^{N} \Theta(r - \|s[i] - s[j]\|) \tag{2.21}$$

where Θ is the Heaviside-function. Grassberger and Procaccia showed that $C(r)$ scales with a power law $C(r) \sim r^\nu$ [23]. The exponent ν is the so called correlation dimension. To compute the correlation-dimension ν via the Grassberger-Procaccia Algorithm, for every data point of the time series its neighbours within r_{min} and r_{max} are counted. Fitting the average count for r_{min} and

	r_{min}	r_{max}	Standard deviation Correlation Dimension
Lorenz	0.39161745	4.16995761	0.016495
Halvorsen	0.14857506	1.4855419	0.033422

Table 2.3: Parameters used for the Grassberger-Procaccia Algorithm.

r_{max} against r^{ν} results in an estimate for ν. If the attractors are scaled r_{min} and r_{max} are scaled accordingly. Both the Rosenstein-Kantz Algorithm and the Grassberger-Procaccia Algorithm relay on nearest neighbour queries. Fast nearest neighbour queries can be efficiently implemented via kd-trees. Sharing the same kd-tree between both algorithms leads to an additional speed-up.

The climate of an attractor consists of all of its statistical properties. Together with the Lyapunov-exponent the correlation dimension is used to test if an ESN is able to reconstruct the climate of an attractor. A predicted trajectory is said to reconstruct the climate of an attractor if its correlation dimension and its Lyapunov-exponent are within five standard deviations of the true trajectory. With this method, successful and unsuccessful long-term predictions can be identified. The standard deviations of the correlation dimension and the parameters for Grassberger-Procaccia Algorithm are listed in table 2.3. The standard deviation of the Lyapunov-exponent is listed in table 2.2. The standard deviations have been computed by running the Rosenstein-Kantz Algorithm and the Grassberger-Procaccia Algorithm on 50 different trajectories.

2.9 Attractor Separation

Multiple attractors can coexist in the state space of the ESN only when they are separated, otherwise it is guaranteed the prediction will fail. This is a consequence of the fact that different discrete time series can not share any of their data points and the linear readout interpolating between the reservoir states.[3] If two reservoir states $\vec{r}_1$ and $\vec{r}_2$ belong to one attractor meaning they are mapped by W_{out} to the attractor points $\vec{s}_1$ and $\vec{s}_2$, then every and reservoir state between these two states can be represented by a linear combination of these two states:

$$\vec{\tilde{r}} = \xi\vec{r}_2 + (1-\xi)\vec{r}_1 \tag{2.22}$$

where $\xi \in [0,1]$. $\vec{\tilde{r}}$ is then mapped by W_{out} to a point between $\vec{s}_1$ and $\vec{s}_2$:

$$W_{out}\vec{\tilde{r}} = \xi W_{out}\vec{r}_2 + (1-\xi)W_{out}\vec{r}_1 = \xi\vec{s}_2 + (1-\xi)\vec{s}_1 \tag{2.23}$$

Despite training with a discrete set of points, the attractors embedded in the high-dimensional reservoir state space occupy a continuous region of the space. In fact it turns out that the complete embedding of the attractor is topologically equivalent to the attractor [24]. Hence it is not enough for two attractors not to share any reservoir states, but additionally the regions they occupy in the reservoir state space must be separated. When the extended Lu reservoir state measurement is included in the considerations, it follows that the measured reservoir states of the attractors must be speparated. But this is only possible when the pure reservoir states are already separated. The minimum distance between the reserovoir states of different attractors encountered during training is called the minimal embedded attractor distance. It is measured for two attractors by the following formula:

$$\min_{n_1,n_2} \|\vec{r}[n_1] - \vec{r}[n_2]\| \tag{2.24}$$

One problem when computing the minimal embedded attractor distance is the high dimensionality of the data. While kd-trees usually help to speed up distance computations, they fail to achieve relevant speed-ups in high dimensions. The only remaining option is to use a brute-force approach comparing every point of one attractor with every point of the second attractor. The time required to measure the attractor separation scales with the square of the available data points. This is not feasible for more than a few thousand data points. In scenarios where many more data points have been generated, the sampled attractor separation is used considering only every l-th point.

$$\min_{\substack{n_1=1,l,2l,\ldots\\ n_2=1,l,2l,\ldots}} \|\vec{r}[n_1] - \vec{r}[n_2]\| \tag{2.25}$$

[3]For the moment any modifications to the readout, e.g. extended Lu-readout are ignored.

2.10 Implementation and Software

The computer simulations done in this thesis use the rescomp package[4]. A lot of functionality has been added on top of this package. Among many other things this includes code for multifunctional training and memory measurements of ESNs. The full modified code is part of the data storage which is included with this thesis. This data storage also includes all code used in specific simulations and the corresponding results. One exception are the simulations in chapter 3. The data analyzed in this chapter is simply to big to be included. The total amount of bits is the number of ESNs (200) times the number of attractors (2) times the number of nodes (500) times the number of reservoir states (9999) times the size of a double value (64 bit). This results in a total of ≈ 15 giga-bytes and is repeated for three different topologies. Unfortunatly this exceeds the capacity of the data storage by far.

[4]https://github.com/GLSRC/rescomp

Chapter 3

Network Topologies and Origins of Multifunctionality

While there is no consensus among researchers on how the human brain works, it is clear that any model of the human brain must explain its capability to perform complex computations. One possibility is to model the human brain as a Turing Machine, a concept well known from computer science [25]. Another approach models the human brain as a dynamical system, where computations emerge from the collective dynamics of its neurons [26]. Following the second approach there is a close connection between biological neural-circuits and ESNs. The model linking these two fields together is given by Liquid State Machines (LSM). Just like ESNs, LSMs also use a network as a reservoir but nodes are represented by integrate-and-fire neurons [2]. Similar to ESNs, the neural network of LSMs is not trained. Instead a single layer of readout neurons extracts the required information. LSMs have been proposed to extend our understanding of computations of biological neural circuits. Later ESNs and LSMs where unified under the umbrella term of "reservoir computing" [4].

LSMs are not the only connection between neuroscience and ESNs. Another aspect is the topology of a network. This is a global property determined by the tail of its degree distribution. There are three common topolgies: The random-network, the scale-free network and the small-world network [27]. The degree distribution is the probability distribution of a network to find a node with a given degree. For random-networks the tail of the degree distribution decreases exponentially while for scale-free networks the tail of the degree distribution decreases with a power-law. Small-world networks are constructed from a ring topology where every node is connected to its k-nearest neighbours. Then, every link is reconnected randomly with a given rewire probability. Depending on the rewire probability, the network is either completely orderd or similar to a random network. Usually, ESNs are constructed using random networks although experiments with other network topologies have been made [28]. The topology of biological neural circuits like the human brain is not so clear. The only aspect where researchers seem to agree is that biological neural circuits are not random graphs. A study from van den Heuvel et al. published in 2008 concludes that a power-law matches the degree distribution of the functionally connected human brain at voxel scale. But they also found a high clustering coefficient typical for small-world networks [29]. Another study published in 2016 by Singh et. al looked at the topological properties of different kind of species. Again they argue that a power-law distribution explains the modular and inter-modular connectivity. But they also found that the topology of higher evolved species is more ordered [30]. Again this would also match with the topology of small-world networks.

There is one more recently discovered aspect connecting neuro science and ESNs. This time the connection is given by the dynamic capabilities of ESNs and biological neural-networks. ESNs have been shown to be capable to switch between different tasks without changing any of their internal connections or output weights. This capability was first published by Herteux and Räth [6] and independently shortly after by Flynn, Tsachouridis and Amann [7]. It were also Flynn, Tsachouridis and Amann who gave this capability the name multifunctionality. Originally, mutli-functionality is a term from neuroscience, first described by Kristan et al. [18]. They observed the neural activity of leeches during shortening, crawling and and swimming. Each of these movements requires a "coordinated movement of the 21 essentially identical midbody segments"[18]. The muscles involved in these movements are controlled by the same set of neurons. When observing the

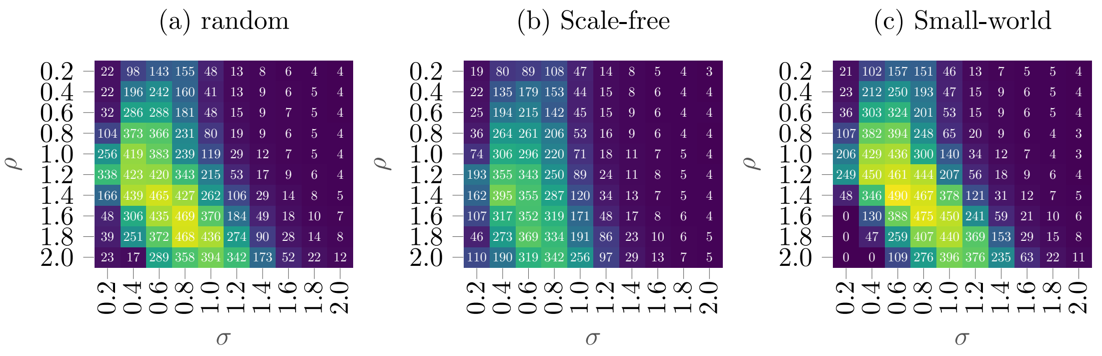

Figure 3.1: Averaged combined forecast horizon with different hyperparameter configurations for ensembles of 20 simplified ESNs

activity of a single neuron, Kristan et. al found different forms of activity patterns during different kinds of movements. Multifunctionality in ESNs and multifunctionality in biological neural circuits have in common, that the same set of nodes or neurons with fixed connections are responsible for different kinds of tasks. But Kristan et al. study does also show the limitations of these analogy. Biological neural-circuits have a hierachical order. The neuron observed by Kristan et. al is a so called "Gating-neuron". Such a neuron is without any equivalence in an ESN. Later Briggman and Kristan published a more detaild investigagtion of multifunctionality in biological neural circuits [8]. They found four different kinds of multifunctionl neural circuits, classified by the overlap of the active neurons during each of the activities. Again they looked at the activity of a single neuron and found different kinds of activity patterns for different kinds of tasks.

3.1 Simplified Echo State Networks

While the network topology is an important feature of a graph, given that the amount of independent parameters in an ESN scales with $\mathcal{O}(N^2)$, a more restricted model is proposed. This limits the degrees of freedom and shifts the focus to the topology of the networks. These kind of models are called simplified ESNs. The first simplification is about the input matrix. All simplified ESNs of the same size share the exact same input matrix. Every row in this simplified input matrix has exactly one randomly chosen value set to one. The next simplification is about the adjacency matrix. The adjacency matrix of a simplified ESN is a symmetric matrix where every entry is either zero or has the same absolute value. A simplified adjacency matrix $\tilde{M}$ can be created from the adjacency matrix of a regular ESN by iterating every entry in the upper triangle of the regular adjacency matrix M, including the diagonal. Whenever $M_{i<=j}$ is not zero, $\tilde{M}_{ij}$ and $\tilde{M}_{ji}$ are set randomly to either plus or minus one. After initialization, $\tilde{M}$ is scaled to the chosen spectral radius. This way it is guaranteed that all topological properties of M are shared by $\tilde{M}$. All simulations in this chapter are done with simplified ESNs.

3.2 Setup and Simulations

The following simulations use a setup similar to the one introduced by Herteux and Räth in "Breaking symmetries of the reservoir equations in echo state networks" [6]. One of the attractors is the Lorenz attractor while the other attractor is the Halvorsen attractor. The ESNs follow the discrete time evolution of 2.11 where the reservoir states are measured by the extended Lu state measurement. The mean of the Lorenz attractor is placed at $(1, 1, 1)^T$ and the mean of the Halvorsen attractor is placed at $(-1, -1, -1)^T$. Both attractors are rescaled such that no point of the time series used during training has a distance greater than one from the mean point of the attractor. The ESNs consist of 500 nodes with an average degree of six and the ridge regression parameter is set to $\beta = 10^{-7}$. In case of the small-world network, the rewire probability is set ot 0.2. For training, 2000 synchronization steps and 10,000 train steps are used. In the prediction phase, switching between the attractors is accomplished by another 2000 synchronization steps. Finally, for every attractor 200,000 time steps are predicted.

In a first step, a grid-search over the spectral-radius and the input strength is performed. For every parameter configuration, an ensemble of 20 different simplified ESNs is trained and evaluated based on the combined forecast horizon.

Figure 3.1 shows the result by visualizing the ensemble averaged combined forecast horizon as a heatmap. For the random network, the best parameter configuration is at $\rho = 1.6$ and $\sigma = 0.8$, for a scale-free network, the best configuration is at $\rho = 1.4$ and $\sigma = 0.4$ and for a small-world network, the best configuration is at $\rho = 1.4$ and $\sigma = 0.6$. The best random and small-world networks perform similar good with an ensemble averaged combined forecast horizon of 469 and 490 respectively. The best configuration of scale-free networks falls behind with an ensemble averaged combined forecast horizon of 395. For all three network topologies, the parameter range of networks with a triple digit combined forecast horizon, is between an input strength of 0.4 and 0.8 as long as the spectral radius is smaller than one. When the spectral radius is greater than one, there is a difference between random and small-world networks in one group and scale-free networks in the other group. For random and small-world networks, the input strength of ESNs with triple digit combined forecast horizons is shifted to the right. For instance when $\rho = 2.0$, ensembles with triple digit combined forecast horizon have input strengths between 0.6 and 1.4. Scale-free networks behave different when the spectral radius is above 1.0. The range of input strengths for ensembles with triple digit combined forecast horizon becomes wider and ranges from 0.2 to 1.0 for $\rho = 2.0$.

In a second step, the 200 ESNs with the best parameter configuration are simulated for each topology type. This time, not just the combined forecast horizon is considered but also the climate reconstruction capabilities. This is done via the correlation dimension and the largest Lyapunov exponent. Both quantities are computed on the last 10,000 time steps of the 200,000 step forecast. As explained in section 2.8 in more detail, an ESN is said to successfully reconstruct the climate of an attractor if the computed climate measures of the predicted trajectory are within five standard deviations of the true values. Table 3.1 lists the most important statistics from the simulations.

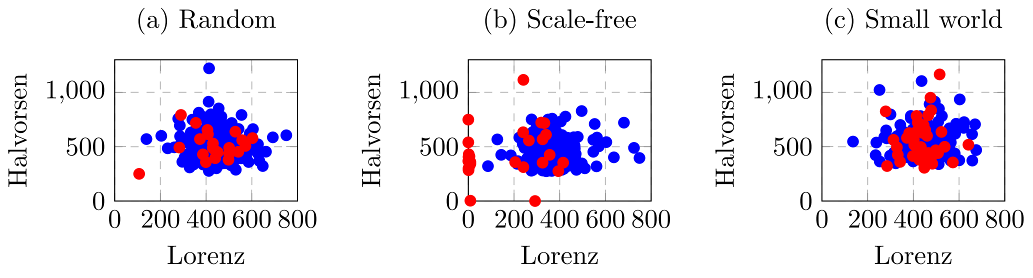

Figure 3.2: Climate reconstruction capabilities of individual simplified ESNs plotted in relation to the forecast horizon. Blue dots represent ESNs which reconstructed the climate of both attractors successfully. Red dots represent ESNs which failed to reconstruct the climate of at least one attractor.

Looking at the column with the combined forecast horizon, the results from the optimization procedure are confirmed. Small-world networks perform best with an average combined forecast horizon of about 484 timesteps. Slightly behind are random networks with an average combined forecast horizon of 475 timesteps. Again scale-free networks fall behind with an average combined forecast horizon of 376 timesteps. When filtering out all the networks, which failed to reconstruct at least one attractor, the average combined forecast horizons of random and small-world networks does not improve. Only for scale-free networks, the average combined forecast horizon improves to about 400 timesteps but is still behind random and small-world networks.

When it comes to attractor reconstructions, small-world networks are the worst network type with less than 80% of the networks being able to reconstruct both attractors. For random and scale-free networks nearly 90% of the networks achieve attractor climate reconstructions. A more detailed analysis is possible when looking at the differences between individual networks. Figure 3.2 plots the forecast horizon of the Lorenz attractor on the x-axis and the forecast horizon of the Halvorsen attractor on the y-axis. Blue dots represent ESNs which achieved climate reconstructions of both attractors while red dots represent ESNs which failed climate reconstruction of at least one attractor. Again random and small-world networks show similar behaviour where failing ESNs are usually at the same spots as successful ESNs. For scale-free networks, a large number of

Topology	Successful climate reconstructions	Average combined forecast horizon	Average combined forecast horizon of successful climate reconstructions
Random	178 (89%)	475.20	476.65
Scale-free	174 (87%)	376.12	401.17
Small-world	156 (78%)	484.04	484.57

Table 3.1: Results from the simualtions with the optimal input-strength and spectral-radius. 200 ESNs are simulated for each topology. Small-world networks perform best when it comes to short-term predictions but fail more often than the other topologies when evaluated on climate reconstruction capabilities. Scale-free networks are the only topology where the average combined forecast horizon improves when only ESNs which succeeded the climate reconstruction are considered.

networks which failed attractor reconstruction also failed to predict more than a few timesteps of the Lorenz attractor. This explains why the average combined forecast horizon improves for scale-free networks, when only ESNs with successful climate reconstruction are considered.

3.3 Activity Patterns and Degree Based Evaluations

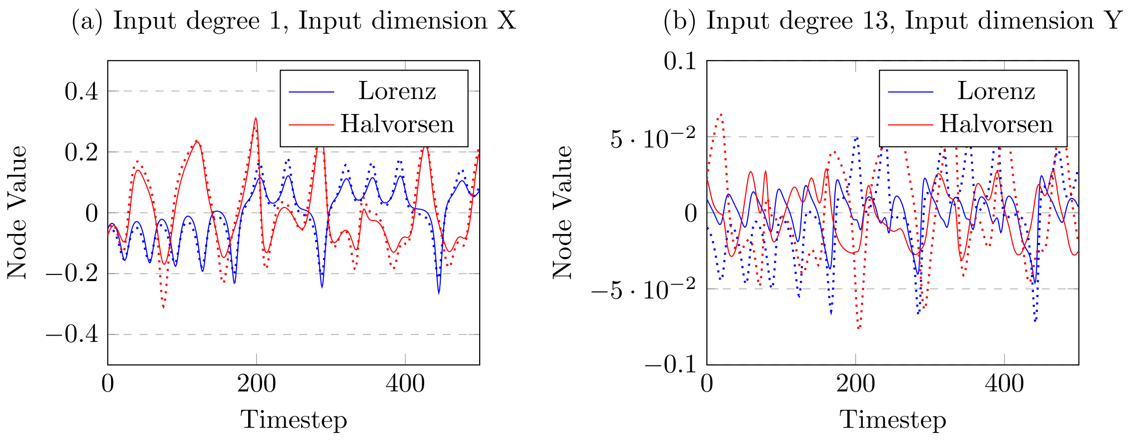

Figure 3.3: The value of nodes with different input degrees at a given time together with its input signal (dotted). All signals have been shifted such that their mean is zero. Additionally the input signals have been scaled by a factor of 0.4 in case of figure (a) and by a factor 0.1 in case of figure (b). The node in figure (a) has an input degree of one and is coupled by W_{in} to the X dimension of the attractors. The node shown in figure (b) has an input degree of 13 and is coupled to the Y dimension of the attractors. Both nodes are from the same random network.

Kristan et al. found the origin of multifunctionality in different kind of activity patterns of a gating neuron in the neural circuit of leeches. While ESNs do not have the same hierachical order as biological neural networks, it is still possible to look at the activity of a certain node by drawing its values over time. Figure 3.3 shows the values of one node with a low input degree and one node with a high input degree in a random network during training. The dotted lines are the values of the input signal of the node. In figure 3.3 a), the input signal is given by the X component and in figure 3.3 b) by the Y component of the Lorenz- and the Halvorsen-attractor. It should be noted that at least for the case of a successful attractor reconstruction, the input signal during training is statistically indistinguishable from the input signal during prediction. Using the reservoir states encountered during training is therefore not a flaw in the methodology. For better visualization purposes, the signals in figure 3.3 have been shifted to a mean of zero and additionally the input signals have been rescaled such that possible overlap is easier to recognize. Similar to the work of Kristan et al. and Briggmann and Kristan, visual inspection reveals different kind of activity patterns when learning different attractors. In case of figure 3.3 a), the blue and the red line peak at different positions with a different frequency and amplitude. It can be concluded that the

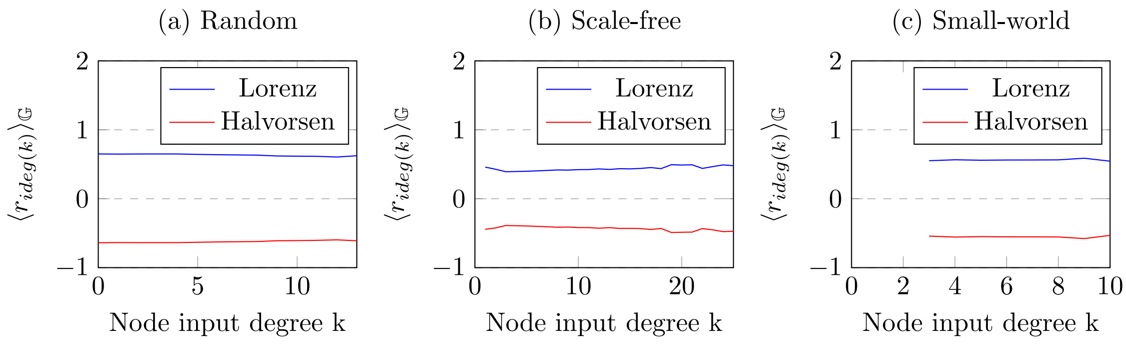

Figure 3.4: The average value of a node in the ensemble with respect to its input degree.

activity of the node changes when learning or predicting different attractors. The reason for the change in activity is also visible from the plot: Since the input signal has a strong influence on the values of the node, the node values are strongly correlated to the input values.

In case of figure 3.3 b), the blue and the red line also peak at different times. But while the frequencies and amplitudes of the peaks are still not the same they are now more similar. Comparing each node signal (solid line) with its associated input signal (dotted line) reveals a loss of influence of the input signal. Compared to the node with a low input degree, the correlation between the input signal and the node-signal is not so strong anymore. Due to the high input degree, the input signal does not dominate the node values anymore.

While visual inspection is a good start to get a first feeling of how things could be, it is not a suitable method to analyze large amounts of data. The next step is to develop a method which gives statistically significant results. The goal of this method is to reveal the impact of nodes with different input degrees. To develop a better understanding of the proposed method, it is first applied to the average value and the standard deviation of the nodes. So let $\mathbb{G}$ be the ensemble of 200 ESNs and label the input degree of a node $n \in G \in \mathbb{G}$ with $ideg(n)$. Then the total number of nodes with an input degree of k is given by:

$$N_k^{\mathbb{G}} = \sum_{G \in \mathbb{G}} \sum_{\substack{n \in G \\ ideg(n)=k}} 1 \tag{3.1}$$

Now the averages over all ESNs with respect to the input degree can be computed. First this is shown for the average node value. The time average of node n is given by $\mathbb{E}(r_n) = \frac{1}{T_{train}-1} \sum_2^{T_{train}} r_n[t]$. The time averaged value of all nodes in the ensemble $\mathbb{G}$ with input degree k is then given by:

$$\langle r_{ideg(k)} \rangle_{\mathbb{G}} = \frac{1}{N_k^{\mathbb{G}}} \sum_{G \in \mathbb{G}} \sum_{\substack{n \in G \\ ideg(n)=k}} \mathbb{E}(r_n) \tag{3.2}$$

One problem arises for nodes with high input degrees. Since the probability of them to occur is low, there are not enough nodes to get meaningful averages. Therefore for each topology all nodes above a certain input degree are grouped together. For random networks this threshold is set to 13, for scale-free networks to 25 and for small-world networks to ten. Figure 3.4 shows that for all three topologies, the average value of nodes with different degrees stay approximatly constant. Since the Lorenz attractor is located in a region where the x-,y-, and z-axis are positive the average value is positive and vice versa for the Halvorsen attractor.

Similar to the average values it is possible to compute the standard deviation of every node during training and average it over all nodes with the same input degree in the ensemble. The standard deviation of a node n is given by $\sigma(r_n) = \frac{1}{T_{train}-1} \sum_2^{T_{train}} (\mathbb{E}(r_n(t)) - r_n[t])^2$. The ensemble averaged standard deviation with respect to the input degree is then given by

$$\langle \sigma_{ideg(k)} \rangle_{\mathbb{G}} = \frac{1}{N_k^{\mathbb{G}}} \sum_{G \in \mathbb{G}} \sum_{\substack{n \in G \\ ideg(n)=k}} \sigma(r_n) \tag{3.3}$$

The result is shown in figure 3.5. Also the ensemble averaged standard deviation is basically constant for nodes with different input degrees.

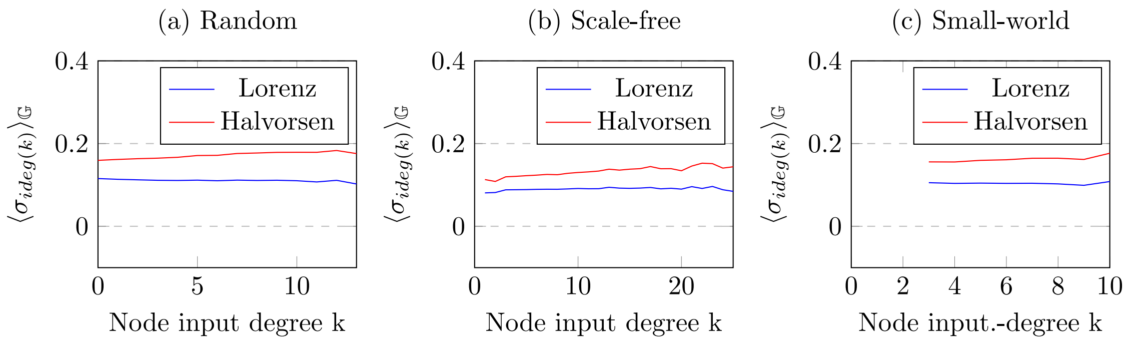

Figure 3.5: The average standard deviation of a node in the ensemble with respect to its input degree.

3.4 The Influence of Nodes on the Outcome

An ESN consists of four different components. There is the input projection matrix W_{in}, the reservoir given by the network, the state measurement which is given by the extended Lu readout and the readout matrix W_{out}. Then there are sevaral hyperparameters associated with each component and the data used to train the ESN. Most of these components are independent of each other. For instance the only restriction imposed by the reservoir on the input matrix is, that the output dimension of W_{in} must match the dimension of the network. But the entries in W_{in} and the entries in the adjacency matrix are independent of each other. W_{out} takes a special role in an ESN because it is the only component which depends either directly or indirectly on every other component, its associated hyperparameters and the training data. This makes the W_{out} the ideal candidate for further analysis.

A reservoir state of an ESN with N nodes $\vec{r} = (r_1, r_2, ..., r_N)^T$ is transformed by the extended Lu state measurement to $(\vec{r}, \vec{r}^2)^T = (r_1, ..., r_N, r_1^2, ..., r_N^2)^T$. The entries of W_{out} determine the influence a node or an extended Lu measured node have on the prediction. A higher absolute value indicates a stronger influence of the associated node. The influence τ a node n has on the prediction is then given by the sum of the absolute values in W_{out} associated with either the node directly or its extended Lu transformed node.

$$\tau_n = \sum_j \|W_{out,(j,n)}\| + \|W_{out,(j,n+N)}\| \tag{3.4}$$

One aspect which is not yet considered in these kind of analysis, is that a node with a large influence τ but values close to zero, still does not have a strong influence on the prediction. But taking previous results into account which showed that on average nodes with different degrees have similar average values, means that averaging the influence of nodes, over an ensemble of ESNs, compensates such effects.

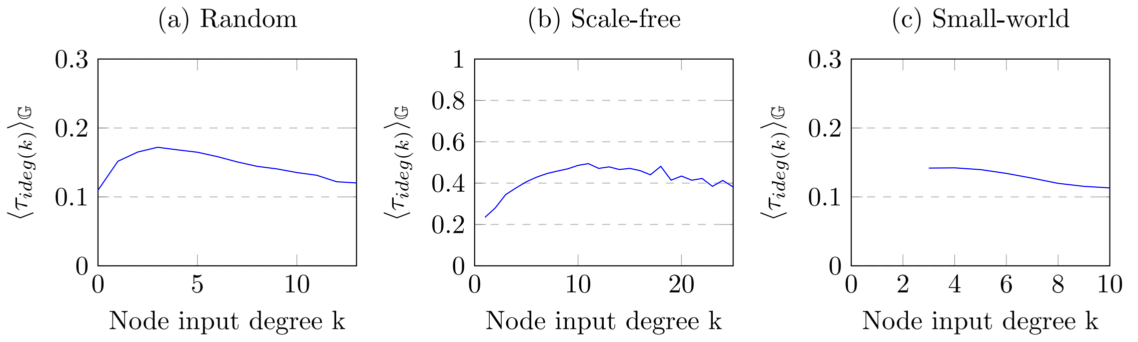

Figure 3.6: Node influence τ on the outcome.

The ensemble averaged influence of a node n with input degree k on the prediction is then given by:

$$\langle \tau_{ideg(k)} \rangle_{\mathbb{G}} = \frac{1}{N_k^{\mathbb{G}}} \sum_{G \in \mathbb{G}} \sum_{\substack{n \in G \\ ideg(n)=k}} \tau_n \tag{3.5}$$

The result can be seen in figure 3.6. In case of the random topology, nodes with low and high input degree have on average the least influence on the outcome. The average influence peaks for input degrees around four and decreases monotonously for higher input degrees.

A similar curve can be seen for scale-free networks. Nodes with low input degrees have the least influence on the outcome. The influence peaks around an input degree of ten and decreases for nodes with higher input degrees. Still, nodes with high input degrees have on average a stronger influences than nodes with low input degrees. One noteable difference is the magnitude of the values. The values of the influence for scale-free networks is more than twice as high as the influences of random or small-world networks.

The curve of the ensemble averaged influence of small-world networks looks different. Here the nodes with the lowest input degree have the strongest influence on the outcome and nodes with high input degrees have the weakest influence.

Depending on their input degree, nodes have different influences on the outcome. But is this information enough to distinguish between ESNs with good prediction performance and ESNs with bad prediction performance? Before answering this question, a thought experiment is proposed: Two different ESNs are trained on the same dataset. The entries in W_{out} of the first ESN are approximatly all of the same absolute value while the entries in W_{out} of the second ESN are distributed over a broad range. While there are without doubts counterexamples, intuitivly the first ESN should perform better during predictions. It distributes the influence of nodes evenly utilizing the full potential of its high dimensional reservoir. The second ESN gives a few nodes a strong influence on the outcome while other nodes have essentially no direct influence on the outcome. The second ESN utilizes its high dimensional reservoir less efficiently.

To measure this effect it is proposed to use the variance of the absolute entries of the readout matrix W_{out}, labeled with $Var(|W_{out}|)$. The results can be seen in figure 3.7. Comparing all three topologies shows that the variance of the absoluet readout matrix of random and small-world networks is significantly lower than the variance of the absolute readout matrix of scale-free networks. But for each topology the relation between the successful networks and $Var(|W_{out}|)$ is weak at best. In case of scale-free networks, most networks which fail attractor reconstruction are in the upper half of figure 3.7 b). In case of small-world networks, the ESNs with the highest combined forecast horizon, are in the lower half of figure 3.7 c).

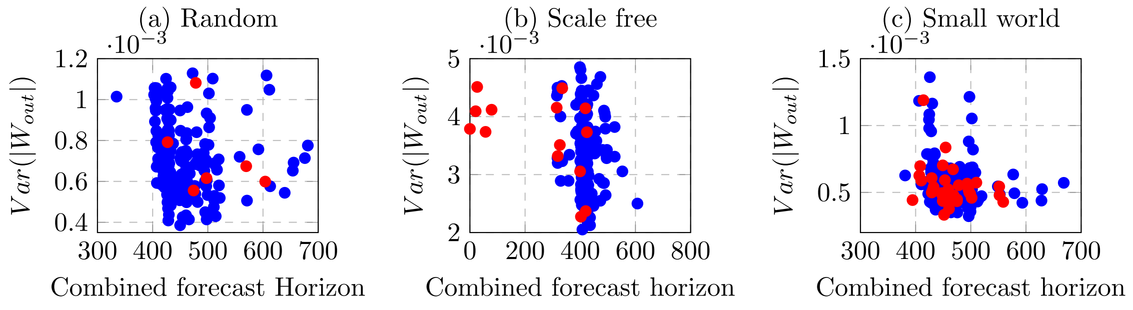

Figure 3.7: The variance of the absolute values of the readout matrix W_{out} with the forecast horizon. Red dots indicate ESNs which failed to reconstruct the climate of both attractors while blue dots indicate ESNs which successfully reconstructed the climate of both attractors.

3.5 Correlation and Mutual Information between Nodes and Input Signals

A possible explanation for the different influences of nodes with different input degrees might be given by different amounts of information nodes have stored about the current state. To test this hypothesis the correlation between nodes and input signals is measured. Since for simplified ESNs, the input projection matrix has only one equally strong non-zero entry per row, the input signal of

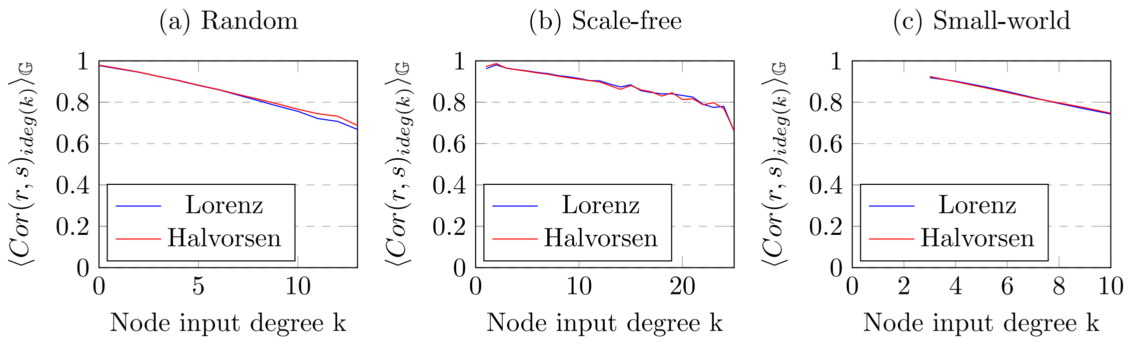

Figure 3.8: Correlation between node values and the input signals of the nodes depending on the input degree.

each node is given by either the X, Y or Z component of the attractor. The correlation coefficient between node n and its input signal s_n is given by

$$Cor(r_n, s_n) = \frac{\mathbb{E}((r_n - \mathbb{E}(r_n))(s_n - \mathbb{E}(s_n)))}{\sigma(r_n) \cdot \sigma(s_n)} \tag{3.6}$$

where $\sigma(\cdot)$ is the standard deviation of the signal. The correlation between two signals is one if changes in one signal lead to proportional changes in the other signal. The ensemble averaged correlation between nodes and their input signal with respect to the input degree is then given by:

$$\langle Cor(r,s)_{ideg(k)}\rangle_{\mathbb{G}} = \frac{1}{N_k^{\mathbb{G}}} \sum_{G\in\mathbb{G}} \sum_{\substack{n\in G \\ ideg(n)=k}} Cor(r_n, s_n) \tag{3.7}$$

The result can be seen in figure 3.8. For all three topologies, nodes with the lowest input degree have the highest correlation with their input signal. The correlation decreases linearly with higher input degress and is independent of the input signal.

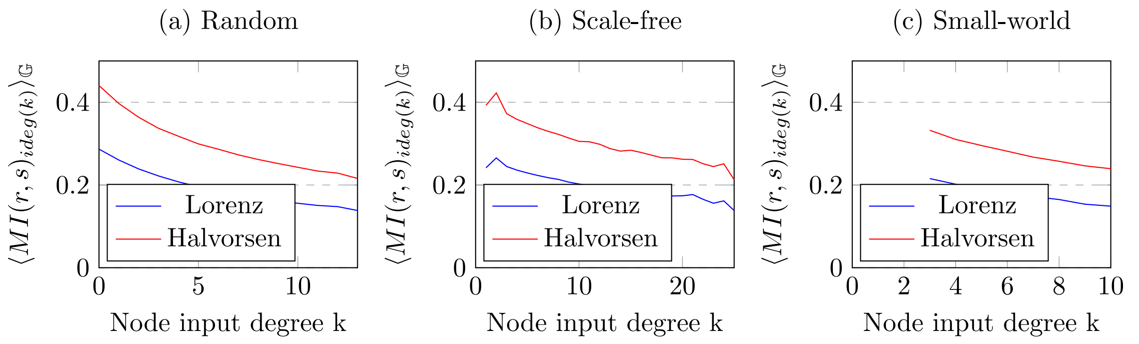

Figure 3.9: Mutual Information between node values and the input signals of the nodes depending on the input degree.

Correlation is a linear measure and can only be used to identify linear relations. A measure which identifies also non-linear dependencies is the mutual information. The mutual information between the node values r_n and its input signal s_n is defined on the (joined) probability distribution:

$$MI(r_n, s_n) = \sum_{r_n, s_n} p(r_n, s_n) \ln(\frac{p(r_n, s_n)}{p(r_n)p(s_n)}) \tag{3.8}$$

For better comparsions, the mutual information is normalized to values between zero and one by dividing with the entropy of the signal.

$$MI(r_n, s_n) = \frac{\sum_{r_n,s_n} p(r_n, s_n) \ln(\frac{p(r_n,s_n)}{p(r_n)p(s_n)})}{\sum_{s_n} -p(s_n)\ln(p(s_n))} \tag{3.9}$$

To compute the mutual information and the entropy, the signals are binned into histograms. From the histograms the probability densities can be derived. For the histograms, a fixed number of bins $\sqrt{\frac{\#\text{data-points}}{4}}$ is chosen. While the binning strategy of the histograms might have an influence on the resulting value, it does not matter when only relative values are compared. The ensemble averaged mutual information for a node with an input degree k is given by:

$$\langle MI(r,s)_{ideg(k)}\rangle_{\mathbb{G}} = \frac{1}{N_k^{\mathbb{G}}}\sum_{G\in\mathbb{G}}\sum_{\substack{n\in G\\ ideg(n)=k}} MI(r_n,s_n) \tag{3.10}$$

The results are presented in figure 3.9. Similar to the previous plots, nodes with the lowest input degree share the most information with their input signal. For all three topologies, nodes hold more information about the Halvorsen attractor than about the Lorenz attractor. A possible effect can be seen in figures 3.2 a) to c). More often than not, the forecast horizon of the Halvorsen attractor is higher than the forecast horizon of the Lorenz attractor.

3.6 Principal Component Nodes

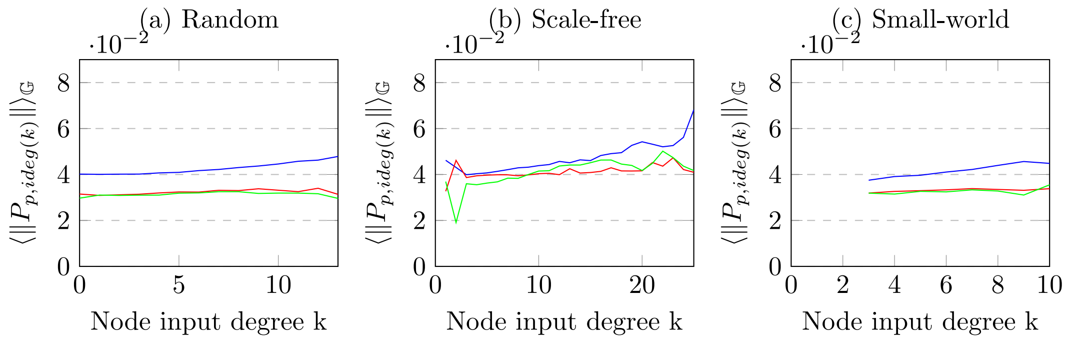

Figure 3.10: The average weighting of a node with input degree k in the principal component matrix of the reservoir states encountered during training. The first principal component is is shown in blue, the second principal component in red and the third principal component in green.

Dimensionality reduction is an importent tool when analyzing time series data. The goal is to find a low dimensional time series which shares as many relevant features with the high dimensional time series as possible. The most popular tool for time series dimensionality reduction is the principle component analysis (PCA). In this section, PCA is applied to reservoir states encountered during training. The PCA corresponds to an eigendecomposition of the covariance matrix. The steps to compute the PCA of a time series are the following:

1. Rearrange the recorded states into a matrix $R \in \mathbb{R}^{N\times 2\cdot(T_{train}-1)}$.
2. From every row subtract its average.
3. Compute the covariance matrix given by RR^T.
4. Compute the eigenvectors and eigenvalues of the covariance matrix.

The eigenvector belonging to the largest eigenvalue is called the first principle component, the eigenvector belonging to the second largest eigenvalue is called the second principle component and so on. The principle component vectors are interpreted as linear combinations of nodes. The linear combination of nodes, corresponding to the the first principle component has the maximum variance among all possible normalized linear combinations of nodes and its variance is given by its associated eigenvalue. Similar the second principle component is the linear combination with the maximum variance among all normalized linear combination which are orthogonal to the first principle component.

Usually, the principal components are used to construct a low-dimensional time series which captures as much variance of the high-dimensional time series as possible. Here, instead of reducing the dimensions, the principal components are analyzed in a similiar way as the readout matrix W_{out} when the influence on the outcome is determined. The PCA-matrix P consists of the principal components row by row. The value assigned to node n for the $p-th$ principal component is then given by P_{pn}. The contribution of a node to the p-th principal component is the absolute value and averaging over the whole ensemble with respect to the input degree results in a formula similar to equation 3.5:

$$\langle \|P_{p,ideg(k)}\| \rangle_{\mathbb{G}} = \frac{1}{N_k^{\mathbb{G}}} \sum_{G\in\mathbb{G}} \sum_{\substack{n\in G \\ ideg(n)=k}} \|P_{pn}\| \tag{3.11}$$

The result for the first three principle components can be seen in figure 3.10. The contribution of the nodes to the first principal component is plotted in blue, for the second principle component in red and for the third principel component in green.

For all three topologies the contribution of nodes to the first principle component increases with the input degree of the nodes. This means that there are nodes with high input degrees that capture more variance of the dataset than nodes with lower input degrees. At first, this might look like a contradiction to the results shown in figure 3.5. But the existance of nodes which have a higher variance does not contradict the fact that on average, the standard deviation of nodes with different input degrees is the constant. The ensemble averaged contribution of the nodes to the second principal component is almost constant for all three topologies. This is also true for the third principal except for scale-free networks, where the contribution to third principal component also increases with the node input degree.

3.7 Examples of Typical Networks

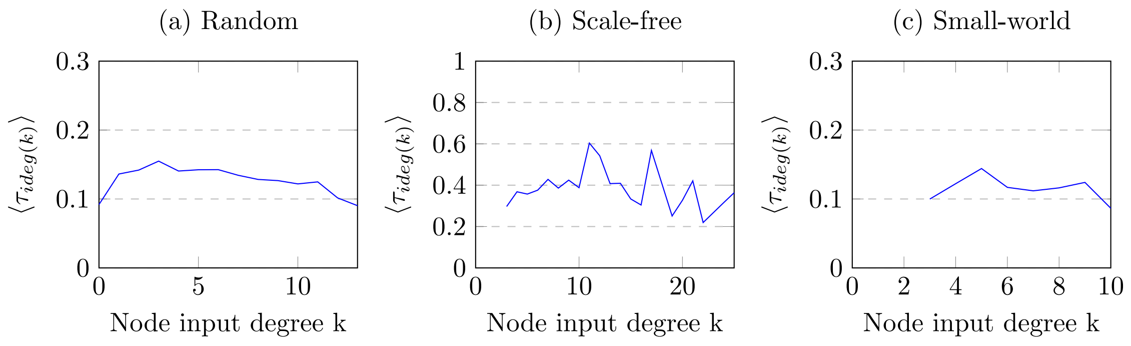

Figure 3.11: Node influence τ on the outcome.

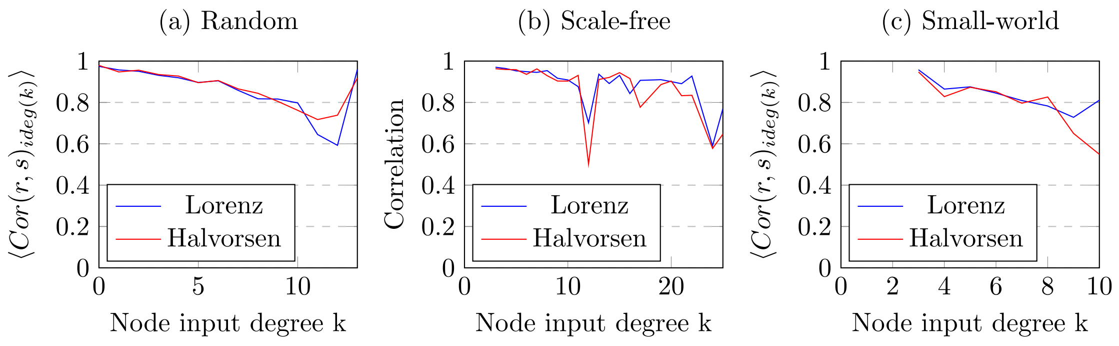

Figure 3.12: Correlation between node values and the input signals of the nodes depending on the input degree.

The previous sections averaged over 200 ESNs. In this section three specific ESNs, one for each topology are considered, to test if the results obtained previously also hold for individual ESNs. All three ESNs reconstructed the climate of both attractors successfully. The ESNs considered in

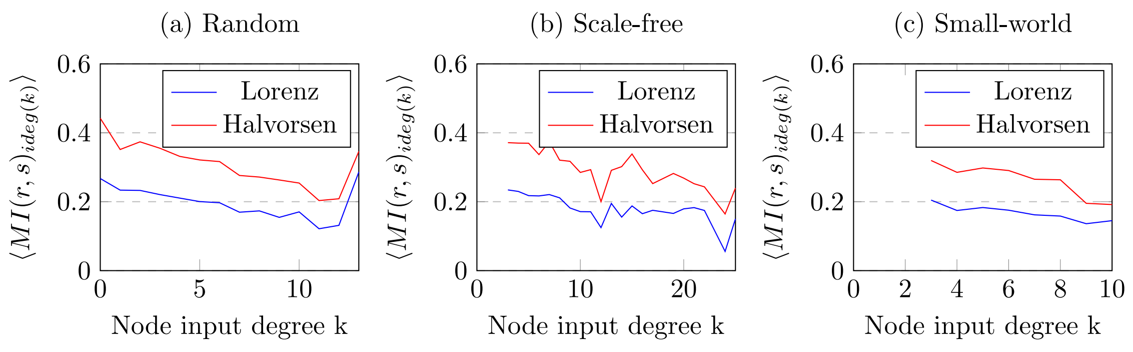

Figure 3.13: Mutual Information between node values and the input signals of the nodes depending on the input degree.

this section are called typical since the influence of their nodes, averaged over the input degree as shown in figure 3.11, is similar to ensemble averaged influence shown in figure 3.6. The influence of the selected random network, shown in figure 3.11 a), is nearly identical with the ensemble averaged influence of random networks shown in 3.6 a). The influence of the selected scale-free network, shown in figure 3.11 b), has more peaks than the averaged influence shown in figure 3.11 b) but its shape is still similar. The peaks cannot be avoided since there are not enough nodes with high input degrees in a single ESN such that the averages are smooth. Apart from two small peaks, the influence of the nodes in the selected small-world network, shown in figure 3.11 c), are nearly identital to the ensemble averaged influence shown in figure 3.6 c). The chosen networks are compared with the ensemble averages based on their correlation and mutual information with the inuput signal.

First the random network is discussed. Figure 3.12 a) shows the correlation and figure 3.13 a) shows the mutual information. In both cases the graph is similar to the ensemble averaged results, except for nodes with input degrees of 13 and above. Here, the correlation and the mutual information increase instead of further decreasing.

The correlation and the mutual information of the scale-free network, shown in figures 3.12 b) and 3.13 b), decrease for higher input degrees but also show to strong downward spikes. The stronger spike belongs to nodes with an input degree of twelve. When matched with the influence, it turns out that these nodes, although their correlation and mutual information with the input signal is low, belong to a peak in the influence.

The correlation and the mutual information between the nodes of the small-world network and their input signal, shown in figure 3.12 c) and figure 3.13 b), are mostly very similar to the ensemble averaged influence. The only difference is the divergence in the correlation between the Lorenz attractor and the Halvorsen attractor for nodes with high input degrees.

3.8 Examples of Atypical Networks

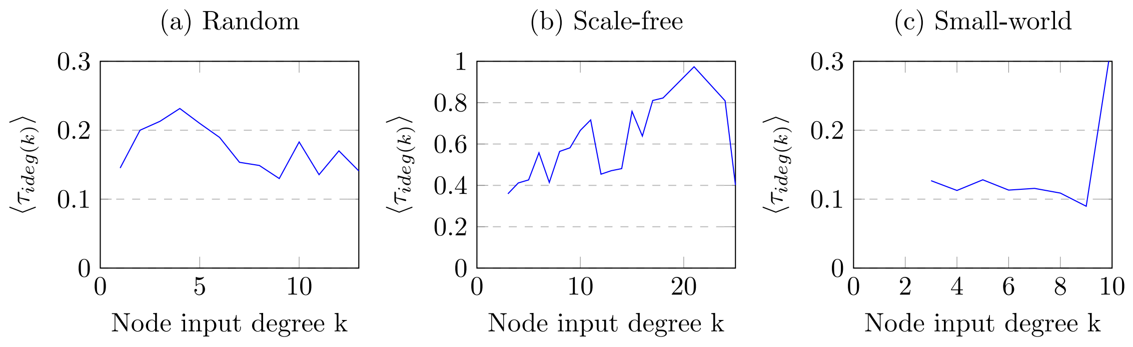

Figure 3.14: Node influence τ on the outcome.

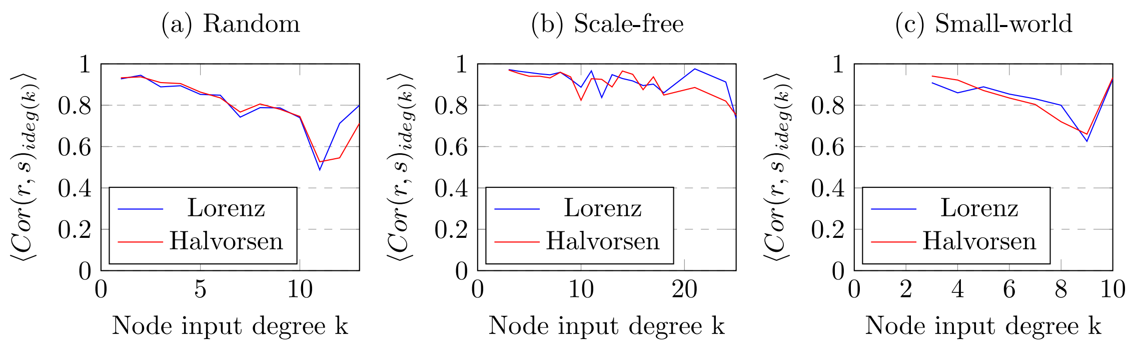

Figure 3.15: Correlation between node values and the input signals of the nodes depending on the input degree.

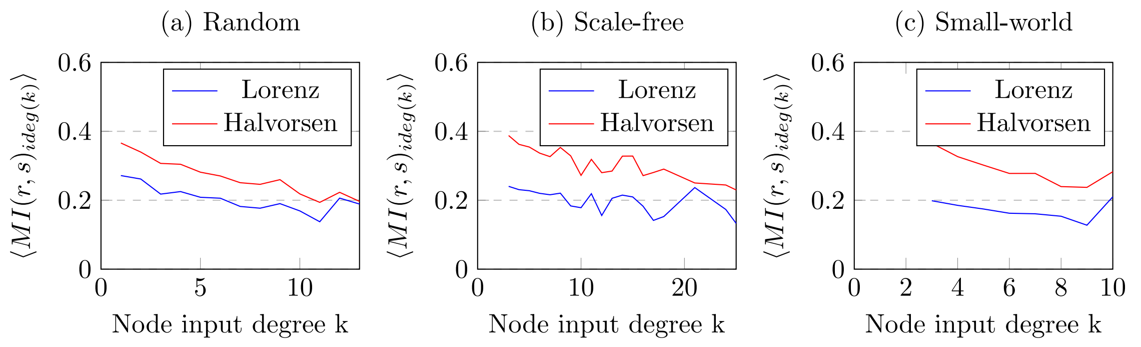

Figure 3.16: Mutual Information between node values and the input signals of the nodes depending on the input degree.

Not all networks are typical. This section investigates ESNs with an atypical influence of its nodes. Again all three ESNs managed to reconstruct the climate of both attractors. The influence of the nodes in the selected random network, shown in figure 3.14 a), network differs from the ensemble averaged influence, shown in figure 3.6 a), for nodes with input degrees above 6. Instead of the decreasing for higher input degree, it stays relative constant with two minor peaks. The influence of the nodes in the scale-free network, shown in figure 3.14 b), grows with increasing input degree. It reaches a peak for nodes with input degree above 20 and then drops rapidly. The influence of the nodes in the selected small-world network, shown in figure 3.14 c), differs from the ensemble averaged result, shown in figure 3.6 c), only for nodes with input degrees of ten and higher. For this nodes, a very strong increase of the input strength is found.

The correlation and the mutual information between the nodes of the random network, shown in figure 3.15 a) and 3.16 a) look relative typical despite the atypical influence. In the correlation plot, the only atypical behaviour is the stronger drop around nodes with an input degree of eleven and the following increase in the correlation. The mutual information differs from its typical graph by the closing gap for nodes with input degrees above twelve.

At first glance, the correlation and the mutual information between the nodes of the scale-free network and their input signal, shown in figures 3.15 b) and 3.16 b), look noisy but typical. But especially the correlation is more interesting when investigated in more detail. Compared to to the ensemble averaged influence its values are relative high. Only nodes with input degrees of 25 and higher have an averaged correlation below 0.8. It is likely that this is one of the reasons for the strange influence graph. For the mutual information, no increased values are found.

In case of the small-world network, the correlation and the mutual information shown in figure 3.15 c) and 3.16 c), behave similar like the influence. For almost all input degrees they are typical. But just like the influence there is a sharp increase for nodes with input degrees of ten and higher.

The results for typical and atypical networks are similar. In both cases, the correlation and mutual information often behaved similar to the ensemble averaged case except for nodes with high degrees. Unfortunatly no conculsions can be drawn from these results. Nodes with high input degrees are rare. As a consequence no meaningful averages could be computed and a lot of noise must be expected. Also, when combining the observations for the influence with the observations of the correlation and mutual information there is no proofable connection. While there are cases where a higher influence comes also with a higher correlation and mutual information, e.g. the atypical scale-free network and the atypical small-world network, there are also just as many examples where this is not the case, e.g. the typical scale-free network and the typical random network.

3.9 Conclusion

The topology of a network is directly related to the degree distribution. In scale-free networks the hubs, nodes with high degrees, accumulate and distribute information efficiently over the whole network. Still, it was shown that the short-term prediction performance of scale-free networks is inferior compared to random and small-world networks. But despite their bad short-term prediction performance, scale-free networks can reconstruct the climate of attractors with the same success rate as random networks. Scale-free networks which failed to reconstruct the climate of the attractors, often also failed the short-term prediction. Small-world networks performed the worst when evaluated on climate reconstructions. The only network topology, successful when evaluated on its short-term prediction capabilities and climate reconstruction capabilities is the random topology.

Initially the question about the impact of the topology was motivated by neuroscience and the topology of the human brain. The results are surprising since neuroscientists exclude random topologies as an option for the topologies of biological neural circuits. Either the analogy between ESNs and neuroscience is further from truth than suspected or there are other reasons why nature does not connect neurons as random networks. The studies cited in the beginning found evidence for scale-free and small-world networks. Further research could try to build ESNs which share more topological properties with biological neural circuits. It is possible that a join of scale-free and small-world networks combines the best of both topologies.

Further investigation of the nodes helped to uncover the role of nodes with different input degrees. When averaged over the ensemble, nodes with high degrees have a more limited influence on the outcome than nodes with less incoming edges. Nodes with high input degrees also have a lower correlation and mutual information with their input signal. Due to the aggregation of signals from their neighbours, these nodes naturally loose information. Instead, the analysis of principal component nodes showed that there are nodes with high input degrees which keep more information about the variance of the input data set.

Chapter 4

Continuous Reservoirs and Memory Capacities

4.1 Continuous Reservoirs

Before defining memory and investigating its properties and effects on ESNs in detail, two extensions are introduced. These extensions are required because they have a strong influence on the memory properties of an ESN. So far, the dynamics of the reservoir are given by the discrete map 2.11. Another approach utilizes a continuous reservoir where the time evolution during training is given by the differential equation

$$\frac{d\vec{r}(t)}{dt} = \gamma(-\vec{r}(t) + \tanh(M\vec{r}(t) + \sigma \cdot W_{in}\vec{s}(t))) \tag{4.1}$$

and during prediction respectively by

$$\frac{d\vec{r}(t)}{dt} = \gamma(-\vec{r}(t) + \tanh(M\vec{r}(t) + \sigma \cdot W_{in}W_{out}q(\vec{s}(t)))) \tag{4.2}$$

where γ is a constant corresponding to a decay rate. Again these equations can only be solved numerically. In literature, a fourth order Runge-Kutta method is used frequently [7]. An alternative approach integrates the equation with a first order Euler method. The time evolution of the reservoir during synchronization and training transforms into:

$$\vec{r}[t+1] = (1-\alpha)\vec{r}[t] + \alpha \tanh(M\vec{r}[t] + \sigma \cdot W_{in}\vec{s}[t]) \tag{4.3}$$

The square brackets indicate that now discrete time points are used. These kinds of reservoir equations were introduced in "Optimization and Applications of Echo State Networks with Leaky Integrator Neurons" [13] and since then have been used in several publications [19] [20]. α relates to γ and the discrete time step Δt by $\alpha = \gamma \cdot \Delta t$. The discrete time step Δt, used to discretize the continuous reservoir equations, must be the same time step as the time step used to discretize the input signal. Changing the α parameter results in ESNs with different kind of continuity properties. Keeping α close to zero reduces the error of the euler-integration/first-order approximation. Equation 4.3 is then a valid approximation of the continuous reservoir equation 4.1, while for $\alpha = 1$ the equation of the discrete reservoir 2.11 is retrieved. By varying α, a continous spectrum of reservoirs can be created, ranging from reservoirs with continuous properties to reservoirs with discrete properties. Due to the numerical discretization the training and prediction phase are with respect to the new time evolution equivalent to the training and prediction phase of a discrete reservoir.

One important result for multifunctional discrete reservoirs also holds for continuous reservoirs. Since different trajectories are not allowed to cross, the embeddings of each attractor must be separated.

4.2 Sparse Input Matrices

The input matrix W_{in} is a $N \times D$ matrix mapping a D-dimensional input state into a N-dimensional reservoir state. In chapter 3, all input matrices are created with one non-zero entry in every row, resulting in a total of N non-zero entries. As a consequence, every node in the ESN is connected to exactly one dimension of the input signal and with every new time step, information about previous states is partially overwritten. Increasing the sparsity of the input matrix by keeping some rows completely zero prevents coupling between some nodes and the input signal. Hence, information about past states is prevailed for longer times. This concept is mathematically described by a new hyperparameter $p_{in-sparse}$. The probability for a row in W_{in} to stay completly empty is $1 - p_{in-sparse}$. The average number of non-zero elements is given by $p_{in-sparse} \cdot N$.

4.3 Short-Term Memory Capacity

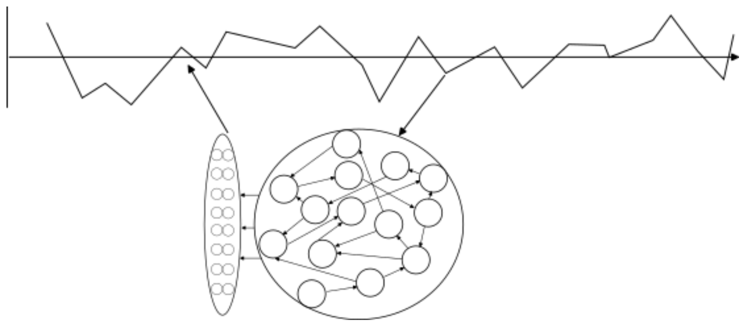

Figure 4.1: The idea behind the short term memory cpapcity is to train an ESN to predict past values of an i.i.d. one-dimensional random signal. A better short-term memory leads to a higher correlation between 'predicted'/remembered and true values.

The short-term memory capacity (STM) of an ESN was introduced by Jaeger in "Short term memory in Echo state networks" [9]. The key idea behind the STM is to train an ESN not to predict future values but to remember past values of an input signal. The setup to measure the STM of an ESN requires multiple training and 'prediction' steps. In each step the so called k-delay STM (STM_k) is computed.

First, the ESN is synchronized with an independet identical distributed (i.i.d.) one-dimensional random signal with values between plus and minus one. If the input matrix of the ESN has incorrect dimensions, a suitable temporary input matrix must be created. The properties of the newly generated matrix should match the properties of the original matrix as closely as possible. All entries in the new input matrix are chosen from the same distribution and if the original input matrix is sparse, the new input matrix should have the same sparsity properties. Following this approach, this temporary matrix has dimension $N \times 1$ and the average number of non-zero elements is still given by $p_{in-sparse} \cdot N$. Now the training for the k-delay STM starts. This is only slightly different from the training procedure for attractor reconstruction. Again after synchronization, the input states and the corresponding reservoir states are collected. The only difference is an additional offset in the optimization. Instead of fitting the n-th reservoir state to the n-th input state, the n-th reservoir state is fitted to the (n-k)-th input state.

$$W^k_{out} = \arg\min_{W^k_{out}} = \sum_t \|W^k_{out} q(\vec{r}[t]) - s[t-k]\| + \beta \|W^k_{out}\| \tag{4.4}$$

E.g. for $k = 1$, W^k_{out} is optimized to map $q(\vec{r}[t])$ to the state one step in the past $s[t-1]$ and for $k = 2$, W^k_{out} is optimized to map $q(\vec{r}[t])$ to $s[t-2]$ and so on. Instead of closing the loop and evolving the ESN as an autonomous system, the 'prediction' phase resembles the training phase and continues to drive the reservoir with an i.i.d. one-dimensional random signal. Again the reservoir states and the input states are recorded. The k-delay STM is then defined as the square of the correlation between 'predicted' states and true states.

$$STM_k = cor(W^k_{out} q(r[t]), s[t-k]))^2 = \tag{4.5}$$

$$\left(\frac{\mathbb{E}((W^k_{out} q(r[t]) - \mathbb{E}(W^k_{out} q(r)) \cdot (s[t-k] - \mathbb{E}(s))))}{\sigma(W^k_{out} q(r)) \cdot \sigma(s)} \right)^2 \tag{4.6}$$

	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 0.8$	$\alpha = 1.0$
STM	6.76	9.49	11.25	12.56	14.04

Table 4.1: STM depeding on α for the ensemble of ESNs presented in figure 4.2 a).

After a sufficient number of k-delay STMs is known, the STM is obtained through the sum over all k-delay STMs

$$STM = \sum_k STM_k \tag{4.7}$$

The better an ESN remembers states from the past, the higher the STM. A numerical implementation can directly follow the steps described in this section. For plots it is convenient to draw a graph with the k-delay STM on the y-axis and the index k on the x-axis.

4.3.1 Numerical Results

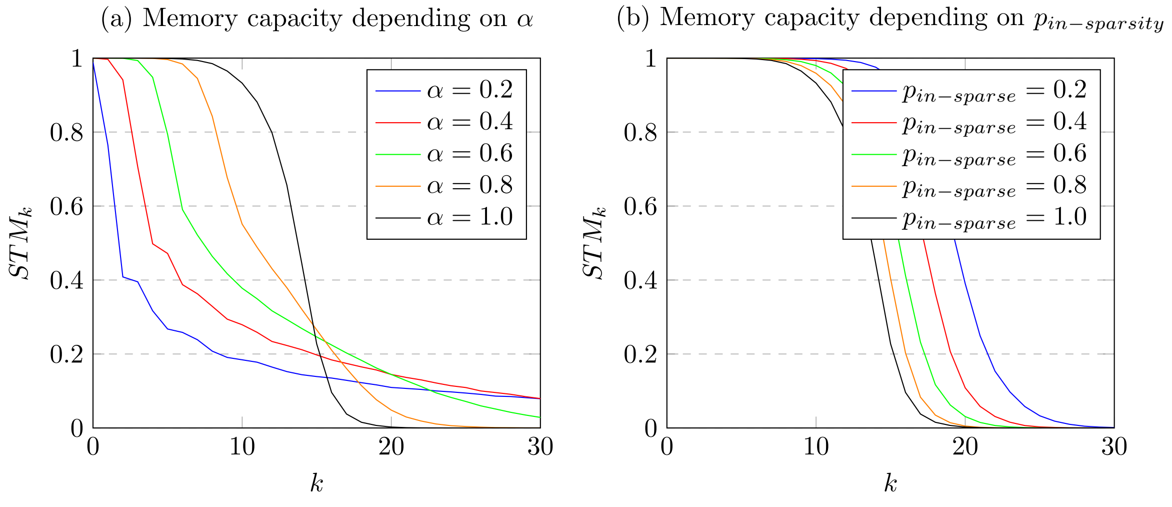

Figure 4.2: a) shows the curve of the k-delay STM for different α values. Each curve is averaged over 20 different STMs. In b) the input matrix sparsity $p_{in-sparse}$ is modified.

For the STM, it is hard to obtain any analytical results. Most analytical results are valid for linear systems only or they lead to very general statements. A detailed investigation with derivations can be found in "Short term memory in Echo State Networks" [9]. The most important results are:

1. The STM of an ESN with N nodes is bounded by N.[1]
2. The k-delay STM of an ESN is monotonically decreasing with k.[2]
3. The STM of a linear ESN is exactly N if the adjacency matrix has full rank.

All of the following numerical simulations use the same setup. The first 2000 data points of the signal are used for synchronization. The next 10,000 data points are used to train the respective W_{out}^k and another 2000 steps to compute the k-delay STM. In total 35 k-delay STMs are computed and then summed up to retrieve the STM. The results are given as the average of 20 different ESNs. The following simulations investigate the influence of hyperparameters on the STM. In each simulation all hyperparameters except one are fixed. Default values are given by $\rho = 0.8$, $\sigma = 1.0$, $\beta = 10^{-7}$, $\alpha = 0.2$ and $p_{in-sparse} = 1.0$. The size of the reservoir is fixed at $N = 500$.

In figure 4.2 a), the k-delay STM for different α realizations is plotted. The figure reveals that α has a very strong influence on the STM. While ESNs with higher α values have an excellent

[1]Jaeger shows that additional non-linearity after the reservoir state is mapped by W_{out} can increase the STM under certain circumstances.
[2]Again it is possible to construct obscure ESNs for which this statement does not hold.

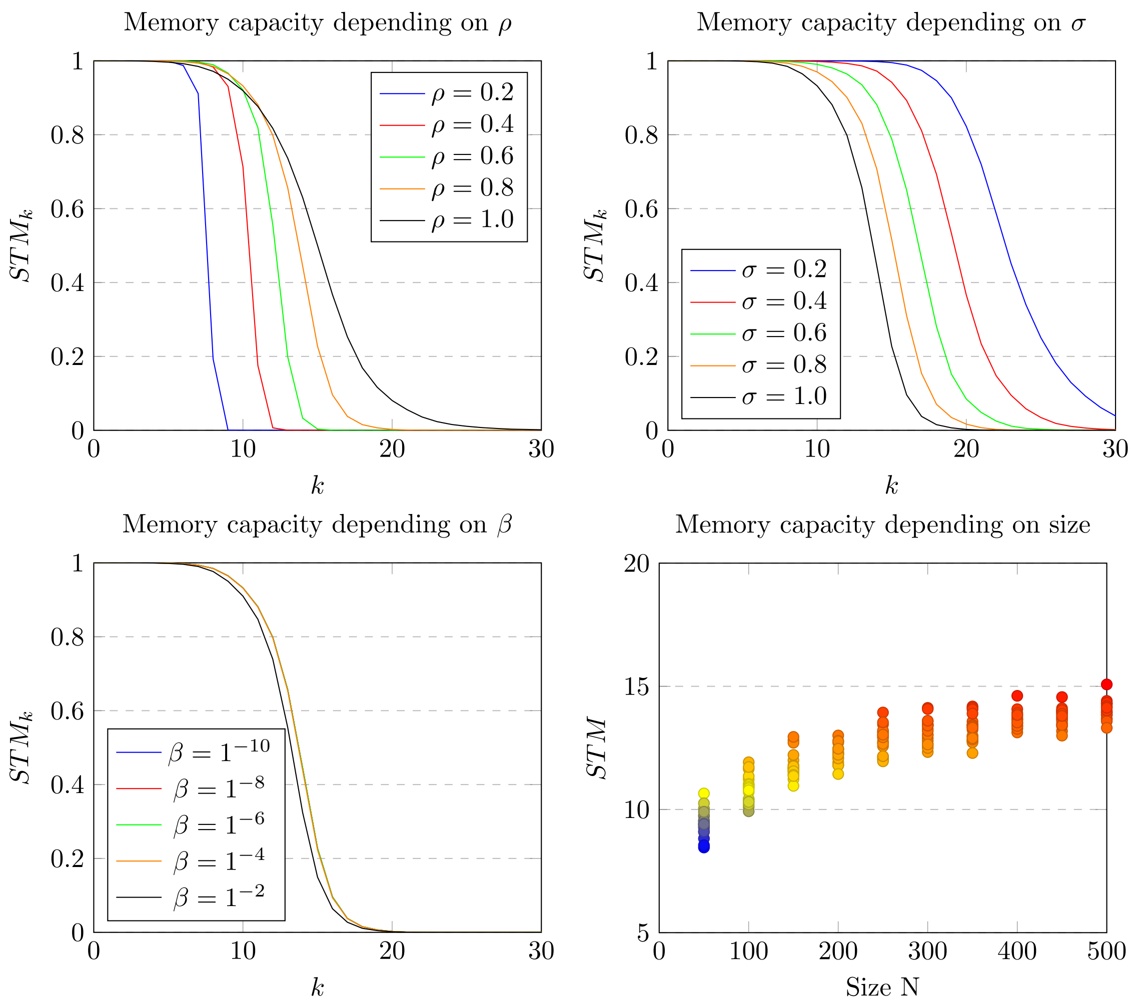

Figure 4.3: a) shows the curve for the k-delay STM for different ρ values. Each curve is averaged over 20 different STMs. In b) the input strength σ varies and in c) the regression parameter β. d) shows the STM for ESNs with different sizes.

memory of the first few states, the k-delay STM drops rapidly to approximately zero once k is greater than 20. For instance, the discrete ESNs with $\alpha = 1$ have correlations with the true states above 0.9 up to delays of ten time steps. But further increasing the delay leads to a steep drop in the k-delay STM until the correlation for delays above 20 is close to zero. In contrast, ESNs with a small α value already have problems reconstructing states with very short delays. For instance, the k-delay STM of the ESNs with $\alpha = 0.2$ drops immediatly. Even 'predictions' for states with a delay as low as three have correlations with their true states below 0.5. Instead, these ESNs are better at reconstructing states with high delays. While the 'predictions' of states with k-delays greater than 20 are far from good, the k-delay STM of continuous ESNs beats discrete ESNs by more than one order of magnitude. In other words, there is still some knowledge about old input states in continuous ESNs. The effect of the α parameter on the k-delay STM can be explained by looking at the definition of the time evolution given by equation 4.3. Due to the first term, a continuous reservoir will have linear correlations in its reservoir states. And since the readout matrix W_{out} is linear, linear correlations in the reservoir states lead to linear correlations in the 'predicted'/remembered output states. But the task of the STM is to reconstruct uncorrelated input signals, therefore linear correlations are not helpful. As a consequence, a continous reservoir is already facing difficulties when trained to reconstruct states with small delays. Still, the linear correlations are beneficial when reconstructing states with longer delays. Information about past states is prevailed for longer times, explaining the longer tail of the k-delay STM for more continuous ESNs.

But not just the k-delay STM also the STM which is proportional to the area under curve can be compared. Table 4.1 lists the STM of the ensemble of ESNs plotted in figure 4.2 a). Although the numbers are not totally correct for ESNs with small α due to the truncation of the tail of the STM_k, it is still evident that discrete ESNs have a higher STM than continuous ESNs.

Figure 4.2 b) shows the relation between the STM and the sparsity of the input matrix. It can be seen that a more sparse input matrix leads to an increase of the STM. In contrast to the case where α is varied, the shape of the individual curves stays approximately the same. Only the delay where the k-delay STM begins to drop moves further to the right. This is not unexpected since a more sparse input matrix overwrites less information in each step. For instance, when $p_{in-sparse} = 0.2$ only 20% of the nodes in the network will be partially overwritten by the input signal.

In figure 4.3 a) the STM for different $\rho < 1$ realizations is plotted. From the Echo state property, it is already known that ρ has a major influence on the memory. But in contrast to the STM, the Echo-state property fails to give a measureable quantity for the memory. As expected increasing ρ does indeed increase the memory capacity.

The input strength σ, shown in figure 4.3 b), has a similar influence on the memory as $p_{in-sparse}$. A stronger input strength overwrites more information in the nodes, decreasing the ability of an ESN to remember past states. The regression parameter β, shown in figure 4.3 c), influences the STM only a little bit. Despite its influence on W^k_{out}, the differences between the measured STMs are small. Last but not least figure 4.3 d) shows the relation between the network size and the STM. The STM of 20 ESNs with different sizes is compared. Every point represents an ESN. The expectation that memory capacity increases with the size of an ESN is fullfilled by the STM.

4.4 Memory as Norm of the Variation

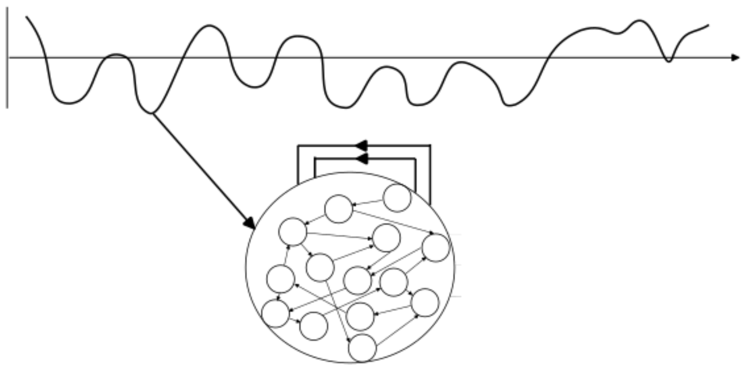

Figure 4.4: The memory capacity proposed by Inubushi and Yoshimura measures the distance between two nearby reservoir state. When the ESN has the Echo state property the two reservoir states will get closer during further synchronization. This process takes more time in an ESN with more memory.

Inubushi and Yoshimura formulate in "Reservoir computing beyond memory-nonlinearity trade-off" the idea to measure the memory of an ESN based on its ability to distinguish between two nearby past reservoir states [10]. To be consistent with another paper comparing different memory capacities [31], this memory capacity will be called Norm of the Variation (NVMC). Given a reservoir state $\vec{r}_1[0]$ which was reached by synchronizing the reservoir with a signal $\vec{s}[t]$ from $-\infty$ to 0, the NVMC tracks the evolution of a small deviation $\vec{\delta}[t]$ during further synchronization. For a time evolution given by 4.3 the deviation between two states synchronized by the same signal can be written as

$$\vec{\delta}[t+1] = [(1-\alpha)\vec{r}_1[t]+\alpha\tanh(M\vec{r}_1[t]+\sigma W_{in}\vec{s}[t])]-[(1-\alpha)\vec{r}_2[t]+\alpha\tanh(M\vec{r}_2[t]+\sigma W_{in}\vec{s}[t])] \quad (4.8)$$

where $\vec{r}_2[t]$ is given by $\vec{r}_2[t] = \vec{r}_1[t] + \vec{\delta}[t]$. For a small deviation, the equation can be linearized via a Taylor expansion at $\vec{\delta}[t] = 0$. This results in

$$\vec{\delta}[t+1] = (1-\alpha)\vec{\delta}[t] + \frac{\alpha}{\cosh^2(M\vec{r}_1[t] + \sigma W_{in}\vec{s}[t])} M\vec{\delta}[t] + \mathcal{O}(\vec{\delta}^2) \quad (4.9)$$

Usually, the direction of the deviation is not of interest. Instead the norm $\|\vec{\delta}[t]\|$ is the relevant quantity. Similar to the STM, the sum over all deviations $\sum_t \|\vec{\delta}[t]\|$ is also of interest. Both of these quantities are called NVMC. The context makes it clear which one is meant, e.g. when plots show graphs over time, $\|\vec{\delta}[t]\|$ is shown, while scatter plots refer to $\sum_t \|\vec{\delta}[t]\|$.

The numerical implementation computes the NVMC by synchronizing the ESN with a given signal. If not stated otherwise, N orthogonal deviations are created by a QR-decomposition of a random matrix. This way, the algortihm does not only track a single deviation in a single direction but the deviations span the full reservoir state space. The deviations are then propagated in time by equation 4.8 and the result is averaged over all N deviations.

4.4.1 Numerical Results

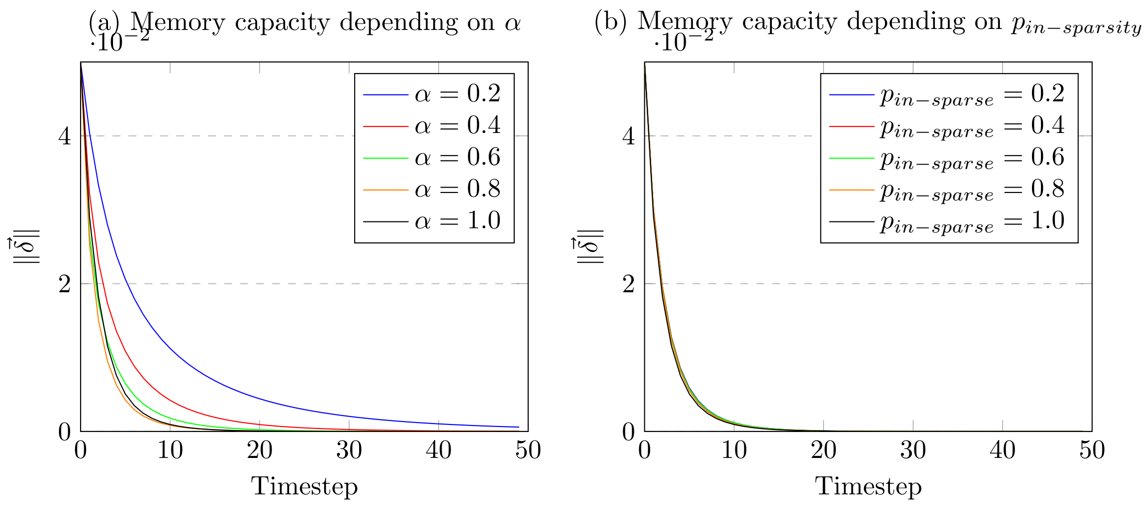

Figure 4.5: NVMC depending on α or $p_{in-sparse}$.

Similar to section 4.3.1, this section presents numerical simulations of the NVMC. In each simulation, one hyperparameter is varied while the other hyperparameters are kept at their default value. The default values of the hyperparameters are given by $\sigma = 1.0$, $\rho = 0.8$, $\beta = 10^{-7}$, $\alpha = 0.2$ and $p_{in-sparse} = 1.0$. The size of the ESNs is again fixed to $N = 500$. For each ESN, the NVMC is measured with data from the Lorenz attractor positioned at the origin and rescaled, such that no point has a greater distance then one from the origin. Since the NVMC depends on the initial reservoir state, the NVMC is measured ten times at ten different positions on the Lorenz attractor. The result is obtained by averaging over the ten positions. Additionally, the NVMC is not measured for just one ESN but averaged over an ensemble of 20 ESNs. The first 2000 time steps are used to synchronize the network with the signal. Then, the deviated reservoir states are created and propagated in time.

Figure 4.5 a) shows the relation between the α parameter introduced in equation 4.3 and the NVMC. Here, increasing α decreases the NVMC. Compared to the STM this is exactly the opposite relation. This time the difference can not be explained by non-linearities since a small deviation is only propagated by linear terms in equation 4.9. In the first term, the contraction of the deviation depends on α only. In the second term, many more factors influence the contraction of the deviation like the spectral radius and the input signal. Especially when the adjacency matrix does not have full rank, contraction can happen much faster.

Also for the sparsity of the input matrix, there is a difference between the NVMC and the STM. As shown in figure 4.5 b), the sparsity of the input matrix has no visible effect on the NVMC. The same holds for the input strength σ shown in figure 4.6 b). The input matrix and σ appear both only in the denominator of the second term in equation 4.9. But since α is set by default to 0.2, the second term has only a marginal impact on the result. If α were higher, the influence of the input matrix and the input-strength would be stronger. A more general discussion on the impact of the input signal on the NVMC is found in the following section.

Figure 4.6 a) shows that increasing the spectral radius does increase the NVMC. It is important to notice, that the spectral-radius is the only hyperparameter which has the same strong influence

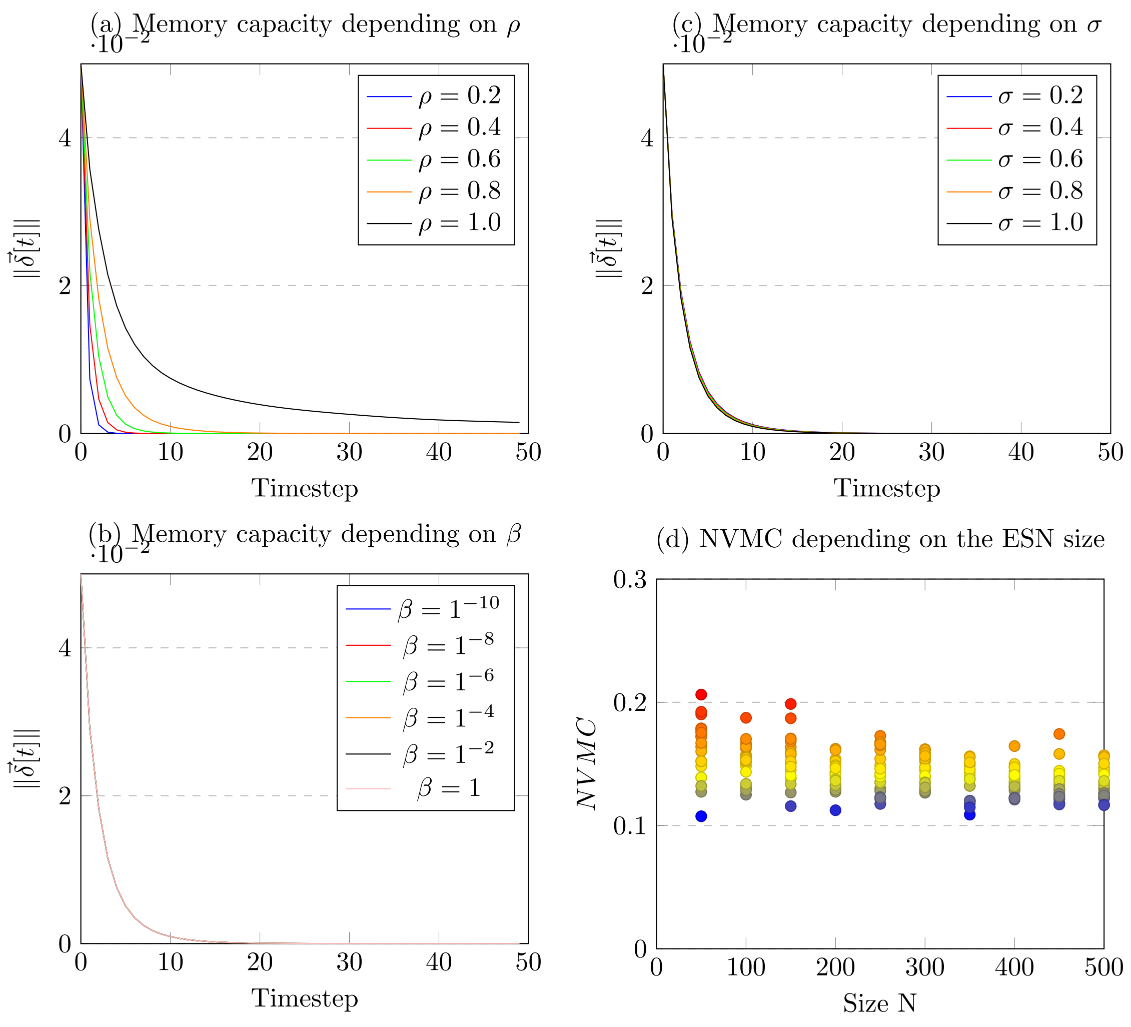

Figure 4.6: NVMC and its relation to different hyperparameters.

on both, the NVMC and on the STM. The reason is the close connection between the spectral radius and the Echo state property and therefore memory. Analytical results derived in the following section give more detailed insights into the role of the spectral radius on the NVMC.

Since the NVMC has no direct or indirect dependency on β, it is not possible for β to have any influence on the NVMC. This is shown in figure 4.6 just for completeness. Plotting the NVMC against the size of the ESN reveals another suprising result. As seen in figure 4.6 d), the NVMC is does not depend on the size. Intuitivly, a larger reservoir should also have a larger memory capacity. But this is clearly not the case here. Again this can be explained by anticipating results from the following section. It turns out that the eigenvalues of the adjacency matrix have a major influence on the NVMC. But a larger matrix does not have larger eigenvalues.

4.4.2 Analytical Results

A big advantage of the NVMC is the fact that at least for simple input signals analytical solutions can be obtained. In the following section, analytical results are derived for discrete and continuous ESNs. The time evolution of a small deviation for a continuous reservoir can be derived by the same steps which were used to derive equation 4.9 for a discrete reservoir.

$$\frac{d\vec{\delta}(t)}{dt} = \gamma(-\vec{r}_1(t) + \tanh(M\vec{r}_1(t) + \sigma \cdot W_{in}\vec{s}(t))) - \gamma(-\vec{r}_2(t) + \tanh(M\vec{r}_2(t) + \sigma \cdot W_{in}\vec{s}(t))) \tag{4.10}$$

$$\frac{d\vec{\delta}(t)}{dt} = -\gamma(\vec{\delta}(t)) + \gamma(\tanh(M\vec{r}_1(t) + \sigma \cdot W_{in}\vec{s}(t))) - \tanh(M\vec{r}_2(t) + \sigma \cdot W_{in}\vec{s}(t)))) \tag{4.11}$$

In a first order approximation, the time evolution of the deviation for the discrete and continuous case is given by:

$$\vec{\delta}[t+1] = \frac{1}{\cosh^2(M\vec{r}_1[t] + \sigma W_{in}\vec{s}[t])} M\vec{\delta}[t] + \mathcal{O}(\vec{\delta}^2) \tag{4.12}$$

$$\dot{\vec{\delta}}(t) = \gamma(-1 + \frac{M}{\cosh^2(M\vec{r}_1[t] + \sigma W_{in}\vec{s}[t])})\vec{\delta}(t) + \mathcal{O}(\vec{\delta}^2) \tag{4.13}$$

The differential equation 4.13 for the continuous reservoir is solved by:

$$\vec{\delta}(t) = e^{-\gamma t} e^{\int_0^t \frac{\gamma M}{\cosh^2(M\vec{r}_1(t)+\sigma W_{in}\vec{s}(t))} dt} \vec{\delta}(0) \tag{4.14}$$

At first, a special case which can be solved analytically is considered. Given a constant input signal $\vec{s}(t) = \vec{s}$ and an ESN with the Echo-state property it follows that the reservoir state is also constant $\vec{r}_1(t) = \vec{r}$.

Proof. Given a reservoir state $\vec{r}(0)$ reached by synchronizing the reservoir from $t = -\infty$ to $t = 0$ with a constant input signal. Further synchronization leads to the reserovir state $\vec{r}(t^\star)$. Since both reservoir states, $\vec{r}(0)$ and $\vec{r}(t^\star)$, have been reached by an infinite long synchronization, their states depend on the input signal only. For both states the signal is an infinte long constant signal and therefore identitcal. From the Echo state property it directly follows that $\vec{r}(0)$ and and $\vec{r}(t^\star)$ must be the same. And since no specific $t^\star$ has been selected this holds for all times. □

In this case, the time evolutions for the deviations can be written as:

$$\vec{\delta}[t+1] = \frac{M}{\mu}\vec{\delta}[t] \tag{4.15}$$

$$\vec{\delta}(t) = exp(-\gamma(1 - \frac{M}{\mu})t)\vec{\delta}(0) \tag{4.16}$$

where $\mu = \cosh^2(M\vec{r}_1 + \sigma W_{in}\vec{s})$ is now a constant. If M is normal, it has an eigendecomposition and any deviation can be expressed as a linear combination of eigenvectors. Restricting the further discussion to deviations which are given by an eigenvector of M leads for a discrete reservoir and the norm of an eigenvector-deviation $\vec{\delta}_\lambda$ at time step t to the following equation:

$$\|\vec{\delta}_\lambda[t]\| = \| \left(\frac{\lambda}{\mu}\right)^t \vec{\delta}_\lambda[0]\| = \left(\frac{\|\lambda\|}{\mu}\right)^t \|\vec{\delta}_\lambda[0]\| \tag{4.17}$$

Therefore the deviation converges towards zero if and only if the largest eigenvalue is smaller than μ. The rate of convergence depends on the ratio of the norm of the largest eigenvalue and μ. It should be noted that for normal matrices every deviation can be represented as a sum of eigenvectors but not every eigenvector can be interpreted as a valid deviation. Because the reservoir equations are formulated on real numbers, deviations are also given by real vectors. But a real matrix can still have complex eigenvalues and eigenvectors. For discrete reservoirs, an explicit discussion of complex eigenvalues or eigenvectors is not necessary. As long as the norm of the largest eigenvalue is smaller than μ, any small deviation will vanish eventually. Due to the hyperbolic cosine, the constant μ is bounded from below by 1 and is exactly one if and only if $\vec{s} = \vec{0}$ and $\vec{r} = \vec{0}$. In this case Jaegers result for the Echo state property is recovered.

When learning or predicting attractors, the input signal is neither constant nor zero. While the dynamics of the input signal has no direct influence on the convergence rate, its values do so. For instance, constant input signals far from the origin increase the value of μ exponentially. The possible upper bound for ρ such that an ESN keeps the Echo state property then increases. This effect can also be seen for constant input signals with different distances from the origin and is visualized in figure 4.7 a). The greater the distance between the constant and the origin, the faster is the decay of the NVMC.

For continuous reservoirs the time evolution of the norm of an eigenvector-deviation is given by:

$$\|\vec{\delta}_\lambda(t)\| = \|exp(-\gamma(1 - \frac{\lambda}{\mu})t)\|\|\vec{\lambda}_\lambda(0)\| \tag{4.18}$$

There is a significant difference for the rate of decay between the discrete and the continuous reservoir. Here, the rate of decay does not only depend on the norm of the eigenvalue but also

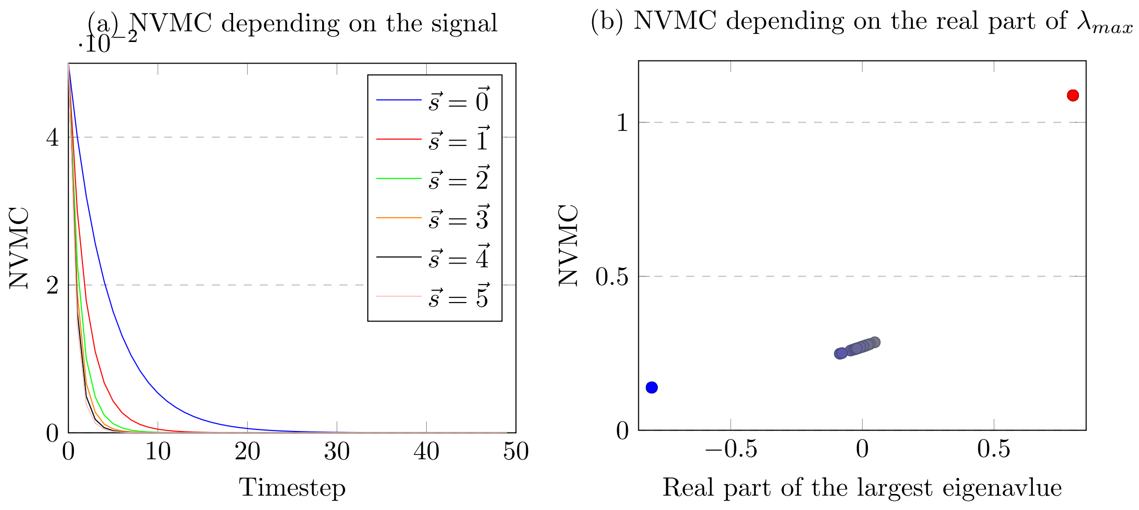

Figure 4.7: a) NVMC for a constant input signal with different distances from the origin. b) NVMC depending on the real part of the eigenvalue with the largest magnitude when the deviation is given by its eigenvector

on its real part. Especially the sign of the real part has a large influence on the NVMC. Plotting the NVMC where the deviation is in the direction of the eigenvector belonging to the eigenvalue with the largest magnitude, splits the considered ESNs into three distinct groups. This is shown in figure 4.7 b). If the eigenvalue is real, then a negative sign leads to a reduced NVMC, while a positive sign results in a higher NVMC. When the real part of the eigenvalue is close to zero, the NVMC of the ESNs is between the two previous cases. Although it should be noted that in this case the eigenvector is definitely given by a complex vector and should not be interpreted as a deviation. An arbitrary deviation is a linear combination of every eigenvector and the NVMC depends therefore on every eigenvalue. It follows that a positive definite adjacency matrix has a higher NVMC. But as long as adjacency matrices are created as sparse random graphs it is hard to exploit this fact. Again the rate of convergence also depends on μ. When the sign of the real part of lambda is negative, an input signal far from the origin can even increase the NVMC.

Now, the general case of chaotic inputs and reservoir states is discussed. Under the assumption that the reservoir state stays bounded as long as the input signal is bounded, it is feasible to come up with further analytical results. For discrete reservoirs, the time evolution of the deviation is now given by

$$\vec{\delta}[t] = \prod_t \frac{M}{\mu[t]} \vec{\delta}[0] \tag{4.19}$$

$\mu[t]$ depends now on the time step and is given by $\mu[t] = \cosh^2(M\vec{r}_1[t] + \sigma W_{in}\vec{s}[t])$ and bounded from below by one. Applying the norm on both sides it is possible to obtain a sufficient condition for convergence. First the compatibility between the euclidean vector norm and spectral norm of a matrix is used. Then the submultiplicative property of the spectral norm is used and in the last step the definition for the spectral norm is inserted.

$$\|\vec{\delta}[t]\| = \|\prod_t \frac{M}{\mu[t]} \vec{\delta}[0]\| \leq \|\prod_t \frac{M}{\mu[t]}\| \|\vec{\delta}[0]\| \leq \prod_t \frac{\|M\|}{\mu[t]} \|\vec{\delta}[0]\| \leq \prod_t \frac{\|\lambda_{max}\|}{\mu[t]} \|\vec{\delta}[0]\| \tag{4.20}$$

The result is similar to the previous case with a constant input signal. If $\prod_t \|\lambda_{max}\|/\mu[t]$ converges towards zero, any small deviation will converge towards zero. The difference is that now the 'if' does only have one direction, whereas in the constant case, the 'if' is an 'if and only if'.

For continuous reservoirs, the time evolution of the deviation is given by:

$$\delta(t) = e^{-\gamma t} e^{\gamma \frac{M}{\mu(t)}} \vec{\delta}(0) \tag{4.21}$$

The time evolution is a product consisting of two parts. The interpretation of the first term $e^{-\gamma t}$ is straightforward. It describes an exponential decay with a decay rate γ. The second term is more

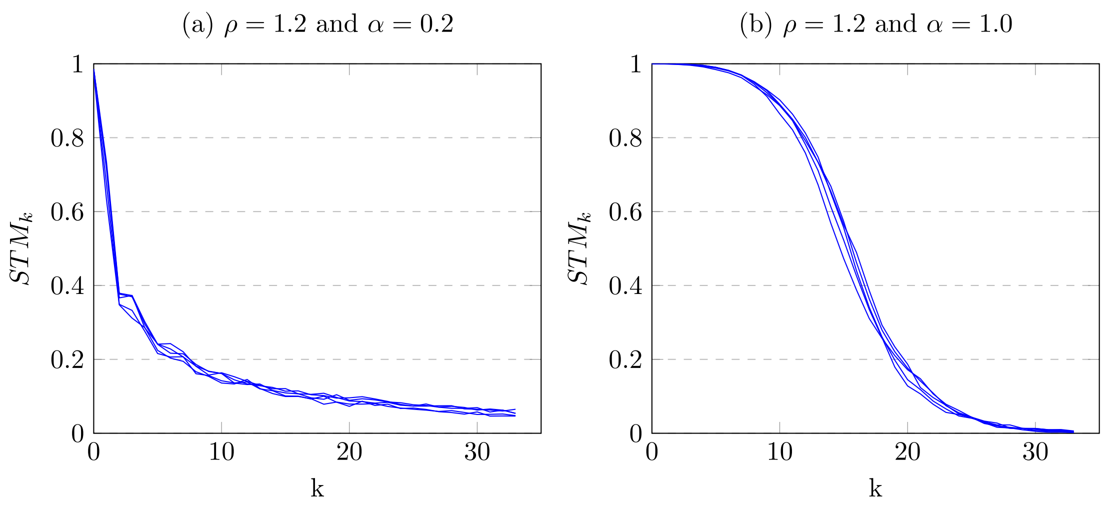

Figure 4.8: STM of selected individual ESNs with $\rho = 1.2$ for a) $alpha = 0.2$ and b) $\alpha = 1.0$. Overall the results are not significantly different from the results for $\rho < 1.0$.

interesting and its action on the deviation depends on the Matrix M and on $\mu(t)$ which is given by the relation:

$$\frac{1}{\mu(t)} = \int_0^t \frac{dt}{\cosh^2(M\vec{r}_1(t) + \sigma W_{in}\vec{s}(t))} \tag{4.22}$$

In the general case, it is not possible to obtain an exact solution of the integral. Approximating it by observing that the denominator in the integral is bounded from below by one, leads to the same results already derived above.

4.5 Numerical Results for $\rho > 1$

The analytical results from the previous section and the Echo state property already indicate that memory and especially the NVMC might behave unintuively, once a spectral radius greater than one is chosen. For a discrete reservoir, equation 4.19 and equation 4.20 show, that the deviation might even grow over time. This is possible when the spectral radius is greater than one. For a continuous reservoir, the situation is a bit more complex. In principal it is possible for the deviation to grow over time when the second exponential term in equation 4.21 is stronger than the first exponential decay term.

For the STM, it is not clear what happens once the spectral radius is greater than one. Fundamentally the STM is still limited by the number of nodes and in contrast to the NVMC, the k-delay STM cannot grow with increasing delay.

The numerical experiments use the same setup as before, except for the spectral-radius which is now set to $\rho = 1.2$ and α which is set to either $\alpha = 0.2$ or $\alpha = 1.0$. Instead of averaging over an ensemble of ESNs, the STM_k and NVMC are plotted for some selected individual ESNs. This is especially important for the NVMC where averaging can hide the growing deviations. First figure 4.8 a) and b) show the STM_k of five selected individual ESNs. The ESNs in a) have α set to 0.2, making them continuous. When comparing this plot with the blue line from figure 4.2 a) no differences are recognizeable. The same holds for discrete reservoirs plotted in figure 4.8 b). When compared to the black line from figure 4.2 a), again no difference is visible.

The same setup is used to simulate the NVMC for ESNs with ρ greater than one. To see the full effect of the larger spectral radius 200 time steps are computed. Figure 4.9 a) presents the NVMC of five selected ESNs with $\alpha = 0.2$. For one of the five ESNs, the NVMC starts to increase after some time. In the other four cases, the NVMC is similar to ESNs with ρ less than one and decreases over time. Still, when comparing figure 4.9 a) with the blue line from figure 4.5 a), it is evident, that the increased spectral radius leads to larger areas under the curves for all ESNs. Figure 4.9 b) shows the result for discrete ESNs. This time the NVMC is increasing for three out of five ESNs. Comparing the time evolution of small deviations for continous ESNs, given by equation 4.21, with the time evolution of small deviations for discrete ESNs, given by

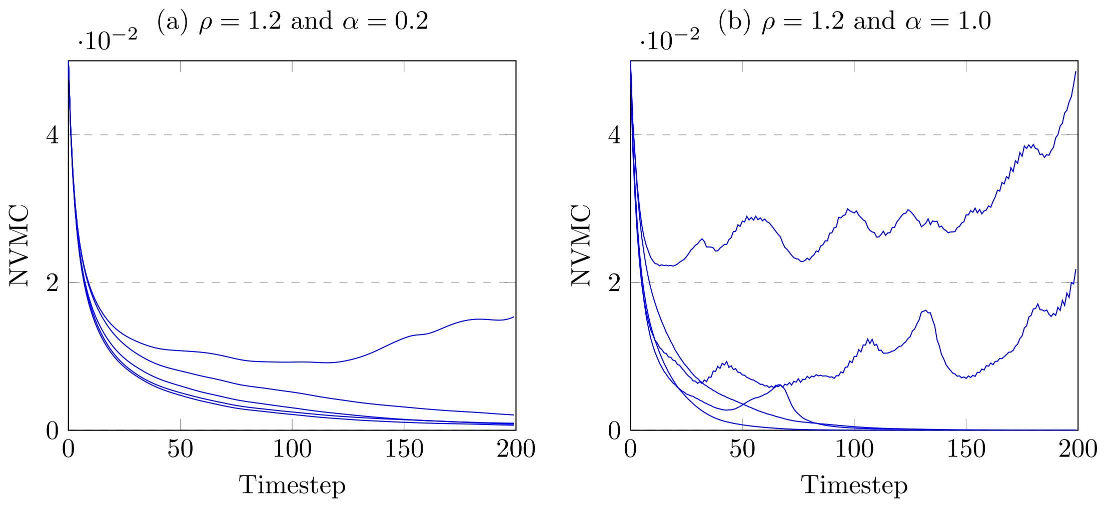

Figure 4.9: NVMC of selected individual ESNs with $\rho = 1.2$ for a) $\alpha = 0.2$ and b) $\alpha = 1.0$. Some networks show growing deviations. The effect is stronger for discrete ESNs.

equation 4.19, explains why ESNs with increasing NVMC appear more often in the discrete case. The NVMC of continous ESNs is always damped by γ while the NVMC of discrete ESNs only depends on the norm of the eigenvalues of the adjacency matrix.

Chapter 5

Predicting Overlapping Attractors

5.1 Prior Work

The STM and its influence on the reconstruction of overlapping circles is shortly discussed in Flynn et al. [14].

5.2 Simulation Setup for Overlapping Attractors

So far, multifunctionality has been tested with attractors at different positions in the state space. Nonetheless, ESNs can also be trained on overlapping trajectories. The first thing to notice is that the overlap can only exist in the low dimensional state space of the attractors. In the high dimensional reservoir state space the attractors must be separated. This is true for discrete and continuous ESNs. Only the projection from the high dimensional reservoir state space to the low dimensional attractor state space via W_{out} let the attractors overlap again. So how can the ESN separate the overlapping trajectories? The only option is to be aware of past points. An ESN must differentiate between trajectories not just based on the current location but also based on the local history of past points. In other words: The ESN needs memory. This chapter applies the memory measurements from the previous chapter to ESNs trained on overlapping attractors. Results from the previous chapter show that both memory capacities have different properties. The relevance of each memory capacity is therefore of particular interest.

The training of the attractors follows the same procedure as the training in previous multifunctional setups. From a trajectory starting within the basin of the attractor, the first 500 points are discarded to ensure all further points are part of the attractor. The synchronization phase uses the next 5000 data points. For training 40,000 data points are used. The attractors are given by the Lorenz attractor and the Halvorsen attractor. All ESNs are trained on the same trajectories. The attractors are shifted such that their mean point is at the origin and scaled such that no data point has a distance greater than one from the origin. The networks have a size of 2500 nodes and an average degree of six. In practice, none of the ESNs is able to reconstruct the climate of both attractors. The geometric mean of the forecast horizon is therefore the only meaningful measure for the quality of the prediction. The result of one prediction can be seen in figure 5.1. At first the predicted trajectories are close to the true trajectories. But after some time, the predicted trajectory of the Lorenz attractor diverges towards the Halvorsen attractor. Apart from the combined forecast horizon the training error and the minimal embedded attractor distance are measured.

The relevant hyperparameters to create ESNs with a broad range of memory properties are the spectral radius ρ, the input matrix sparsity $p_{in-sparse}$ and α. While the input strength σ has a similar effect on the memory capacity as $p_{in-sparse}$, it does also change the input signal. Changes in the outcome when varying the input strength can therefore not necessarily be attributed to changes in the memory capacity. The input strength is fixed at $\sigma = 3.0$, the regression parameter is set to $\beta = 5^{-10}$ and the average degree is six. Both memory capacities are computed based on the following 50 timesteps.

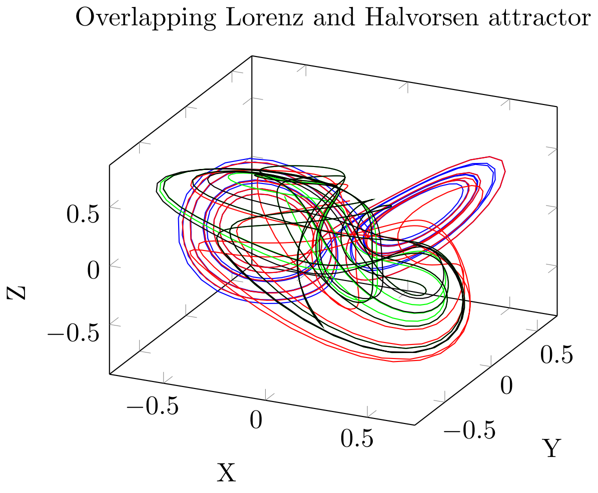

Figure 5.1: Plot of the first 700 time steps of the true and predicted trajectories of overlapping attractors. Blue is the true trajectory of the Lorenz attractor and red its prediction. For the Halvorsen attractor its true trajectory is plotted in green and its predicted trajectory in black. On the Lorenz attractor on the right side of the plot, it can be seen how the predicted trajectory diverges from the Lorenz attractor to the Halvorsen attractor.

5.3 Results for $\rho < 1.0$

Figures 5.2 a) and b) show the arithmetic mean of the combined divergence times for ensembles containing 20 ESNs. Apart from the spectral radius, the ESN ensembles used in both simulations are tile by tile exactly the same. Comparing the tiles from figure 5.2 a), where $\rho = 0.6$ with equivalent tiles in figure 5.2 b), where $\rho = 0.8$ shows that most of the time ESNs with $\rho = 0.8$ perform better. Since the spectral radius is higher these are also the ESNs which have on average a higher STM and NVMC. In both cases, ESNs with $\alpha = 0.2$ usually outperformed ESNs with higher α values. The parameter configuration of the best ensemble is $\rho = 0.8$, $\alpha = 0.2$ and $p_{in-sparse} = 1.0$. This gives first hints about the role of memory in these simulations. While the NVMC is among the highest of all considered ESN, due to $\alpha = 0.2$, the STM is among the lowest. For $p_{in-sparse}$ it is harder to identify an influence on the prediction performance. Comparing each heatmap row by row shows that the row with $p_{in-sparse} = 0.6$ performs pretty good for all α values. On average the ESNs in these row have neither a particular high nor a particular low STM.

To be able to draw more detailed conclussions about the role of memory capacities on the prediction perfromance, it is required to look at the individual ESNs. Figures 5.3 a) and b) show the STM of each ESN with the combined divergence time. Once for $\rho = 0.6$ and once for $\rho = 0.8$. Both figures show a clear trend between the STM and the combined forecast horizon as long as the STM is small. The ESNs performing the best are also the ESNs having low STM values. A small increase in the STM results in a strong drop of the combined forecast horizon. In both figures, ESNs which performed the worst are located in the middle with an STM between four and eleven. But even in this region there are also ESNs with average combined forecast horizons. ESNs with a higher STM perform neither extraordinary good nor extraordinary bad.

Figures 5.4 a) and b) show the same relation as figures 5.3 a) and b) but this time for the NVMC instead of the STM. Here, the best performing ESNs are ESNs with the highest NVMC. Especially ESNs with low forecast horizons only exist when the NVMC is realtive low. In both plots, the data points are grouped into distinct clusters. Since for fixed ρ, only α has a strong influence on the NVMC, each cluster must represent a group of one or more specific α values. When clusters are close together they appear as one resulting in less than five visible clusters. From the plots, it can be recognized that clusters with lower NVMC values are closer together. As a result, for $\rho = 0.6$, the clusters with $\alpha = 1.0$ and $\alpha = 0.8$ are merged together and for $\rho = 0.8$, the clusters with $\alpha = 1.0$, $\alpha = 0.8$ and $\alpha = 0.6$ are merged into a single cluster.

With the results so far, it is not clear if memory has an important impact on the short term prediction performance. While the combined forecast horizon is higher for ESNs with high NVMC, it can also be argued that continuous ESNs are more suitable for the task. This hypothesis is

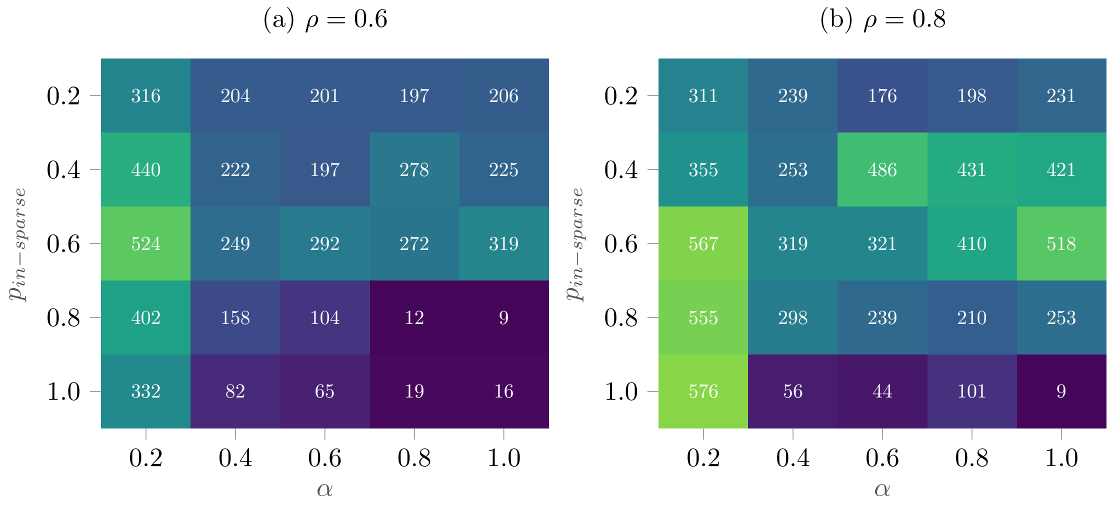

Figure 5.2: Both figures show the combined forecast horizon with the $p_{in-sparse}$ parameter on the x-axis and α on the y-axis. In figure a) the spectral radius is $\rho = 0.6$ and in figure b) the spectral radius is $\rho = 0.8$. Apart from the spectral radius the ESNs in both figures are exactly the same.

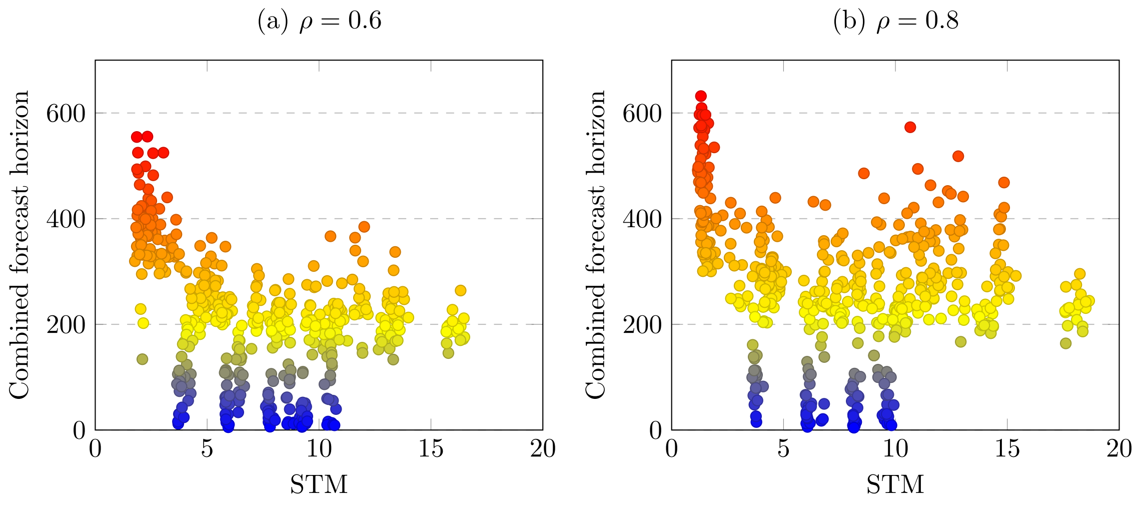

Figure 5.3: Combined forecast horizon together with the STM of each ESN for a) $\rho = 0.6$ and b) $\rho = 0.8$.

supported by two observations. First by the splitting into discrete clusters in figure 5.4 and second by the fact that ESNs with low STM values perform better. Further insights can be gained by looking at ESNs with ρ greater than one. Then it is not necessarily true anymore that discrete ESNs have a lower NVMC than continuous ESNs.

5.4 Results for $\rho \geq 1.0$

The first thing to look at is the averaged combined forecast horizon depending on the hyperparameters α and $p_{in-sparse}$. The two heatmaps in figure 5.5 a) and b) visualize the results for $\rho = 1$ and $\rho = 1.2$ in the same way, figure 5.2 does for $\rho < 1$. Previously it was possible to identify trends between the tested hyperparameters and the performance in each heatmap. Now this is hardly possible. For $\rho = 1.0$, the best tiles are nearby the diagonal going from the bottom left to the top right. Ensembles above this diagonal perform worse than ensembles below this diagonal. For $\rho = 1.2$, it is even harder to identify a structure. Usually, ensembles with $p_{in-sparse}$ above 0.4 perform better. Especially when compared to simulations where $\rho < 1$, α lost its dominating influence on the performance. Comparing both heatmaps tile by tile shows that $\rho = 1$ is most of the time superior. This is still the case when all four heatmaps are compared. The best performing

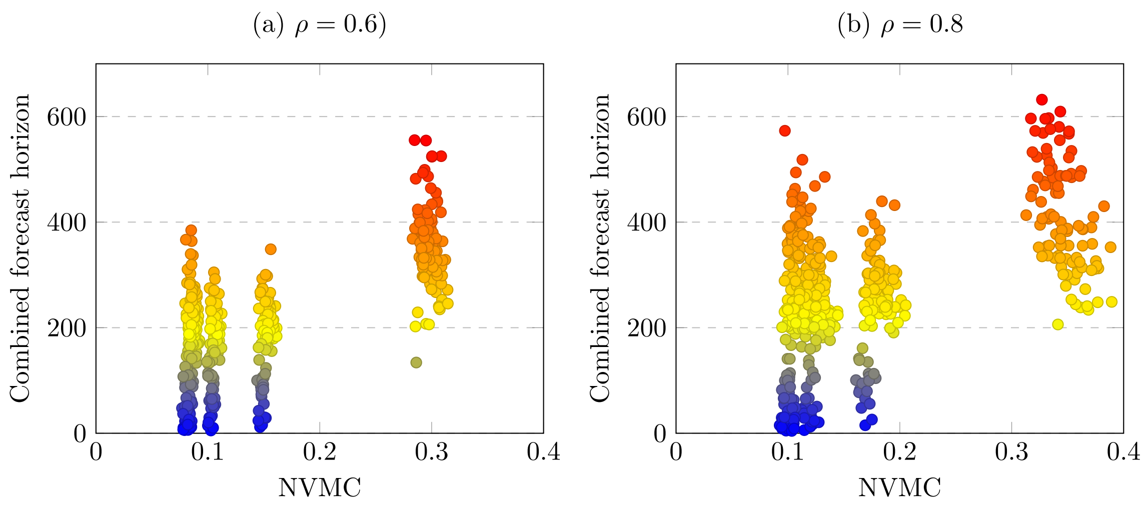

Figure 5.4: Combined forecast horizon together with the NVMC of each ESN for a) $\rho = 0.6$ and b) $\rho = 0.8$.

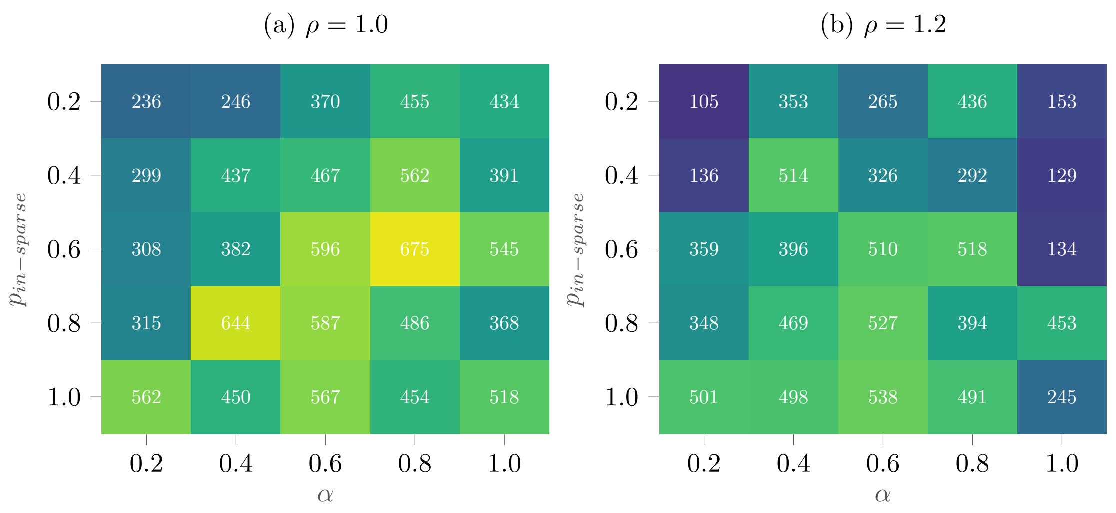

Figure 5.5: Similar to figure 5.2. In a) the spectral radius is 1.0 while in b) the spectral radius is 1.2. again every tile shows the averaged combined forecast horizon of 20 ESNs. Apart from the hyperparameters the ESNs are exactly the same for every tile.

ensemble with an average combined forecast horizon of 675 is found for $\rho = 1$ with $\alpha = 0.8$ and $p_{in-sparse} = 0.6$. Still, it is not possible to draw any conclusions about the influence of memory on the short term prediction capabilities of ESNs just from these plots. Instead, it is again required to look at the individual ESNs.

Figure 5.6 a) and b) plot the STM together with the combined forecast horizon, once for $\rho = 1.0$ and once for $\rho = 1.2$. In the first case when $\rho = 1.0$, the points of the scatter plot form a rectangle with no relation between the STM and the combined forecast horizon. In the second case when $\rho = 1.2$ there is some structure. The best performing ESNs are located left to the middle with an STM value of about six. But for every measured STM, there is still a wide range of possible combined forecast horizons.

Figures 5.7 a) and b) show the NVMC together with the combined forecast horizons. Again for $\rho = 1.0$ shown in figure 5.7 a), there is no relation between the NVMC and the combined forecast horizon. Figure 5.7 b) is the most interesting so far. There is a very strong influence of the NVMC on the prediction performance. But defying all expectations, a greater NVMC leads clearly to a worse combined forecast horizon.

Combining the results with the results from the previous section allows only one conclusion: Memory is in not relevant for the prediction performance. In fact when comparing figure 5.8,

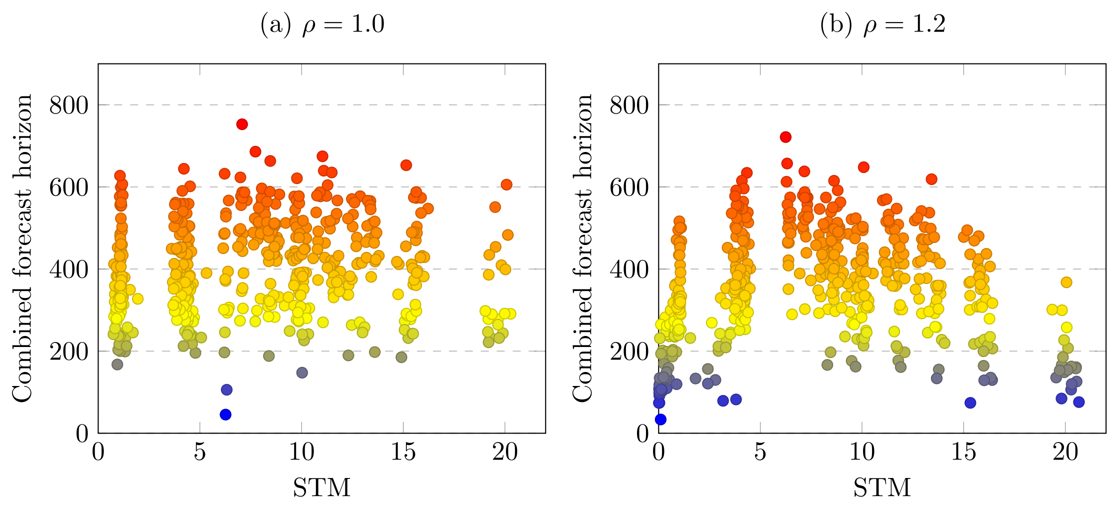

Figure 5.6: Combined forecast horizon of each ESN together with the STM. In a) for $\rho = 1.0$ and in b) for $\rho = 1.2$.

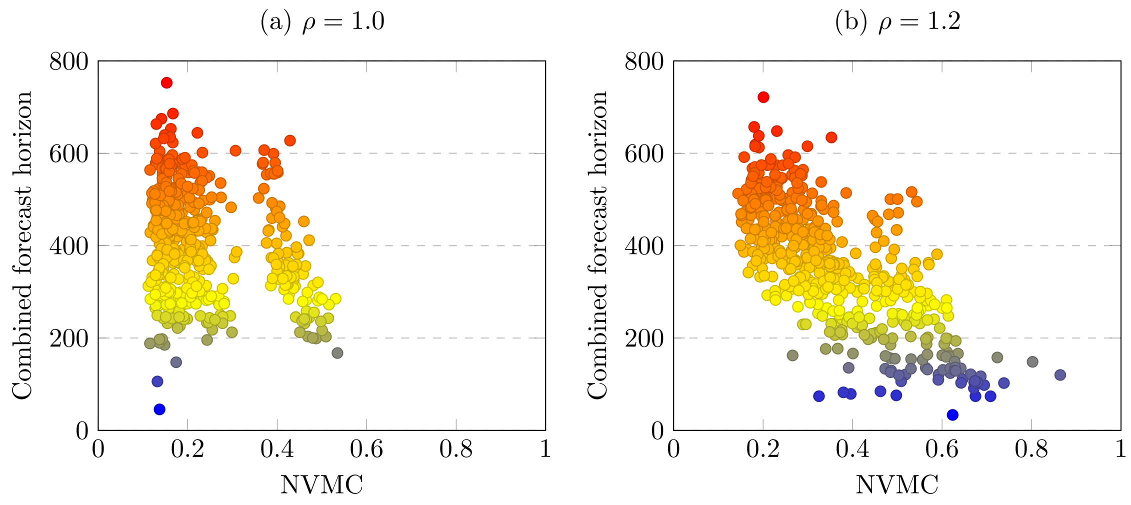

Figure 5.7: Combined forecast horizon of each ESN together with the NVMC. In a) for $\rho = 1.0$ and in b) for $\rho = 1.2$.

which shows the relation between the α hyperparameter and the combined forecast horizon, with the results so far, it is obvious that for $\rho \leq 1.0$, all changes in prediction performance can be explained by changes in α. When $\rho = 1.2$ the situation is not so obvious. But figure 5.7 b) proves that a higher NVMC is not always better. In this case, the ESNs with the highest NVMC are ESNs with growing NVMC. Figure 5.8 d) shows that indeed discrete ESNs perform worse but the pest performing ESNs are not continuous ESNs with $\alpha = 0.2$, instead they are somewhere inbetween.

5.5 Memory and Attractor Separation

Not just the influence of memory on the prediction performance is interesting, but also its influence on other quantities can give further insights into ESNs. For a successful prediction, the embeddings of the attractor in the reservoir state space need to be separated. The minimal distance of these embeddings is therefore an important quantity. Since T_{train} is set to 40,000, it is not feasible to compute the minimal embedded attractor distance by comparing every reservoir states encountered during training with the Lorenz attractor with every reservoir state encountered during training with the Halvorsen attrator. Instead, the sampled minimal embedded attractor distance is used where only every fourth reservoir state is considered. For an ESN, it should be easier to differentiate between two attractors when they are further apart. The following hypothesis is proposed: Since

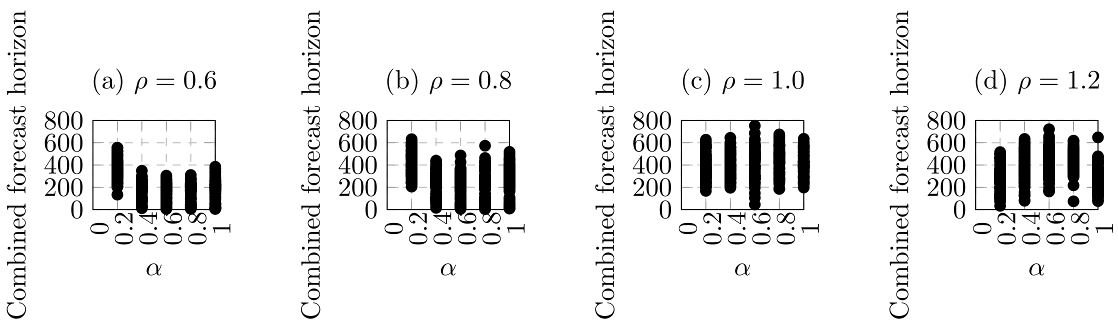

Figure 5.8: The combined forecast horizon together with α of individual ESNs. The relation between α and the combined forecast horizon can help to explain the relation between memory and the combind forecast horizon.

memory helps to distinguish between different trajectories, a higher memory should lead to greater attractor separation.

The relation between the STM and attractor separation is shown in figure 5.9. Figure 5.9 a) and b) show the results for $\rho = 0.6$ and $\rho = 0.8$. In dispute to the hypothesis, the minimal embedded attractor distance shrinks with increasing STM. This is still valid for $\rho = 1.0$, shown in figure 5.9 c), although the relation is not so strong anymore. For $\rho = 1.2$, shown in figure 5.9 d), the plots becomes chaotic and no influence of the NVMC on the minimal attractor distance is recognizeable.

The NVMC has one advantage compared to the STM. It shares all the components of an ESN used during training. Since the minimal embedded attractor distance is measured during training, it is plausible that the NVMC has a stronger impact on the minimal embedded attractor distance and might align better with the hypothesis formulated above. Figures 5.10 a) and b) and c) show that the minimal embedded attractor distance indeed grows with the NVMC. This is in agreement with the hypothesis formulated above. For $\rho = 1.2$, shown in figure 5.10 d), the plot becomes chaotic, but the same trend although much weaker can still be identified.

Especially the fact, that the STM and the NVMC have opposite relations with the minimal embedded attractor distance shows that the root cause for the observed data, is not the NVMC or the STM but again the continuousness/discreteness of the ESNs. For continuous ESNs and $\rho \leq 1.0$, the attractor distance is higher explaining why the NVMC has a positive effect while the STM has a negative effect. But the data also shows that the minimal embedded attractor distance grows with the spectral radius. So memory has a positive impact.

5.6 Memory and Training Error

Before discussing the plots which show the relation between the STM and the training error, it should be noted that there is a striking similarity between these two quantities. The training error measures the sum of absolute deviations between predicted and true states. The STM measures the sum of squared correlations between 'predicted'/remembered and true states. Due to this similarity is expected that higher training errors correspond to lower STM values.

Simulations support this hypothesis at least partially. Figures 5.11 a) and b) show the training error and the STM for ESNs with $\rho = 0.6$ and $\rho = 0.8$ respectively. Both plots are pretty similar. Most ESNs have training errors very close to zero. A few ESNs with higher training errors exist only for STMS up to twelve. So far this only weak support for the previously formulated hypothesis. For $\rho = 1.0$ something remarkable happens. As shown in figure 5.11 c), all the ESNs with high training erros are on the very left side of the plot, confirming the hypothesis. But unfortunatley it breaks down when the spectral radius is set to $\rho = 1.2$. The plot becomes pretty chaotic and while there are still some ESNs with high training error and STM close to zero, most ESNs with high training errors are located in the right side of the plot. It is also noteable that the training error increased by two orders of magnitudes and reaches now values of more than 40.

Figure 5.12 a) and b) plot the NVMC together with the training error of individual ESNs for $\rho = 0.6$ and $\rho = 0.8$ respectively. In both plots, ESNs with the highest training error have a low NVMC value. This is now the first time the NVMC and the STM have a similar influence on

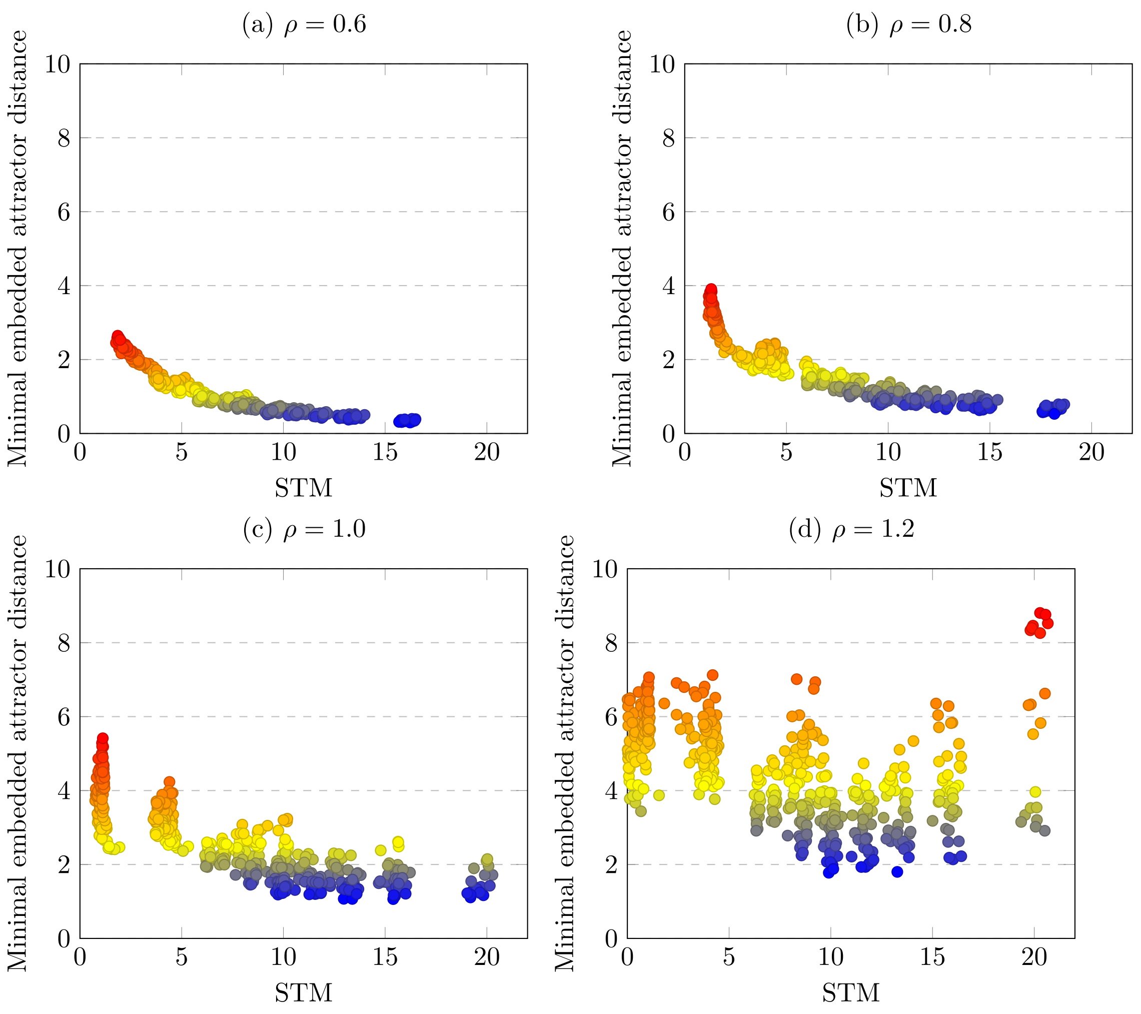

Figure 5.9: Relation between the STM and the embedded attractor distance.

another measured quantity. It directly follows that the oberservations cannot be explained by the continuousness/discreteness of the ESNs. Again, there is a strong change in the relation when the spectral radius is increased to one and above. For $\rho = 1.0$ and $\rho = 1.2$, shown in figure 5.12 c) and d), increasing the NVMC leads to an increase in the training error.

It is worth it to further discuss the case when $\rho = 1.0$. With this spectral radius, the NVMC cannot grow and shows the typical exponential decay for any input signal except a constant zero. It is therefore expected, that plots relating the NVMC to other quantities, are similar to plots where $\rho < 1.0$. In previous simulations, this is the case. But here, these ESNs behave more similar to ESNs with $\rho > 1$. A possible explanation is that figure 5.10 shows the trade-off between linear memory and input processing. The existance of such a trade-off was proposed by Dambre et al. [21]. For small deviations the NVMC is linear and better input processing capability reduce the training error.

5.7 Conclusion

Chapter 4 showed that memory is a to abstract concept to be meaningful. Although both, the STM and the NVMC match with our intuitive understanding of memory, they are not compatible. Ranking different ESNs based on memory leads to different orders depending on the memory measure in use. The only hyperparameter which has the same effect on both memory measures is the spectral radius. For other hyperparameters, the effect is either opposed or changed only one of the memory measures. But the connection between the spectral radius and the memory of an ESN is only well understood as long as the spectral radius is less or equal to one. For a spectral radius above one, chapter 4 proofs that the input signal has a major influence if the Echo

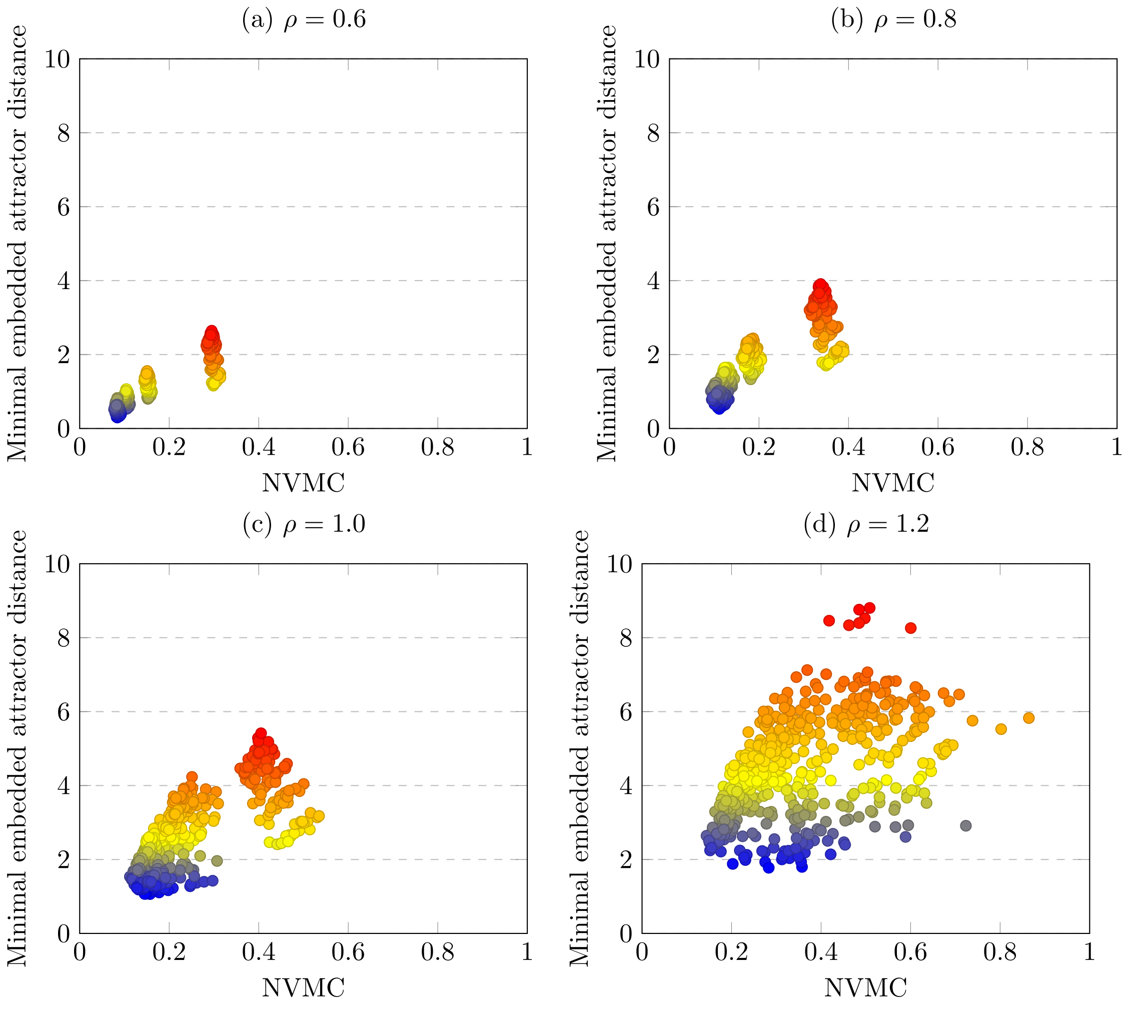

Figure 5.10: Relation between the NVMC and the minimal embedded attractor distance.

state property still holds. But even for the same input signal and a spectral radius of $\rho = 1.2$ experiments find ESNs with and without the Echo state property. When interpreting the results in terms of a memory capacity, growing deviations and infinite memory are problematic. Also the fact that the NVMC does not increase with the size of the ESN is counterintuitive. A big advantage of the NVMC is the fact that analytical results can be obtained. Despite their simple mathematical formulation, ESNs are black boxes and the can NVMC help deepen the mathematical understanding. Contrarily to the NVMC, the STM is limited and grows with the size of the ESN. A drawback of the STM is the independence of the input signal. A temporary input matrix needs to be created changing the measured the ESN.

Although the simulations used in chapter 5 require memory by construction, the results clearly show that neither the NVMC nor the STM have a consistent impact on the short term prediction performance. Even though in some setups ESNs with a higher NVMC perform better, a correct interpretation of the results finds the root cause in the continuousness of the ESN. Also the relation between memory and the minimal embedded attractor distance can be explained by the continuousness/discreteness of the ESNs. Only for $\rho > 1.0$ these explanations break down. The influence of the STM on the training error is explained by their similarity. Relations between the NVMC and the training error could be partially explained by the trade-off between memory and input processing.

Regarding overlapping attractors this chapter showed that short-term predictions are possible. For long-term predictions the stability of the prediction needs to be increased. Stability is already mentioned in section 2.3 as one of the four fundamental requirements for a successful prediction. Unfortunatly stability is also the most difficult requirement to guarantee. Overall, the findings of this chapter can be summed up in the following two sentences: When building an ESN for a

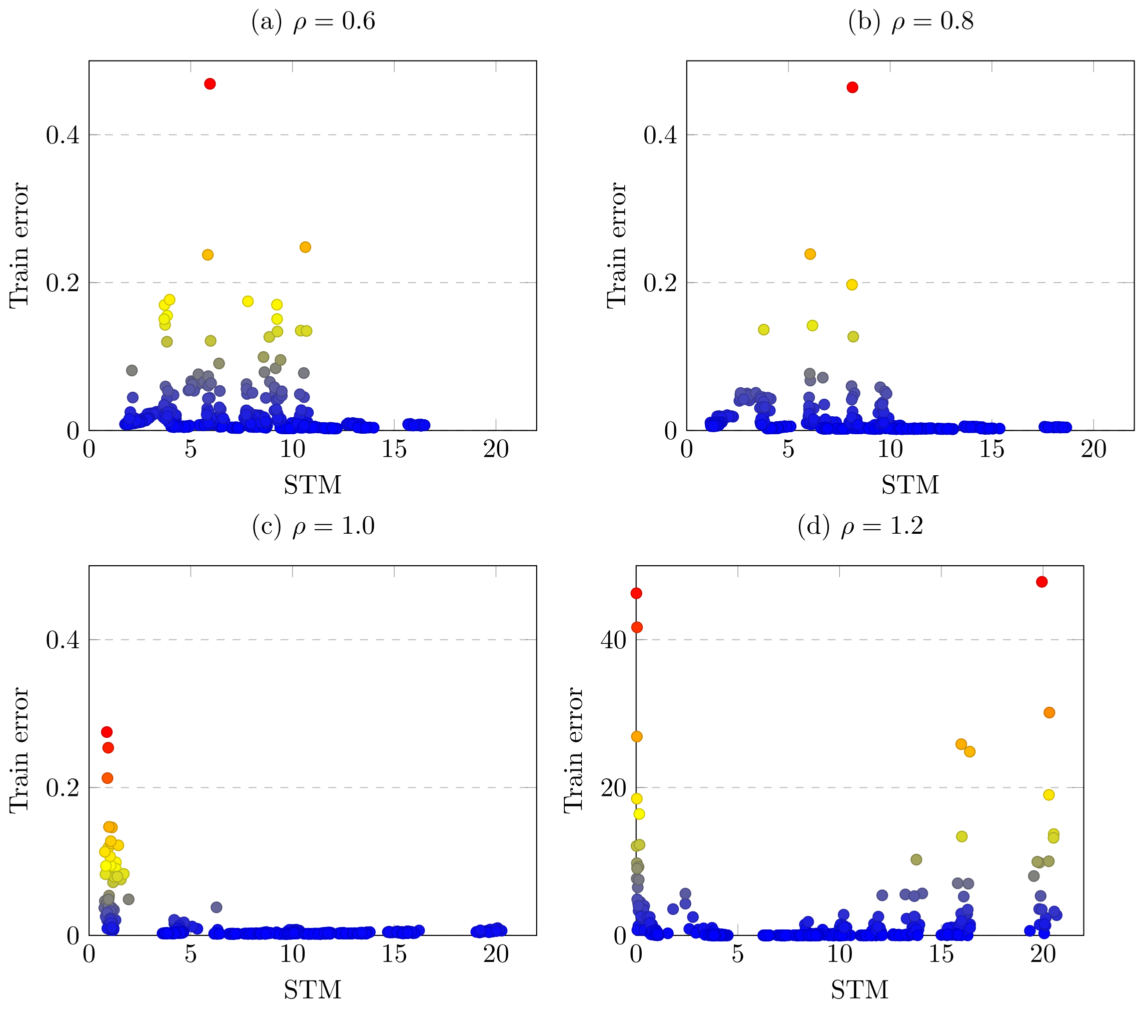

Figure 5.11: Relation between the STM and training error.

specific task, memory should only be part of the consideration through the spectral radius, not by targeting a specific value for the NVMC or the STM. Instead, a stronger focus should be set on the continuousness/discreteness of the ESN.

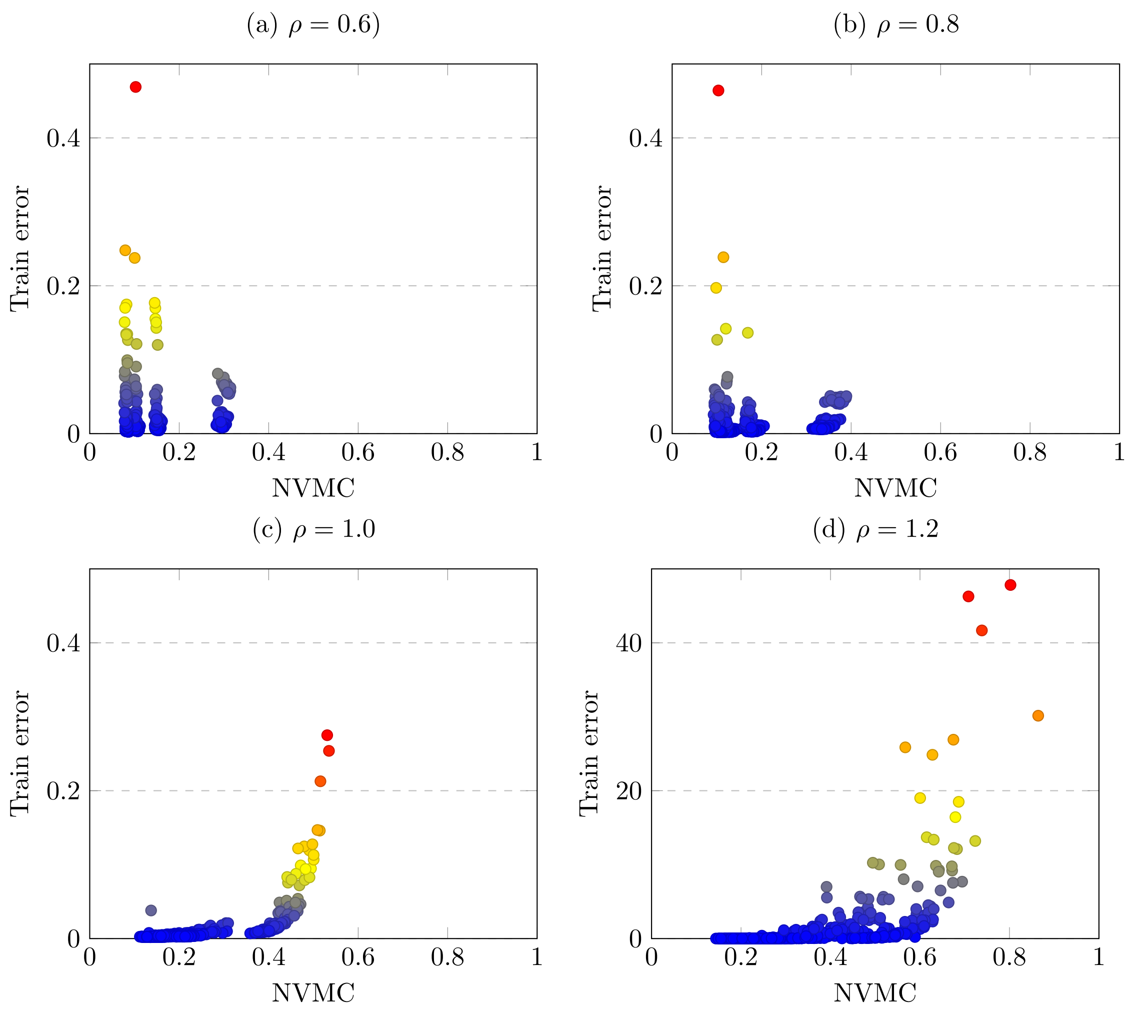

Figure 5.12: Relation between the NVMC and the training error.

Bibliography

[1] J. Lanchester, *How to speak money.* Faber & Faber Ltd, 2014.

[2] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, vol. 14, pp. 2531–60, 12 2002.

[3] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report*, vol. 148, 01 2001.

[4] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural networks*, vol. 20, pp. 391–403, 05 2007.

[5] E. N. Lorenz, "Deterministic nonperiodic flow," *Journal of Atmospheric Sciences*, vol. 20, no. 2, pp. 130–141, 1963.

[6] J. Herteux and C. Räth, "Breaking symmetries of the reservoir equations in echo state networks," *Chaos*, vol. 30, p. 123142, 12 2020.

[7] A. Flynn, V. A. Tsachouridis, and A. Amann, "Multifunctionality in a reservoir computer," *Chaos*, vol. 31, no. 1, p. 013125, 2021.

[8] K. Briggman and W. Kristan, "Multifunctional pattern-generating circuits," *Annual review of neuroscience*, vol. 31, pp. 271–94, 08 2008.

[9] H. Jaeger, "Short term memory in echo state networks," 01 2002.

[10] M. Inubushi and K. Yoshimura, "Reservoir computing beyond memory-nonlinearity trade-off," *Scientific Reports*, vol. 7, 12 2017.

[11] S. Mallat, "Understanding deep convolutional networks," *Philosophical Transactions of the Royal Society A*, vol. 374, 01 2016.

[12] A. Flynn, J. Herteux, V. Tsachouridis, C. Räth, and A. Amann, "Symmetry kills the square in a multifunctional reservoir computer," *Chaos*, vol. 31, p. 073122, 07 2021.

[13] H. Jaeger, M. Lukosevicius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural networks*, vol. 20, pp. 335–52, 05 2007.

[14] A. Flynn, O. Heilmann, D. Köglmayr, V. Tsachouridis, C. Räth, and A. Amann, "Exploring the limits of multifunctionality across different reservoir computers," pp. 1–8, 07 2022.

[15] N. Rulkov, M. Sushchik, L. Tsimring, and H. Abarbanel, "Generalized synchronization of chaos in directionally coupled chaotic systems," *Physical review. E*, vol. 51, pp. 980–994, 03 1995.

[16] Z. Lu, B. Hunt, and E. Ott, "Attractor reconstruction by machine learning," *Chaos*, vol. 28, 06 2018.

[17] Z. Lu and D. Bassett, "Invertible generalized synchronization: A putative mechanism for implicit learning in neural systems," *Chaos*, vol. 30, p. 063133, 06 2020.

[18] W. Kristan, G. Wittenberg, M. Nusbaum, and W. Stern-Tomlinson, “Multifunctional interneurons in behavioral circuits of the medicinal leech,” *Experientia*, vol. 44, pp. 383–9, 06 1988.

[19] S. Krishnagopal, M. Girvan, E. Ott, and B. Hunt, “Separation of chaotic signals by reservoir computing,” *Chaos*, vol. 30, p. 023123, 02 2020.

[20] T. Carroll, “Path length statistics in reservoir computers,” *Chaos*, vol. 30, p. 083130, 08 2020.

[21] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, “Information processing capacity of dynamical systems,” *Scientific reports*, vol. 2, p. 514, 07 2012.

[22] H. Kantz, “A robust method to estimate the maximal lyapunov exponent of a time series,” *Physics Letters A*, vol. 185, no. 1, pp. 77–87, 1994.

[23] P. Grassberger and I. Procaccia, “Characterization of strange attractors,” *Phys. Rev. Lett.*, vol. 50, pp. 346–349, Jan 1983.

[24] A. Hart, J. Hook, and J. Dawes, “Embedding and approximation theorems for echo state networks,” *Neural Networks*, vol. 128, pp. 234–247, 2020.

[25] M. Rescorla, *From Ockham to Turing - and Back Again*, pp. 279–304. Cham: Springer International Publishing, 2017.

[26] J. Hopfield, “Neural networks and physical systems with emergent collective computational abilities,” *Proceedings of the National Academy of Sciences of the United States of America*, vol. 79, pp. 2554–8, 05 1982.

[27] R. Albert and A.-L. Barabási, “Statistical mechanics of complex networks,” *Rev. Mod. Phys.*, vol. 74, pp. 47–97, Jan 2002.

[28] A. Haluszczynski and C. Räth, “Good and bad predictions: Assessing and improving the replication of chaotic attractors by means of reservoir computing,” *Chaos*, vol. 29, no. 10, p. 103143, 2019.

[29] M. van den Heuvel, C. Stam, M. Boersma, and H. Hulshoff Pol, “Small-world and scale-free organization of voxel-based resting-state functional connectivity in the human brain,” *NeuroImage*, vol. 43, no. 3, pp. 528–539, 2008.

[30] S. S. Singh, B. Khundrakpam, A. Reid, J. Lewis, A. Evans, R. Ishrat, I. Sharma, and R. Singh, “Scaling in topological properties of brain networks,” *Scientific Reports*, vol. 6, p. 24926, 04 2016.

[31] T. Carroll, “Optimizing memory in reservoir computers,” *Chaos*, vol. 32, p. 023123, 02 2022.

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.