Neural Networks 204 (2026) 109209

Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Full Length Article

# Two-shot learning of multiple strange attractors

Daniel Köglmayr [a,1,*], Miralem Spahic [b,1], Andrew Flynn [c,d], Christoph Räth [e,b]

[a] Institut für KI-Sicherheit, Deutsches Zentrum für Luft- und Raumfahrt (DLR), Ulm, Germany
[b] Ludwig-Maximilians-Universität (LMU), München, Germany
[c] School of Mathematical Sciences, University College Cork, Cork, T12 XF62, Ireland
[d] INFANT Research Centre, University College Cork, Cork, T12 DC4A, Ireland
[e] Institut für Frontier Materials auf der Erde und im Weltraum, Deutsches Zentrum für Luft- und Raumfahrt (DLR), Köln, 51140, Germany

ARTICLE INFO

*Keywords:*
Reservoir computing
Machine learning
Chaotic systems
Attractor reconstruction
Multifunctionality
Extremely randomized trees

ABSTRACT

The brain combines short- and long-term memory to process, store, and recall multiple different pieces of information. Inspired by this and recent results on multifunctional and parameter-aware learning, we extend a new machine learning technique that combines short- and long-term memory units, specifically, a system consisting of a next-generation reservoir computer (NGRC) and extremely randomized trees (ERT), to process, store, and recall multiple different strange attractors. We train the combined NGRC + ERT system using a two-shot learning approach which significantly improves performance by filtering out unnecessary features, thereby avoiding extensive hyperparameter optimization. We first show that an NGRC + ERT system achieves highly accurate reconstruction of the short- and long-term dynamics of both the Lorenz and Halvorsen chaotic attractors when using an exponential filtering scheme. We validate these findings by training the NGRC + ERT system to reconstruct 16 different attractors and show that sufficient index-based separation in feature space suppresses unwanted switching dynamics, thus stabilizing long-term memory recall. Finally, we identify that defects in short-term memory processing can provoke failure modes in long-term memory recall resulting in confabulation.

## 1. Introduction

The brain has been a constant source of inspiration for advancing and understanding the capabilities of artificially intelligent systems. The mathematics of dynamical systems theory often acts as a bridge between these worlds where developments in one area can inform the other (Raut et al., 2025). In this paper we take inspiration from the brain's ability to combine short- and long-term memory to rapidly process, store, and recall multiple complex temporal patterns. To do this, certain biological neural networks use 'multifunctionality', when a single network exploits its capacity for multistable dynamics to perform multiple different tasks (Dickinson, 1995; Getting, 1989; Marder & Calabrese, 1996). With advances in machine learning, researchers are increasingly focused on developing computational models that capture key characteristics of biological neural networks, including their ability to process, store and recall temporal information or adapt to changing inputs (Kim et al., 2021; Yamazaki et al., 2022). These efforts bridge neuroscience and modern machine learning, offering new tools for modeling neural computation. In particular, recent studies demonstrate that multifunctional reservoir computers (RCs) in the form of recurrent neural networks (Flynn et al., 2022, 2021a,b; Herteux & Räth, 2020; Kong et al., 2024) and tenseg-rity robots (Terajima et al., 2025), provide an effective framework for storing and retrieving temporal information from multiple dynamical systems within a single system. While traditional networks, such as Hopfield networks are limited to storing static patterns, RCs can process, store, and recall multiple strange attractors through either index-based (location-addressable/parameter-aware) or multistability-based (content-addressable/multifunctional) approaches (Kong et al., 2024). Building on these memory storage capabilities, researchers have began investigating the associated failure modes and limitations of RCs (O'Hagan et al., 2025).

Training a single RC to reconstruct more than one attractor comes with significant challenges. In the case of multifunctionality, optimizing the RC requires extensive hyperparameter tuning, particularly when reconstructing overlapping attractors (Flynn et al., 2022, 2023). Classical RCs use high-dimensional recurrent networks with fixed weights and a trainable readout layer, typically trained using ridge regression. Recent work proved the mathematical equivalence of nonlinear autoregressive models and RCs (Bollt, 2021), inspiring next-generation reservoir computing which replaces recurrent networks with explicit monomial expansions of time-delayed data points, offering reduced computational cost, fewer hyperparameters, and improved interpretability

* Corresponding author.
*E-mail address:* daniel.koeglmayr@dlr.de (D. Köglmayr).
[1] These authors contributed equally to this work.

https://doi.org/10.1016/j.neunet.2026.109209
Received 3 February 2026; Received in revised form 20 May 2026; Accepted 1 June 2026
Available online 4 June 2026

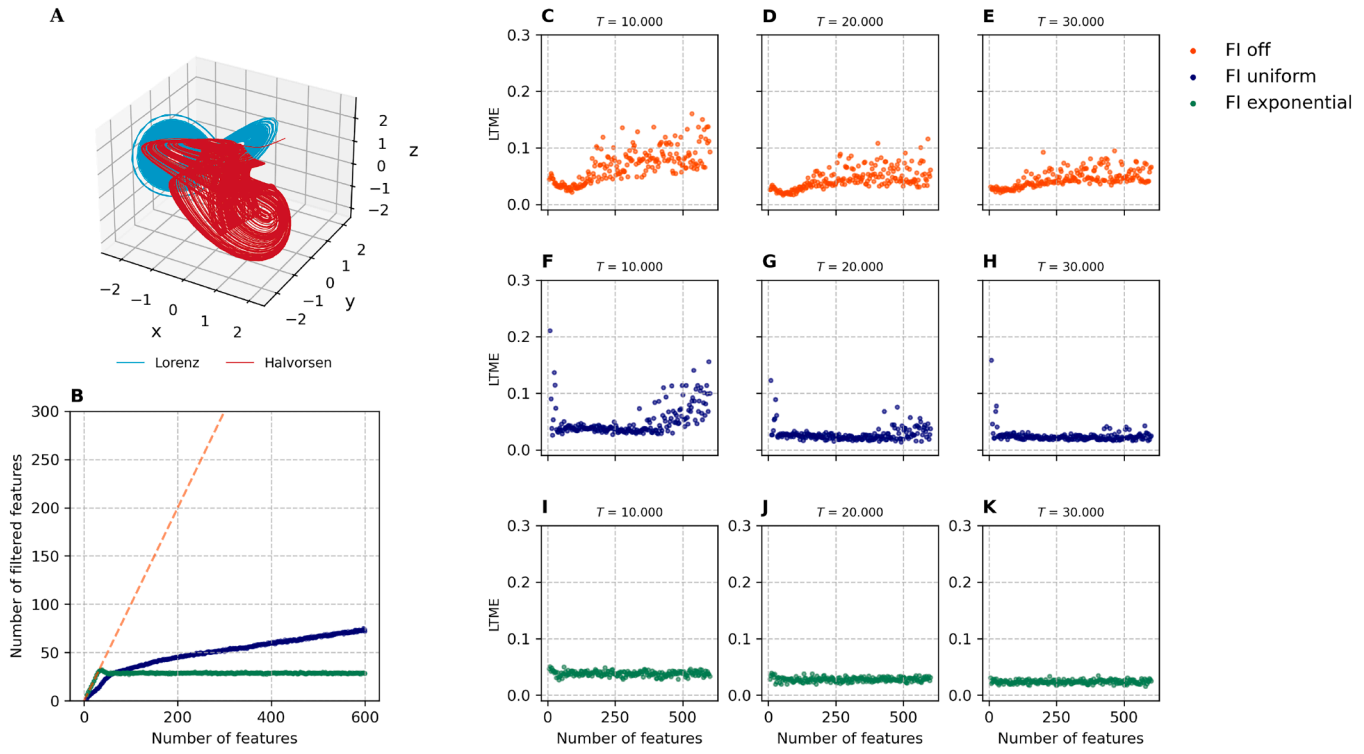


**Fig. 1.** Feature importance filtering improves attractor reconstruction in VAR feature model for the Lorenz and Halvorsen task. **A** Overlapping trajectories of the Lorenz and Halvorsen attractors after normalization. **B** Comparison of filtered versus initial features for three approaches: No filtering (base model) (top), uniform filtering (middle), and exponential filtering (bottom) for measured feature importance FI. **C-E** Long-term measurement error for VAR hyperparameter configurations without FI filtering. **F-G** Long-term measurement error using uniform filtering. **I-K** Long-term measurement error (LTME) using exponential filtering. Each data point represents the mean over ten random initializations of ERT instances.

(Gauthier et al., 2021). More recently, Giammarese et al. (2025) demonstrated that the output layer of a next-generation reservoir computer (NGRC) can be successfully trained using Extremely Randomized Trees (ERT), a tree-based ensemble method, rather than conventional ridge regression and show their approach exhibiting improved robustness and high-fidelity prediction of chaotic attractors. Crucially, the tree-based approach enables extraction of feature importance measures directly from the trained trees, allowing for substantial improvements in hyperparameter tuning by retraining with features identified as relevant for the specific task.

Given that NGRC + ERT systems achieve superior performance with less hyperparameter tuning than traditional RCs, and that traditional RCs require extensive hyperparameter tuning for multifunctional and parameter-aware learning, in this paper we extend the capabilities of NGRC + ERT systems to these domains, yielding a new machine learning framework that efficiently processes, stores, and recalls multiple complex temporal patterns without extensive tuning. Further, in the brain, short-term memory is primarily associated with the prefrontal cortex, long-term memory with the cerebral cortex, and the hippocampus serves as the critical bridge between these regions to enable memory consolidation (Queensland Brain Institute , n.d.; Tulving, 1995). In our case, the NGRC acts a short-term memory unit, the ERT as a long-term memory unit, and by combining these we produce a NGRC + ERT system capable of achieving accurate short-term predictions and long-term reconstruction of multiple strange attractors. We introduce a two-shot learning methodology which significantly improves performance and reduces the need of extensive hyperparameter tuning than single-shot learning.

For the first-shot, we use NGRC hyperparameter configurations to generate an initial feature representation of the input data and train an ERT model. For the second-shot, we employ feature importance metrics to identify important features, to train a new and refined NGRC + ERT system which significantly outperforms the initial first-shot NGRC + ERT system. Interestingly, the refined NGRC + ERT systems place a greater emphasis on features relating to short-term memory processing. To scale this approach for learning multiple dynamical attractors, we use an index-based (location-addressable) approach similar to Kong et al. (2024) which allows the NGRC + ERT system to operate in either a multifunctional or parameter-aware style setup.

Our overall approach offers two key advantages over current methods. First, it enables interpretable separation of short-term and long-term memory capabilities of a single system, the NGRC + ERT system, with the NGRC unit handling the short-term memory processing through time-delay features and the ERT unit handling the long-term memory recall of the temporal dynamics. Second, it enables optimization via the tree-based feature importance measures, making extensive hyperparameter optimization superfluous. We develop and test the approach on the well-studied overlapping Lorenz and Halvorsen task (Flynn et al., 2022; Herteux & Räth, 2020; O'Hagan et al., 2025) and validate our findings on similar Chua and Halvorsen, and Rucklidge and Windmi tasks. When applied to a 16-overlapping-attractor task, we find that without sufficient feature separation the NGRC + ERT system suffers from unwanted switching dynamics like in Flynn and Amann (2024) and Kabayama et al. (2025), indicating long-term memory recall problems. We further apply this method on a 49-attractor task and show that the NGRC + ERT system can perform location-addressable memory itinerancy by changing the active index during recall. Consequently, the interpretable nature of short-term memory processing through time-delay features allows us to analyze how defects in short-term memory, resembling deterioration of the system in 'on-chip' applications, trigger long-term memory recall errors, causing the NGRC + ERT system to generate confabulations in the form of 'generated attractors' - a malfunction that may represent a fundamental characteristic shared across various learning systems (O'Hagan et al., 2025).

## 2. Results

### 2.1. Two attractor task

This subsection presents results for the overlapping Lorenz and Halvorsen task (see Methods). We train an NGRC + ERT system to reconstruct both the Lorenz and Halvorsen chaotic attractors (Fig. 1**A**) using different amounts of linear time-delayed features (VAR feature model, see Methods). We compare the performance of NGRC + ERT systems that are trained without two-shot learning and with two-shot learning using different feature importance metrics. The number of features created for each NGRC + ERT system after the first shot of learning is $N_f = k \times d$, where $k$ is a time-delay parameter used to denote the number of time-delayed data points and $d = 3$ represents the dimension of the corresponding Lorenz and Halvorsen systems. We evaluated the performance of each NGRC + ERT system for different choices of $k$ from 2 to 200. Note, we keep the time steps between the time-delayed data points fixed at $s = 1$. Here, we employ an index-based separation strategy to distinguish between attractors in feature space. Specifically, we add a scaled constant $\theta_0 = -1$ times scaling parameter $\gamma = 1$ to each feature derived from the Lorenz data, likewise $\theta_1 = 1$ times $\gamma = 1$ to the features derived from the Halvorsen data. This modification enables the NGRC + ERT system to differentiate between the two attractors during both training and prediction (see Supplementary Note S11). Hence, the NGRC + ERT system reconstructs a different attractor depending on the choice of $\theta$. We assess attractor reconstruction accuracy using metrics that quantify short- and long-term prediction behavior, as detailed in the Methods section. Results on short-term prediction behavior are provided in Figure S2 of the Supplementary Materials, along with supplementary analysis for the NVAR and NML feature models. Fig. 1(**C**-**E**) show the long-term measurement error (LTME) of the reconstructed attractors vs. $N_f$ for different training sizes $T$ when the NGRC + ERT system is trained without two-shot learning. We observe a minimum in the error metric for a similar $N_f$ across all training sizes, indicating that there is an optimal choice of $k$ that enables the NGRC + ERT system to perform accurate long-term reconstruction of each attractor. Notably, this performance deteriorates significantly when $k$ is too large but can be improved by using larger $T$.

In the following analyses we train NGRC + ERT systems that are trained with our two-shot learning approach. We evaluate the feature importance (FI) measure inherent to the ERT (see Methods) to optimize the feature configuration. In short, we train a new NGRC + ERT system with features whose importance exceeds the threshold $\mathrm{I}_c$. We employ the 'uniform filtering scheme' introduced in the TreeDOX framework (Giammarese et al., 2025), in which the threshold is defined as $\mathrm{I}_c = 1/N_f$. Fig. 1(**F** − **H**) show the resultant LTME from uniform filtering. We find that long-term reconstruction improves in all analyzed training sizes, with the improvement being particularly prominent for larger training sizes. Comparing Fig. 1(**C**) to (**F**), we observe that uniform filtering broadens the range of $N_f$ values where similar long-term measurement errors are obtained, specifically $N_f$ between approximately 60 and 350. For small values of $N_f$, the uniform filtering approach results in higher errors, while for $N_f > 350$ no notable improvement is observed. Results on short-term prediction behavior and supplementary analysis for the NVAR and NML feature models are provided in Figure S3 of the Supplementary Materials.

To counteract the increasing error distribution observed, we introduce an 'exponential filtering scheme'. We set the cut-off threshold to the mean lifetime of exponential decay, calculated as $\mathrm{I}_c = \mathrm{I}_{\max}/e$, where $\mathrm{I}_{\max}$ is the maximum measured feature importance value. This way the threshold is positioned at the steep decay between clusters of high-importance features and the tails of low-importance features in the feature importance distributions observed (Supplementary Note S2). We compare this threshold in Supplementary Note S3 against alternative tree-based importance metrics derived from SHAP and permutation importance.

In Fig. 1(**I** − **K**) we plot the resultant LTME from exponential filtering versus $N_f$. These show that the exponential filtering scheme provides low and constant LTME for each $N_f$ and that increasing $T$ results in marginally smaller LTME.

Furthermore, our results reveal other distinct differences between the uniform and exponential filtering schemes. For instance, Fig. 1**B** shows the number of filtered features versus the number of features that are used to initially train the NGRC + ERT system. For uniform filtering (blue curve) we observe an almost linear relationship for $N_f > 100$. In contrast, exponential filtering (green curve) converges toward a constant number as $N_f$ increases. Further, for small $N_f$, exponential filtering performs no feature reduction, whereas uniform filtering does, i.e., deviates from the orange curve. Based on these results, the increase in LTME for $N_f > 350$ in Fig. 1(**C**-**E**) may be correlated with the number of filtered features increasing past a certain point in the uniform filter case. Supplementary Figures S4– S6 show the short-term prediction behavior, reconstructed attractors obtained with exponential filtering, and relative improvements achieved by both filtering schemes compared to results without filtering, respectively. The exponential filtering scheme achieves LTMEs across all $N_f$ similar to the minimal LTME obtained without filtering, thereby eliminating the need for extensive hyperparameter optimization.

In the Supplementary Materials we add further weight to our statement that two-shot learning with exponential filtering eliminates the need for extensive hyperparameter optimization. We validate our findings by showing that similar behavior is observed across hyperparameter configurations before filtering when using different attractors as training data, specifically, we analyze the proposed exponential filter scheme on overlapping Chua and Halvorsen attractors as well as on an overlapping Rucklidge and Windmi attractors in the Supplementary Notes S4 and S5.

Supplementary Note S2 complements these results by showing the feature importance distributions obtained after the first-shot for the three overlapping attractor tasks, and comparing the resulting uniform and exponential thresholds and how the approaches differ in retaining features. Additionally, we benchmark the two-shot learning approach with exponential filtering against multifunctional traditional RC (Herteux & Räth, 2020) and multifunctional NGRC + Ridge (Köglmayr, 2022) on the well-studied non-overlapping Lorenz and Halvorsen task in Supplementary Note S6. The NGRC + ERT system reproduces the climate of both attractors within the published tolerances and substantially outperforms NGRC + Ridge in multifunctional success rate, though its short-term prediction accuracy on the non-overlapping Lorenz and Halverson task does not match that of the reported results for RC.

### 2.2. Multiple attractor task

This subsection presents results from training an NGRC + ERT system to reconstruct more than two attractors. Specifically, we train the NGRC + ERT system on the 16-overlapping-attractor task specified in the Methods section, see Fig. 2**A** for a plot of these attractors overlapping in the same space. Informed by Fig. 1, we keep the training parameters as $T = 20{,}000$, $k = 100$, $s = 1$, choose $\gamma = 25$ and assign a unique identifier $\theta \in \{-8, -7, \ldots, 7, 8\} \setminus \{0\}$ to each attractor. We train the NGRC + ERT system using the same two-shot learning approach with the exponential filtering scheme. For this task we provide a more detailed breakdown of the individual terms used to compute the LTME from before and additional error metrics. Specifically, we assess performance by computing the forecast horizon, largest Lyapunov exponent (LY), and correlation dimension (CD) of each reconstructed attractor and compare them with the metrics obtained from the test data. We compute the mean and standard deviation of these metrics for each attractor based on the trajectories obtained from 100 different random initializations of the ERT unit within the NGRC + ERT system.

In Table 1 we list the attractors we consider and the integration time step used to generate a trajectory on the attractor in the first and second

**Table 1**
Performance metrics for the 16 overlapping attractor task. Measurements performed on 100 predicted trajectories (Each using a different random seed for the ERT regressor) and 100 test time series initialized from distinct starting points.

| | | Predictions | | | Test Data | |
|---|---|---|---|---|---|---|
| Attractor | Δ t | Forecast Horizon | Larg. Lyapunov Exp. (LY) | Corr. Dim. (CD) | LY | CD |
| *C.Butterfly_0* | 0.08 | 5.07 ± 0.59 | 0.13 ± 0.01 | 2.19 ± 0.04 | 0.17 ± 0.0 | 2.25 ± 0.01 |
| *Roessler_1* | 0.1 | 4.65 ± 0.45 | 0.08 ± 0.01 | 1.88 ± 0.07 | 0.08 ± 0.0 | 1.85 ± 0.01 |
| *Chua_2* | 0.02 | 2.55 ± 0.37 | 0.3 ± 0.03 | 1.98 ± 0.07 | 0.35 ± 0.01 | 1.93 ± 0.01 |
| *Chen_3* | 0.005 | 2.98 ± 0.74 | 2.02 ± 0.18 | 2.12 ± 0.03 | 1.94 ± 0.1 | 2.12 ± 0.01 |
| *Halvorsen_4* | 0.02 | 1.61 ± 0.07 | 0.69 ± 0.04 | 2.11 ± 0.0 | 0.79 ± 0.02 | 2.11 ± 0.0 |
| *Chua_5* | 0.02 | 2.65 ± 0.43 | 0.3 ± 0.04 | 1.96 ± 0.05 | 0.35 ± 0.01 | 1.93 ± 0.01 |
| *Lorenz_6* | 0.02 | 5.41 ± 1.25 | 0.85 ± 0.03 | 2.06 ± 0.01 | 0.84 ± 0.01 | 2.06 ± 0.01 |
| *Roessler_7* | 0.1 | 4.04 ± 0.38 | 0.07 ± 0.01 | 1.9 ± 0.07 | 0.08 ± 0.0 | 1.85 ± 0.01 |
| *Windmi_8* | 0.08 | 2.41 ± 1.02 | 0.08 ± 0.01 | 1.99 ± 0.04 | 0.07 ± 0.01 | 1.96 ± 0.03 |
| *Rucklidge_9* | 0.08 | 3.28 ± 0.61 | 0.2 ± 0.01 | 1.94 ± 0.05 | 0.21 ± 0.01 | 1.94 ± 0.03 |
| *Chen_10* | 0.005 | 2.61 ± 0.18 | 1.85 ± 0.31 | 2.12 ± 0.09 | 1.94 ± 0.1 | 2.12 ± 0.01 |
| *Windmi_11* | 0.08 | 2.43 ± 0.77 | 0.07 ± 0.01 | 1.95 ± 0.05 | 0.07 ± 0.01 | 1.96 ± 0.03 |
| *C.Butterfly_12* | 0.08 | 4.86 ± 0.14 | 0.13 ± 0.01 | 2.21 ± 0.04 | 0.17 ± 0.0 | 2.25 ± 0.01 |
| *Halvorsen_13* | 0.02 | 1.63 ± 0.07 | 0.69 ± 0.05 | 2.11 ± 0.0 | 0.79 ± 0.02 | 2.11 ± 0.0 |
| *Rucklidge_14* | 0.08 | 3.25 ± 0.62 | 0.19 ± 0.01 | 1.93 ± 0.05 | 0.21 ± 0.01 | 1.94 ± 0.03 |
| *Lorenz_15* | 0.02 | 4.75 ± 1.05 | 0.8 ± 0.03 | 2.06 ± 0.01 | 0.84 ± 0.01 | 2.06 ± 0.01 |

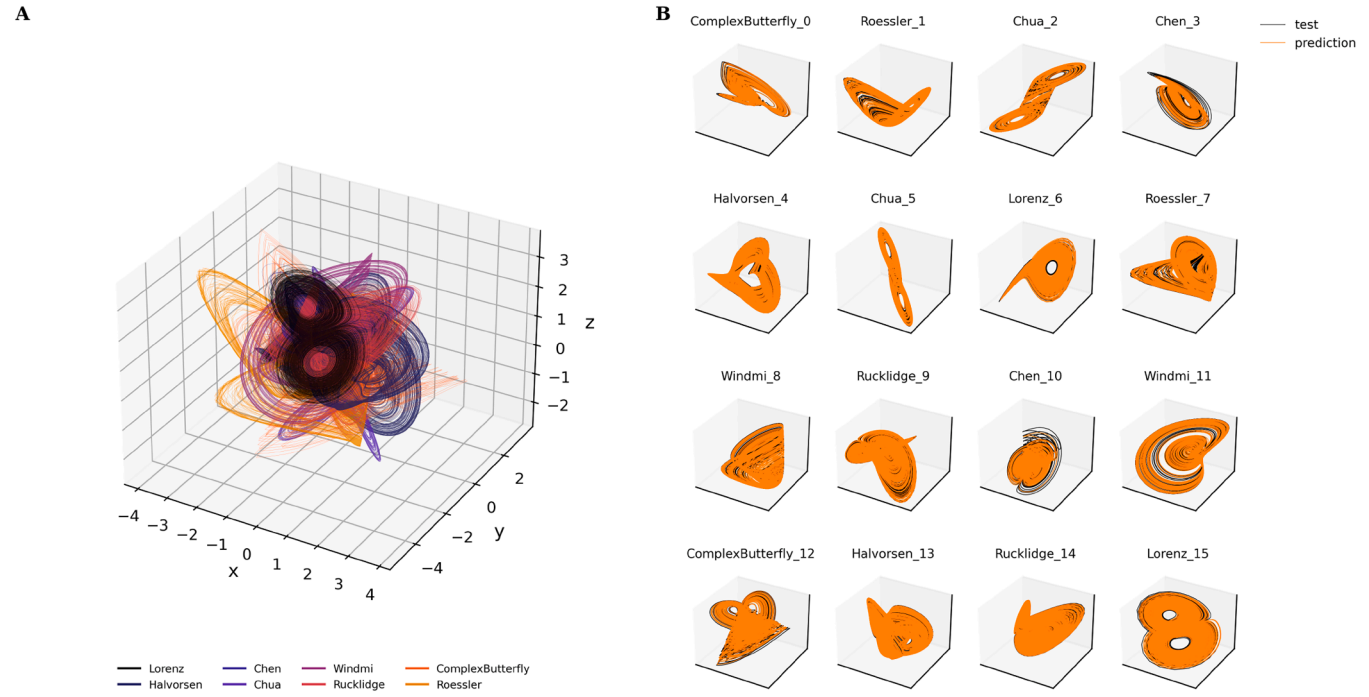


**Fig. 2.** Multi-attractor reconstruction. **A** Experimental setup defined by 16 overlapping attractors in state space, each normalized to zero mean, unit standard deviation and random rotation. **B** NGRC + ERT predictions with exponential filtering applied.

columns, the mean forecast horizon, LY, and CD of the reconstructed attractor, together with their standard deviations, in the third, fourth, and fifth column and compare these results to the corresponding LY and CD of the original attractors in the sixth and seventh columns. NGRC + ERT with exponential filtering achieves a mean forecast horizon of 3.39 Lyapunov times, a LY mean absolute error of 0.041, and a CD mean absolute error of 0.019 averaged across all attractor reconstructions, demonstrating strong short-term prediction accuracy and successful long-term reconstruction of the chosen attractors.

We find that by choosing the scaling parameter $\gamma$ to be small, predictions become susceptible to failure modes such as switching dynamics or recalling attractors from different $\theta$. Fig. 3**B** shows examples of this switching behavior for a purely content-addressable approach (implemented by setting $\gamma = 0$); see panels titled 'Chua_2', 'Chen_3', 'Chua_5', 'Lorenz_6', 'Windmi_11', and 'Lorenz_15. These switching dynamics resemble an attractor consisting of two quasi-attracting regions that correspond to two attractors that may have previously coexisted for a different choice of hyperparameters. We note that similar phenomena has been found when training traditional RCs to achieve multifunctionality; see Flynn and Amann (2024) and Kabayama et al. (2025) for further details.

To obtain further insight into these switching dynamics for specific $\gamma$, we analyze which output leaves of the ERT are traversed during prediction of a given attractor. Specifically, we define an overlap measure as the ratio of leaves active during prediction to those associated with each unique identifier $\theta$ during training. An overlap smaller than 1 indicates that parts of the predicted trajectory follow different dynamics to those used during training, which we classify as switching dynamics. We quantify this behavior for scaling parameters $\gamma$ ranging from 0 to 25. In Fig. 3**A**, we plot the corresponding overlap measure for each

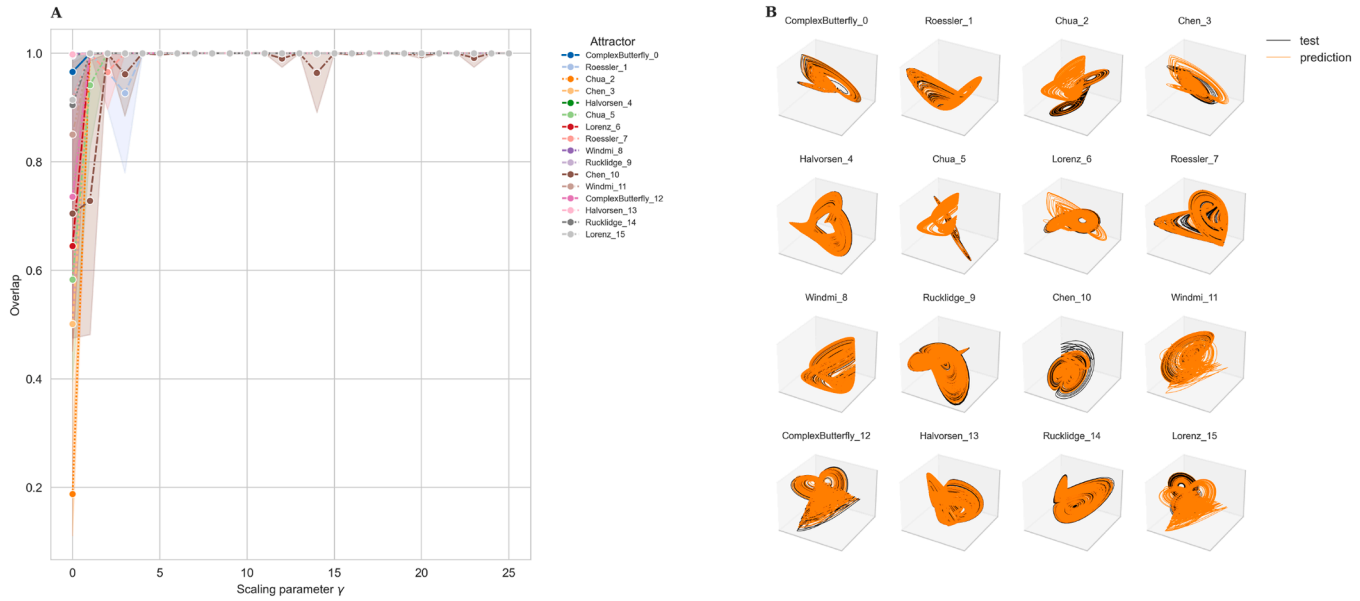


**Fig. 3.** Effect of scaling parameter $\gamma$ on attractor reconstruction. **A** Overlap metric as a function of $\gamma$ for all 16 attractors in the multi-attractor reconstruction. **B** Predicted trajectories for 16 attractors with $\gamma = 0$ (content-addressable approach), showing spontaneous switching between trained attractors in panels titled 'Chua_2', 'Chen_3', 'Chua_5', 'Lorenz_6', 'Windmi_11',and 'Lorenz_15'.

attractor across different scaling parameters, taking into account the inherent randomness of ERT by performing the analysis over 10 random initializations of the ensemble. The results show that larger scaling parameters reduce the frequency of switching behavior, with some large scaling parameters completely suppressing the switching behavior across all reconstructed attractors. Large scaling parameters, and hence larger separation of dynamics in feature space, provide a mechanism to stabilize predictions by controlling the susceptibility of the ERT unit to switching between leaves assigned to different attractors during prediction. Supplementary Note S11 shows how increasing $\gamma$ separates the attractors in feature space, suppressing the switching behavior and enabling stable long-term memory recall. Although we do not claim that this method achieves the highest precision forecasts in systems trained to perform multi-attractor reconstruction, our results demonstrate that this index-based approach enables the NGRC + ERT system to successfully reconstruct the dynamics of multiple attractors without extensive hyperparameter optimization. We extend this to a 49-attractor task in Supplementary Note S7, where the attractors are arranged on a $7 \times 7$ grid. There we show that memory itinerancy (Kong et al., 2024) is achievable by changing the unique identifier during prediction to recall target attractors.

### 2.3. Failure modes from defects in short-term memory processing

In this subsection we conduct a more detailed analysis on the retained features from the two-shot learning, i.e., features that remain after filtering. More specifically, we examine how the retained features are distributed across the time-delay features of the x, y, and z components of the output after applying (i) the exponential filtering scheme and (ii) the uniform filtering scheme. We consider six sub-tasks involving the reconstruction of 4, 9, 16, 25, 36, and 49 overlapping attractors, respectively. The hyperparameter configuration is fixed across all sub-tasks to $k = 100$, $s = 1$, $\gamma = 25$ and $T = 20{,}000$ for each attractor. In Fig. 4 we show the optimized feature configurations for each sub-task, by decomposing the retained features into the constituent z (top), y (middle), and x (bottom) components. We refer to this decomposed representation as the optimized short-term memory processing being performed by the NGRC + ERT system. Additionally, we display for each retained feature the obtained feature importance in green, where a darker shade corresponds to a greater importance. We find that the feature importance measure tends to assign the largest importance to features closest to the current time for both schemes. For the exponential filtering scheme, we find that retained features tend to be similar across each sub-task and do not involve time-delay terms greater than 15, i.e., no features beyond $z(t-15)$ are used. In contrast, for the uniform filtering scheme, we find a more diverse range of retained features across each sub-task with much larger time-delay terms used, where even $z(t-51)$ is deemed to be of importance.

From here we analyze how defects in short-term memory processing influence long-term memory recall. By defects we mean that features are manually placed at incorrect positions within the retained feature vector the ERT was trained with to perform long-term memory recall. We conduct the analysis by keeping the trained ERT unit fixed, while varying the feature configuration. This experimental paradigm models how variations in short-term memory processing can induce untrained behavior in learning systems, with implications for hardware-level applications of our approach. We investigate the 4-attractor sub-task with the exponential filtering scheme applied (Fig. 5). Starting from the retained feature configuration (Fig. 5**A1**), we simulate processing delays of $n$ time steps for features associated with $z$, e.g., feature $f_j$ processes $z(t-n)$ instead of $z(t)$. We find that by introducing this defect, the system starts recalling deformations of the attractors it was trained on, exhibiting confabulations in the form of 'generated attractors' (O'Hagan et al., 2025). Fig. 5 (middle and bottom row) shows the predicted dynamics with processing delays of $n = 3$ and $n = 17$. Comparing some of the changes from the top row, in Fig. 5 (**C2**) we see the inner loop of the Windmi_1 attractor vanishes and the period of the limit cycle reduces. In contrast, in Fig. 5(**C3**) the Windmi_1 attractor displays chaotic behavior. Key changes to the other attractors also occur for different $n$. For the Windmi_1 attractor we systematically analyze the dynamics for increasing processing delays up to $n = 40$ and visualize in Fig. 6 the attractor recall by plotting the changes in the local maxima versus processing delay $n$ in the $z$ component, see Figures S21– S22 in the Supplementary Materials for corresponding state space diagrams. Interestingly, near $n = 3,\ 20$, and 30, we observe a major change in the generated dynamics; specifically, a sudden contraction/expansion in the size of the attractor, which is indicative of an interior crisis taking place. We find that these defects can result in various generated attractors, including

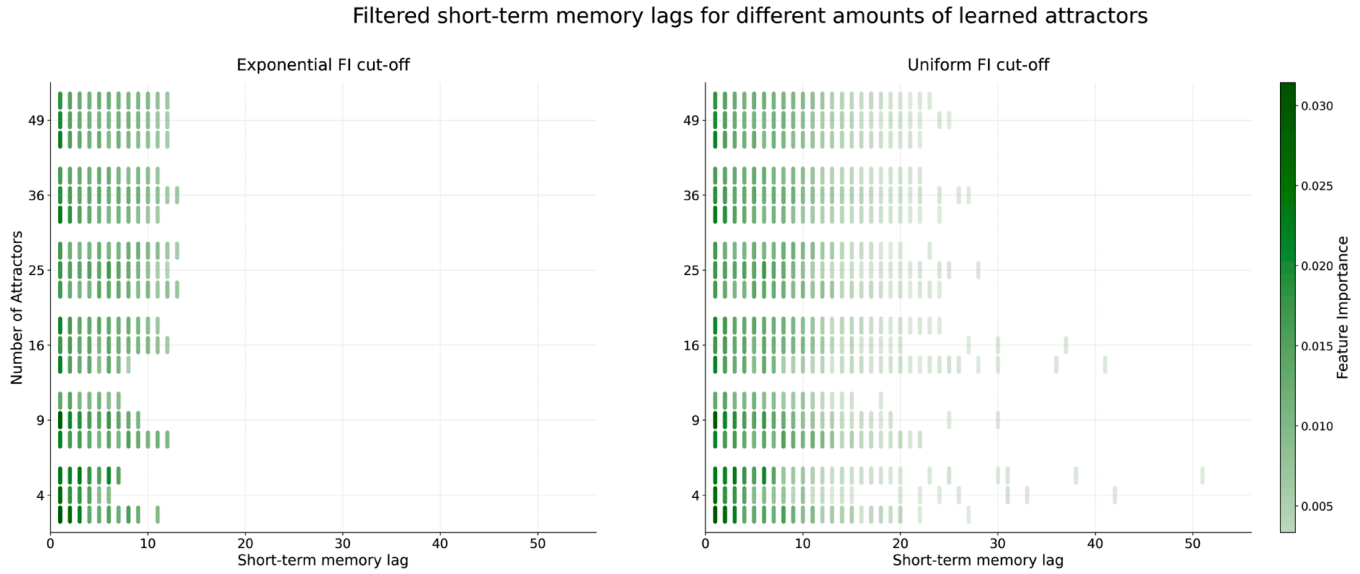


**Fig. 4.** Filtered short-term memory representation for different attractor tasks. Left panels show filtered short-term memory representation using exponential feature importance filtering for varying numbers of learned attractors, displayed for groups consisting of z (top), y (middle), and x (bottom) components where each group corresponds to the different number of attractors. Right panels show corresponding results using uniform filtering. Each lag represents one time step in the past.

periodic and chaotic attractors, that the system was never trained on. Further, interior crises take place at various locations in these in Fig. 6, thus highlighting the significant role played by unstable equilibria in state space whose presence we observe indirectly. In Supplementary Figures S19-S20, we observe similar behavior when processing delays are applied to two retained features across all x, y, and z components.

## 3. Discussion

In this paper we extend a two-shot learning approach introduced by Giammarese et al. (2025) to train a single machine learning system to store and recall multiple strange attractors. In short, we apply this approach to a system consisting of a short-term and long-term memory unit, specifically a next generation reservoir computer (NGRC) and an extremely randomized tree (ERT) ensemble, and take inspiration from multifunctional and parameter-aware reservoir computing approaches to multi-attractor learning (Flynn, 2023; Flynn & Amann, 2024; Flynn et al., 2022, 2021b, 2023; Kim et al., 2021; Köglmayr et al., 2026; Köglmayr & Räth, 2024; Kong et al., 2021, 2023; Morra et al., 2023) to train the combined NGRC + ERT system to reconstruct multiple strange attractors. The two-shot learning approach significantly improves performance without the need for extensive hyperparameter optimization by (i) identifying important features of the NGRC + ERT system and (ii) training a new NGRC + ERT system based on these features. This is achieved by the exponential filtering method we introduce. The separation between short-term memory processing (NGRC) and long-term memory recall (ERT) also enables greater interpretation of the two-shot learning process. We show how the information obtained from the first-shot informs the second, specifically, after an initial optimization we find subsequent training focuses exclusively on short-term memory features that prove important for the given task, thereby improving performance without further hyperparameter tuning. In short, we show the NGRC + ERT achieves excellent performance and does so for a wide range of NGRC parameter settings when the two-shot learning method with exponential filtering is applied. This result motivates a broader investigation of feature selection methods for reservoir computing more generally, where feature filtering has so far played a limited role.

Moreover, our results shed light on the relationship between short- and long-term memory and how they work together to store and recall memories, in our case several different strange attractors. More specifically, we study the factors that disrupt this relationship by highlighting that stable storage and recall of multiple attractors requires clear separation in feature space. When the separation is insufficient, the NGRC + ERT system exhibits a tendency to recall or 'switch between' different trained attractors. We show that this behavior can be prevented through adequate feature separation. Building on this, we investigate recall properties when defects are introduced during short-term memory processing. We find that the system exhibits momentary confabulations in the form of generated attractors, attractors that partially resemble the original attractor. This is a qualitatively different outcome than recalling the wrong trained attractor, rather, the NGRC + ERT confabulates in the sense of O'Hagan et al. (2025). Additionally, we show that memory itinerancy (Kong et al., 2024) is achievable by changing the identifier during recall.

Our findings may inspire future studies on the relationship between short- and long-term memory in other learning systems, like classic reservoir computers, where short-term memory processing is facilitated through a network structure and influenced by node saturation through the activation function. Our approach performs long-term memory recall through trees, whereas classical reservoir computing compresses this task into a matrix typically obtained from ridge regression. These represent qualitatively different learning methodologies as trees can grow dynamically, while matrices are finite-dimensional. However, it is important to note that despite their structural differences, these methodologies exhibit similar failure modes, like the emergence of generated attractors and switching dynamics. This may point towards the existence of a more universal principle governing failure modes in learning systems.

## 4. Methods

### 4.1. Extremely randomized trees

This subsection introduces the key principles of the Extremely Randomized Trees (ERT) algorithm (Geurts et al., 2006), a machine learning-based approach for solving classification and regression tasks.

#### 4.1.1. Overview of decision trees

A **decision tree** is a non-parametric model made up of hierarchically structured *nodes* connected by *branches* that represent if-then-else

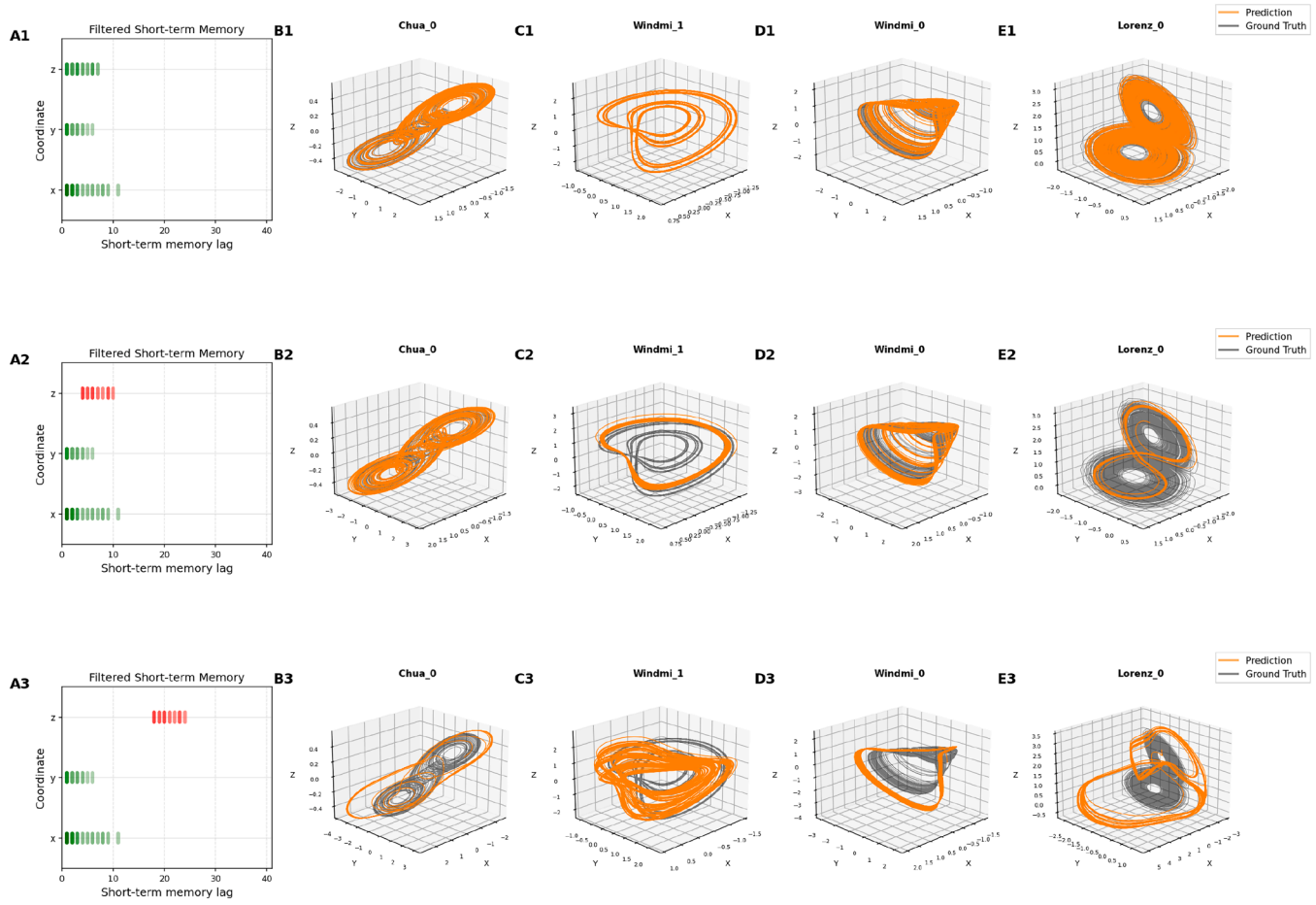


**Fig. 5.** Two-shot learning of four attractor system and processing delay effects. **Top:** Optimized short-term memory representation obtained from two-shot learning using exponential filtering. Panels show the four attractors: chaotic Chua, periodic Windmi, chaotic Windmi, and chaotic Lorenz. **Middle:** Lag 3 - Processing defects in the z coordinate induce transitions to generated attractors during long-term memory recall. **Bottom:** Lag 17 - Increased processing defects generate diverse generated attractor behaviors during recall.

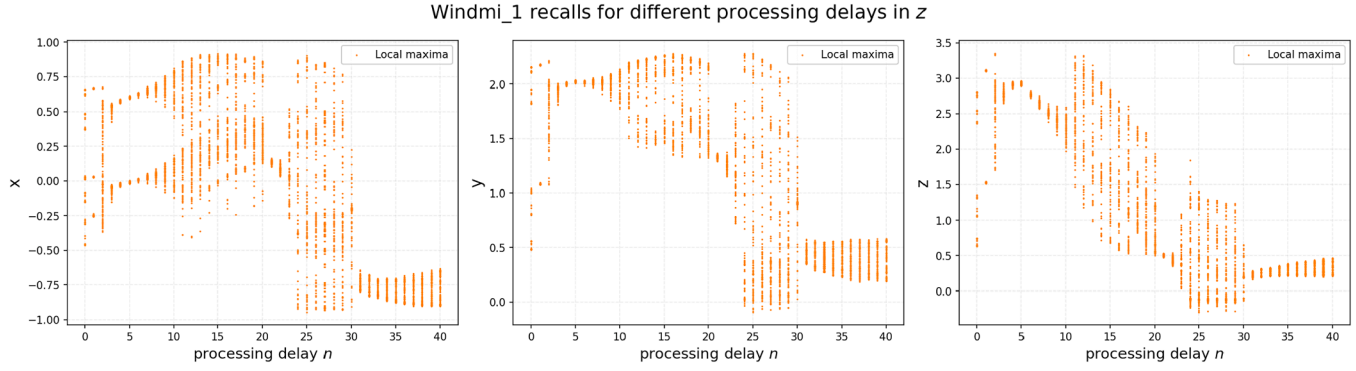


**Fig. 6.** Attractor recalls of Windmi_1 for processing delays. Changes in local maxima in NGRC + ERT system's output of x (a), y (b), and z (c) components for changes in processing delays in $z$ reveals rich generated dynamical behaviors including periodic and chaotic regimes.

rules. We introduce the following notation to describe a decision tree in greater detail. Let $D = \{(\mathbf{r}_i, \mathbf{y}_i)\}_{i=1}^{T}$ be the training data set, with $T$ elements. Elements of this set, known as training samples, consist of the following pair, a feature vector, $\mathbf{r}_i \in \mathbb{R}^Z$, where $Z \in \mathbb{N}$ is the number of features, and a target vector $\mathbf{y}_i \in \mathbb{R}^d$, where $d \in \mathbb{N}$ is the number of components of the vector. A feature is a quantity derived from elements of the training data set and used as input to, in this instance, the decision tree. To simplify the notation below, we define a set of indices $\mathcal{I} = \{1, \ldots, T\}$ to reference the training samples, where $i \in \mathcal{I}$ references the training sample $(\mathbf{r}_i, \mathbf{y}_i)$. A tree starts with a single node, called the *root node*, which represents the entire set of indices $\mathcal{I}$, hence the entire training data set.

**Training the tree:** Starting from the root node, the tree 'grows' by recursively partitioning the training data set into two smaller subsets, forming the left and right child nodes. Each child node is split again, and the process continues until a stopping criterion is reached, such as a maximum depth of the tree, defined as the number of instances that new branches are created, or a minimum number of training samples

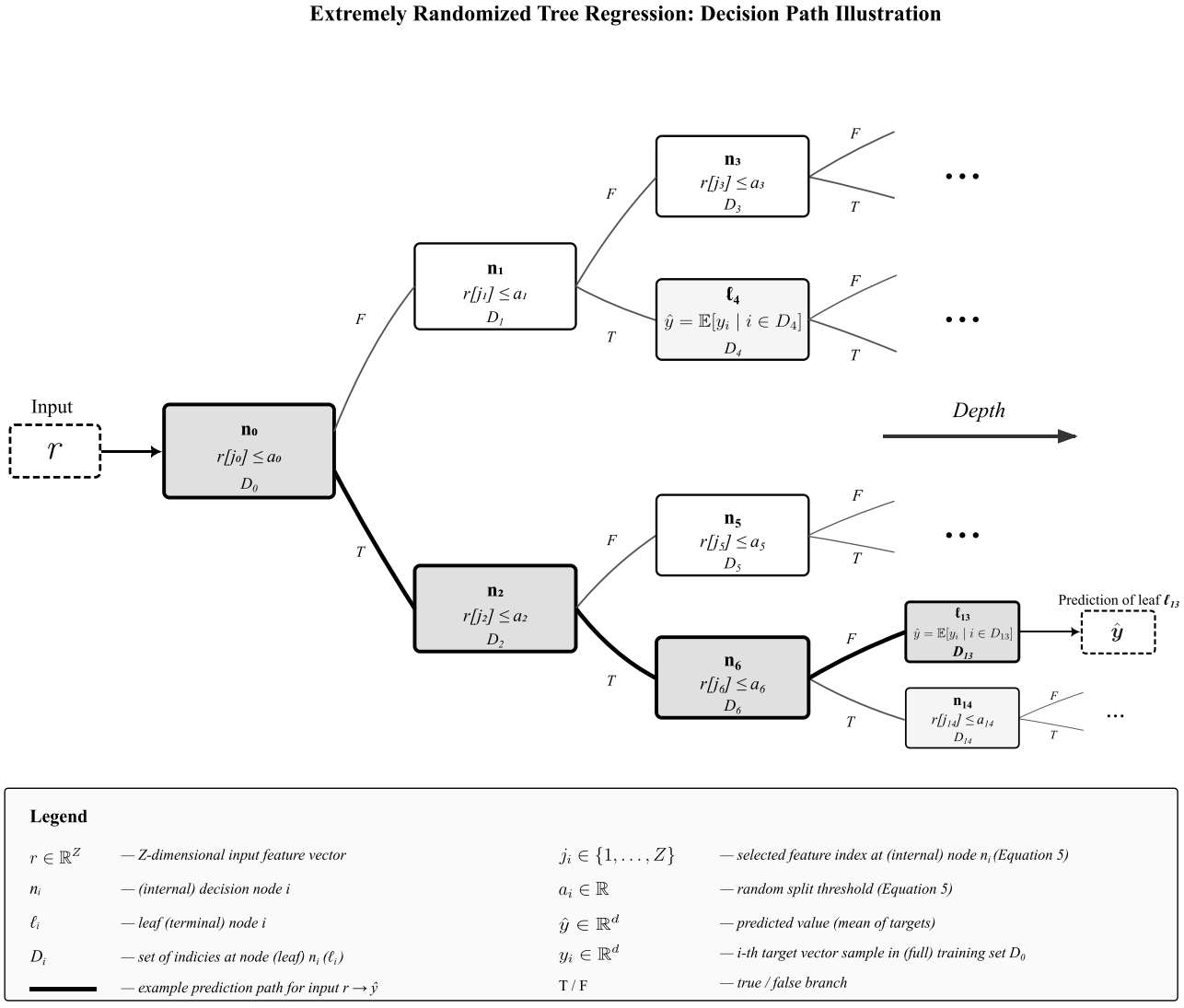


**Fig. 7.** Structure of an extremely randomized tree. The tree receives an input $\mathbf{r}$, which traverses the branches by comparing input values with feature thresholds at each node until reaching a leaf, where it outputs a prediction $\hat{\mathbf{y}}$.

per node. A node that is not split further is called a leaf node, or simply a leaf. Each leaf stores a subset of training samples, identified by their indices. Once the stopping criterion is reached, the tree-building process is complete, i.e., the tree is trained. The splitting rule at each node depends on the specific tree method, we introduce the splitting rule for ERT later.

**Making predictions with the tree:** Given a new input $\mathbf{r} \in \mathbb{R}^Z$, the tree makes a prediction by 'routing' $\mathbf{r}$ from the root node through the tree's branches to the respective leaf. More specifically, at each node, the splitting condition learned during the tree building process determines whether $\mathbf{r}$ is forwarded to the left or right child node. This process continues until a leaf node is reached. In a regression setting, the prediction of a tree for an input $\mathbf{r}$ is the arithmetic mean of the target vectors of the training samples that reached the same leaf. If $\mathbf{r}$ reaches leaf $l$ the prediction is given by,

$$tree(\mathbf{r}) = \frac{1}{|\mathcal{I}_l|} \sum_{i \in \mathcal{I}_l} \mathbf{y}_i = \mathbb{E}[\mathbf{y_i} \mid i \in \mathcal{I}_l] = \hat{\mathbf{y}}, \tag{1}$$

where $\mathcal{I}_l$ denotes the set of indices of training samples assigned to leaf $l$. $\mathbb{E}$ denotes the arithmetic mean of the target vector samples in $\mathcal{I}_l$. Specifically, the sum is element-wise for target vectors with $d$ components.

#### 4.1.2. Overview of extremely randomized trees

**Extremely randomized trees (ERT):** Many tree-based machine learning methods suffer from high prediction variance (Geurts et al., 2006), a consequence of their hierarchical structure. As the splitting process depends on the training data, small perturbations in the training samples can produce different splits, and these differences propagate through the tree, making the learning process sensitive to small changes in the training data. The ERT algorithm trains an ensemble of independent trees, averaging their predictions after training to address this issue.

**Training extremely randomized trees:** The ERT algorithm builds $M$ trees independently. Each tree follows the same construction rules but introduces randomness at each split, resulting in $M$ distinct trees. Let $D = \{(\mathbf{r}_i, \mathbf{y}_i)\}_{i=1}^T$ denote the training data set, where $\mathbf{r}_i \in \mathbb{R}^Z$ is the feature vector and $\mathbf{y}_i \in \mathbb{R}^d$ is the target vector for the $i$th sample. Equivalently, we define the feature matrix $\mathbf{R} \in \mathbb{R}^{Z \times T}$ and the target matrix $\mathbf{Y} \in \mathbb{R}^{d \times T}$, where $Z$ is the number of features, $T$ is the number of samples, and $d$ is the dimension of the 'target vector space', with $Z, T, d \in \mathbb{N}$. In our case, the feature vectors are constructed using NGRC. During tree construction, the training data set is partitioned into subsets, each tracked by a corresponding index set. Let $\mathcal{I} \subseteq \{1, \ldots, T\}$ denote such an index set.

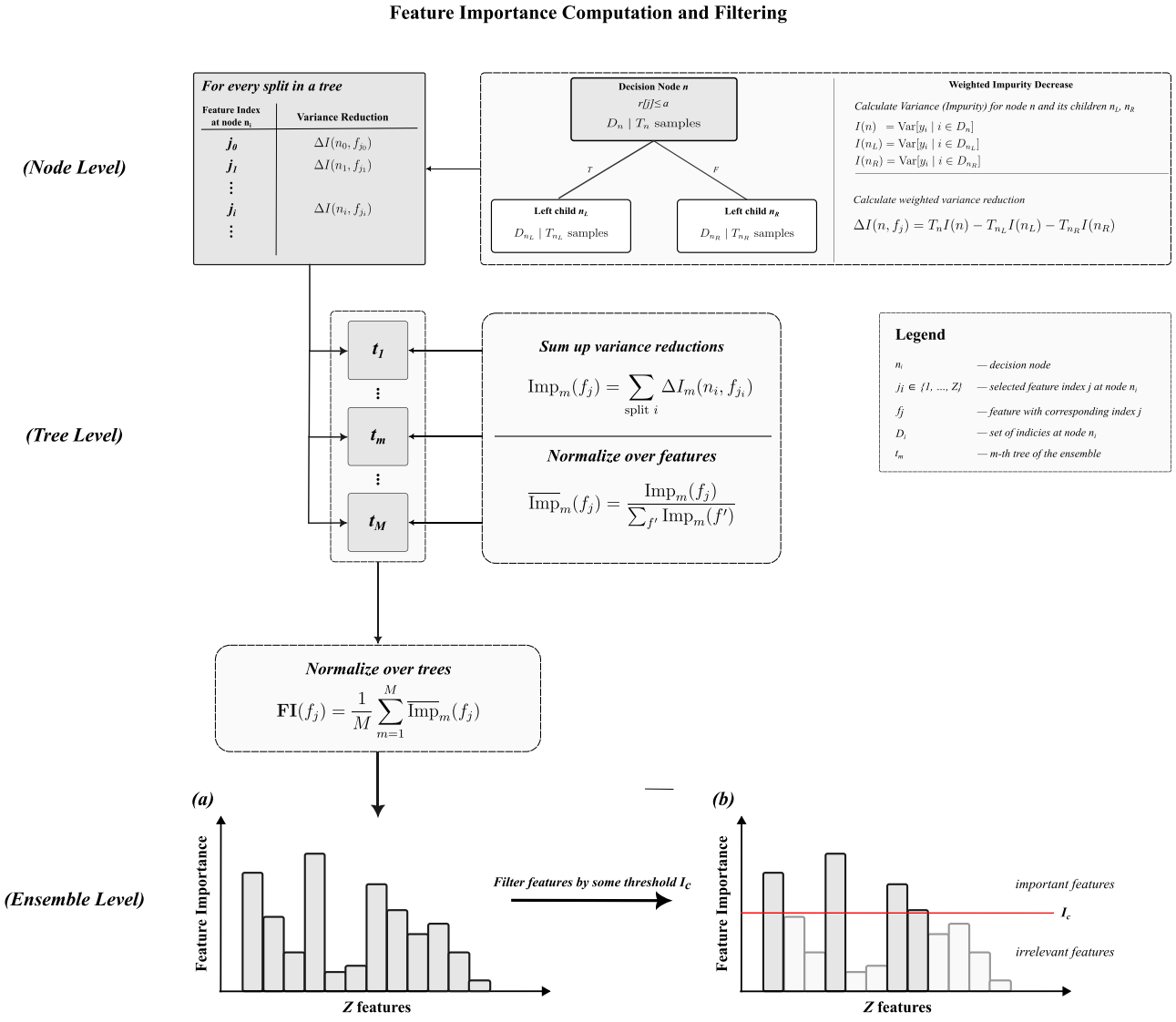


**Fig. 8.** Feature importance computation and filtering in extremely randomized trees. At each node, the weighted variance reduction is calculated for every split. These values are summed across the tree and normalized so that the total importance equals one. At the ensemble level, importance measures are averaged across all trees, yieldinga final feature importance value for each feature. Features are then classified as important or irrelevant by comparison with a threshold $I_c$.

At each node $n$, let $T' = |\mathcal{I}|$ denote the number of training samples that reach this node, and let $\mathbf{R}' \in \mathbb{R}^{Z \times T'}$ denote the submatrix of $\mathbf{R}$ restricted to these samples. The splitting rule at each node is defined as follows. From the $Z$ features, $K \leq Z$ features are randomly selected, indexed by $\{f_1, \ldots, f_K\}$. For each selected feature $f_i$, let $\mathbf{R}'_{f_i} \in \mathbb{R}^{1 \times T'}$ denote the corresponding feature row of $\mathbf{R}'$. A random threshold $a_i$ is then drawn uniformly from the interval $[\min(\mathbf{R}'_{f_i}), \max(\mathbf{R}'_{f_i})]$, yielding $K$ candidate splits $\{(f_1, a_1), \ldots, (f_K, a_K)\}$. Each candidate split partitions the training data at node $n$ into two subsets:

$$D_i^{(L)} = \{(\mathbf{r}_j, \mathbf{y}_j) \mid j \in \mathcal{I},\ \mathbf{R}_{f_i,j} \leq a_i\}, \quad D_i^{(R)} = \{(\mathbf{r}_j, \mathbf{y}_j) \mid j \in \mathcal{I},\ \mathbf{R}_{f_i,j} > a_i\}, \tag{2}$$

corresponding to the left (L) and right (R) candidate child nodes. Let $T_i^{(L)} = |D_i^{(L)}|$ and $T_i^{(R)} = |D_i^{(R)}|$ denote the number of samples in each candidate subset, with $T_i^{(L)} + T_i^{(R)} = T'$. Each candidate split is evaluated using the squared error loss,

$$L(f_i, a_i) = \sum_{j \in \mathcal{I}^{(L)}} (\mathbf{y}_j - \hat{\mathbf{y}}_L)^2 + \sum_{j \in \mathcal{I}^{(R)}} (\mathbf{y}_j - \hat{\mathbf{y}}_R)^2, \tag{3}$$

where $\hat{\mathbf{y}}_L, \hat{\mathbf{y}}_R \in \mathbb{R}^d$ are the mean target vectors in each subset defined by

$$\hat{\mathbf{y}}_p = \frac{1}{T_p} \sum_{j \in \mathcal{I}^{(p)}} \mathbf{y}_j\,, \quad p = L,\ R. \tag{4}$$

Minimizing this loss is equivalent to maximizing the variance reduction. The best splitting pair and with that the learned splitting rule at node $n$ is

$$(f^*, a^*) = \arg \min_{i \in \{1, \ldots, K\}} L(f_i, a_i). \tag{5}$$

The partitioning is recursively repeated at each new node until the tree's stopping criterion is reached.

**Making predictions with the extremely randomized tree:** Once all $M$ trees are constructed, they can be used for prediction. For an input $\mathbf{r} \in \mathbb{R}^Z$ passing through node $n$, let $(f_{j_n}, a_n)$ denote the splitting pair selected during training, where $j_n \in \{1, \ldots, Z\}$ is the index of the selected feature. If $\mathbf{r}[j_n] \leq a_n$, the input is routed to the left child, otherwise, it is routed to the right child. This process continues recursively until the

input reaches a leaf node, where the prediction is given by Eq. (1). Each of the $M$ trees independently process the input $\mathbf{r}$ and produces a prediction. The final ERT prediction is the average across all trees:

$$\mathrm{ERT}(\mathbf{r}) = \frac{1}{M}\sum_{m=1}^{M} tree_m(\mathbf{r}) \tag{6}$$

A schematic of the above process for one extremely randomized tree is provided in Fig. 7.

**Implementation:** We implement the ERT algorithm using the `scikit-learn` python library (Pedregosa et al., 2011). We use the default tree hyperparameters, except for setting the number of trees to $M = 100$ for two-attractor tasks and $M = 50$ for multiple-attractor tasks, and $K = Z$. The tree depth is not fixed, which leads to fully grown trees within the ensemble.

### 4.2. Feature importance

The *feature importance* (Altmann et al., 2010; Breiman, 2001; Geurts et al., 2006; Pedregosa et al., 2011) quantifies how much each feature contributes to reducing the loss function in Eq. (3) across all splits in the ensemble. We use this measure to optimize the feature configuration of the NGRC framework, eliminating the need for extensive hyperparameter tuning typically required to obtain an optimized feature configuration. To describe how feature importance is calculated, we first recall that at each node $n$, the selected split $(f_j, a_j)$ partitions the samples into child nodes $n_L$ and $n_R$. Let $T_n$, $T_{n_L}$ and $T_{n_R}$ denote the number of samples in these nodes and let $I(\cdot)$ denote the *impurity at a given node*, which is the variance of the target values within these nodes. The contribution of the split to the feature importance calculation of $f_j$ is the weighted variance reduction,

$$\Delta I_m(n, f_j) = T_n I(n) - T_{n_L} I(n_L) - T_{n_R} I(n_R). \tag{7}$$

This quantity is computed for every split in the $m$th tree and normalized across all features, such that

$$\mathrm{Imp}_m(f_j) = \sum_{\text{split } i} \Delta I_m(n_i, f_{j_i}), \tag{8}$$

$$\overline{\mathrm{Imp}}_m(f_j) = \frac{\mathrm{Imp}_m(f_j)}{\sum_{f'} \mathrm{Imp}_m(f')} \tag{9}$$

where the sum in Eq. (8) is computed over all nodes split by $f_j$ and Eq. (9) specifies the normalization across all features. To obtain the feature importance of $f_j$, we take the average of the normalized importances computed in Eq. (9) across the ensemble, yielding the feature importance (FI)

$$\mathbf{FI}(f_j) = \frac{1}{M}\sum_{m=1}^{M} \overline{\mathrm{Imp}}_m(f_j). \tag{10}$$

A schematic of the above process is provided in Fig. 8.

### 4.3. Next-generation reservoir computing

Next-generation reservoir computing is a machine learning approach introduced by Gauthier et al. (2021) with applications in, for instance, time series prediction, attractor reconstruction, and control (Barbosa & Gauthier, 2022; Gauthier et al., 2021; Haluszczynski et al., 2023). Unlike traditional reservoir computers (RCs) that rely on random networks (Jaeger & Haas, 2004; Maass et al., 2002; Verstraeten et al., 2007), feature vectors for next-generation reservoir computers (NGRCs) are constructed in a deterministic manner by computing unique monomials of time-shifted input variables.

While RCs and NGRCs are typically trained via ridge regression, Giammarese et al. (2025) recently introduced a new technique for training NGRCs which improves their attractor reconstruction capabilities by incorporating an 'Extremely Randomized Tree' (ERT) unit as a trainable output layer using their TreeDOX framework (**Tree**-based **D**elay **O**verembedded e**X**plicit memory learning of chaos). In this paper, we extend the results of Giammarese et al. (2025) by introducing a new technique for training NGRCs to reconstruct multiple attractors from different dynamical systems, thereby extending the TreeDOX framework to multifunctional (Flynn & Amann, 2024; Flynn et al., 2022, 2021b) and parameter-aware (or index-based) learning (Kim et al., 2021; Köglmayr et al., 2026; Köglmayr & Räth, 2024; Kong et al., 2021, 2023).

**Training an NGRC system to perform attractor reconstruction:**

The aim in attractor reconstruction tasks is to use a time series describing a trajectory on a given attractor to train some system to reconstruct the future 'climate' of the attractor, i.e., mimic the long-term dynamics, while at the same time provide reasonable short-term predictions on the trajectory's future.

When using an NGRC for attractor reconstruction tasks, the first step is to construct features using the procedure outlined below.

For a given data point in the 'input time series', $\mathbf{X} \in \mathbb{R}^{d\times T}$, containing $T > 0$ data points each consisting of $d > 0$ components, which corresponds to discrete-time samples of a trajectory on a given attractor, we first construct an initial 'linear time-delay feature vector' using $k \geq 1$ past data points each separated by $s \geq 1$ time steps. Therefore, there is a fixed 'warm-up phase' of $\delta t = (k-1)\cdot s + 1$ time steps needed to create the initial feature vector, reducing the effective training data set to $T - \delta t$ samples. The 'final feature vector' $\mathbf{r}_i$ at time $t_i$ is obtained by computing unique monomials up to order $O$ from the components of this linear time-delay vector, introducing nonlinearity into the feature construction process. Here, unique means that each monomial term appears exactly once. Thus, when given $\mathbf{X}$, we define the 'feature matrix' as,

$$\mathbf{R}(\mathbf{X}, k, s, O) = [\mathbf{r}_{\delta t}, \mathbf{r}_{\delta t+1}, \dots, \mathbf{r}_{T-1}] \in \mathbb{R}^{Z\times(T-1-\delta t)}, \tag{11}$$

where $Z$ denotes the number of features as before. Note, there are $T - 1 - \delta t$ columns in this matrix, we explain why below. The type of features used depend on the choice of $O$ and $k$. In this paper we construct features in three distinct ways and refer to each of the resulting sets of features as a 'feature model type' to further stress that the features are a 'model', i.e., mapping, of the input data; see Figure S1 in the Supplementary Materials for an illustration of each feature model type. In short, the feature model types are created by (i) setting $O = 1$ and $k > 1$, which reduces the NGRC to a vector autoregression (VAR) model containing only linear time-delayed features, (ii) setting $O > 1$ and $k > 1$, thereby including higher-order monomials which yields a nonlinear vector autoregression (NVAR) model, and (iii) setting $O > 1$ and $k = 1$, which produces a nonlinear memoryless model (NML) that discards temporal information. Given the feature matrix $\mathbf{R}$, the training target matrix $\mathbf{Y}$ specifies what the NGRC is trained to predict the future of. Specifically, we employ the NGRC as a 'one-step-ahead integrator' like in Gauthier et al. (2021), training it to predict the difference between future consecutive states. Accounting for the warm-up phase, the 'target matrix' is hence defined as

$$\mathbf{Y} = [\Delta\mathbf{x}_{\delta t}, \Delta\mathbf{x}_{\delta t+1}, \dots, \Delta\mathbf{x}_{T-1}] \in \mathbb{R}^{d\times(T-1-\delta t)}, \tag{12}$$

where we define $\Delta\mathbf{x}_i = \mathbf{x}_{i+1} - \mathbf{x}_i$ which, by construction, enables the NGRC to act as a one-step-ahead integrator and, as a consequence, only $T - 1 - \delta t$ data points of $\mathbf{X}$ are used to train the NGRC. In other words, we train the NGRC to 'learn' the mapping $\mathbf{r}_i \mapsto \Delta\mathbf{x}_i$. After training, we obtain a predicted trajectory on the attractor by iterating the following 'NGRC system' forward through time,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + G(\mathbf{r}_i), \tag{13}$$

where $G$ describes the trained output layer and maps feature vectors to the difference between successive predicted points on the trajectory. To be more explicit, we use the known value of $\mathbf{x}_i$ at time $i = T$ to obtain a prediction of $\mathbf{x}_i$ at time $T + 1$, and so on.

**Training an NGRC + ERT system to perform attractor reconstruction:** While standard NGRCs compute $G$ via ridge regression, the TreeDOX framework employs Extremely Randomized Trees (ERT) to learn

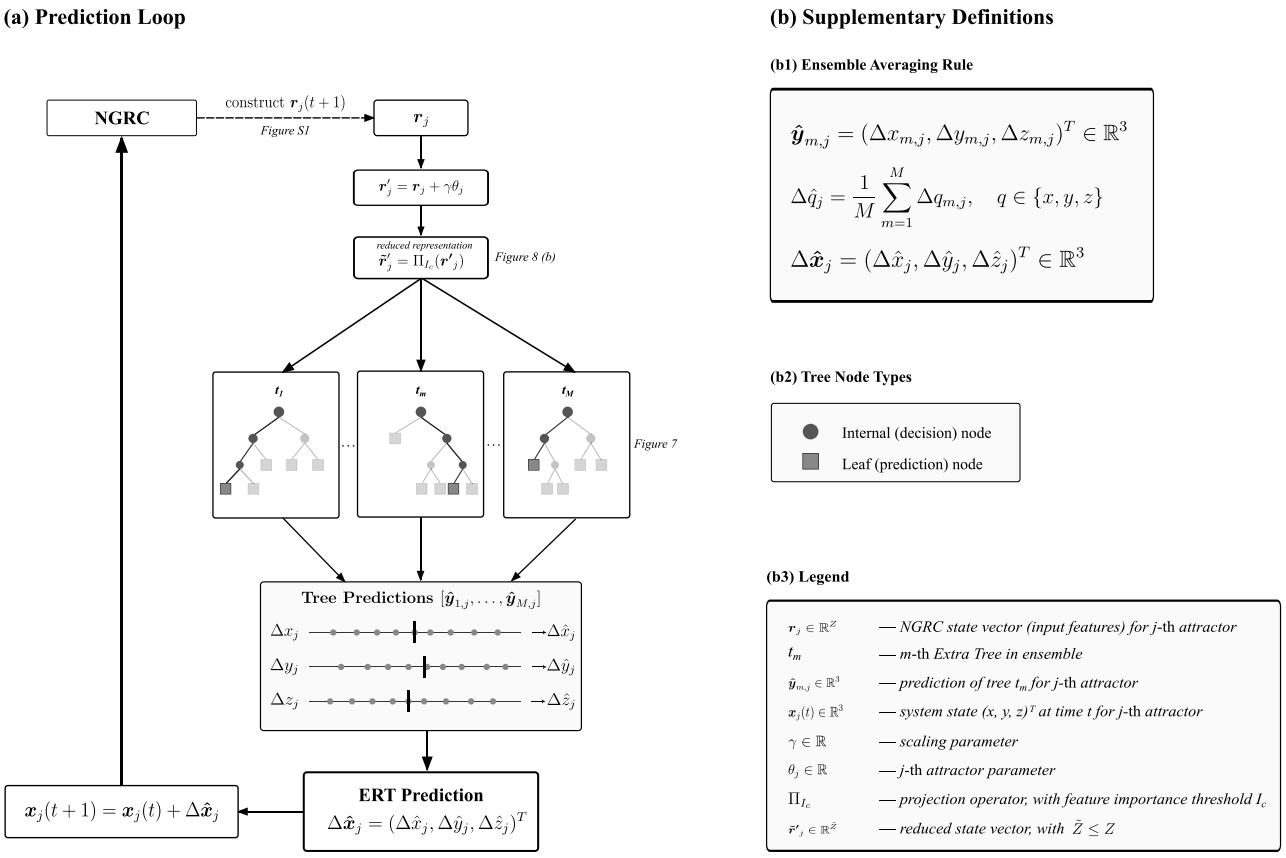


**Fig. 9.** Prediction loop of the NGRC + ERT system for multi-attractor reconstruction after training, the model predicts the next state of the $j$th attractor by constructing a feature vector using the NGRC framework, augmented with the attractor-specific parameter $\theta_j$ scaled by $\gamma$. This vector is then projected onto a reducedfeature space containing only the important features identified during training. Each tree in the ensemble receives the reduced vector and produces a prediction. The final output is obtained by averaging across all trees and adding the result to the current state.

this mapping. We train using the same steps as above and use the ERT algorithm outlined in previous sections to obtain a predicted trajectory on the attractor by iterating the following 'NGRC + ERT system' forward through time,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{ERT}(\mathbf{r}_i). \tag{14}$$

**Training an NGRC + ERT system to perform multi-attractor reconstruction:** We extend the work of Giammarese et al. (2025) to multi-attractor reconstruction by integrating their TreeDOX framework with concepts from multifunctionality (Flynn et al., 2021b) and parameter-aware learning (Kim et al., 2021; Kong et al., 2021). This enables the NGRC + ERT system to reconstruct the dynamics of $N$ distinct attractors with a single trained output layer. Each attractor $j$ is assigned a unique identifier $\theta_j \in \mathbb{N}$, which is multiplied with a scaling parameter $\gamma \in \mathbb{N}_0$ and added to every element of the attractor's corresponding feature vector, yielding the parametrized representation $\mathbf{r}'_{j,i} = \mathbf{r}_i + \gamma\theta_j$. The system operates in a multifunctional sense for $\gamma = 0$ and parameter-aware sense for $\gamma > 0$. For each $\mathbf{X}_j$, the corresponding parametrized feature matrix $\mathbf{R}_j = [\mathbf{r}'_{j,\delta t}, \ldots, \mathbf{r}'_{j,T-1}]$ and target matrix $\mathbf{Y}_j$ are constructed as before. All $\mathbf{R}_j$ and $\mathbf{Y}_j$ matrices are concatenated into larger $\mathbf{R}_N$ and $\mathbf{Y}_N$ matrices for training, enabling the NGRC + ERT to learn the mapping $\mathbf{R}_j \mapsto \mathbf{Y}_j$ for each $j$ simultaneously. For prediction, selecting a specific identifier $\theta_j$ enables the system to recall the corresponding attractor dynamics by iterating the following forward through time

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathrm{ERT}(\mathbf{r}'_{j,i}). \tag{15}$$

**Two-shot learning:** We introduce a 'two-shot' learning approach in our experiments, training the NGRC + ERT system and then updated it to produce a more optimal system. Specifically, we use the feature importance measure **FI** provided by the trained ERT ensemble to filter the feature space for an optimized feature configuration. Only features whose importance exceeds a cut-off threshold $\mathrm{I}_c$ are allowed to remain active in the updated NGRC + ERT system. For a given feature vector $\mathbf{r}' \in \mathbb{R}^Z$, we construct the reduced representation $\tilde{\mathbf{r}}' = \Pi_{\mathrm{I}_c}(\mathbf{r}')$ through a projection operator that retains only features $f$ satisfying $\mathbf{FI}(f) > \mathrm{I}_c$ (Fig. 8(b)), reducing the effective number of features from $Z$ to $\tilde{Z} = |\{f \mid \mathrm{I}_f > \mathrm{I}_c\}|$. We then train a new ERT using the retained feature vectors, yielding a refined ERT that we denote by $\widetilde{\mathrm{ERT}}$. In the Results section, we analyze the performance of this two-shot approach using the following two feature importance thresholds $\mathrm{I}_c$. The first is a 'uniform threshold', similar to Giammarese et al. (2025), defined as $\mathrm{I}_c = 1/Z$, where $Z$ is the total number of features. The second is an exponential threshold defined as $\mathrm{I}_c = \max(\mathbf{FI}(f))/e$, where $e$ is Euler's number. The reconstruction of attractor $j$ is hence performed via,

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \widetilde{\mathrm{ERT}}(\tilde{\mathbf{r}}'_{j,i}), \tag{16}$$

and the same procedure is used to reconstruct the dynamics of each of the $N$ attractors (Fig. 9). We would like to remark that 'feature optimization' is an emerging area of interest in next generation reservoir computing; see Cestnik and Martens (2026) for an alternative approach.

### 4.4. Description of tasks the NGRC + ERT system is trained to solve

We examine the performance of a NGRC + ERT system in different tasks, each of which involve training a NGRC + ERT system to reconstruct the dynamics of $N$ attractors. All of the attractors we consider are subsets of the state spaces of the various continuous-time dynamical systems listed in the Attractors section below. Training data, i.e., elements of the input time series $\mathbf{X}$ corresponding to each attractor, are obtained

by integrating the state of a given dynamical system from a chosen initial condition up to time $(T + 5000)dt$, using the $4^{th}$-order Runge-Kutta method with integration time step $\Delta t$, and then discarding the transient of the first 5000 time steps. From this we obtain a time series of length $T$ which describes a trajectory on a given attractor that we use as $\mathbf{X}$. We continue to integrate the dynamical system for $P$ additional time step and reserve the corresponding time series of length $P$ as the 'test', i.e., ground truth, for assessing the NGRC + ERT system's performance in predicting the dynamical system over a prediction length of $P$ time steps.

#### 4.4.1. Lorenz and Halvorsen task

We consider the Lorenz system described in Eqs. (31)–(33) and the Halvorsen system described in Eqs. (28)–(30), and obtain trajectories on the corresponding Lorenz and Halvorsen attractors by integrating both systems using the 4th-order Runge-Kutta (RK4) method with time step size $\Delta t = 0.02$. We normalize both obtained trajectories to zero mean and unit standard deviation so that they overlap in state space to provide an additional level of difficulty; see Flynn et al. (2023) and Kong et al. (2024). For all tests, we apply the attractor reconstruction metrics (specified later) over a prediction length of $P = 15{,}000$ time steps.

#### 4.4.2. 16-Overlapping-attractor task

For the 16-overlapping-attractor task, we generate time series from two realizations of trajectories on attractors from each of the eight dynamical systems described in the Attractors section by integrating each system from two different initial conditions for 20,000 time steps each. Each trajectory we obtain is normalized to have zero mean and unit standard deviation. To add an extra level of difficulty to the multi-attractor task, we rotate each trajectory counter-clockwise around the $x$, $y$, and $z$ axes by angles $\theta_x$, $\theta_y$, $\theta_z$. The NGRC + ERT system is trained on the rotated trajectories and evaluated against the correspondingly rotated test data, verifying that the model successfully learns the rotation imposed on the training data. The specific rotation angles used for each attractor are provided in Supplementary Note S10, Table S2. Additionally, to further illustrate the NGRC + ERT system's ability to reconstruct attractors from different dynamical systems, we add another level of difficulty by using different time step sizes for each system, specifically: $\Delta t = 0.005$ for Chen systems, $\Delta t = 0.02$ for Chua Circuit, Halvorsen, and Lorenz systems, $\Delta t = 0.08$ for Complex Butterfly, Windmi, and Rucklidge systems, and $\Delta t = 0.1$ for Roessler systems. We apply the attractor reconstruction metrics, described later, over a prediction length of $P = 15{,}000$ time steps across all tests.

#### 4.4.3. Multiple-overlapping-attractor task

The task involves 4, 9, 16, 25, 36, and 49 overlapping attractors from different dynamical systems. The training data from the different attractors we use in this task are obtained by cycling through the available dynamical systems (described in the Attractors subsection) ensuring roughly equal representation of each attractor. Each trajectory is normalized to zero mean and unit standard deviation, and rotated along every coordinate axis. The time step $\Delta t$ for each system is the same as specified in the 16-overlapping-attractor task description.

### 4.5. Attractor reconstruction metrics

We now describe the two metrics we use to assess how accurate the NGRC + ERT system learns the **two attractor tasks** in terms of long-term and short-term behavior.

**Long-term behavior:** We define the 'long-term measurement error' in terms of how the largest Lyapunov exponent (Rosenstein et al., 1993) (LY) $\lambda$ and correlation dimension (Grassberger & Procaccia, 2004) (CD) $\nu$ of a given reconstructed attractor compares to its corresponding ground truth across all $N$ analyzed attractors here. More specifically, we compute the absolute relative error of the LY and CD of a given reconstructed attractor as,

$$\Delta\lambda = \frac{|\lambda^{\text{test}} - \lambda^{\text{pred}}|}{\lambda^{\text{test}}}, \qquad \Delta\nu = \frac{|\nu^{\text{test}} - \nu^{\text{pred}}|}{\nu^{\text{test}}},$$

where *test* correspond to the ground truth and *pred* to the reconstructed attractor over the prediction interval defined by $P$. We define the long-term measurement error (LTME) as,

$$\text{Long-term measurement error} = \frac{1}{2N}\sum_{i=1}^{N}(\Delta\lambda_i + \Delta\nu_i). \tag{17}$$

We consider the NGRC + ERT system to reconstruct the 'climate' and performs accurate long-term memory recall of both attractors in the sense of Lu et al. (2018) if the LTME $\leq 0.1$ which was found empirically. If the LTME $> 0.1$ the reconstructions begin to break down.

**Short-term behavior:** We define the 'short-term measurement error' in terms of the 'mean forecast horizon' (MFH) across all $N$ attractors. More specifically, we compute the 'forecast horizon' (FH) of a given reconstructed attractor and its corresponding ground truth both over the interval defined by $P$ as the number of time steps for which the prediction time series, $\mathbf{X}_{\text{pred}}$, remains within a threshold $\boldsymbol{\epsilon} \in \mathbb{R}^d$ of the test time series $\mathbf{X}_{\text{test}}$ across all $d$ variables:

$$\tau = \arg\max_t(|\,\mathbf{x}_{train}(t) - \mathbf{x}_{test}(t)| < \boldsymbol{\epsilon})$$

The threshold vector $\boldsymbol{\epsilon}$ is defined element-wise as a fraction of the difference between the maximum and minimum values of $\mathbf{x}_{\text{test}}(t)$ in each variable. In our case we found the following

$$\epsilon = 0.15 \cdot (\max \mathbf{x}_{test}(t) - \min \mathbf{x}_{test}(t)),$$

to be sufficient, where the max and min operators are applied separately to each variable. To quantify the short-term behavior for multi-attractor reconstruction, we compute the mean forecast horizon across all $N$ attractors:

$$\text{Mean Forecast Horizon} = \frac{1}{N}\sum_{i=1}^{N}\tau_i', \tag{18}$$

where each forecast horizon is expressed in Lyapunov time $\tau' = \tau \cdot \lambda^{\text{test}} \cdot \Delta t$. This normalization allows direct comparison across dynamical systems with different time scales ($\Delta t$) and largest Lyapunov exponents ($\lambda^{\text{test}}$).

For the **16-overlapping-attractor task**, we compare the LY, CD, and compute the FH of each reconstructed attractor individually against the corresponding ground truth over the prediction length $P$.

Note, we do not assess the performance of the NGRC + ERT system using the so-called 'memory capacity metric' as this does not provide performance insights into what we consider as memory in this paper, the ability to process, store, and recall different attractors. More specifically, we consider each attractor that the NGRC + ERT system reconstructs as a memory that has been encoded in the system due to training using the terminology developed by Kong et al. (2024). The performance metrics we use assess the quality of this memory, i.e., how similar is the reconstructed attractor to the original attractor. Moreover, in Flynn et al. (2022) it was shown that the memory capacity metric provides no insight on tasks involving the reconstruction of multiple attractors.

### 4.6. Attractors

#### 4.6.1. Chen system

$$\dot{x} = a(y - x) \tag{19}$$

$$\dot{y} = (c - a)x - xz + cy \tag{20}$$

$$\dot{z} = xy - bz \tag{21}$$

with parameters $a = 35$, $b = 3$, $c = 28$ for the 16-overlapping-attractor task and an additional parametrization of $a = 33$, $b = 3$, $c = 28$ for the multiple-overlapping-attractor task.

#### 4.6.2. Chua circuit

$$\dot{x} = \alpha(y - x + bx + 0.5(a - b)((|x| + 1) - (|x| - 1))) \quad (22)$$
$$\dot{y} = x - y + z \quad (23)$$
$$\dot{z} = -\beta y \quad (24)$$

with parameters $\alpha = 9$, $\beta = 100/7$, $a = 8/7$, $b = 5/7$ for the two-attractor task and the 16-overlapping-attractor task and an additional parametrization of $\alpha = 8.5$, $\beta = 100/7$, $a = 8/7$, $b = 5/7$ for the multiple-overlapping-attractor task.

#### 4.6.3. Complex butterfly system

$$\dot{x} = a(y - x) \quad (25)$$
$$\dot{y} = -z\frac{x}{|x|} \quad (26)$$
$$\dot{z} = |x| - 1 \quad (27)$$

with parameter $a = 0.55$ across all tasks.

#### 4.6.4. Halvorsen system

$$\dot{x} = -\sigma x - 4y - 4z - y^2 \quad (28)$$
$$\dot{y} = -\sigma y - 4z - 4x - z^2 \quad (29)$$
$$\dot{z} = -\sigma z - 4x - 4y - x^2 \quad (30)$$

with parameter $\sigma = 1.27$ for the two-attractor task and the 16-overlapping-attractor task and an additional parametrization of $\sigma = 1.7$ for the multiple-overlapping-attractor task.

#### 4.6.5. Lorenz system

$$\dot{x} = \sigma(y - x) \quad (31)$$
$$\dot{y} = x(\rho - z) - y \quad (32)$$
$$\dot{z} = xy - \beta z \quad (33)$$

with parameters $\sigma = 10$, $\rho = 28$, $\beta = 8/3$ for the two-attractor task and the 16-overlapping-attractor task and an additional parametrization of $\sigma = 10$, $\rho = 350$, $\beta = 8/3$ for the multiple-overlapping-attractor task.

#### 4.6.6. Roessler system

$$\dot{x} = -y - z \quad (34)$$
$$\dot{y} = x + ay \quad (35)$$
$$\dot{z} = b + z(x - c) \quad (36)$$

with parameters $a = 0.2$, $b = 0.2$, $c = 5.7$ for the two-attractor task and the 16-overlapping-attractor task.

#### 4.6.7. Rucklidge system

$$\dot{x} = -\kappa x + \lambda y - yz \quad (37)$$
$$\dot{y} = x \quad (38)$$
$$\dot{z} = -z + y^2 \quad (39)$$

with parameters $\kappa = 2$, $\lambda = 6.7$ for the two-attractor task and for the multiple-overlapping-attractor task.

#### 4.6.8. Windmi system

$$\dot{x} = y \quad (40)$$
$$\dot{y} = z \quad (41)$$
$$\dot{z} = -az - y + b - e^x \quad (42)$$

with parameters $a = 0.7$, $b = 2.5$ for the two-attractor task and the 16-overlapping-attractor task and an additional parametrization of $a = 0.7$, $b = 1.75$ for the multiple-overlapping-attractor task.

## CRediT authorship contribution statement

**Daniel Köglmayr:** Writing – review & editing, Visualization, Software, Project administration, Methodology, Conceptualization; **Miralem Spahic:** Writing – review & editing, Writing – original draft, Visualization, Validation, Software, Methodology, Investigation; **Andrew Flynn:** Writing – review & editing, Writing – original draft, Formal analysis; **Christoph Räth:** Writing – review & editing, Writing – original draft, Formal analysis.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Acknowledgement

D. K. gratefully acknowledges the funding provided by Allianz Global Investors (AGI). We dedicate this work to the late Erik Bollt who pursued similar research and pioneered the field. May he rest in peace.

## Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.neunet.2026.109209.

## References

Altmann, A., Toloşi, L., Sander, O., & Lengauer, T. (2010). Permutation importance: A corrected feature importance measure. *Bioinformatics*, *26*(10), 1340–1347.
Barbosa, W. A. S., & Gauthier, D. J. (2022). Learning spatiotemporal chaos using next-generation reservoir computing. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, *32*(9), 093137.
Bollt, E. (2021). On explaining the surprising success of reservoir computing forecaster of chaos? The universal machine learning dynamical system with contrast to VAR and DMD. *Chaos*, *31*(1), 013108.
Breiman, L. (2001). Random forests. *Machine Learning*, *45*(1), 5–32.
Cestnik, R., & Martens, E. A. (2026). Next-generation reservoir computing for dynamical inference. *Chaos*, *36*(1), 013115.
Dickinson, P. S. (1995). Interactions among neural networks for behavior. *Current Opinion in Neurobiology*, *5*(6), 792–798.
Flynn, A. (2023). Theory and applications of multifunctional reservoir computers. University College Cork (PhD Thesis).
Flynn, A., & Amann, A. (2024). Exploring the origins of switching dynamics in a multifunctional reservoir computer. *Frontiers in Network Physiology*, *4*, 1451812.
Flynn, A., Heilmann, O., Köglmayr, D., Tsachouridis, V. A., Räth, C., & Amann, A. (2022). Exploring the limits of multifunctionality across different reservoir computers. In *2022 International joint conference on neural networks (IJCNN)* (pp. 1–8). IEEE.
Flynn, A., Herteux, J., Tsachouridis, V. A., Räth, C., & Amann, A. (2021a). Symmetry kills the square in a multifunctional reservoir computer. *Chaos*, *31*(7), 073122.
Flynn, A., Tsachouridis, V. A., & Amann, A. (2021b). Multifunctionality in a reservoir computer. *Chaos*, *31*(1), 013125.
Flynn, A., Tsachouridis, V. A., & Amann, A. (2023). Seeing double with a multifunctional reservoir computer. *Chaos*, *33*(11), 113115.
Gauthier, D. J., Bollt, E., Griffith, A., & Barbosa, W. A. S. (2021). Next generation reservoir computing. *Nature Communications*, *12*(1), 1–8.
Getting, P. A. (1989). Emerging principles governing the operation of neural networks. *Annual Review of Neuroscience*, *12*(1), 185–204.
Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42.
Giammarese, A., Rana, K., Bollt, E. M., & Malik, N. (2025). Tree-based learning for high-fidelity prediction of chaos. *Scientific Reports*, *15*(1), 41967.

Grassberger, P., & Procaccia, I. (2004). Measuring the strangeness of strange attractors. In *The theory of chaotic attractors* (pp. 170–189). Springer.

Haluszczynski, A., Koeglmayr, D., & Räth, C. (2023). 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)* (pp. 1–7). IEEE.

Herteux, J., & Räth, C. (2020). Breaking symmetries of the reservoir equations in echo state networks. *Chaos*, *30*(12), 123142.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication. *Science*, *304*(5667), 78–80.

Kabayama, T., Komuro, M., Kuniyoshi, Y., Aihara, K., & Nakajima, K. (2025). Crisis-induced intermittency in reservoir computing. *Physical Review Research*, *7*(3), L032058.

Kim, J. Z., Lu, Z., Nozari, E., Pappas, G. J., & Bassett, D. S. (2021). Teaching recurrent neural networks to infer global temporal structure from local examples. *Nature Machine Intelligence*, *3*(4), 316–323.

Köglmayr, D. (2022). Reservoir computing based cryptography and exploration of the limits of multifunctionality in NG-RC. Master's thesis, Ludwig-Maximilians-Universität München.

Köglmayr, D., Haluszczynski, A., & Räth, C. (2026). Controlling dynamical systems into unseen target states using machine learning. *Advanced Intelligent Systems*, *8*(5), e202501319. Advance online publication.

Köglmayr, D., & Räth, C. (2024). Extrapolating tipping points and simulating non-stationary dynamics of complex systems using efficient machine learning. *Scientific Reports*, *14*(1), 507.

Kong, L.-W., Brewer, G. A., & Lai, Y.-C. (2024). Reservoir-computing based associative memory and itinerancy for complex dynamical attractors. *Nature Communications*, *15*(1), 4840.

Kong, L.-W., Fan, H.-W., Grebogi, C., & Lai, Y.-C. (2021). Machine learning prediction of critical transition and system collapse. *Physical Review Research*, *3*(1), 013090.

Kong, L.-W., Weng, Y., Glaz, B., Haile, M., & Lai, Y.-C. (2023). Reservoir computing as digital twins for nonlinear dynamical systems. *Chaos*, *33*(3), 033111.

Lu, Z., Hunt, B. R., & Ott, E. (2018). Attractor reconstruction by machine learning. *Chaos*, *28*(6), 061104.

Maass, W., Natschläger, T., & Markram, H. (2002). Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural Computation*, *14*(11), 2531–2560.

Marder, E., & Calabrese, R. L. (1996). Principles of rhythmic motor pattern generation. *Physiological Reviews*, *76*(3), 687–717.

Morra, J., Flynn, A., Amann, A., & Daley, M. (2023). Multifunctionality in a connectome-based reservoir computer. In *2023 IEEE International conference on systems, man, and cybernetics (SMC)* (pp. 4961–4966). IEEE.

O'Hagan, J., Keane, A., & Flynn, A. (2025). Confabulation dynamics in a reservoir computer: Filling in the gaps with untrained attractors. *Chaos*, *35*(9), 093130.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, *12*, 2825–2830.

Queensland Brain Institute (n.d.). Online article: Where are memories stored in the brain?, Last accessed on 18-10-2025 https://qbi.uq.edu.au/memory/where-are-memories-stored.

Raut, R. V., Rosenthal, Z. P., Wang, X., Miao, H., Zhang, Z., Lee, J.-M., Raichle, M. E., Bauer, A. Q., Brunton, S. L., Brunton, B. W. et al. (2025). Arousal as a universal embedding for spatiotemporal brain dynamics. *Nature*, *647*(8089), 454–461.

Rosenstein, M. T., Collins, J. J., & De Luca, C. J. (1993). A practical method for calculating largest Lyapunov exponents from small data sets. *Physica D: Nonlinear Phenomena*, *65*(1-2), 117–134.

Terajima, R., Inoue, K., Nakajima, K., & Kuniyoshi, Y. (2025). Multifunctional physical reservoir computing in soft tensegrity robots. *Chaos*, *35*(8), 083111.

Tulving, E. (1995). Organization of memory: Quo vadis? In M. S. Gazzaniga (Ed.), *The cognitive neurosciences* (pp. 839–847). MIT Press.

Verstraeten, D., Schrauwen, B., d'Haene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.

Yamazaki, K., Vo-Ho, V.-K., Bulsara, D., & Le, N. (2022). Spiking neural networks and their applications: A review. *Brain Sciences*, *12*(7), 863.