DEPARTMENT OF PHYSICS

LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Master's Thesis

Exploring Multifunctionality in Tree-based Reservoir Computing

Untersuchung von Multifunktionalität in Baum-basierten Reservoir Computing

Author: Miralem Spahic
Supervisor: Dr. Christoph Räth
Submission Date: 02.05.2025

Contents

1. Introduction

Forecasting future events has been a human desire since ancient times. The philosopher Aristotle wrote the book *Meteorologica*, the oldest formal study on meteorology [1], in which he gathered his observations on different weather phenomenons and constructed theories on their formation. Predicting the future of weather has been based on such empirical observations and philosophic ideas for a long time. With the rise of new technologies that collected large amounts of data and statistical methods like machine learning, it was possible to analyse complex dynamical systems more rigorously. In [2], Takens showed that the state of a dynamical system can be reconstructed from partially observed data. Machine learning methods such as recurrent neural networks (RNN) were developed to learn sequential and time dependent data [3]. They consist of connected nodes in a feedback loop and are utilized as models of brain activity in neuroscience [4, 5]. RNNs use a gradient based approach, but because of their sequential architecture this turned out be a disadvantage for learning long term dependencies [6]. To mitigate this problem methods based on RNNs have been developed from the machine learning field [7] and from computational neuroscience [8], which have been unified under the name "Reservoir Computer" (RC) [9]. RCs have successfully been used in many tasks [10, 11], most notably for reconstructing chaotic attractors described by nonlinear dynamical systems [12]. An extension of this is learning multiple attractors with one RC [13, 14], referred to as multifunctionality. Other architectures have built upon the framework of reservoir computing and developed different methods to enhance performance, such as minimal RC [15] or Hybrid RC [16] as well as physical realizations [17]. This thesis uses the Next-Generation RC (NGRC) framework [18], which eliminates randomness introduced in traditional RC , requires fewer parameters to tune and shows comparable performance with less training data.

Training a reservoir computer is typically done using regularized linear regression in the output layer and requires optimizing the regularization parameter. In multifunctional setups with multiple attractors overlapping in space, this turns out be a hard task and is crucially dependent on parameters of the RC [19]. In this thesis, a different approach based on tree ensemble as regressors will be explored. A machine learning method called Extremely Randomized Trees (ERT) [20, 21] is used for training instead, which is an ensemble method that aggregates many trees for prediction. The advantage of the ERT algorithm is that it is robust with respect to its hyperparameters in

regression problems as well as being computationally efficient. An approach to learn chaotic dynamical systems using the ERT algorithm has been done in [22]. Another method extends the NGRC algortihm by employing piecewise-polynomial regression trees [23]. The focus in this thesis is on combining NGRC with the ERT regressor to learn multiple attractors and achieving multifunctionality.
In Chapter 2 the NGRC architecture and the algorithm will be introduced, as well as measures to characterize dynamical systems exhibiting chaos. In Chapter 3 different setups of a multifunctional NGRC with the ERT as regressor will be examined.

2. Methods

2.1. Dynamical Systems

Mathematically a dynamical system of dimension d is governed by a set of differential equations:

$$\frac{dx_1}{dt} = f_1(x_1, x_2, \ldots, x_d) \tag{2.1}$$

$$\frac{dx_2}{dt} = f_2(x_1, x_2, \ldots, x_d) \tag{2.2}$$

$$\frac{dx_d}{dt} = f_d(x_1, x_2, \ldots, x_d), \tag{2.3}$$

with $\boldsymbol{x} = (x_1, x_2, \ldots, x_d) \in \mathbb{R}^d$. The functions f_i may additionally several time independent parameters that allow for different dynamical behaviour for the same set of differential equations. Given an initial point $(x)(t_0)$, a solution to the differential equations is called a trajectory. A trajectory could either diverge or be bounded in phase space, the space of all possible dynamical states, where the bounded region is called an attractor. By the well known Poincaré–Bendixson theorem for continuous dynamical systems two trajectories can not cross each other [24]. Consequently, a trajectory either reaches a fix point, follows a periodic orbit or is attracted to a region with aperiodic motion. In the third case, if two trajectories start infinitesimally close to each other but diverge exponentially fast while being bounded, the system exhibits chaotic behaviour and is called a strange attractor [25]. Even though the system dynamics are deterministic and bounded, it is sensitive to initial conditions and therefore hard to predict without the exact initial conditions. The functions $f_i(\vec{x})$ that describe chaotic systems typically have at least one nonlinear term, which makes them difficult or even impossible to solve analytically. The prime example of a strange attractor is the Lorenz attractor [26], seen in Fig.2.1, which is given by the following

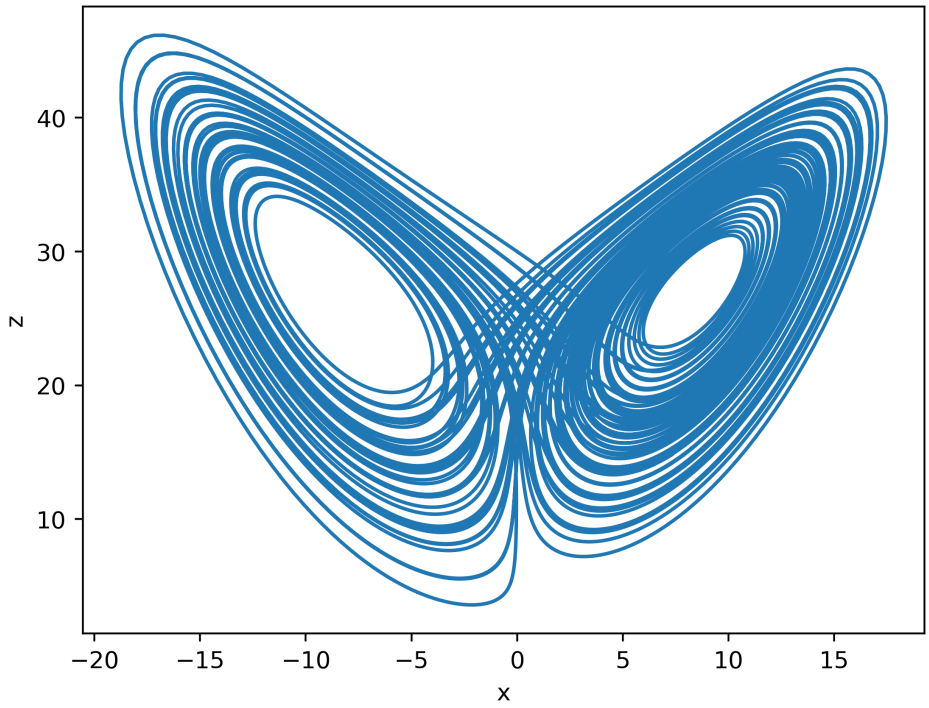


Figure 2.1.: xz view of the Lorenz attractor

equations:

$$\frac{dz}{dt} = \sigma(y - x) \tag{2.4}$$

$$\frac{dy}{dt} = x(\rho - z) - y \tag{2.5}$$

$$\frac{dz}{dt} = xy - \beta z \tag{2.6}$$

As no analytical solution exist, the equations of motion can numerically be integrated using the Runge-Kutta 4 (RK4) method. For every system considered in this thesis the RK4 method is used to generate a trajectory.
Depending on the specific values of the parameter set (σ, ρ, β) the Lorenz system undergoes different types of dynamical patterns. Geometrically strange attractors are reminiscent of fractals, showing self similar structure[27]. Even without knowing the full solutions of the differential equations the statistical climate of a strange attractor can be evaluated with the Lyapunov exponent and the correlation dimension which will be introduced in section 2.3.2 and 2.3.3, respectively. The chaotic systems used for training are Lorenz [26], Halvorsen [28], Rucklidge [29], Roessler [30], Chua [31], Chen [32], Windmi [33] and Compelx Butterfly [34] attractors.

2.2. NGRC

A resevoir computer consists of three layers [35]: The input layer, the reservoir and the output layer. The input layer is connected to the reservoir by fixed random weights. The reservoir is a network consisting of nodes with randomly chosen connections. The reservoir embeds the input into a high dimensional space the reservoir state is driven by the input signal. An adjustable weight matrix connects the reservoir to the output layer. Training these weights is done using ordinary least squares with Thikinov regularization. After the training phase the reservoir acts as an autonomous dynamical system that can used for prediction.
It was shown that, under some mild conditions, an RC is mathematically equivalent to a nonlinear vector autoregression model (NVAR)[18], as well as being a universal approximator[36]. On this basis a new RC framework, called Next-Generation Reservoir Computing (NGRC)[37]. Instead of a reservoir with interconnected nodes, NGRC uses past data points as well as monomials of them to construct the linear and nonlinear features. To map the input to the feature space two operators are necessary, L_s^k and $P^{\mathcal{D}(p)}$. The first operator has two (hyper-)parameters, k and s. For a data point $\boldsymbol{x}_t$ in an observed time series $\boldsymbol{u}(t)$, L_s^k maps the current and $(k-1)$ past observations, with a spacing value s between sequential steps:

$$L_s^k(\boldsymbol{x}_t) = \boldsymbol{x}_t \oplus \boldsymbol{x}_{t-s} \oplus \cdots \oplus \boldsymbol{x}_{t-(k-1)s} \tag{2.7}$$

where $\oplus$ is a concatenation operation. This is the linear part of the feature vector. The nonlinear part consists of creating monomials up to order p out of the linear part:

$$P^{\mathcal{D}(p)}(L_s^k(\boldsymbol{x}_t)) = \underbrace{L_s^k(\boldsymbol{x}_t)\lceil\otimes\rceil \ldots \lceil\otimes\rceil L_s^k(\boldsymbol{x}_t)}_{p \text{ times}} \tag{2.8}$$

The operator $\lceil\otimes\rceil$ acts in two ways. First it builds the outer product of the linear operators, resulting in a symmetric tensor and then collects the unique monomials and concatenates them into a vector. A dictionary $\mathcal{D}(p)$ specifies which powers are computed for a feature.
As an example, for one dimensional time series with an observation x_t, setting $k = 2, s = 1$ and $\mathcal{D}(p) = [1, 2]$, this results in:

$$L_1^2(x_t) = (x_t, x_{t-1})$$
$$P^{\mathcal{D}(2)}(L_1^2(x_t)) = \begin{bmatrix} x_t x_t \\ x_t x_{t-1} \\ x_{t-1} x_{t-1} \end{bmatrix}$$

In general the feature vector is then given by $S = c \oplus P^{\mathcal{D}(p)}$, with a bias term c. The bias term is set to $c = 0$. If instead of a single time series the NGRC takes N

different ones corresponding to separate dynamical systems the feature vector turns into a feature matrix $X = [S_1, S_2, \ldots, S_N]$. Additionally, a set of unique parameters $\{\theta_1, \theta_2, \ldots, \theta_N\}$ is introduced, where each θ is associated to a dynamical system. The attractor parameter is scaled by a scaling parameter γ and added to each feature vector. This was originally introduced for extrapolating tipping points in nonlinear dynamical systems in [38] but serves a different purpose later on. The final feature vector $\tilde{S}_i$ for time series $\boldsymbol{u}_i(t)$ is:

$$\tilde{S}_i = S_i + \gamma\theta_i \tag{2.9}$$

For the training in the output layer they are concatenated into a single feature matrix $\mathcal{S} = [\tilde{S}_1, \ldots, \tilde{S}_N]$ as well as their respective target matrix $\tilde{y} = [\Delta\boldsymbol{u}_1, \ldots, \Delta\boldsymbol{y}_N]$, where $\Delta\boldsymbol{u_i}(t_i) = y_{t_{i+1}} - y_{t_i}$. This setup is similar to [39], where it is utilized for controlling dynamical systems into different target states. The NGRC can be stated as a one-step ahead integrator, in the form of

$$\boldsymbol{y}_{t+1} = \boldsymbol{y}_t + \mathcal{R}(\tilde{S}_t) \tag{2.10}$$

where $\mathcal{R}$ is the regressor, which process a feature vector and predicts the difference between the next state and current state in time. Thus an optimized regressor learns the flow of a dynamical system. Depending on the choice of time delays k , spacing parameter s and the dictionary $\mathcal{D}(p)$, three NGRC types can be defined: If the feature vector f consists of only k time delayed observations with spacing s, $p = 1$, then f is linear and this is referred to as a vector autoregression model (VAR) that has a similar form as in [18]:

$$x_{t+1} = w_1 x_t + w_2 x_{t-s} + \cdots + w_k x_{t-(k-1)s} = \boldsymbol{W} L_s^k(x_t) \tag{2.11}$$

with weights $\boldsymbol{W} = (w_1, w_2, \ldots, w_k)^T$ and $L_s^k(x_t) = (x_t, x_{t-s}, \ldots, x_{t-(k-1)s})$ as defined above. If higher order monomials with $p > 1$ are included then f contains additionally nonlinear terms. This is the classical NGRC configuration and where the relation to nonlinear vector autoregression comes from. To keep the general NGRC architecture from a realization with these parameter configurations, whenever features consist of time delays and their respective higher order terms, we will refer to this as NVAR. The last possibility is when f is made up of only nonlinear terms and no time delays. Predicting the next step in a time series depends then solely on monomials of the current step. Accordingly, this will be referred to as a nonlinear memoryless model (NML), in the sense that the history of past states besides the current one do not influence the next state and the feature vector is made up mainly of high order monomials. Then Eq. 2.11 can be more generally stated as Eq. 2.10 where the weight matrix is replaced with a regressor function that predicts the flow instead of the next

VAR	NVAR	NML
$k \in \mathbb{N}$	$k > 1 \in \mathbb{N}$	$k = 1$
$s \in \mathbb{N}$	$s \in \mathbb{N}$	$s \in \mathbb{N}$
$p = 1$	$p \geq 1$	$p > 1$

Table 2.1.: Different NGRC parameter configurations lead to different NGRC models

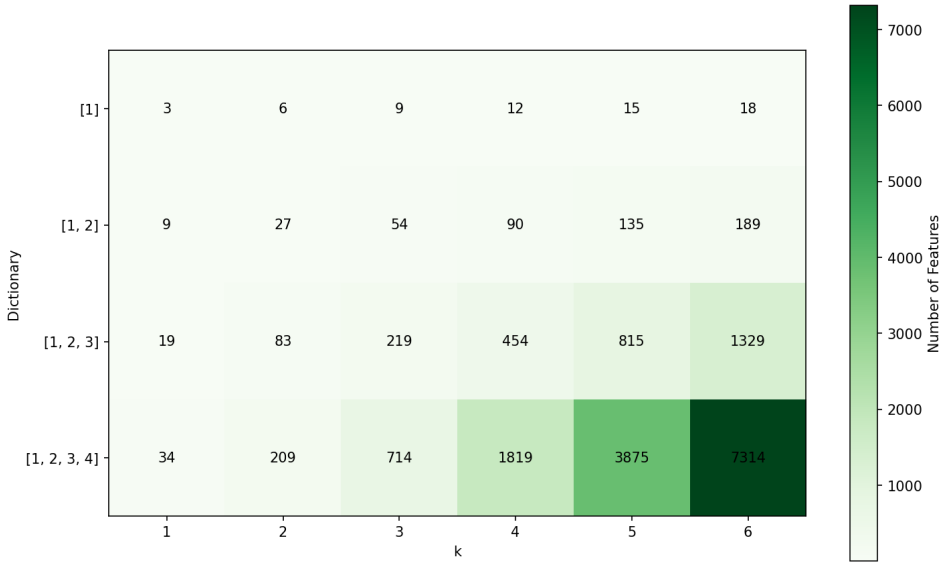


Figure 2.2.: Size of feature dimension for different k valuesa and dictionaries $\mathcal{D}(p)$

state. The parameter configuration of (k, s, p) that defines these models can be looked up in table 2.1. The dimension of the feature space is given as:

$$\dim \mathcal{F} = \sum_{p \in \mathcal{D}(p)} \frac{(dk + p - 1)!}{p!(dk - 1)!} \tag{2.12}$$

with d as the dimension of the dynamical system. This is a sum over the formula for combinations with replacement, as each linear feature of which there are dk is multiplied with itself and any other to compute monomials of order p. The number of features for the VAR model increases linearly with k. For $k > 1$ and $p >> 1$ the feature dimension grows rapidly. With no time delays $k = 1, p >> 1$ the increase is minimal, as seen in Fig. 2.2.

The standard choice of the regressor $\mathcal{R}$ in any RC setup is typically the ridge regressor, which is defined below. Ridge regression is a linear regression model with regularization which can be computed efficiently, requiring no derivates. As a linear model it makes an assumption about the relationship between feature vector and

the target. Non-parametric machine learning models do not assume any functional form, making them more flexible. The ERT algorithm is such a non-parametric model. Ridge regression is computationally efficient and in part interpretable, but training the output layer with other regression techniques could be advantageous in some cases. Hence, a tree based approach to regression will be examined in this thesis.

2.2.1. Ridge Regression

Assuming a linear relationship $\boldsymbol{y} = \boldsymbol{W}_{out}\boldsymbol{X}$, where $\boldsymbol{X}$ is a feature matrix and $\boldsymbol{W}$ the weights, the solution can be calculated analytically by using linear regression with regularization, called ridge regression. $\boldsymbol{W}_{out}$ is solved by:

$$\boldsymbol{W}_{out} = \arg\min_{W}[(\boldsymbol{y} - \boldsymbol{W}\boldsymbol{X})^T(\boldsymbol{y} - \boldsymbol{W}\boldsymbol{X}) + \alpha\boldsymbol{W}^T\boldsymbol{W}] \tag{2.13}$$

where α is the regularization parameter which needs to be optimized. A closed form solution of the above expression exists:

$$\boldsymbol{W_{out}} = (\boldsymbol{X}^T\boldsymbol{X} + \alpha\mathbf{1})^{-1}\boldsymbol{X}^T\boldsymbol{y} \tag{2.14}$$

2.2.2. Tree Learning

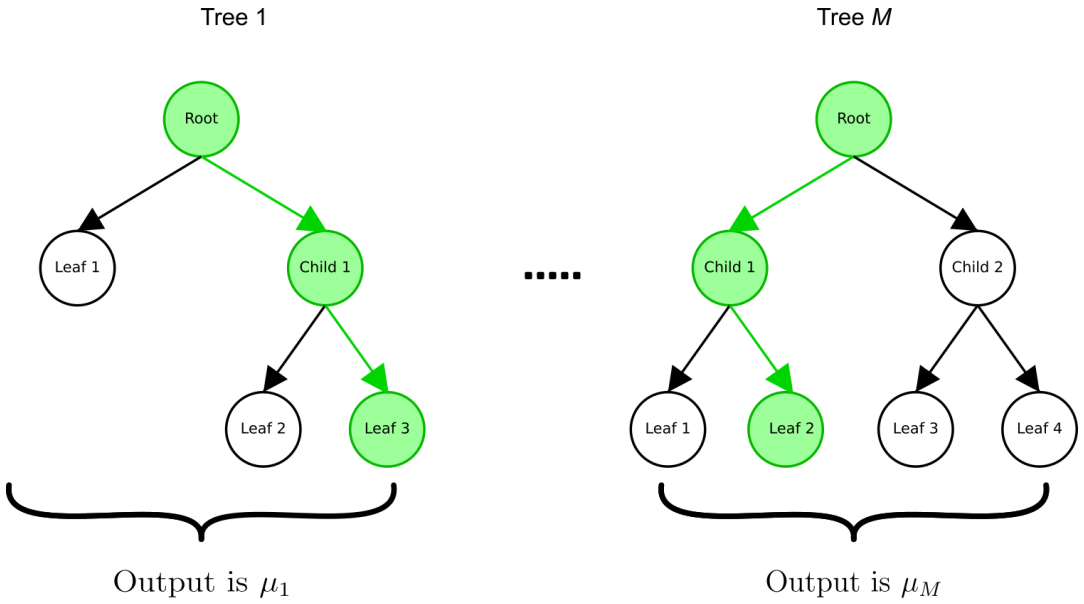
Given a dataset $D(\boldsymbol{X}, \boldsymbol{y})$ with N samples, with feature matrix $\boldsymbol{X} \in \mathbb{R}^{(N,d_f)}$ containing the feature vectors f with dimension d_f and the target matrix $\boldsymbol{y} \in \mathbb{R}^{N,d}$, tree based methods recursively partition the data in an $if - else$ fashion. Starting from a root node, the whole dataset is split according to predefined splitting rules into two subsets D_{left} and D_{right}, which again get split until some terminal condition is met. The terminal nodes, called leaf nodes or just leaves contain those target samples $\boldsymbol{y}$ that remain in the data subsets. For each tree t a simple model of the following form is fitted:

$$f_t(X) = \sum_i \mu_i I(X \in R_i) \tag{2.15}$$

where R_i is the set containing the data points in leaf i and μ_i is a constant. The number of leaves depends on the splitting criterion. $I(\cdot)$ is the step function:

$$I_A(x) = \begin{cases} 0, x \notin A \\ 1, x \in A \end{cases} \tag{2.16}$$

The splitting rules depend on the type of tree algorithm. Since we are dealing with chaotic, continuous time series and thus a regression problem, the governing rules, such as the splitting criterion and the output of the tree, should reflect that. Usually



$$\Rightarrow \text{Ensemble prediction } f = \mu_1 + \cdots + \mu_M$$

Figure 2.3.: Sketch of an ensemble of trees. A feature sample follows a specific path down each tree and ends up in one leaf, shown in green. The full prediction is given as the average over all leaf outcomes.

trees have hard boundaries, which means that a sample has a fixed path down the tree. E.g. a sample X_{test} can end up only in one leaf, say in R_j, the output of the tree is then $f(X_{text}) = \mu_j$, see Fig. 2.3.

Many variations of decision tree algorithms exists, for an overview of different tree based machine learning methods see[40, 41].
Decision tree algorithms tend to have high variance, mainly due to the hard boundary when splitting continuos features [42]. To mitigate that several strategies exist [43]. One method is the Extremely Randomized Trees algorithm (ERT)[20], which combines multiple trees into an ensemble, where the output is averaged over all trees, thus reducing the variance of the model. A tree in an ERT ensemble is built as follows: At the root node construct K feature-threshold pairs $(f_i, a_i), i = 1, \ldots, K \leq d_f$. The feature is chosen at random from the whole set of features and the threshold is uniformly drawn in an interval $a_i \in [a_{min}, a_{max}]$, which are the smallest and largest values for the column X_i of the feature matrix, respectively. This divides the dataset into two subsets $D^{(i)}_{left} = \{(\boldsymbol{X}, \boldsymbol{y}) \mid X_i \leq a_i\}$ and $D^{(i)}_{right} = \{(\boldsymbol{X}, \boldsymbol{y}) \mid X_i > a_i\}$. A loss function such as the mean squared loss is evaluated for each feature-threshold pair:

$$L(f_i, a_i) = \sum_{\boldsymbol{X}_i \in D_{left}} (\boldsymbol{y}_i - \mu_{left})^2 + \sum_{\boldsymbol{X}_i \in D_{right}} (\boldsymbol{y}_i - \mu_{right})^2 \tag{2.17}$$

The root is split into two children nodes and the pair (f_j, a_j) for which L is minimized is used as a decision criterion for an input vector. By taking the derivative of the loss function L with respect to m the solution is given by:

$$\mu = \frac{1}{n} \sum_{i}^{n} \boldsymbol{y}_i, \tag{2.18}$$

with n samples in a subset. A particular leaf can be reached by an input vector fulfilling the decision criterion set by (f_s, a_s) at each split s. The constant μ in (2.15) is determined by (2.18). The prediction of the ensemble is then given as:

$$f(X) = \sum_{t}^{M} f_t(X)$$

2.3. Measures

To characterize how well our machine learning can learn a dynamical system, we are interested in two types of measures. First, how well the prediction compares to the true trajectory if it would be evolved beyond the train data. This is the short term behaviour of the prediction. For a chaotic dynamical system the distance between predicted and true trajectory should increase exponentially and a measure

that captures chaos is necessary. Besides quantifying chaos a way to capture the fractal structure of a strange attractor is needed as well. Therefore, two measures will be introduced to characterize the long term behaviour. The true values of these two measures for the systems considered in this thesis can be looked up in [28].

2.3.1. Forecast Horizon

Since we are dealing with solutions of dynamical systems that were computed using numerical means, one can compare the prediction with the test data and evaluate the distance between them as they evolve in time. Due to the chaotic nature of the systems at hand two initially close trajectories will eventually depart from another exponentially fast. We record how the Euclidean distance $d(t)$ between two points at any time evolve and if it exceeds some threshold, then the trajectories eventually follow different paths on the attractor. This can be stated as $|\boldsymbol{u}(t) - \boldsymbol{u}^{'}(t)| < \boldsymbol{\epsilon}$. The forecast horizon is defined as the number of time steps until this inequality holds[44]. A reasonable choice for $\boldsymbol{\epsilon}$ is to be 15% of the difference between the maximal and minimal value for each dimension, which defines a cuboid. The exact dimensions of the cuboid are not important since the distance between predicted and true trajectory outgrows the size of the cuboid exponentially.

2.3.2. Lyapunov Exponent

Chaotic systems tend to show complex, unpredictable behaviour, yet they obey a set of deterministic differential equations. When chaos occurs, it is expected that, for two initial conditions x_0 and $x_0 + \delta$, where δ is infinitesimal, the distance of two nearby solutions diverge exponentially fast. This characteristic is captured in the following equation: $d(t) = d(0)e^{\lambda t}$, where λ is known as the Lyapunov exponent. A d dimensional system can be described by d Lyapunov exponents. Most notably, the sign shows what kind of dynamics the system undergoes. If at least for one dimension the Lyapunov exponent is positive, the system is said to be chaotic. This is the reason why the largest Lyapunov exponent λ_{max} is also the most interesting if chaos is expected. If the systems differential equations are given then an analytical approach is viable. However, this is not always the case. Therefore, many algorithms exist to calculate λ_{max} from data alone. Some of these as well as the analytical calculation can be found in [45]. For our purposes the Rosenstein algorithm[46] will be used. As this is a data driven approach, it requires a time series $\boldsymbol{u}(t)$, with Δt time units between successive observations. For any random point x_t in the time series the nearest neighbour is determined. The nearest neighbour should be above some temporal threshold, for example the mean period of the trajectory. This ensures that each neighbour can be considered to be on a nearby trajectory. Successive observations

of x_t as well as of its neighbour are followed and the distance d_j for the jth pair is calculated at time $i\Delta t$. For a chaotic system the distance grows exponentially in time:

$$d_j(i) = d(0)e^{\lambda i \Delta t}$$

To calculate λ take the logarithm on both sides and reorder the equation:

$$\lambda = \frac{1}{i\Delta t}(\ln(d_j(i)) + d(0)) \tag{2.19}$$

A linear fit to the line of Eq. 2.19 gives λ as the slope of the log separation of the distances. The initial distance $d(0)$ is proportional to the intercept and therefore not important for the calculation. Since any point and its nearest neighbour of the time series is temporally separated and thus uncorrelated an average over multiple initial points is taken for better results. Eventually the diverging distance saturates to the size of the attractor, and to get rid of transients at the beginning, it is necessary to define a time interval for which the fit is computed, as shown for the Lorenz attractor Fig. 2.4. Both measures are done with a step size of 15000 time steps and $\Delta t = 0.02$. The time interval is either set by hand for attractors like the Rucklidge system and Roessler system, since their Lyapunov exponents are close to zero and a larger time window is necessary to capture their chaotic nature. Since many simulations will be done with different parameter configurations the time window is typically set to be proportional to Δt and to the attractor size, yielding close enough results to distinguish the systems and compare the prediction against the true trajectory.

2.3.3. Correlation Dimension

To capture the strangeness of an attractor and separate it from a periodic orbit with a long period a measure that assess the local structure is needed. The geometry of fractals can be described by the fractal dimension. There are many methods to estimate the fractal structure of a strange attractor, a more general overview can be found in [47]. For a given trajectory a geometrical measure that characterizes the complex shape it evolves to in phase space is the correlation dimension (CD) [48]. Two close points on the attractor will be mostly uncorrelated temporally because of the chaotic nature of the system but spatially correlated, and this can be measured by a correlation sum in the discrete version:

$$C(r) = \lim_{N\to\infty} \frac{1}{N^2} \sum_{i,j,i\neq j}^{\infty} \Theta(r - \|x_i - x_j\|)$$

where $\Theta(x)$ is the Heaviside step function and r is some threshold distance. For some interval $[r_{min}, r_{max}]$ the correlation sum follows a power law:

$$C(r) \sim r^{\nu}$$

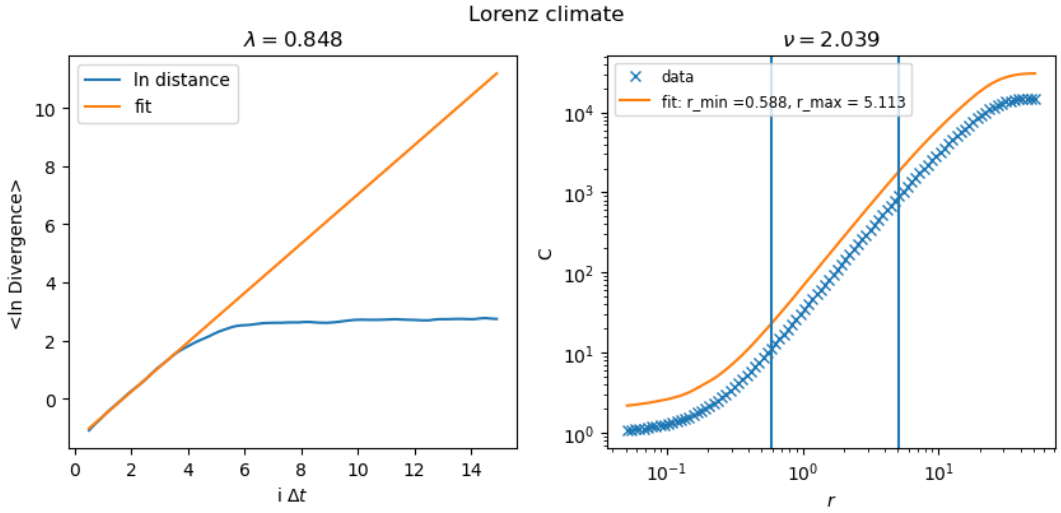


Figure 2.4.: Long term measures for the Lorenz system

To calculate the correlation dimension ν the correlation sum $C(r)$ is calculated over a wide span of r values. On a log scale a linear line is fitted and ν is given by the slope. The fit is done for a smaller range of r values, because for a small r there are not enough points for a meaningful measure, while a large r is not sensitive to local structures. To find a useful region where a line can be fitted a grid search is performed on a predefined interval I of r values. Then a loss function can be defined to find the best pair of (r_{min}, r_{max}):

$$L(r_{min}, r_{max}) = a_{fit} E_{fit} + a_{length} \eta^{-3}$$

The normalized root squared error E is calculated with respect to the fitted line. Additionaly, a term is introduced to penalize small intervals, with $\eta \sim 1/|I|$. The procedure described is taken from [49]. An example for the Lorenz attractor is shown on the right side of Fig. 2.4.

3. Multifunctionality

The term multifunctionality originates from neuroscience, where it was shown that neurons are capable of learning multiple behaviour patterns and switch between different tasks [50, 51, 52]. For example, a study using leeches discovered that the same set of neurons can trigger shortening, breathing and swimming [53]. In dynamical system theory this is also referred to as multistability, the existence of multiple stable in a system under the same configuration. In other fields multistability was observed as well, such as in lasers [54] or in semiconductors [55], and many others [56]. In reservoir computing multifunctionality can occur in two ways. In the traditional RC architecture, the reservoir state has a symmetry that can lead to the prediction of a mirror attractor which was not originally trained for [57]. In contrast, when multifunctionality is intended by design, multiple trajectories of different dynamical systems are aggregated into a single training set. Such multifunctional RCs have been realized in [13, 14]. In phase space where the attractors reside they can either be separate or overlapping. The latter case proves to be a harder problem [19]. Additionally, tuning the RC parameters is crucial for achieving multifunctionality [58]. Therefore, a change in the framework may be necessary to mitigate the parameter dependence and difficulties in an overlapping case.

3.1. Two Attractors

To evaluate the ERT regressors performance on a multifunctional setup, many parameters will be examined. The hyperparameters of the regressor are fixed in this section and set to $K = 1$ and number of trees $M = 100$, which typically yield sufficiently good results in regression settings [20]. The stopping criterion is not specified, leading to fully grown trees. In section 3.4 the ERT parameters will be examined. For the NGRC parameters k, s and p are varied. Depending on the choice they describe a specific model as defined in section 2.2. For any NGRC type we build the Cartesian product $(k, s, \mathcal{D}(p)) \mapsto (k \times s \times \mathcal{D}(p))$, which results in different feature space dimensions as given by Eq. 2.12. Since the feature space $\mathcal{F}$ can grow quickly, the dimension of the feature space is limited to $\dim \mathcal{F} \leq 1000$. The parameters for the different setups are shown in table 3.1: For example, in the NVAR case, the parameter combination $k = 6, \mathcal{D}(p) = [1, 2, 3]$ has a feature dimension size $\dim \mathcal{F} = 1329$ and as such is

Model type/Parameters	k	s	$\mathcal{D}$(p)
VAR	$[2,\ldots,200]$	1	1
NVAR	[2,3,4,5,6]	$[1,\ldots,10]$	[1,2,3,4]
NML	1	1	[1,2,3,4,5,6,7,8]

Table 3.1.: Parameter configuration of the NGRC

not considered here, but $k = 6, \mathcal{D}(p) = [1,2]$ with $\dim \mathcal{F} = 189$ is. All parameter combinations of the VAR and NML model already fulfill this condition. To set up a multifunctional NGRC a solution to the Lorenz system and the Halvorsen system is computed using RK4. Each time series is normalized to a mean value of 0 and a standard deviation of 1. The attractor parameters θ are set to -1 and 1 for the Lorenz and Halvorsen attractor, respectively. The scaling parameter is set to $\gamma = 1$. In section 3.3 the effects of these parameters will be examined. The NGRC architecture requires less training data for comparable performance to a traditional RC [37, 59, 60]. Nevertheless, there is no fixed lower bound on the amount of the training data guaranteeing successful learning of the attractors and the question remains how training data size affects the performance. Hence, different training sizes will be considered given the same parameter configuration. The training data sizes chosen are $T = \{10000, 20000, 30000\}$. The ERT algorithm is random in nature and different random realizations may have an impact on attractor reconstruction. Therefore, ten random realizations for every configuration will be examined. Each system is evolved additionally for 15000 time steps to produce a test trajectory. The prediction length is set to this value as well. This should be enough to capture the chaotic behaviour and fractal structure of the attractor and to compare measures of the prediction against the test trajectory. To visualize the correlation dimension (CD) and the Lyapunov exponent λ_{max} against the true values, the mean over the random realizations is calculated and shown. The standard deviation can be found in B. The red ellipse has its centre at the mean values of the correlation dimension and Lyapunov exponent, computed for 100 different starting points for the Lorenz and Halvorsen attractor. The diameter is given by the 3σ error.

3.1.1. Separated attractors

For the separating case the attractors are translated diagonally away from each other as shown in Fig 3.1. The influence of a realization of the NGRC parameters $(k, s, \mathcal{D}(p))$ is under the effect on short term behaviour quantified by the forecast horizon as well as on the long term behaviour characterized by the largest Lyapunov exponent λ_{max} and the correlation dimension ν. The impact of different training sizes T is also inspected, which is typically not a tunable parameter. Nevertheless, the NGRC

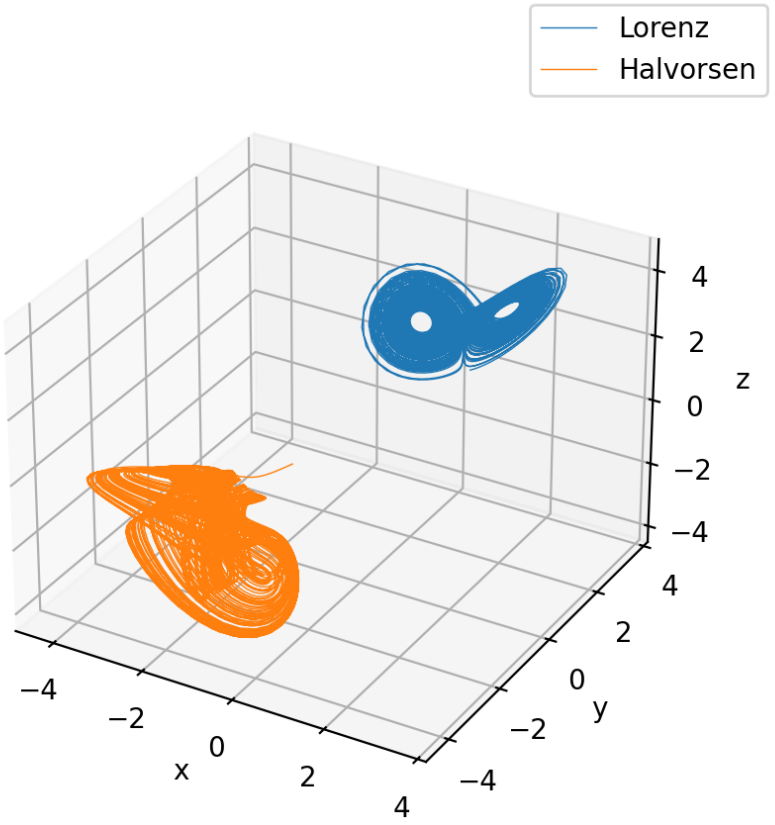


Figure 3.1.: Lorenz and Halvorsen, separated in space

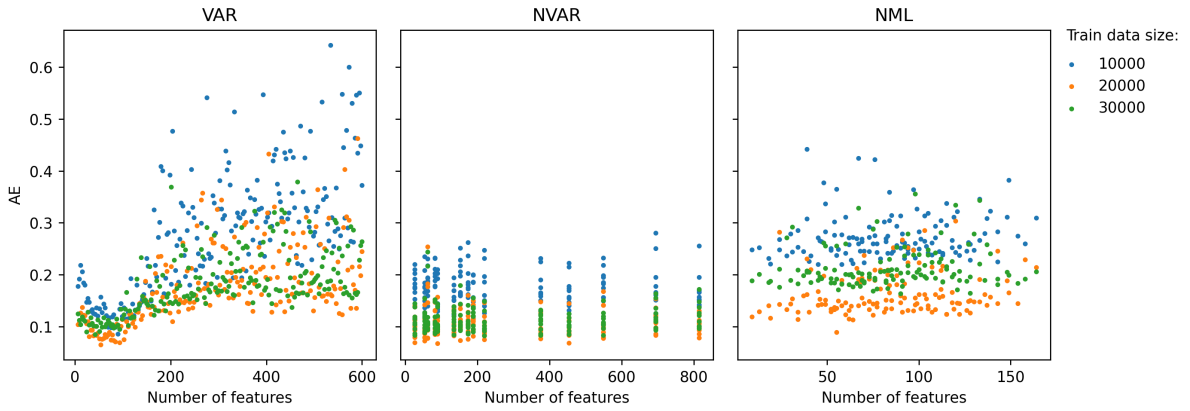


Figure 3.2.: AE measure for long term behaviour of two separated attractors for different NGRC models.

architecture uses information based on the input data to construct the feature space. Therefore, T may have an substantial effect on the learning process.
Since there are several measures quantifying the reconstruction of a dynamical system, the question arises on how to determine successful learning in a multifunctional system. For that, a single measure is defined as a combination of the different measures, one each for the short term and long behaviour. For the short term behaviour the product of each forecast horizons is defined:

$$FHP := \prod_{i}^{\#Attractors} FH^i \tag{3.1}$$

As for comparing the prediction against the true trajectory, the relative error of the measure $\eta = \{\lambda_{max}, \nu\}$ is computed:

$$\eta_{rel} = \frac{\eta_{test} - \eta_{pred}}{\eta_{test}} \tag{3.2}$$

The measure for long term behaviour is a sum of the absolute values (AE) of the relative errors:

$$AE := \sum_{i}^{\#Attractors} |\lambda^i_{rel} + \nu^i_{rel}| \tag{3.3}$$

In Fig. 3.2 AE is plotted against the number of features.

The overall best attractor reconstruction with respect to the measure is for an intermediate training size $T = 20000$ across all NGRC types. Increasing the training size at first leads to an improvement, irrespective of the number of features and whether only linear or nonlinear, or both features are present. For $T = 30000$

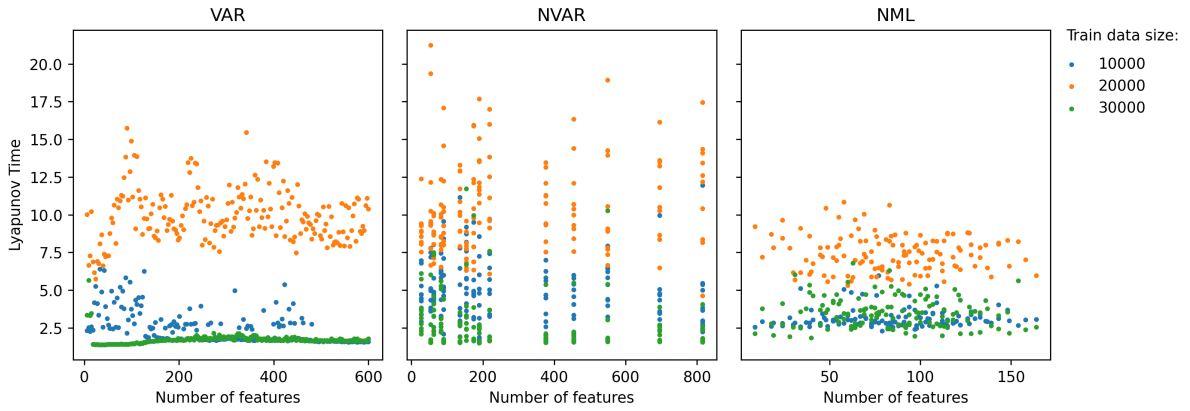


Figure 3.3.: *FHP* measure for short term behaviour of two separated attractors for different NGRC models

the performance degrades, suggesting that after some amount more data may be detrimental to the learning process.
While the models that include nonlinear features are less sensitive to the dimension of feature space, in the VAR model there is a clear optimum, where an appropriate amount of features performs the best. Having only a few features might lead to bad splits since only a limited amount of information is available and consecutive splits rely on the same small set of features. A positive correlation between linear features and the sum of errors is observed across all training sizes for the VAR model. A large feature space consisting of only linear features has a negative impact on performance. The candidates of feature-threshold pair considered at a split are inherently random in the ERT algorithm. Therefore, the probability that a split is defined by less relevant feature-threshold pair increases with the dimension of the feature space. In contrast to nonlinear terms included in the NVAR and NML model, linear features consist of past observations and occupy a smaller region in feature space and consequently a small change in the threshold has a bigger influence in a split.
The short term behaviour in Fig. 3.3 improves too for an intermediate training size across all models and parameter configurations.

Notably, the forecast horizon product tends to be the lowest for the largest training size across all models. By choosing the right amount of training data both short term and long term prediction benefit. While increasing the amount of training size of each attractor leads to more information about the system the models can learn, it is not clear why at some point having more knowledge should decrease the performance. It was shown that more data can lead to instabilities in the NGRC architecture with regularized regression[61]. Since a similar observation can be seen here, it may be irrespective of the chosen regressor. Inspecting the individual measures for each

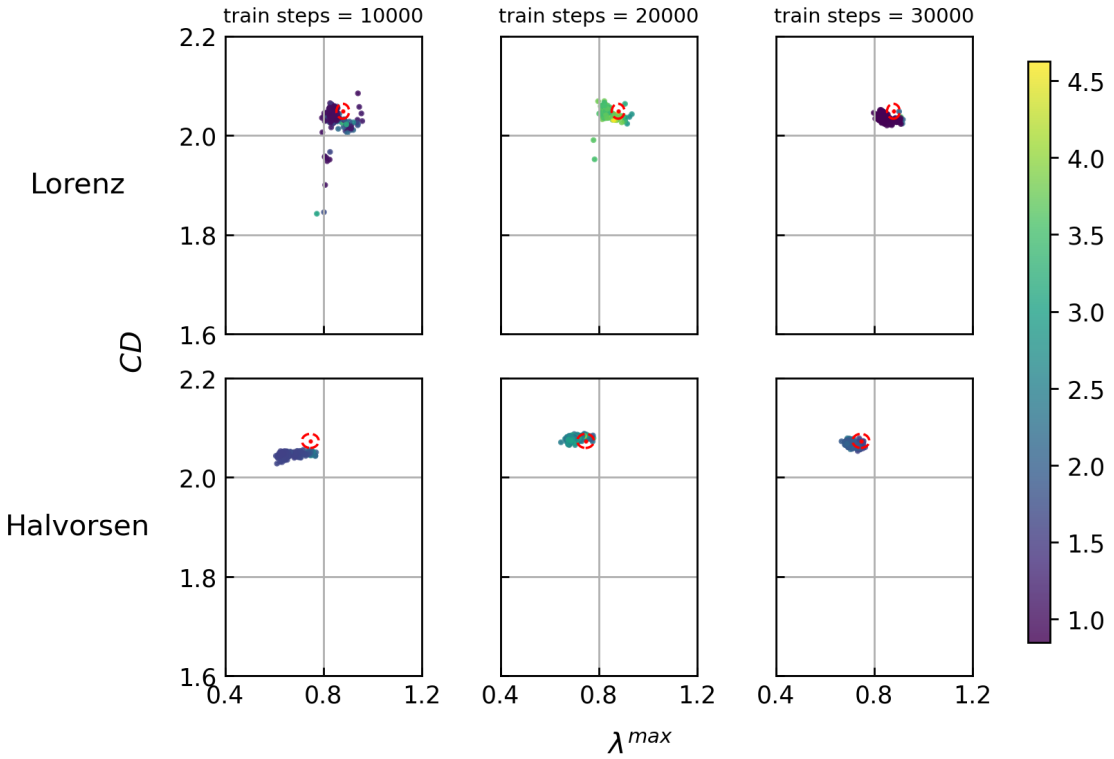


Figure 3.4.: VAR model for two separated attractors.

attractor, for the VAR and NVAR model in Fig. 3.4 and Fig. 3.5, respectively, adding more training data causes the regressor to be less sensitive to the choices of the NGRC parameters, the spread in each measure direction tends to decrease.

In the NML case in Fig.3.6 which consists of only higher order monomials with no time delay, the choice of the dictionary $\mathcal{D}(p)$ has a small influence in most cases, only a larger training size leads to measures close the true values.

While all three models are viable options that lead to potentially good results, the VAR model needs a suitable feature space dimension to work best. Any less or more than an optimal choice might hurt the learning process. By adding nonlinear features each attractor occupies a larger, more distinct region in feature space. If features are less overlapping, a tree can find splits that separate features from different systems, thus learning to distinguish an input from one system to another. Removing linear features as in the NML case is also an option, but requires potentially more data. Especially the short term behaviour in Fig.3.3 improves when a mixture of linear and nonlinear features exists.

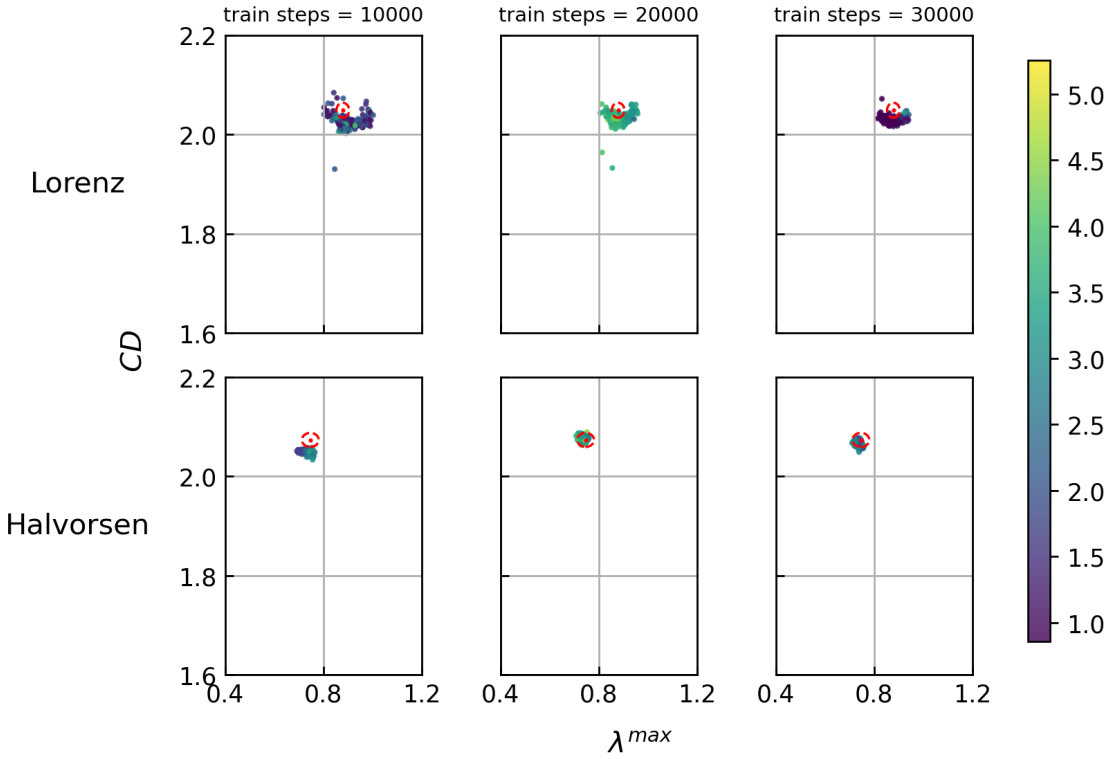


Figure 3.5.: NVAR model for separated attractors

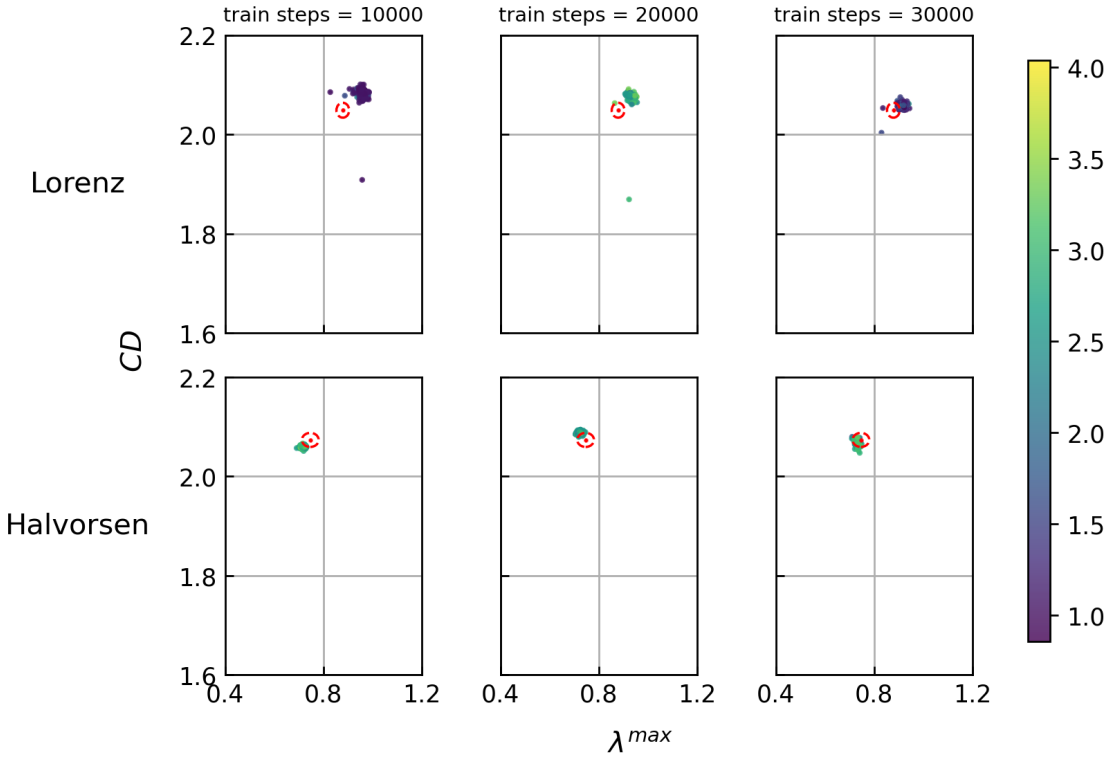


Figure 3.6.: NML model for separated attractors

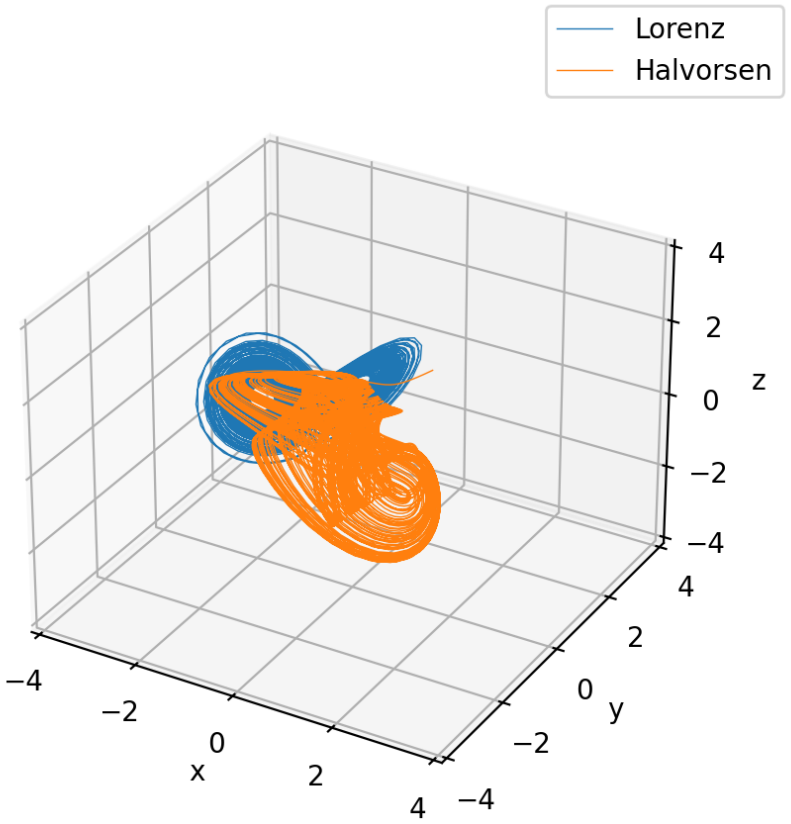


Figure 3.7.: Lorenz and Halvorsen, overlapping in space

3.1.2. Overlapping attractors

In the overlapping case the attractors have a mean value of 0 and a standard deviation of 1, as in Fig. 3.7. Overlapping attractors are harder to learn [19], because they occupy some portion of the region at the same time , thus making it harder to distinguish between which dynamics belong to the true attractor. The same set of configurations was examined as in the separating case.

Comparing the long term behaviour in Fig.3.8 and the forecsat in Fig.3.9 for the overlapping case with the separating one the the sum of errors is generally higher and the forecast horizon product shows faster divergence of prediction from the test trajectory, as expected, but essentially shows the same qualitative behaviour for all three models.

For a suitable choice of parameter configurations through optimization the predicted measures can be close to the true value, thus achieving multifunctionality is

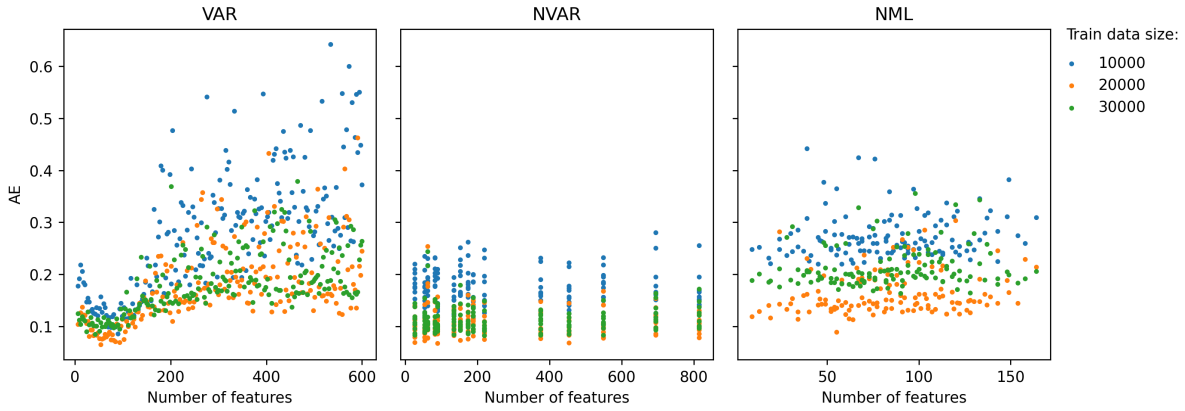


Figure 3.8.: *AE* measure for long term behaviour for two overlapping attractors

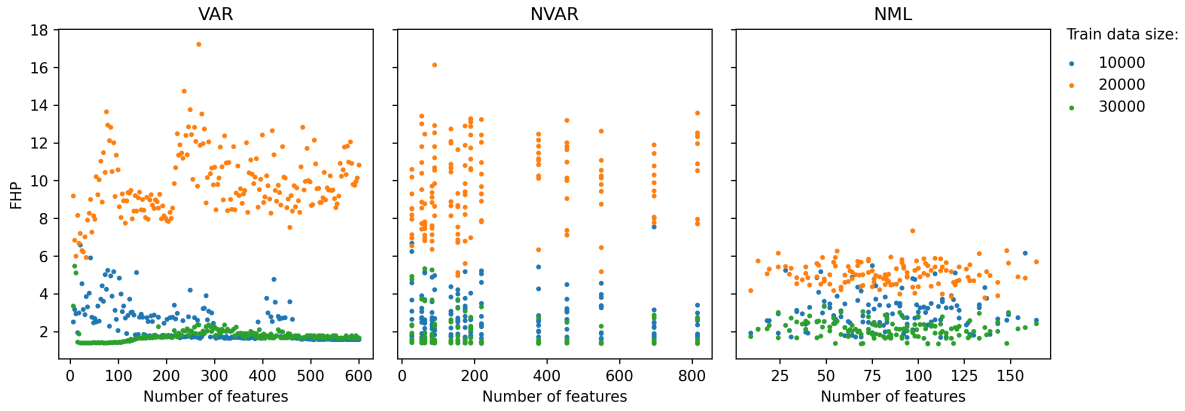


Figure 3.9.: *FHP* measure for long term behaviour for two overlapping attractors

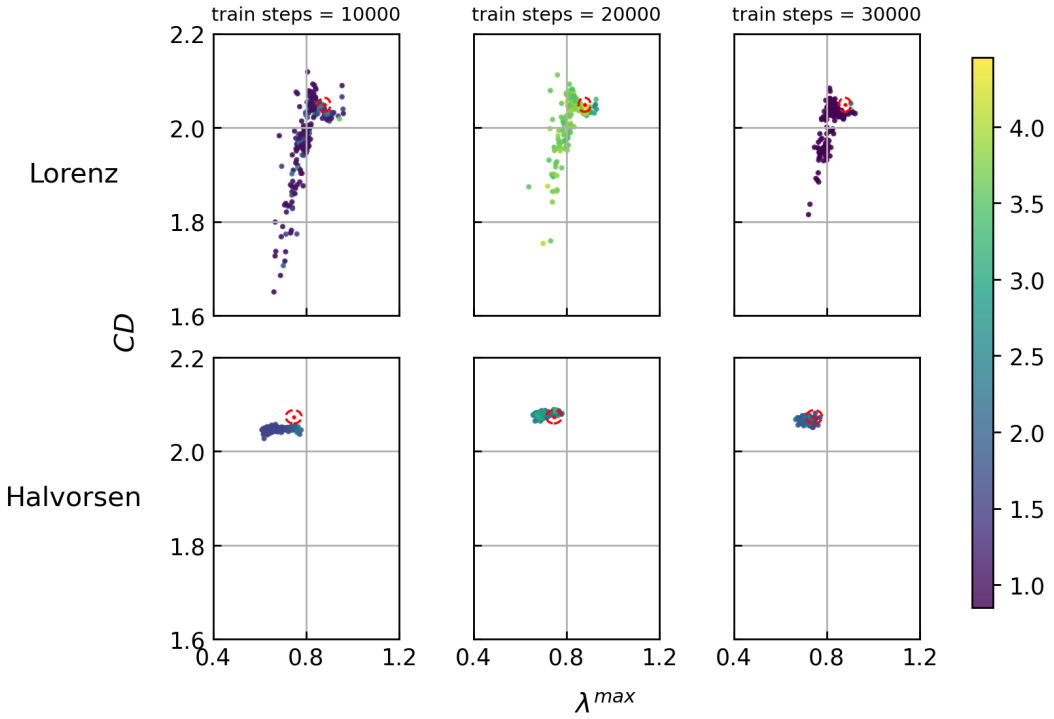


Figure 3.10.: VAR model for two overlapping attractors

possible with regards overlapping attractors.

Inspecting the individual measures in Fig.3.10, Fig.3.11 and Fig.3.12 for VAR, NVAR and NML, respectively, we can see that especially the Lorenz attractor has a large spread in the correlation dimension, and a relatively smaller one in the Lyapunov exponent. Increasing the training size reduces the overall spread but for the VAR model in Fig.3.10 only a fraction of parameter configurations comes close to the true values.

The NVAR approach in Fig.3.11 shows improvement in terms of relative error for the Lorenz attractor, as the points tend to group around the true values.

Consequently adding nonlinear features improvse the learning capacity as seen in the NVAR case in Fig. 3.12. Even then, the correlation dimension ν for the Lorenz system shows that in some cases the model was not able to reconstruct the geometrical properties of the attractor.

On the other hand the prediction of the Halvorsen attractor is indifferent to being separated or overlapping with the Lorenz attractor. There seems to be a tendency for the ERT to prefer splits that favour a good reconstruction of the Halvorsen system. Nevertheless, multifunctionality is only achieved if both attractors can be reconstructed. An example where both attractors are reconstructed with the NVAR

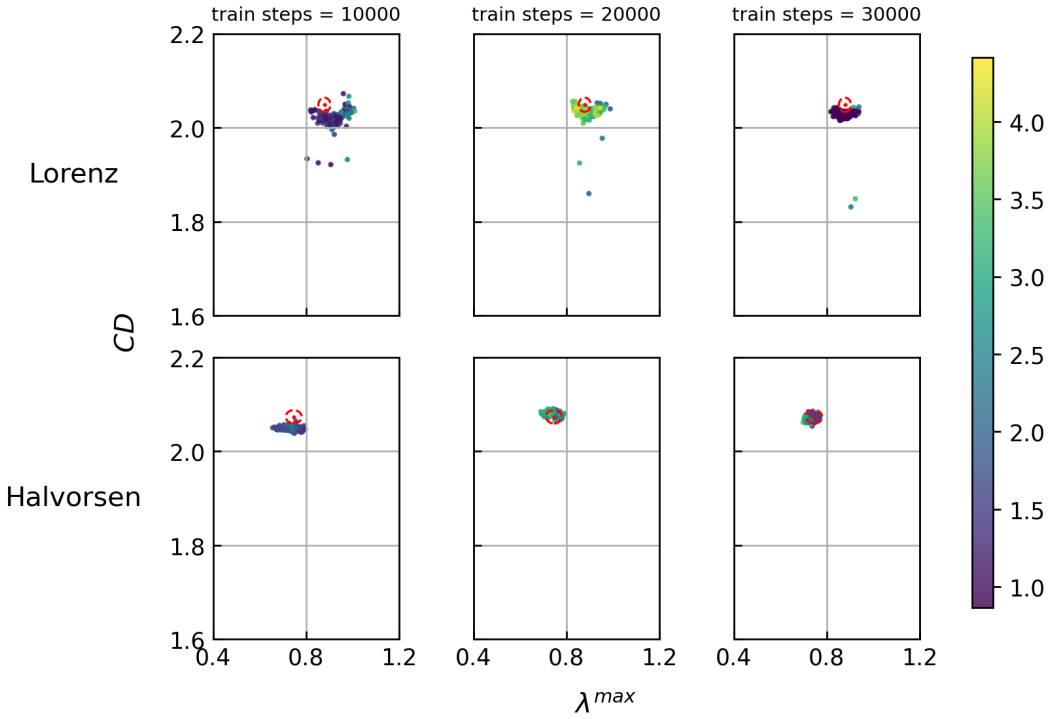


Figure 3.11.: NVAR model for two overlapping attractors

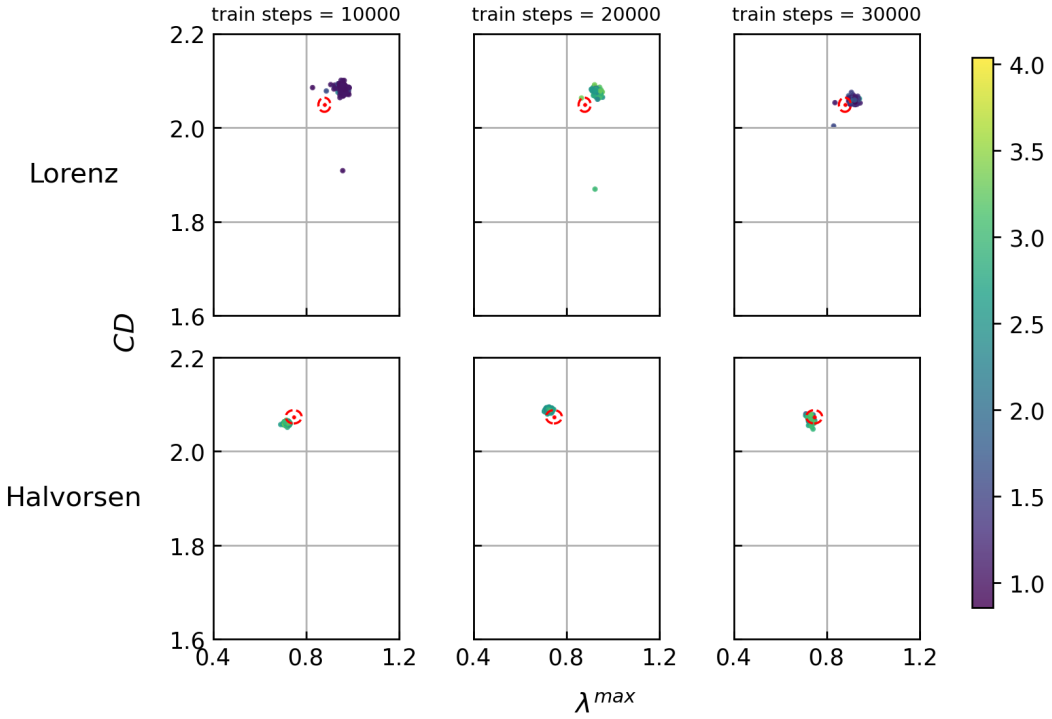


Figure 3.12.: NML model for two overlapping attractors

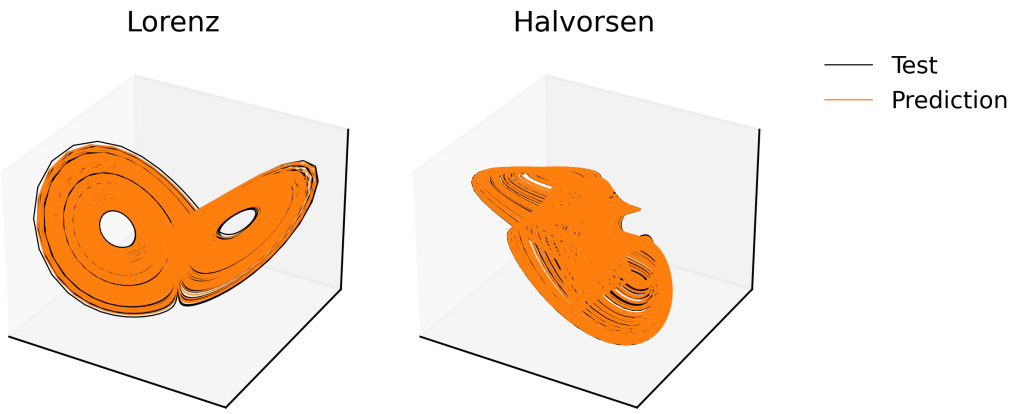


Figure 3.13.: Reconstruction of overlapping attractors for $k = 5, s = 1, \mathcal{D}(p) = [1, 2]$

model can be seen in Fig. 3.13.

For both setups it can be concluded that increasing training size improves long term prediction, but more data might may have a negative impact on short term behaviour. The ERT showed to be robust with changes in the NGRC paramaeters if nonlinear features are included. For the VAR model, since linear features are just time delayed versions of the actual attractor, overlapping in phase space means also overlapping in feature space. Nonlinear features cover a broader region in feature space. At the same time, the range of values they can take increases. As such, regions can exist that are not shared between attractors. In contrast to the Lorenz attractor, the reconstruction of the Halvorsen attractor is less sensitive to different parameter settings. Thus, depending on the dynamics of the underlying system, different NGRC parameter have higher impact on learning.. In section 3.3 it will be clear that both cases can be regarded as the same problem with the right scaling parameter, which is a substantial advantage for the ERT regressor over linear regression.

3.2. Feature Importance

Ideally the number of parameters that needs to be set by hand is zero or at least as few as possible, otherwise it is necessary to optimize over different parameter configurations. While in the NVAR and NML model, where nonlinear features are present, the hyperparameters of the NGRC overall have similar learning capabilities, in the VAR case with just linear features a minimum in the cost function AE is observed for this particular multifunctional setup. If the number of features is high, then consequently the probability of having variables that have negative influence on the prediction capabilities increases and a way to filter these out would benefit the

learning process.
For each depth of the ERT algortihm a split s, defined by a feature f and a cut-off value a, is chosen based on the amount of reduction $\Delta I_i(s)$ in the mean squared loss it achieved. Going through each split for each tree we can add up all achieved reductions for all features, such that the total reduction for each feature is given as:

$$I_i = \sum_{m}^{M} \sum_{s} \Delta I_i(s_m) \tag{3.4}$$

If the split is not defined on feature f_i then $\Delta I_i(s) = 0$. A higher total reduction implies a more important feature, thus we can rank the features. The set $\{I_1, I_2, \ldots, I_d\}$ with $\dim \mathcal{F} = d$ is normalized to one. This is then referred to as feature importance. Based on some cut-off value b everything below it can be considered as irrelevant. If each feature is as important as any other, then the feature importance would yield $I_i = 1/d, \forall i \in \{1, \ldots, d\}$. Therefore, a simple way to define the cut-off value is $b = \frac{1}{d}$ [22], everything above b is considered as an important feature. After fitting an ensemble to the data, we can filter out irrelevant variables using the feature importance and re-fit the whole ensemble using just the features that are above the cut-off value. This is essentially a feature reduction method. With less irrelevant features the performance may improve and remove the optimization over the NGRC parameters. For the VAR model in the separating as well as overlapping situation a case can be made that for larger k more irrelevant variables are introduced, decreasing the overall performance. As such, we will examine, for the overlapping attractors, with $T = 20000$ and ten random realizations, if an improvement can be seen by using the feature importance approach. In Fig. 3.14 the relative errors of the long term measures for both attractors is shown.

On the left side the difference between the initially same number of features is seen, with feature importance turned off and on. On the right side of each plot the ten smallest errors on average according to 3.3 are shown as comparison to the learning with a filtered feature space. An automatically reduced feature space produces on average a similar error as the best parameter configurations for all measures describing long term behaviour. As for the forecast horizon in Fig. 3.15, the short term behaviour is at least as good as for the best parameter configuration.

Hence, instead of searching for an optimized set of parameters a comparable performance can be achieved by the feature importance approach. Instead of optimizing the NGRC + ERT setup with a hyperparameter search, a more efficient way is adding both linear and nonlinear features and let the ERT algorithm figure out the most relevant set of features through feature importance. More sophisticated methods for tree ensembles exist, such as [62] and for a review of different feature reduction methods see [63]. In the following sections an NVAR model will be used. The feature space will be reduced in all cases and then a new ensemble is fitted for prediction.

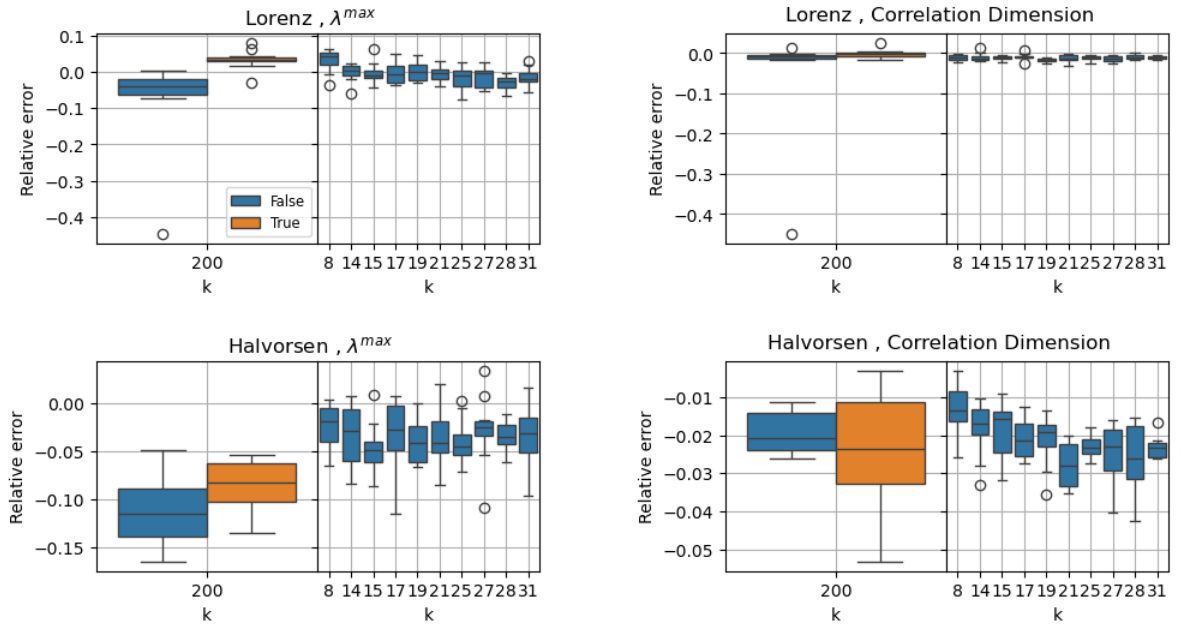


Figure 3.14.: Measures for long term behaviour for the optimization run versus a run with feature importance

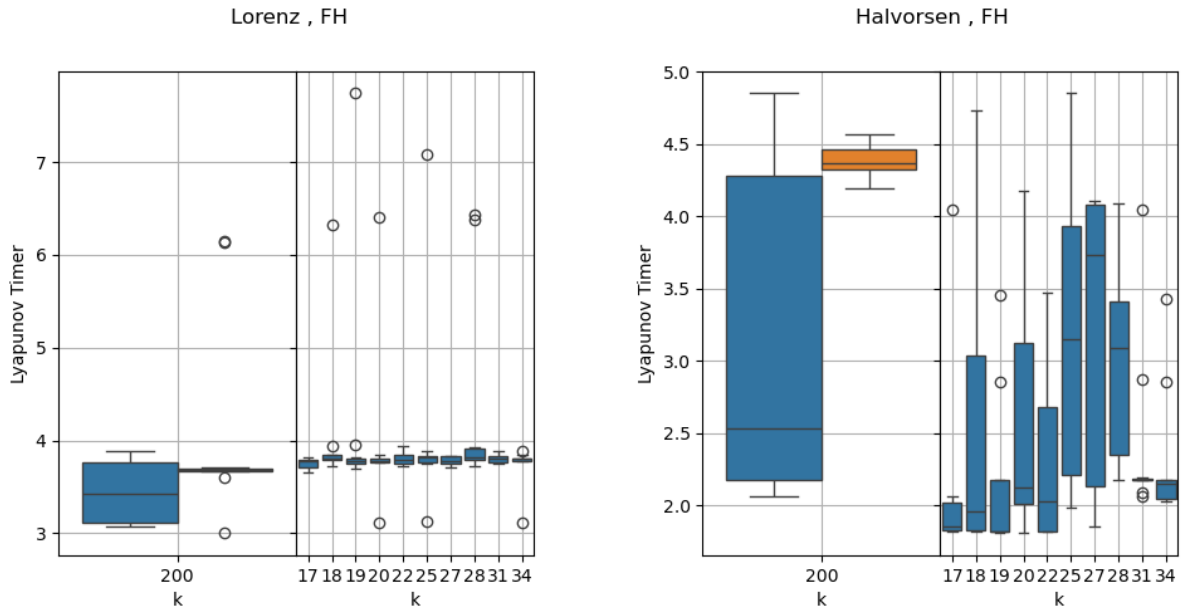


Figure 3.15.: Measures for short term behaviour for the optimization run versus a run with feature importance

3.3. Spontaneous Switching and Scaling in Feature Space

In a multifunctional setup features from different attractors possibly overlap in the feature space and a path down the tree could lead to a region that contains a mixture of samples from more than one attractor. The prediction from the tree leaf corresponding to such a region then leads the trajectory off course, resulting in either a diverging trajectory or to a different attractor. Spontaneous switching between different cognitive tasks, or more abstractly, dynamical patterns occurs in biological networks[64]. In this context this is also known as chaotic itinerancy and may be important for information processing and memory formation.[65, 66, 67] In this setup of NGRC and ERT this can happen if the feature vector for different attractors, which is essentially the warm-up time the NGRC needs, that goes in the ERT regressor follows the same path down the tree, ending up in the same region and thus the same output.
Concretely, let's have a look at an example, with four overlapping attractors. For each attractor we assign an attractor parameter ν. For Rucklidge, Chen, Halvorsen, Lorenz these are $(-2, -1, 1, 2)$, respectively, and the scaling parameter $\gamma = 0$. In that case the same large portion of the feature space is occupied by most attractor features, see 3.16 **B**. During the learning process, if a tree is not able to find distinct boundaries between features belonging to different attractors, then eventually a decision path will lead to a region with an output that potentially kicks out the trajectory of its intended track. In **C** an overlap matrix shown. The ijth element displays the fraction of feature inputs for a prediction step of attractor $\mathcal{A}_i$ that fell into leaves across all trees corresponding to the trained attractor $\mathcal{A}_j$. Hence, a row adds up to 100% for the full predicted trajectory and off diagonal elements correspond to a prediction associated with a different attractor.

In Fig. 3.16 **A**, each trajectory eventually goes off track and lands in another attractor. A solution to avoid random switching dynamics is choosing a larger scaling parameter γ, as is done in Fig. 3.17. In feature space a clear separation is made and each attractor gets its own designated region on which splits can be found corresponding to the actual dynamical behaviour. With that there is no spilling over to wrong leaves.

This is not a general statement, evidently from the overlapping case in section 3.1.2 where good performance is still possible even with partly overlapping features given a suitable set of parameters. Rather, due to the randomization in the cut-off values in a split the probability of dividing the regions into chunks associated with each attractor increases. For full-grown trees as is the case here a good split might nevertheless be found, but for a more coarse grained division in feature space where a leaf prediction consists of more samples, this will be more relevant. This happens when the tree only splits up to a specific depth that was selected beforehand. But

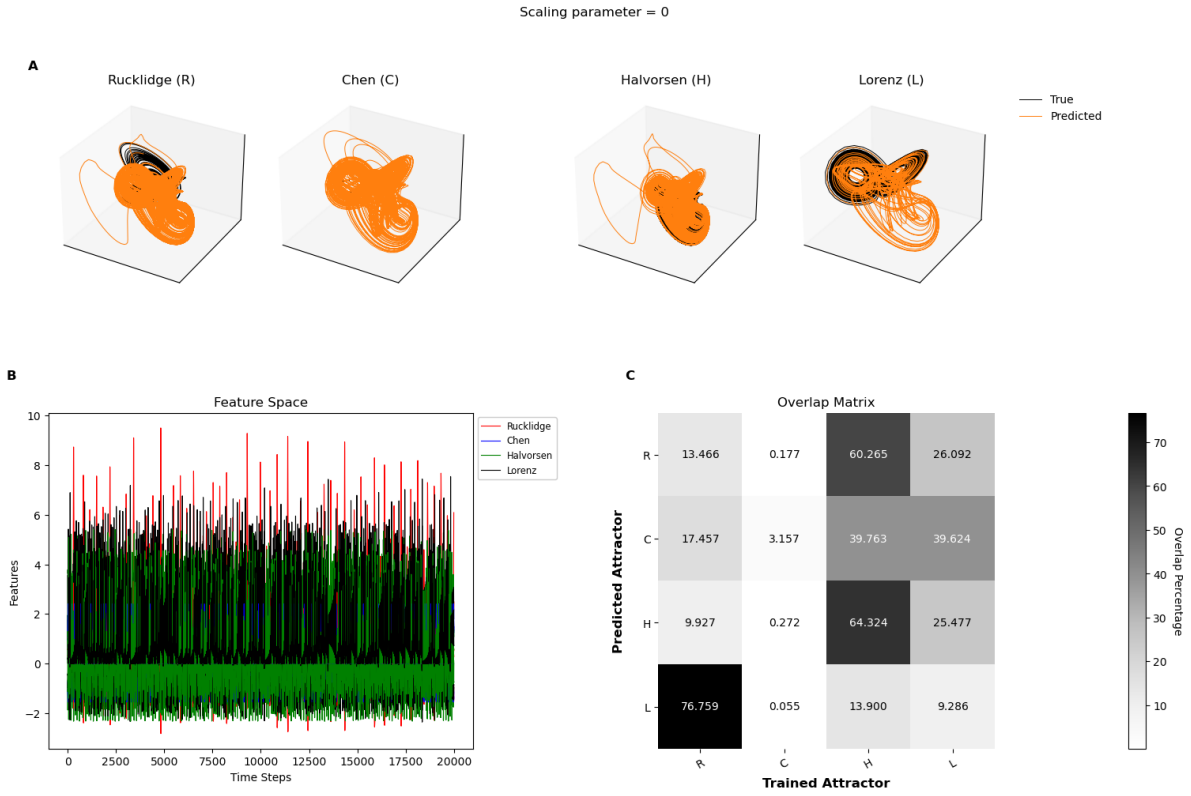


Figure 3.16.: **A**: Attractor prediction, spontaneous switching occurs in all four attractors. **B**: Features are overlapping in the feature space, a region found by the tree ensemble could potentially contain samples from different attractors or the region is shared by multiple attractors and a path down the trees result in switching. **C**: Row i displays the fraction of feature inputs for $\mathcal{A}_i$ that result in an output associated with a different attractor.

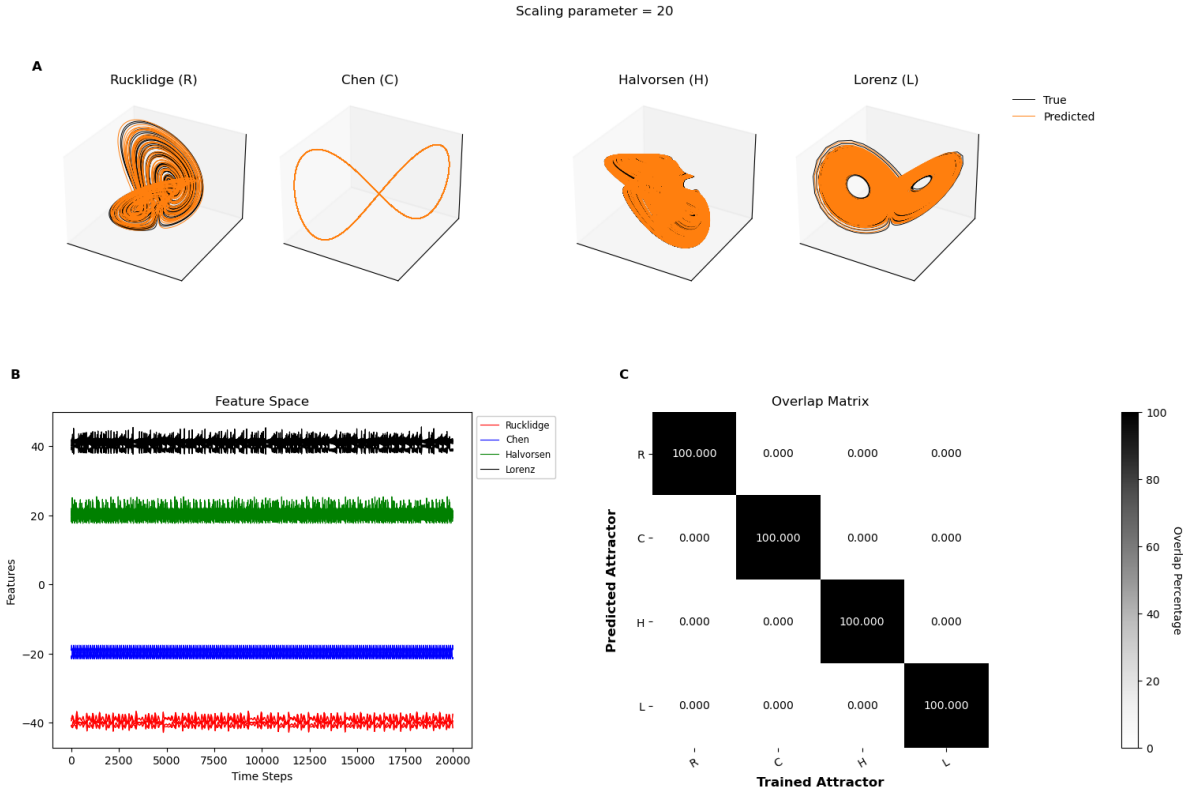


Figure 3.17.: **A**: Attractor prediction, no switching occurs. **B**: Features are separated in feature space, allowing for better splits. **C**: The prediction falls within leaves that are associated with the corresponding trained attractor dynamics. The off diagonal elements are zero, each input gave an output corresponding the trained attractor

k	s	$\mathcal{D}(p)$	Number of features	Fraction of all features	Number of trees M
5	1	[1,2]	135	[0.1, 0.2, . . . , 0.9, 1.0]	[10, 20, . . . , 90, 100}

Table 3.2.: NGRC and ERT parameters

since it not a priori known whether decision paths eventually lead to bad predictions, it is useful to separate features by choosing an appropriate scaling parameter.

3.4. Optimizing the Regressor

For the default ERT parameters the parameter configurations $(k, s, \mathcal{D}(p))$ of the NGRC framework had no large impact, even for only linear features when the feature importance procedure outlined in section 3.2 is applied, a comparable performance ca be reached with respect to an optimization run. The regressor itself can be optimized as well. Three hyperparameters can be configured that define how much randomness is introduced in the building process of a tree and the size of it as well as oh the whole ensemble. How many features are considered in a split, given by K, the number of trees M in the ensemble and the stopping criterion, either through requiring a node to contain a minimum number of samples to be a leaf or setting the maximal depth. Only the first two, K and M will be considered here, but in any serious parameter optimization run the stopping criterion is a factor to consider as well. In some sense K and M work against each other. A small K introduces more randomness into the model, as the actual best feature at some split might not be part of the set under consideration. For $K = d$ all features are considered, the cut-off values are still uniformly chosen, but more important features are expected to be selected by a split on average.
Adding more trees to the ensemble counteracts the variance introduced by the randomization and as such the error tends to decrease by increasing M. However at some point the reduction in error might saturate after adding a certain number of trees[68]. To investigate the influence of K and M a grid search is performed for the parameters in table 3.2

The best values, when minimzing Eq. 3.3, were $K_{fraction} = 0.2, M = 40$ and the predicted attractors are shown in Fig. 3.18, with their measured values in table 3.3.

The cost function EE in Fig.3.19 depends implicitly on $K_{fraction}$. The feature matrix is the same across all parameters, but any set of features considered at a split is random. $K_{fraction} = 1$ takes all features into account, but the cut-off value a is distinct for two different $K_{fraction}$ values, even if they contain the same feature. Therefore, the cut-off values are significant for defining a split. They cannot be set by a parameter and are only defined within an interval given the features of the training data. The

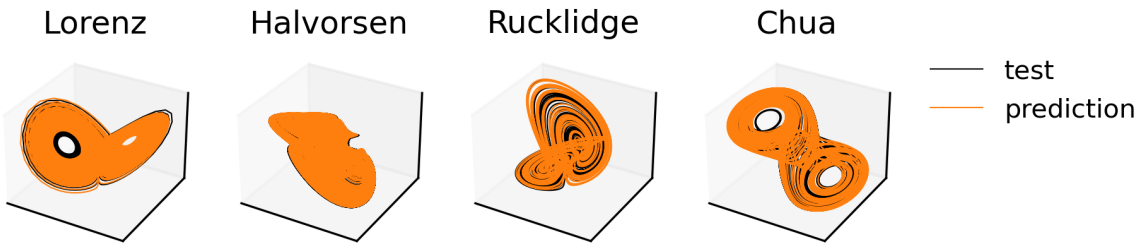


Figure 3.18.: Prediction using the optimized ERT parameters

	Lorenz		**Halvorsen**		**Rucklidge**		**Chua**	
	Train	Test	Train	Test	Train	Test	Train	Test
λ^{max}	0.92	0.85	0.79	0.80	0.24	0.23	0.37	0.39
C	2.04	2.03	1.89	1.91	1.90	1.88	1.87	1.86
FH	3.67	-	3.12	-	6.23	-	1.52	-

Table 3.3.: Measure values for optimized ERT parameters

number of trees is can be neglected when an adequate choice for $K_{fraction}$ or K is made. From Fig. 3.19 it can be seen that even a tenth of the maximal number of trees considered is as good or even better with the amount of randomization. Also, for short term behaviour, fewer trees could have substantial improvement over models with a larger ensemble for some $K_{fraction}$. Inspecting the individual measures for each attractor in Fig.B.7 most of the error comes from the Rucklidge attractor, both in largest Lyapunov exponent and correlation dimension, thus an optimization would improve especially the prediction of this particular attractor.

The other three dynamical systems are not less influenced by different configurations. As for the forecast horizon it can be seen more clearly when looking at the separate attractors that it follows for the most part a discrete pattern where it jumps to a specific value. Short term behaviour might be more impacted by the choices of the NGRC and training size than the ERT regressor.

To sum up, tuning the ERT regressor can be worthwhile, especially for the Rucklidge attractor substantial improvement can be made. Depending on the speed and memory available, more randomization by decreasing K leads to faster learning, as fewer features need to be considered at each split. Training a larger ensemble requires more memory, but as seen a big model is not necessary. As stated in [68] a good strategy can be to choose as many trees as computationally feasible, thus reducing the number of tunable parameters and from there optimize the randomization, and ideally the stopping criterion as well.

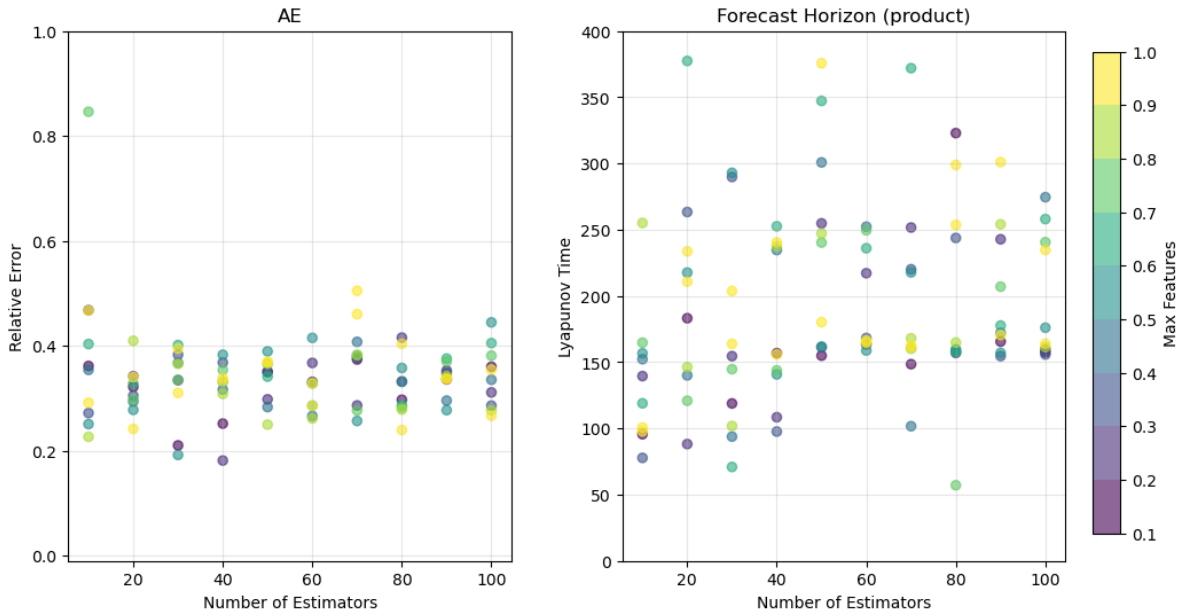


Figure 3.19.: Relative measures are plotted for different values of $K_{fraction}$ and M

3.5. Controlled Switching

A system capable of learning multiple attractors is a multifunctional system that stores the individual dynamical patterns by some means and retrieves them, among possible other ways, by receiving an input in the form of past states. As such, this system is also a memory device. The NGRC architecture naturally preserves information of an attractor by designing features, or states, that consist of time delays and their monomial expansions. The ERT regressor, or for the matter any similar tree based regressor, finds a suitable representation of the dynamic behaviour by dividing the high dimensional feature space into local regions and fits a simple model. Retrieval of long term memory can be categorized in two ways [69], "content-addressable" and "location addressable". The first requires some content associated with the memory to be fetched. In our case this would be a short time series that is associated with the underlying dynamics of an attractor, which is embedded in a higher dimensional space through the NGRC framework and the ERT regressor predicts the flow along the attractor, thus a memory is accessed. Since this is a closed loop the signal may change the dynamics of the trajectory, and different memories are accessed, as seen in section 3.3. The second retrieval mechanism works by giving a location of the memory to be fetched, which is how a computer basically access its memory. Instead of specifying an exact address a suitable cue is enough to recall memories in neural networks.[70, 71] In the NGRC + ERT framework the cue comes in the form of the

Train size	k	s	$\mathcal{D}(p)$	γ	Number of trees	$K_{fraction}$
20000	5	1	[1, 2]	50	40	0.7

Table 3.4.: Parameter configuration of NGRC + ERT for 16 overlapping attractors

attractor parameter. As mentioned in section 3.3 a cue in the form of an additional parameter besides the warm-up time is not strictly needed, even for completely overlapping attractors, but to avoid spontaneous switching it is advantageous to divide the higher dimensional feature space into separate regions for each attractor. The scaling parameter can be regarded as the cue strength. A higher cue strength should naturally lead to better memory recall.
To demonstrate controlled switching via a cue signal, 16 attractors, for eight distinct dynamical systems with two trajectories each randomly rotated in space, are trained. The setup is given in Fig. 3.20 and the parameter configuration follows from table 3.4.

Only $K_{fraction}$ was optimized, with a fixed number of trees, as the focus here lies on controlled switching. For a small number ensemble size and little optimization every attractor was reconstructed as shown in Fig. 3.21. The measures as well as each attractor parameter can be looked up in table B.1.

To predict an attractor $\mathcal{A}_i$, a warm-up time consisting of the past ks time steps is given as well as the associated attractor parameter η_i. The prediction length is 15000 time steps as before. To switch to a different attractor $\mathcal{A}_j$, a cue is given in the form of a parameter change $\eta_i \to \eta_j$. Then the retrieval of a trajectory on $\mathcal{A}_j$ consists of a warm-up time of the last ks of the time series predicted on $\mathcal{A}_i$ and a cue η_j. A successful switching between different attractors or "memories" is shown in Fig. 3.22. Being close to each other is clearly an advantage since the transition time is short and the trajectory is quickly on the respective attractor, but this depends on the starting point at the switch as well as the time between consecutive time steps. The *Chen*_10 evolves slower with a smaller Δt and the starting point is further from the attractor, consequently the transition is longer.

The point at which switching is turned on is a crucial one. Each dynamical system has a basin of attraction, if the initial point is within that region it converges to that particular attractor. The basin of attraction is learned by the NGRC + ERT setup and is subjected to simple decision rules at each split, but each tree stores its own representation, which results in a complex structure in feature space. While the trajectory transverse through the phase space it could potentially be attracted to the wrong attractor. Therefore, it is important to examine the effects of different points on the starting attractor and whether a switch is successful or not. In Fig. 3.23 switching from a Chua system to a Lorenz system is shown from different parts of the initial attractor.

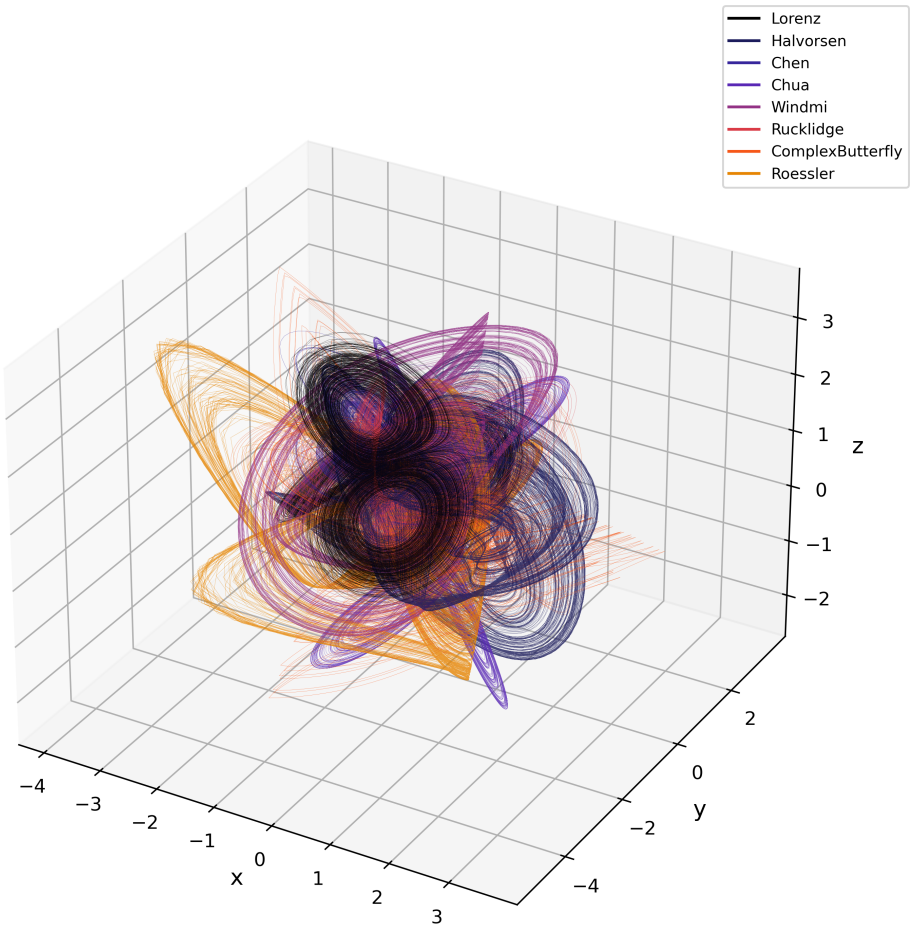


Figure 3.20.: 16 overlapping attractors

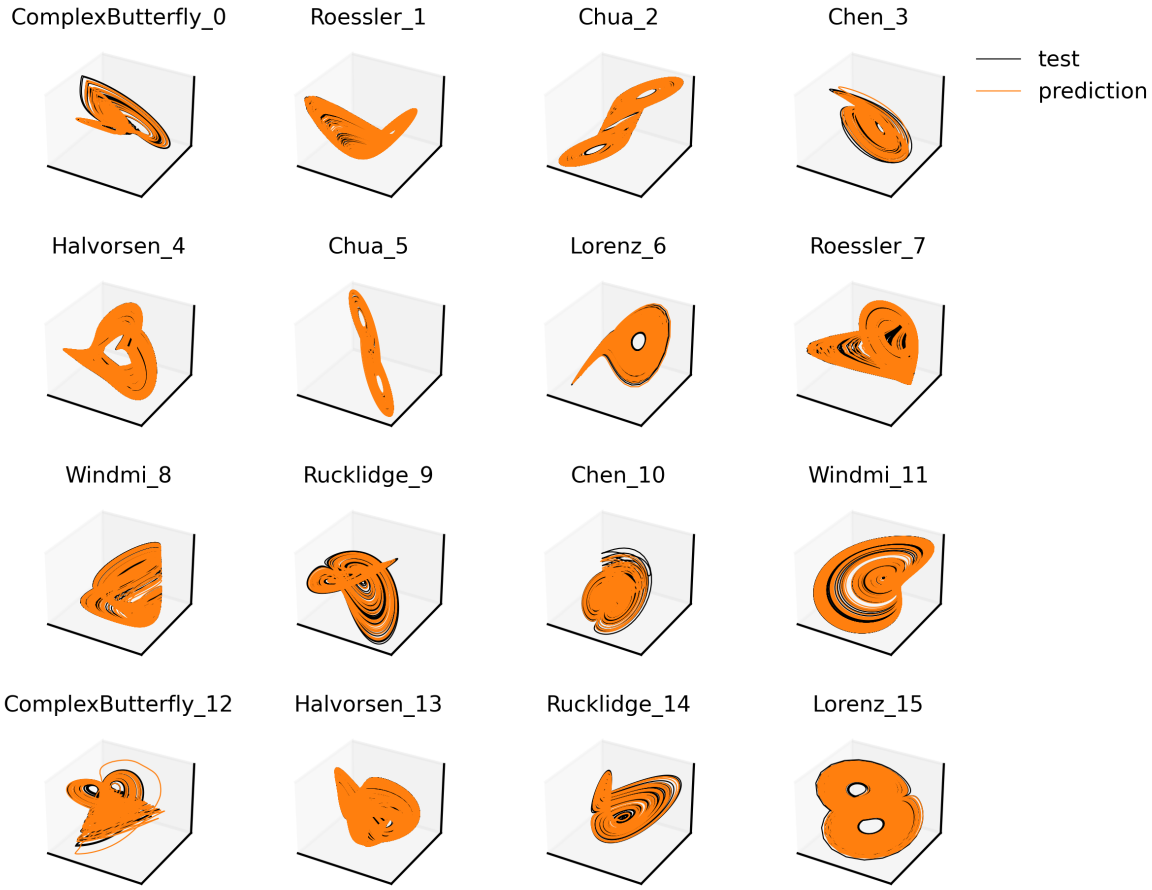


Figure 3.21.: Prediction of 16 overlapped attractors

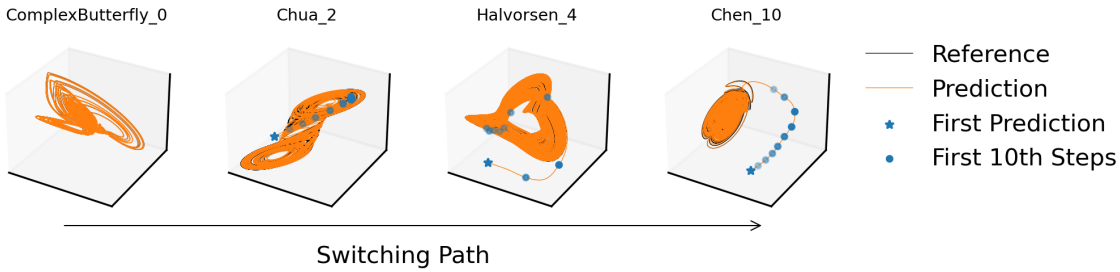


Figure 3.22.: Switching between different attractors

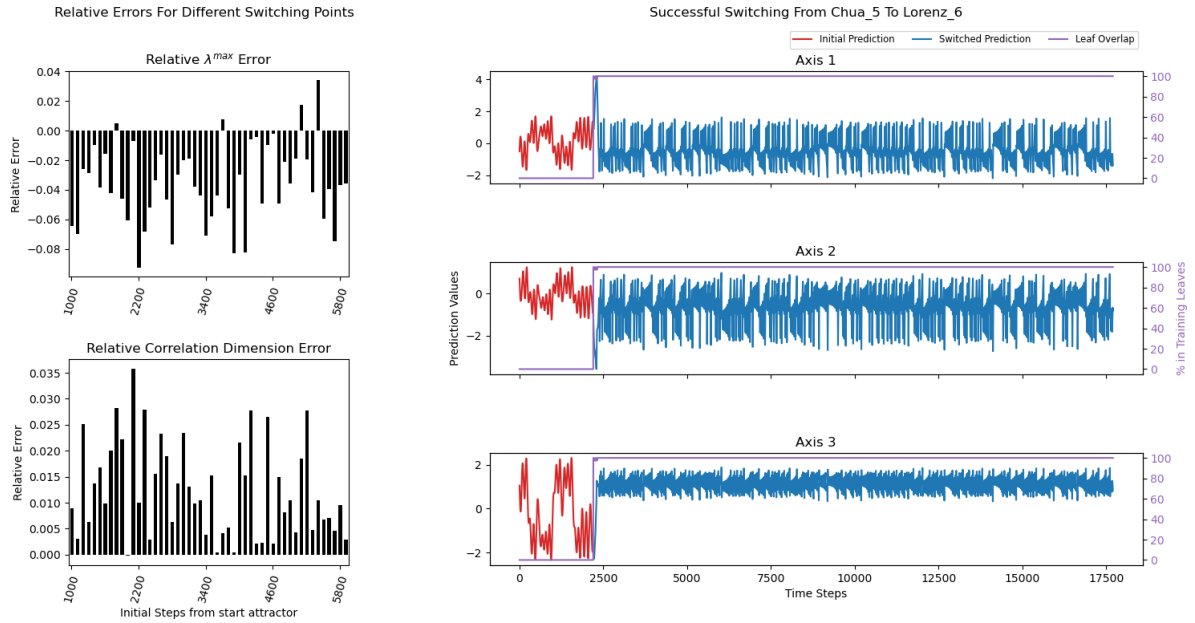


Figure 3.23.: Successful switching. The ERT predicts the flow and follows a trajectory that reconstructs the switched attractor.

The relative errors of both λ_{max} and the correlation dimension is low, suggesting that for the most part the switching was a success. Exactly at the switch point the predictions fall within the leaves corresponding to the dynamics that belong to the Lorenz system.
On the other hand an unsuccessful switching can be observed in 3.24 from the Complex Butterfly system to the Rucklidge attractor. The error bars suggest that there are regions on the initial system that would not lead to the desired state. On left side of Fig. 3.24, for a time after the switch, the ERT predicts a flow belonging to the Rucklidge attractor, but the trajectory eventually diverges. Even though for some time the ERT identifies the input as belonging to the switched attractor, the trajectory is not reconstructing the Rucklidge system. The ERT follows the flow of the Rucklidge system it learned, but the prediction follows an untrained dynamical behaviour.

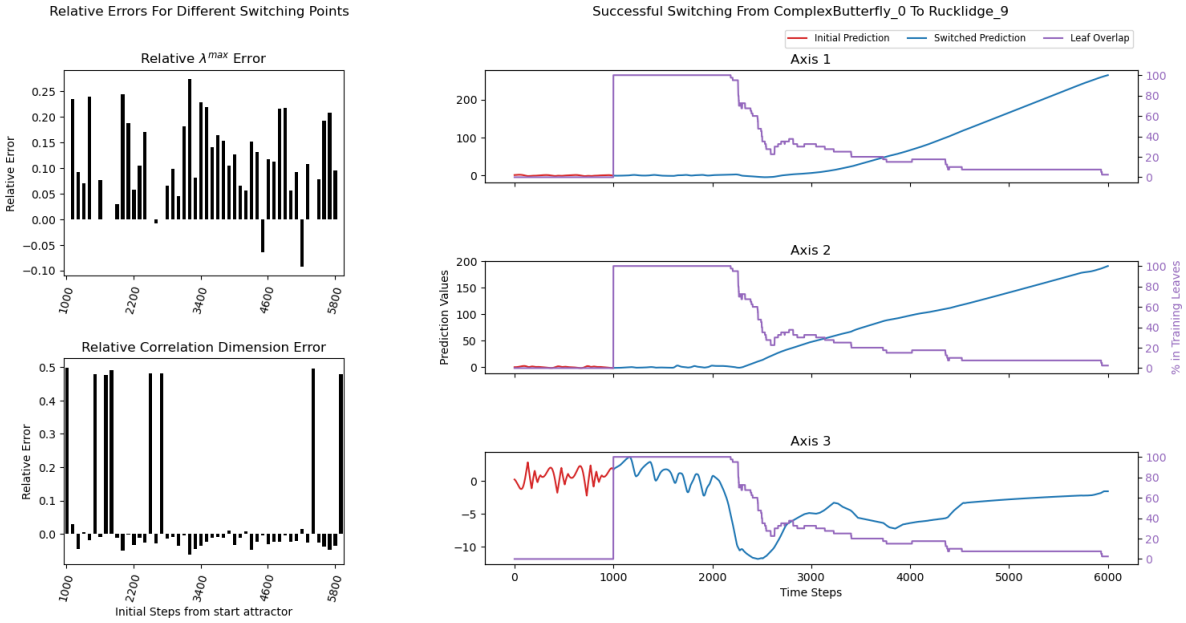


Figure 3.24.: Unsuccessful switching. Even though the ERT predicts the flow of the Rucklidge system, it eventually diverges.

4. Conclusion

The goal of thesis was to explore whether reservoir computing with different regression techniques is capable of reconstructing attractors. For that, a variation of RC was employed, called Next-Generation Reservoir Computing, in combination with a tree based approach instead of regularized linear regression. As a regressor, the Extremely Randomized Trees algorithm was chosen. The focus was then on learning multiple attractors, and thus achieving multifunctionality with one model, which has its roots in neuroscience, but found in other scientific fields as well. In chapter 3 a multifunctional setup consisting of two attractors was examined for many NGRC parameters. With fixed regressor parameters, it was shown that for separated as well as for overlapping attractors multifunctionality can be achieved. Additionally, with a feature importance approach, irrelevant features can be filtered out and a comparable performance can be achieved with respect to a parameter search, eliminating for the most part the tuning of the NGRC architecture. With a mixture of linear and nonlinear features, that was denoted as the NVAR model, the NGRC + ERT architecture performed the best and was more resilient towards parameter changes. As such, to achieve multifunctionality an NVAR model should be preferred. Setting a suitable dimension for the feature space and applying feature importance can remove irrelevant variables, which may introduced when adding linear features. An optimization over NGRC parameters might therefore not be necessary.
With the addition of an attractor parameter and scaling parameter, which are added to the feature vector, the NGRC + ERT can leverage the tree architecture to separate features in feature space with a suitable choice of the scaling parameter. Consequently, spontaneous switching between attractors can be avoided.
Although the ERT algorithm introduces several new hyperparameters, good performance can be achieved without optimization. Nevertheless, in some cases it may be necessary to find a suitable set of parameters to achieve multifunctionality. The size of the ensemble can in principle be fixed, thus reducing the number of hyperparameters. Instead, it can be linked to the amount of memory available and the number of features K considered in a split can be tuned.
The last section showed that introducing an attractor parameter can be used in an additional way. The attractor parameter acts as a cue, allowing for controlled switching between different attractors. Hence, different memories can be dynamically accessed. Because the trees in the ensemble learned each a representation of the flow of the

attractors, the basin of attraction is complex and switching can fail if it is initiated in some specific regions.
To summarize the findings, the NGRC + ERT architecture showed to be advantageous for learning multiple attractors and thus achieve multifunctionality. While for the standard RC framework with linear regression, especially the overlapping case turned out be hard to learn. In contrast, the ERT regressor does not differentiate between separating and overlapping attractors, if attractor parameters are introduced and an appropriate choice for the scaling parameter is made. Furthermore, the attractor parameters can be used to switch between different attractors on purpose. Comparing ERT to regularized ridge regression, in principle, the ERT regressor shows suitable prediction power with default parameter setting, while the regularization parameter needs to be adjusted. In any case, the ERT can be optimized for systems that are harder to learn. The number of attractors an ERT is capable of learning should be linear with respect to the memory available. Scaling the feature space accordingly assigns effectively a region to each attractor, and due to the self similar structure of trees, a tree can reach any depth needed to separate the dynamics of attractors. Because splits are in part random in nature, the probability that unfavourable splits are chosen increases, and in turn the number of trainable attractors decreases. This could be avoided by adding more trees, reducing the variance of the ensemble. Further research should look into other tree based approaches with different algorithmic structure. The fundamental structure of trees stays largely the same, but different decision rules in splits as well as the output of leaves can lead to changes in the learning process.

A. Equations of Dynamical Systems

The equations for the chaotic systems can be found in [28].

Lorenz System

$$\begin{aligned} \dot{x} &= \sigma(y - z) \\ \dot{y} &= x(\rho - z) - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{A.1}$$

Parameters: $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, $\Delta t = 0.02$.

Halvorsen System

$$\begin{aligned} \dot{x} &= -\sigma x - 4y - 4z - y^2 \\ \dot{y} &= -\sigma y - 4z - 4x - z^2 \\ \dot{z} &= -\sigma z - 4x - 4y - x^2 \end{aligned} \tag{A.2}$$

Parameters: $\sigma = 1.27$, $\Delta t = 0.02$.

Rucklidge System

$$\begin{aligned} \dot{x} &= -\kappa x + \lambda y - yz \\ \dot{y} &= x \\ \dot{z} &= -z + y^2 \end{aligned} \tag{A.3}$$

Parameters: $\kappa = 2$, $\lambda = 6.7$, $\Delta t = 0.02$.

Roessler System

$$\begin{aligned}\dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c)\end{aligned} \tag{A.4}$$

Parameters: $a = 0.2$, $b = 0.2$, $c = 5.7$, $\Delta t = 0.1$.

Chua Circuit

$$\begin{aligned}\dot{x} &= \alpha \left(y - x + bx + \frac{1}{2}(a - b)(|x + 1| - |x - 1|) \right) \\ \dot{y} &= x - y + z \\ \dot{z} &= -\beta y\end{aligned} \tag{A.5}$$

Parameters: $\alpha = 9$, $\beta = \frac{100}{7}$, $a = \frac{8}{7}$, $b = \frac{5}{7}$, $\Delta t = 0.02$.

Complex Butterfly Attractor

$$\begin{aligned}\dot{x} &= a(y - x) \\ \dot{y} &= -z \operatorname{sign}(x) \\ \dot{z} &= |x - 1|\end{aligned} \tag{A.6}$$

Parameter: $a = 0.55$, $\Delta t = 0.05$.

Chen System

$$\begin{aligned}\dot{x} &= a(y - x) \\ \dot{y} &= (c - a)x - yz + cy \\ \dot{z} &= xy - bz\end{aligned} \tag{A.7}$$

Parameters: $a = 35$, $b = 3$, $c = 28$, $\Delta t = 0.005$.
In section 3.3, the parameter a set to $a = 33$, which results in a periodic orbit.

Windmi System

$$\begin{aligned} \dot{x} &= y \\ \dot{y} &= z \\ \dot{z} &= -az - y + b - e^x \end{aligned} \tag{A.8}$$

Parameters: $a = 0.7$, $b = 2.5$, , $\Delta t = 0.08$.

B. Additional Figures

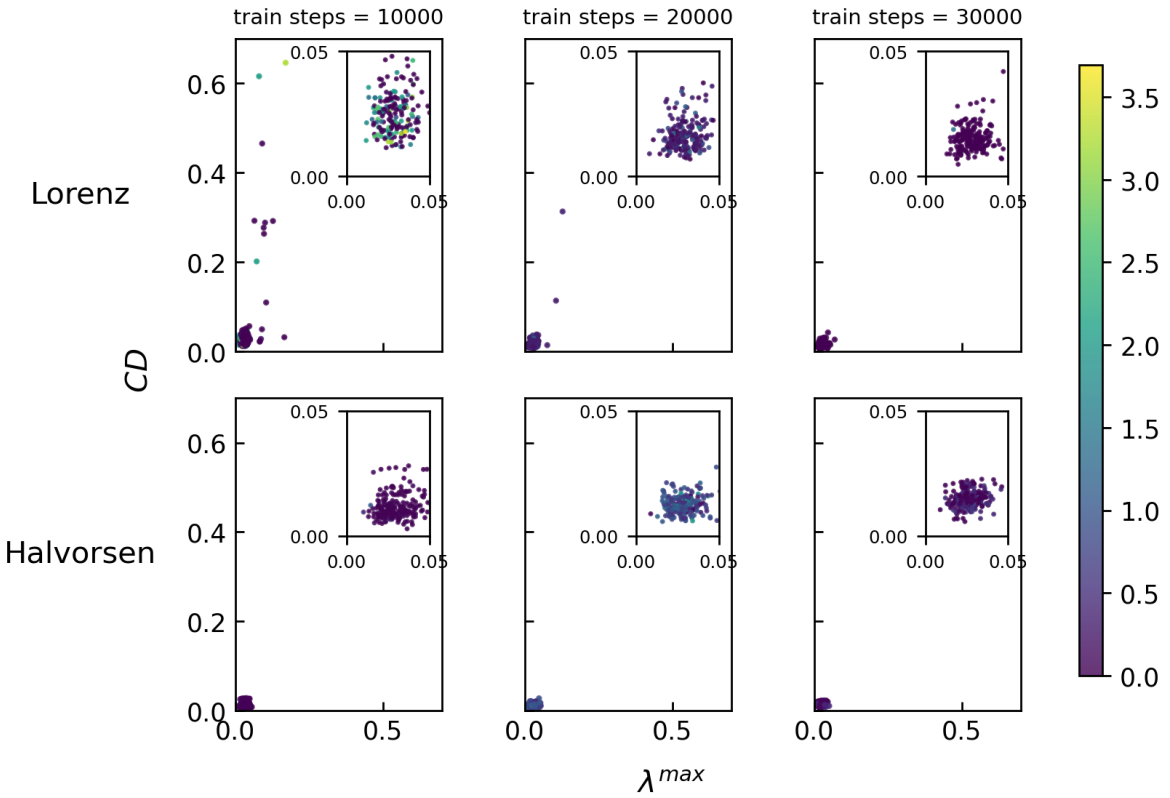


Figure B.1.: Standard deviation for separated attractors in the VAR model

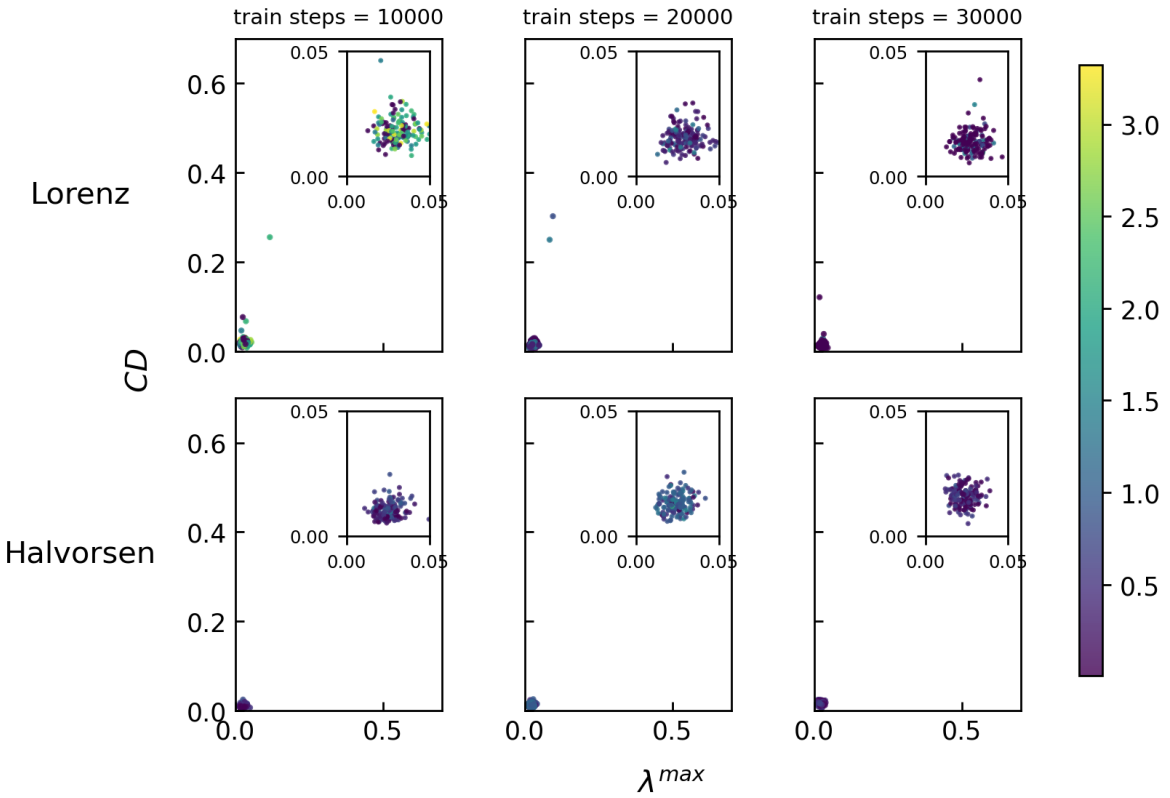


Figure B.2.: Standard deviation for separated attractors in the NVAR model

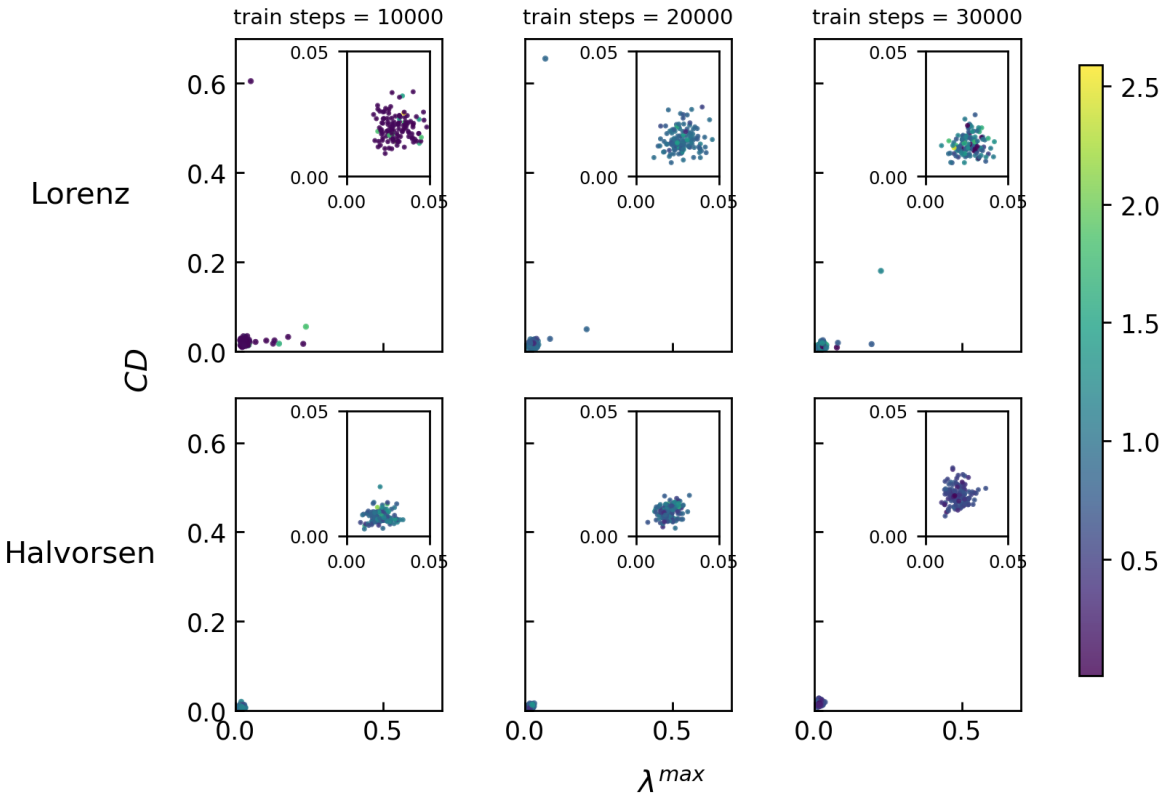


Figure B.3.: Standard deviation for separated attractors in the NML model

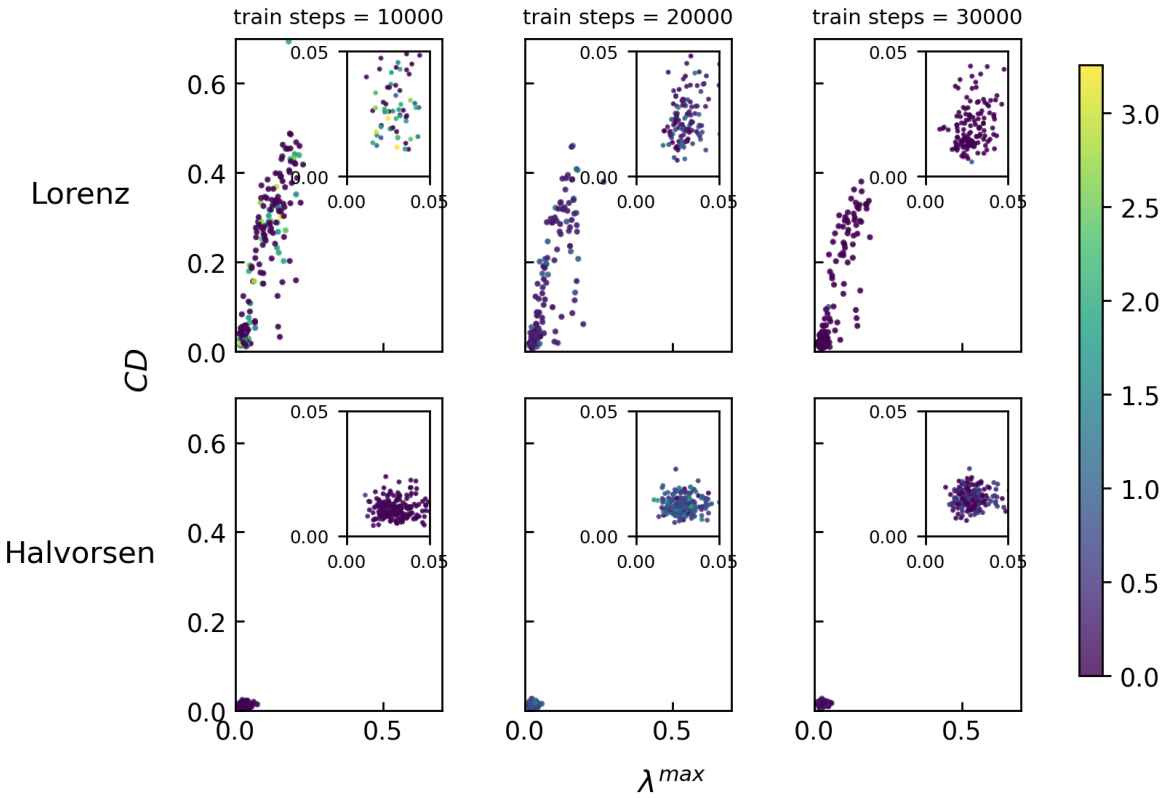


Figure B.4.: Standard deviation for overlapping attractors in the VAR model

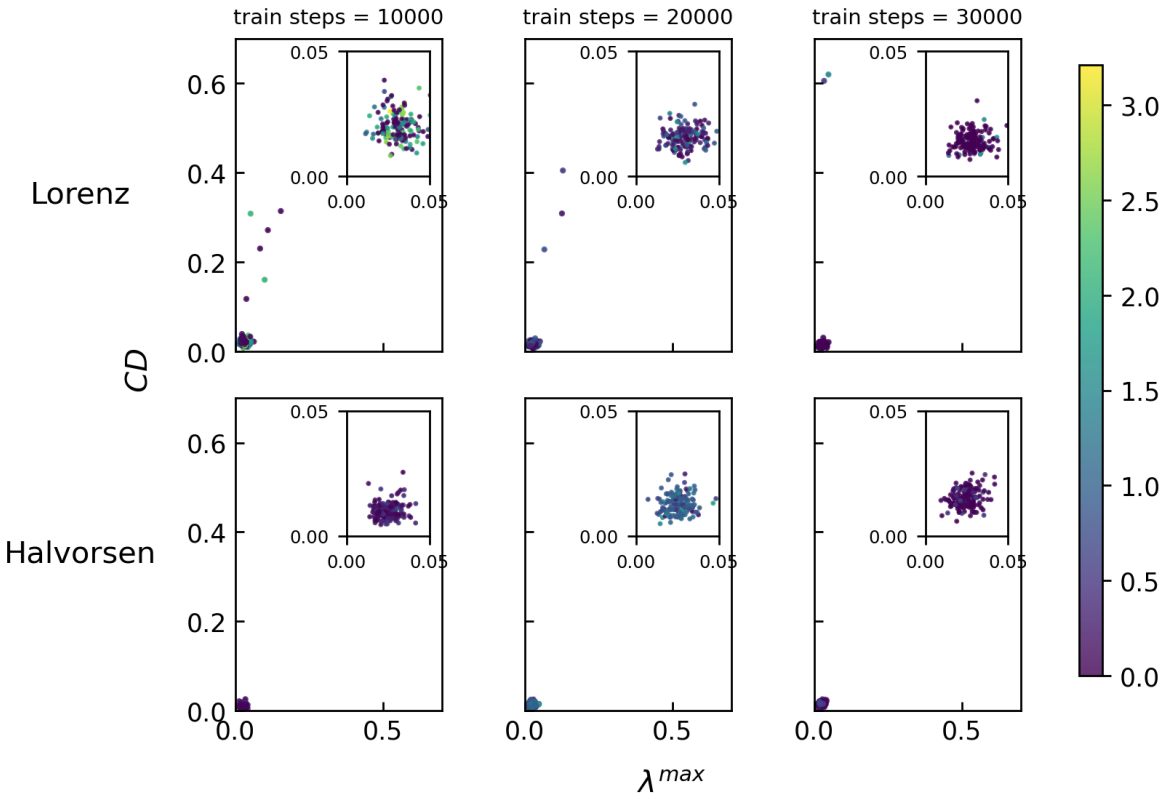


Figure B.5.: Standard deviation for overlapping attractors in the NVAR model

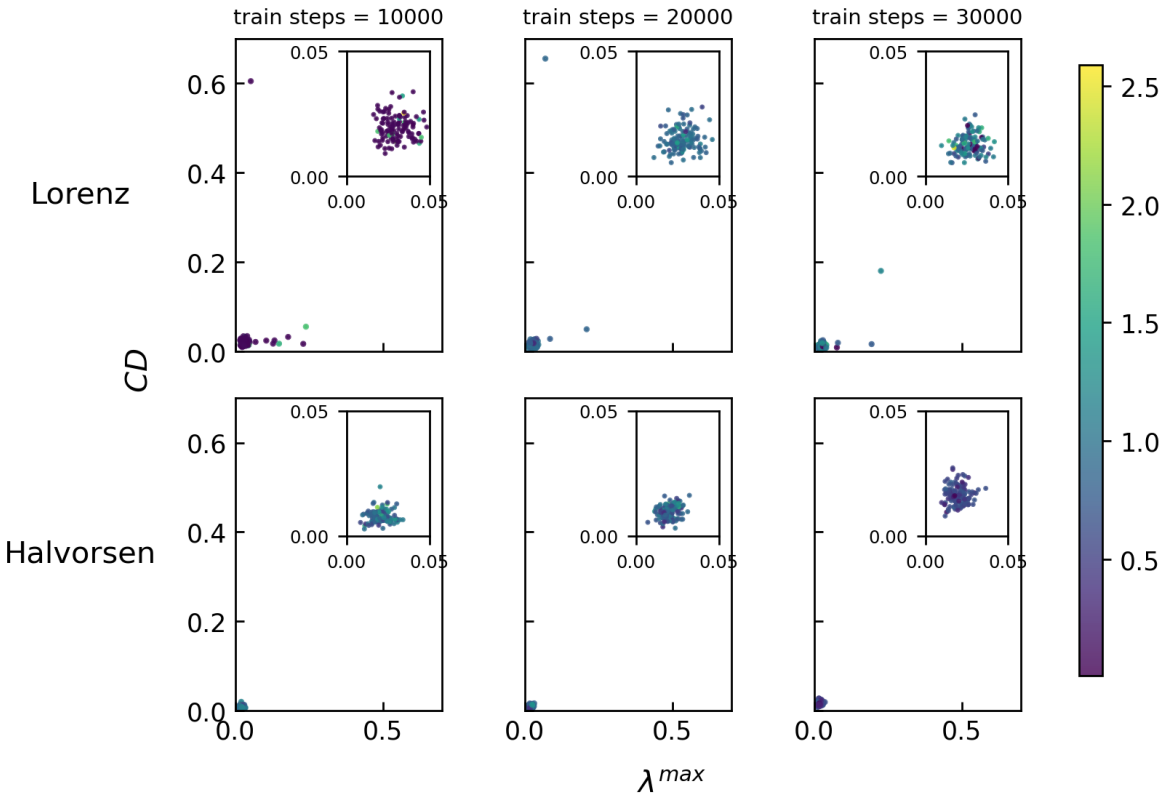


Figure B.6.: Standard deviation for overlapping attractors in the NML model

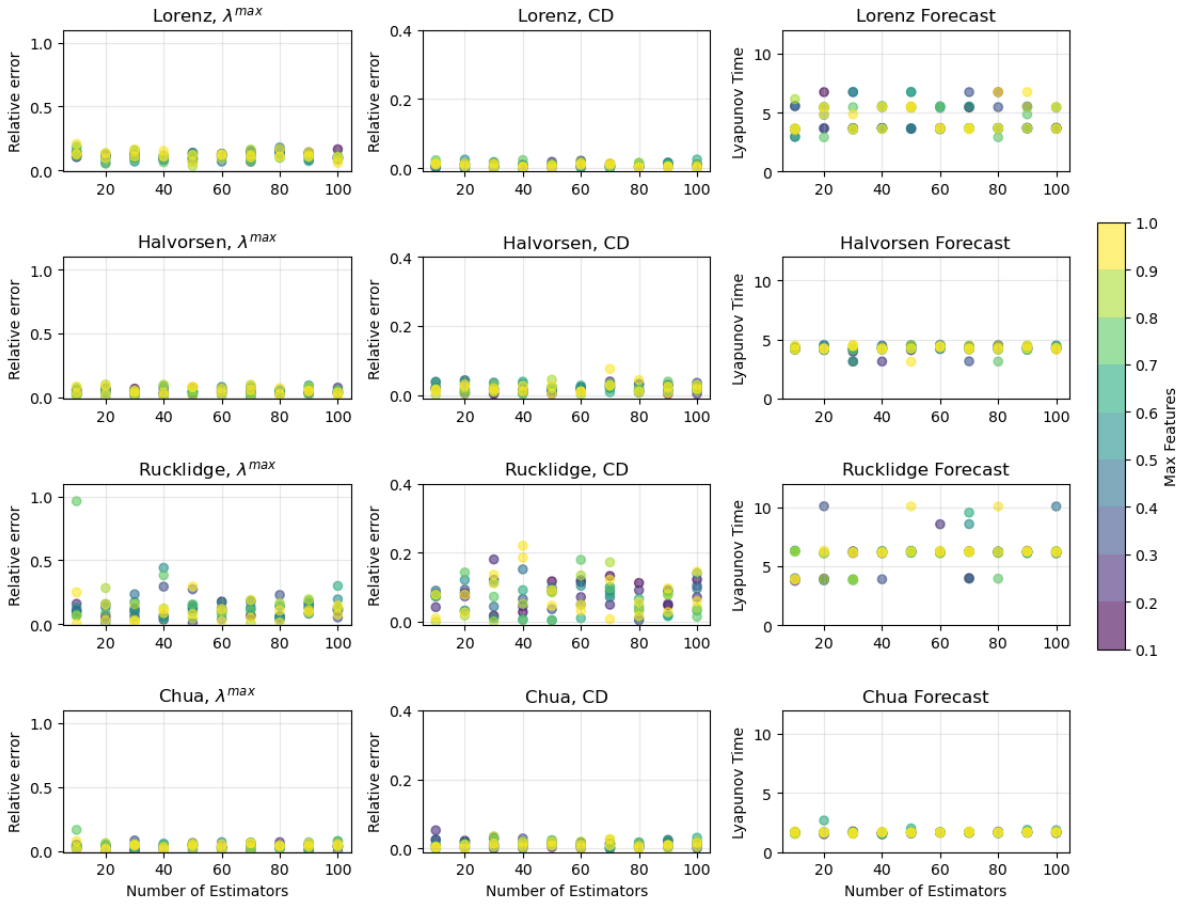


Figure B.7.: Relative errors for all attractors

Attractors / Measures	λ_{test}	λ_{pred}	C_{test}	C_{pred}	FH	Attractor parameter
ComplexButterfly_0	0.157	0.141	2.112	1.929	4.73	-20
Roessler_1	0.064	0.065	1.838	1.783	3.216	-18
Chua_2	0.379	0.431	1.88	1.881	4.831	-15
Chen_3	1.485	1.546	2.116	2.039	0.594	-12
Halvorsen_4	0.702	0.759	1.943	1.859	1.671	-10
Chua_5	0.379	0.381	1.88	1.881	3.248	-7
Lorenz_6	0.851	0.872	2.042	2.006	3.149	-4
Roessler_7	0.064	0.062	1.837	1.72	3.223	-2
Windmi_8	0.072	0.08	1.894	1.834	1.855	1
Rucklidge_9	0.237	0.238	1.91	1.827	1.737	4
Chen_10	1.485	1.618	2.116	2.079	1.871	6
Windmi_11	0.072	0.08	1.894	1.838	2.339	9
ComplexButterfly_12	0.157	0.139	2.112	2.029	5.202	12
Halvorsen_13	0.719	0.785	1.946	1.824	1.697	14
Rucklidge_14	0.237	0.22	1.909	1.863	1.77	17
Lorenz_15	0.851	0.923	2.04	2.011	3.029	20

Table B.1.: Measures for all 16 attractors

Bibliography

[1] H. H. Frisinger. "Aristotle and His "Meteorologica"". In: *Bulletin of the American Meteorological Society* 53.7 (July 1, 1972), pp. 634–638. ISSN: 0003-0007, 1520-0477. DOI: 10.1175/1520-0477(1972)053<0634:AAH>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/bams/53/7/1520-0477_1972_053_0634_aah_2_0_co_2.xml (visited on 05/01/2025).

[2] F. Takens. "Detecting Strange Attractors in Turbulence". In: *Dynamical Systems and Turbulence, Warwick 1980*. Ed. by D. Rand and L.-S. Young. Berlin, Heidelberg: Springer, 1981, pp. 366–381. ISBN: 978-3-540-38945-3. DOI: 10.1007/BFb0091924.

[3] Z. C. Lipton, J. Berkowitz, and C. Elkan. *A Critical Review of Recurrent Neural Networks for Sequence Learning*. Oct. 17, 2015. DOI: 10.48550/arXiv.1506.00019. arXiv: 1506.00019 [cs]. URL: http://arxiv.org/abs/1506.00019 (visited on 04/30/2025). Pre-published.

[4] U. Güçlü and M. A. J. van Gerven. "Modeling the Dynamics of Human Brain Activity with Recurrent Neural Networks". In: *Frontiers in Computational Neuroscience* 11 (2017), p. 7. ISSN: 1662-5188. DOI: 10.3389/fncom.2017.00007. PMID: 28232797.

[5] T. C. Kietzmann, C. J. Spoerer, L. K. A. Sörensen, R. M. Cichy, O. Hauk, and N. Kriegeskorte. "Recurrence Is Required to Capture the Representational Dynamics of the Human Visual System". In: *Proceedings of the National Academy of Sciences* 116.43 (Oct. 22, 2019), pp. 21854–21863. DOI: 10.1073/pnas.1905544116. URL: https://www.pnas.org/doi/10.1073/pnas.1905544116 (visited on 04/30/2025).

[6] Y. Bengio, P. Simard, and P. Frasconi. "Learning Long-Term Dependencies with Gradient Descent Is Difficult". In: *IEEE Transactions on Neural Networks* 5.2 (Mar. 1994), pp. 157–166. ISSN: 1941-0093. DOI: 10.1109/72.279181. URL: https://ieeexplore.ieee.org/document/279181 (visited on 04/22/2025).

[7] H. Jaeger. "The "Echo State" Approach to Analysing and Training Recurrent Neural Networks". In: (2001). DOI: 10.24406/publica-fhg-291111. URL: https://publica.fraunhofer.de/handle/publica/291111 (visited on 04/28/2025).

[8] W. Maass, T. Natschläger, and H. Markram. "Real-Time Computing without Stable States: A New Framework for Neural Computation Based on Perturbations". In: *Neural Computation* 14.11 (Nov. 2002), pp. 2531–2560. ISSN: 0899-7667. DOI: 10.1162/089976602760407955. PMID: 12433288.

[9] D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt. "An Experimental Unification of Reservoir Computing Methods". In: *Neural Networks*. Echo State Networks and Liquid State Machines 20.3 (Apr. 1, 2007), pp. 391–403. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2007.04.003. URL: https://www.sciencedirect.com/science/article/pii/S089360800700038X (visited on 04/30/2025).

[10] H. Zhang and D. V. Vargas. "A Survey on Reservoir Computing and Its Interdisciplinary Applications Beyond Traditional Machine Learning". In: *IEEE Access* 11 (2023), pp. 81033–81070. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2023.3299296. URL: https://ieeexplore.ieee.org/document/10196105 (visited on 05/01/2025).

[11] M. Yan, C. Huang, P. Bienstman, P. Tino, W. Lin, and J. Sun. "Emerging Opportunities and Challenges for the Future of Reservoir Computing". In: *Nature Communications* 15.1 (Mar. 6, 2024), p. 2056. ISSN: 2041-1723. DOI: 10.1038/s41467-024-45187-1. URL: https://www.nature.com/articles/s41467-024-45187-1 (visited on 04/22/2025).

[12] Z. Lu, B. R. Hunt, and E. Ott. "Attractor Reconstruction by Machine Learning". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6 (June 22, 2018), p. 061104. ISSN: 1054-1500. DOI: 10.1063/1.5039508. URL: https://doi.org/10.1063/1.5039508 (visited on 04/30/2025).

[13] A. Flynn, V. A. Tsachouridis, and A. Amann. "Multifunctionality in a Reservoir Computer". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (Jan. 12, 2021), p. 013125. ISSN: 1054-1500. DOI: 10.1063/5.0019974. URL: https://doi.org/10.1063/5.0019974 (visited on 04/07/2025).

[14] Z. Lu and D. S. Bassett. "Invertible Generalized Synchronization: A Putative Mechanism for Implicit Learning in Neural Systems". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.6 (June 16, 2020), p. 063133. ISSN: 1054-1500. DOI: 10.1063/5.0004344. URL: https://doi.org/10.1063/5.0004344 (visited on 04/30/2025).

[15] H. Ma, D. Prosperino, and C. Räth. "A Novel Approach to Minimal Reservoir Computing". In: *Scientific Reports* 13.1 (Aug. 10, 2023), p. 12970. ISSN: 2045-2322. DOI: 10.1038/s41598-023-39886-w. URL: https://www.nature.com/articles/s41598-023-39886-w (visited on 04/30/2025).

[16] D. Duncan and C. Räth. "Optimizing the Combination of Data-Driven and Model-Based Elements in Hybrid Reservoir Computing". In: *Chaos (Woodbury, N.Y.)* 33.10 (Oct. 1, 2023), p. 103109. ISSN: 1089-7682. DOI: 10.1063/5.0164013. PMID: 37831789.

[17] K. Nakajima. "Physical Reservoir Computing – An Introductory Perspective". In: *Japanese Journal of Applied Physics* 59.6 (June 1, 2020), p. 060501. ISSN: 0021-4922, 1347-4065. DOI: 10.35848/1347-4065/ab8d4f. arXiv: 2005.00992 [nlin]. URL: http://arxiv.org/abs/2005.00992 (visited on 04/22/2025).

[18] E. Bollt. "On Explaining the Surprising Success of Reservoir Computing Forecaster of Chaos? The Universal Machine Learning Dynamical System with Contrast to VAR and DMD". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 31.1 (Jan. 4, 2021), p. 013108. ISSN: 1054-1500. DOI: 10.1063/5.0024890. URL: https://doi.org/10.1063/5.0024890 (visited on 04/13/2025).

[19] A. Flynn, O. Heilmann, D. Köglmayr, V. Tsachouridis, C. Räth, and A. Amann. "Exploring the Limits of Multifunctionality across Different Reservoir Computers". In: *2022 International Joint Conference on Neural Networks, IJCNN 2022*. 2022 International Joint Conference on Neural Networks (IJCNN). Padua, Italien: IEEE, 2022. ISBN: 978-1-7281-8671-9. URL: https://ieeexplore.ieee.org/document/9892203 (visited on 04/22/2025).

[20] P. Geurts, D. Ernst, and L. Wehenkel. "Extremely Randomized Trees". In: *Machine Learning* 63.1 (Apr. 1, 2006), pp. 3–42. ISSN: 1573-0565. DOI: 10.1007/s10994-006-6226-1. URL: https://doi.org/10.1007/s10994-006-6226-1 (visited on 04/07/2025).

[21] scikit-learn. *sklearn.ensemble.ExtraTreesRegressor, version 1.6.1*. https://scikit-learn.org. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html. 2025.

[22] A. Giammarese, K. Rana, E. M. Bollt, and N. Malik. *Tree-Based Learning for High-Fidelity Prediction of Chaos*. Mar. 12, 2024. DOI: 10.48550/arXiv.2403.13836. arXiv: 2403.13836 [cs]. URL: http://arxiv.org/abs/2403.13836 (visited on 04/21/2025). Pre-published.

[23] D. J. Gauthier, A. Pomerance, and E. Bollt. *Locality Blended Next Generation Reservoir Computing For Attention Accuracy*. Mar. 30, 2025. DOI: 10.48550/arXiv.2503.23457. arXiv: 2503.23457 [nlin]. URL: http://arxiv.org/abs/2503.23457 (visited on 04/21/2025). Pre-published.

[24] T. C. Sideris. *Ordinary Differential Equations and Dynamical Systems*. 2013.

[25] J. .-. Eckmann and D. Ruelle. "Ergodic Theory of Chaos and Strange Attractors". In: *Reviews of Modern Physics* 57.3 (July 1, 1985), pp. 617–656. DOI: 10.1103/RevModPhys.57.617. URL: https://link.aps.org/doi/10.1103/RevModPhys.57.617 (visited on 04/22/2025).

[26] E. N. Lorenz. "Deterministic Nonperiodic Flow". In: *Journal of the Atmospheric Sciences* 20.2 (Mar. 1, 1963), pp. 130–141. ISSN: 0022-4928, 1520-0469. DOI: 10.1175/1520-0469(1963)020<0130:DNF>2.0.CO;2. URL: https://journals.ametsoc.org/view/journals/atsc/20/2/1520-0469_1963_020_0130_dnf_2_0_co_2.xml (visited on 04/27/2025).

[27] E. Ott. *Chaos in Dynamical Systems*. 2nd ed. Cambridge: Cambridge University Press, 2002. ISBN: 978-0-521-01084-9. DOI: 10.1017/CBO9780511803260. URL: https://www.cambridge.org/core/books/chaos-in-dynamical-systems/7A0749AE3FBBF4312A54D7573C2DAAB5 (visited on 04/27/2025).

[28] J. C. Sprott. *Chaos and Time-Series Analysis*. Oxford University Press, Jan. 2003. ISBN: 978-0-19-850839-7. DOI: 10.1093/oso/9780198508397.001.0001. URL: https://doi.org/10.1093/oso/9780198508397.001.0001.

[29] A. M. Rucklidge. "Chaos in Models of Double Convection". In: *Journal of Fluid Mechanics* 237 (Apr. 1992), pp. 209–229. ISSN: 1469-7645, 0022-1120. DOI: 10.1017/S0022112092003392. URL: https://www.cambridge.org/core/journals/journal-of-fluid-mechanics/article/abs/chaos-in-models-of-double-convection/29A1D82111941E50EE0D4A31DF09BE1D (visited on 04/22/2025).

[30] O. E. Rössler. "An Equation for Continuous Chaos". In: *Physics Letters A* 57.5 (July 12, 1976), pp. 397–398. ISSN: 0375-9601. DOI: 10.1016/0375-9601(76)90101-8. URL: https://www.sciencedirect.com/science/article/pii/0375960176901018 (visited on 04/22/2025).

[31] T. Matsumoto, L. Chua, and M. Komuro. "The Double Scroll". In: *IEEE Transactions on Circuits and Systems* 32.8 (Aug. 1985), pp. 797–818. ISSN: 1558-1276. DOI: 10.1109/TCS.1985.1085791. URL: https://ieeexplore.ieee.org/document/1085791 (visited on 04/22/2025).

[32] G. Chen and T. Ueta. "Yet Another Chaotic Attractor". In: *International Journal of Bifurcation and Chaos* 09.07 (July 1999), pp. 1465–1466. ISSN: 0218-1274. DOI: 10.1142/S0218127499001024. URL: https://www.worldscientific.com/doi/10.1142/S0218127499001024 (visited on 04/22/2025).

[33] W. Horton and I. Doxas. "A Low-Dimensional Dynamical Model for the Solar Wind Driven Geotail-Ionosphere System". In: *Journal of Geophysical Research: Space Physics* 103.A3 (1998), pp. 4561–4572. ISSN: 2156-2202. DOI: 10.1029/

97JA02417. URL: https://onlinelibrary.wiley.com/doi/abs/10.1029/97JA02417 (visited on 05/02/2025).

[34] A. S. Elwakil, S. Özoğuz, and M. P. Kennedy. "Creation of a Complex Butterfly Attractor Using a Novel Lorenz-type System". In: *IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications* 49.4 (Apr. 1, 2002), pp. 527–530. DOI: 10.1109/81.995671. URL: https://digitalcommons.memphis.edu/facpubs/17392.

[35] M. Cucchi, S. Abreu, G. Ciccone, D. Brunner, and H. Kleemann. "Hands-on Reservoir Computing: A Tutorial for Practical Implementation". In: *Neuromorphic Computing and Engineering* 2.3 (Aug. 2022), p. 032002. ISSN: 2634-4386. DOI: 10.1088/2634-4386/ac7db7. URL: https://dx.doi.org/10.1088/2634-4386/ac7db7 (visited on 05/01/2025).

[36] L. Gonon and J.-P. Ortega. "Reservoir Computing Universality With Stochastic Inputs". In: *IEEE transactions on neural networks and learning systems* 31.1 (Jan. 2020), pp. 100–112. ISSN: 2162-2388. DOI: 10.1109/TNNLS.2019.2899649. PMID: 30892244.

[37] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. S. Barbosa. "Next Generation Reservoir Computing". In: *Nature Communications* 12.1 (Sept. 21, 2021), p. 5564. ISSN: 2041-1723. DOI: 10.1038/s41467-021-25801-2. URL: https://www.nature.com/articles/s41467-021-25801-2 (visited on 04/07/2025).

[38] D. Köglmayr and C. Räth. "Extrapolating Tipping Points and Simulating Non-Stationary Dynamics of Complex Systems Using Efficient Machine Learning". In: *Scientific Reports* 14.1 (Jan. 4, 2024), p. 507. ISSN: 2045-2322. DOI: 10.1038/s41598-023-50726-9. URL: https://www.nature.com/articles/s41598-023-50726-9 (visited on 04/30/2025).

[39] D. Köglmayr, A. Haluszczynski, and C. Räth. *Controlling Dynamical Systems into Unseen Target States Using Machine Learning*. Dec. 13, 2024. DOI: 10.48550/arXiv.2412.10251. arXiv: 2412.10251 [nlin]. URL: http://arxiv.org/abs/2412.10251 (visited on 04/30/2025). Pre-published.

[40] A. Navada, A. N. Ansari, S. Patil, and B. A. Sonkamble. "Overview of Use of Decision Tree Algorithms in Machine Learning". In: *2011 IEEE Control and System Graduate Research Colloquium*. 2011 IEEE Control and System Graduate Research Colloquium. June 2011, pp. 37–42. DOI: 10.1109/ICSGRC.2011.5991826. URL: https://ieeexplore.ieee.org/document/5991826 (visited on 04/28/2025).

[41] C. Kern, T. Klausch, and F. Kreuter. "Tree-Based Machine Learning Methods for Survey Research". In: *Survey research methods* 13.1 (Apr. 11, 2019), pp. 73–93. ISSN: 1864-3361. PMID: 32802211. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7425836/ (visited on 04/28/2025).

[42] P. Geurts and L. Wehenkel. "Investigation and Reduction of Discretization Variance in Decision Tree Induction". In: *Machine Learning: ECML 2000*. Ed. by R. López de Mántaras and E. Plaza. Berlin, Heidelberg: Springer, 2000, pp. 162–170. ISBN: 978-3-540-45164-8. DOI: 10.1007/3-540-45164-1_17.

[43] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY: Springer, 2009. ISBN: 978-0-387-84857-0. DOI: 10.1007/978-0-387-84858-7. URL: http://link.springer.com/10.1007/978-0-387-84858-7 (visited on 05/01/2025).

[44] A. Haluszczynski and C. Räth. "Good and Bad Predictions: Assessing and Improving the Replication of Chaotic Attractors by Means of Reservoir Computing". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.10 (Oct. 25, 2019), p. 103143. ISSN: 1054-1500. DOI: 10.1063/1.5118725. URL: https://doi.org/10.1063/1.5118725 (visited on 04/28/2025).

[45] K. Geist, U. Parlitz, and W. Lauterborn. "Comparison of Different Methods for Computing Lyapunov Exponents". In: *Progress of Theoretical Physics* 83.5 (May 1, 1990), pp. 875–893. ISSN: 0033-068X, 1347-4081. DOI: 10.1143/PTP.83.875. URL: https://academic.oup.com/ptp/article-lookup/doi/10.1143/PTP.83.875 (visited on 04/28/2025).

[46] M. T. Rosenstein, J. J. Collins, and C. J. De Luca. "A Practical Method for Calculating Largest Lyapunov Exponents from Small Data Sets". In: *Physica D: Nonlinear Phenomena* 65.1 (May 15, 1993), pp. 117–134. ISSN: 0167-2789. DOI: 10.1016/0167-2789(93)90009-P. URL: https://www.sciencedirect.com/science/article/pii/016727899390009P (visited on 04/07/2025).

[47] J. Theiler. "Estimating Fractal Dimension". In: *Journal of the Optical Society of America A* 7 (June 1, 1990), pp. 1055–1073. ISSN: 0740-3232. DOI: 10.1364/JOSAA.7.001055. URL: https://ui.adsabs.harvard.edu/abs/1990JOSAA...7.1055T (visited on 04/28/2025).

[48] P. Grassberger and I. Procaccia. "Measuring the Strangeness of Strange Attractors". In: *Physica D: Nonlinear Phenomena* 9.1 (Oct. 1, 1983), pp. 189–208. ISSN: 0167-2789. DOI: 10.1016/0167-2789(83)90298-1. URL: https://www.sciencedirect.com/science/article/pii/0167278983902981 (visited on 04/07/2025).

[49] J. Herteux. "The Influence of the Activation Function on Reservoir Computers". MA thesis. Ludwig-Maximilians-Universität München, 2021. URL: https://elib.dlr.de/141730/ (visited on 04/28/2025).

[50] P. A. Getting. "Emerging Principles Governing the Operation of Neural Networks". In: *Annual Review of Neuroscience* 12 (1989), pp. 185–204. ISSN: 0147-006X. DOI: 10.1146/annurev.ne.12.030189.001153. PMID: 2648949.

[51] K. L. Briggman and W. B. Kristan. "Imaging Dedicated and Multifunctional Neural Circuits Generating Distinct Behaviors". In: *Journal of Neuroscience* 26.42 (Oct. 18, 2006), pp. 10925–10933. ISSN: 0270-6474, 1529-2401. DOI: 10.1523/JNEUROSCI.3265-06.2006. PMID: 17050731. URL: https://www.jneurosci.org/content/26/42/10925 (visited on 04/22/2025).

[52] P. S. Dickinson. "Interactions among Neural Networks for Behavior". In: *Current Opinion in Neurobiology* 5.6 (Dec. 1995), pp. 792–798. ISSN: 0959-4388. DOI: 10.1016/0959-4388(95)80108-1. PMID: 8805420.

[53] W. B. Kristan, G. Wittenberg, M. P. Nusbaum, and W. Stern-Tomlinson. "Multifunctional Interneurons in Behavioral Circuits of the Medicinal Leech". In: *Experientia* 44.5 (May 15, 1988), pp. 383–389. ISSN: 0014-4754. DOI: 10.1007/BF01940531. PMID: 3286283.

[54] F. T. Arecchi, R. Meucci, G. Puccioni, and J. Tredicce. "Experimental Evidence of Subharmonic Bifurcations, Multistability, and Turbulence in a Q-Switched Gas Laser". In: *Physical Review Letters* 49.17 (Oct. 25, 1982), pp. 1217–1220. DOI: 10.1103/PhysRevLett.49.1217. URL: https://link.aps.org/doi/10.1103/PhysRevLett.49.1217 (visited on 05/01/2025).

[55] G. Schwarz, C. Lehmann, and E. Schöll. "Self-Organized Symmetry-Breaking Current Filamentation and Multistability in Corbino Disks". In: *Physical Review B* 61.15 (Apr. 15, 2000), pp. 10194–10200. DOI: 10.1103/PhysRevB.61.10194. URL: https://link.aps.org/doi/10.1103/PhysRevB.61.10194 (visited on 05/01/2025).

[56] A. N. Pisarchik and U. Feudel. "Control of Multistability". In: *Physics Reports*. Control of Multistability 540.4 (July 30, 2014), pp. 167–218. ISSN: 0370-1573. DOI: 10.1016/j.physrep.2014.02.007. URL: https://www.sciencedirect.com/science/article/pii/S0370157314000453 (visited on 04/21/2025).

[57] J. Herteux and C. Räth. "Breaking Symmetries of the Reservoir Equations in Echo State Networks". In: *Chaos* 30.12 (2020), p. 123142. DOI: 10.1063/5.0028993.

[58] A. Flynn, V. A. Tsachouridis, and A. Amann. "Seeing Double with a Multifunctional Reservoir Computer". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 33.11 (Nov. 1, 2023), p. 113115. ISSN: 1054-1500, 1089-7682. DOI: 10.1063/5.0157648. URL: https://pubs.aip.org/cha/article/33/11/113115/2920250/Seeing-double-with-a-multifunctional-reservoir (visited on 04/07/2025).

[59] A. Haluszczynski, D. Köglmayr, and C. Räth. "Controlling Dynamical Systems to Complex Target States Using Machine Learning: Next-Generation vs. Classical Reservoir Computing". In: *2023 International Joint Conference on Neural Networks, IJCNN 2023*. International Joint Conference on Neural Networks (IJCNN). Gold Coast, Australien: IEEE, 2023. ISBN: 978-1-6654-8867-9. URL: https://ieeexplore.ieee.org/document/10191257 (visited on 04/30/2025).

[60] W. A. S. Barbosa and D. J. Gauthier. "Learning Spatiotemporal Chaos Using Next-Generation Reservoir Computing". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 32.9 (Sept. 26, 2022), p. 093137. ISSN: 1054-1500. DOI: 10.1063/5.0098707. URL: https://doi.org/10.1063/5.0098707 (visited on 04/30/2025).

[61] *How More Data Can Hurt: Instability and Regularization in next-Generation Reservoir Computing*. URL: https://arxiv.org/html/2407.08641v1 (visited on 04/22/2025).

[62] S. M. Lundberg and S.-I. Lee. *Consistent Feature Attribution for Tree Ensembles*. Feb. 17, 2018. DOI: 10.48550/arXiv.1706.06060. arXiv: 1706.06060 [cs]. URL: http://arxiv.org/abs/1706.06060 (visited on 04/30/2025). Pre-published.

[63] P. Wei, Z. Lu, and J. Song. "Variable Importance Analysis: A Comprehensive Review". In: *Reliability Engineering & System Safety* 142 (Oct. 1, 2015), pp. 399–432. ISSN: 0951-8320. DOI: 10.1016/j.ress.2015.05.018. URL: https://www.sciencedirect.com/science/article/pii/S0951832015001672 (visited on 04/30/2025).

[64] Y. Kessler, Y. Shencar, and N. Meiran. "Choosing to Switch: Spontaneous Task Switching despite Associated Behavioral Costs". In: *Acta Psychologica* 131.2 (June 1, 2009), pp. 120–128. ISSN: 0001-6918. DOI: 10.1016/j.actpsy.2009.03.005. URL: https://www.sciencedirect.com/science/article/pii/S0001691809000389 (visited on 04/26/2025).

[65] I. Tsuda. "Chaotic Itinerancy as a Dynamical Basis of Hermeneutics in Brain and Mind". In: *World Futures* 32.2–3 (Sept. 1, 1991), pp. 167–184. ISSN: 0260-4027. DOI: 10.1080/02604027.1991.9972257. URL: https://doi.org/10.1080/02604027.1991.9972257 (visited on 04/26/2025).

[66] I. Tsuda. "Chaotic Itinerancy and Its Roles in Cognitive Neurodynamics". In: *Current Opinion in Neurobiology*. SI: Brain Rhythms and Dynamic Coordination 31 (Apr. 1, 2015), pp. 67–71. ISSN: 0959-4388. DOI: 10.1016/j.conb.2014.08.011. URL: https://www.sciencedirect.com/science/article/pii/S0959438814001731 (visited on 04/26/2025).

[67] C. A. Skarda and W. J. Freeman. "How Brains Make Chaos in Order to Make Sense of the World". In: *Behavioral and Brain Sciences* 10.2 (1987), pp. 161–195. ISSN: 1469-1825. DOI: 10.1017/S0140525X00047336.

[68] P. Probst and A.-L. Boulesteix. *To Tune or Not to Tune the Number of Trees in Random Forest?* May 16, 2017. DOI: 10.48550/arXiv.1705.05654. arXiv: 1705.05654 [stat]. URL: http://arxiv.org/abs/1705.05654 (visited on 04/21/2025). Pre-published.

[69] R. M. Shiffrin and R. C. Atkinson. "Storage and Retrieval Processes in Long-Term Memory". In: *Psychological Review* 76.2 (1969), pp. 179–193. ISSN: 1939-1471. DOI: 10.1037/h0027277.

[70] E. Tulving and Z. Pearlstone. "Availability versus Accessibility of Information in Memory for Words". In: *Journal of Verbal Learning and Verbal Behavior* 5.4 (Aug. 1, 1966), pp. 381–391. ISSN: 0022-5371. DOI: 10.1016/S0022-5371(66)80048-8. URL: https://www.sciencedirect.com/science/article/pii/S0022537166800488 (visited on 04/26/2025).

[71] P. W. Frankland, S. A. Josselyn, and S. Köhler. "The Neurobiological Foundation of Memory Retrieval". In: *Nature Neuroscience* 22.10 (Oct. 2019), pp. 1576–1585. ISSN: 1546-1726. DOI: 10.1038/s41593-019-0493-1. URL: https://www.nature.com/articles/s41593-019-0493-1 (visited on 04/26/2025).

I confirm that this master's thesis is my own work and I have documented all sources and material used.

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.

Munich, 02.05.2025 Miralem Spahic