

Confabulation dynamics in a reservoir computer: Filling in the gaps with untrained attractors

Jack O'Hagan,¹ Andrew Keane,¹ and Andrew Flynn^{1, 2, a)}

¹⁾School of Mathematical Sciences, University College Cork, Cork T12 XF62, Ireland.

²⁾INFANT Research Centre, University College Cork, T12 DC4A Cork, Ireland.

(Dated: 5 August 2025)

Artificial Intelligence has advanced significantly in recent years thanks to innovations in the design and training of artificial neural networks (ANNs). Despite these advancements, we still understand relatively little about how elementary forms of ANNs learn, fail to learn, and generate false information without the intent to deceive, a phenomenon known as ‘confabulation’. To provide some foundational insight, in this paper we analyse how confabulation occurs in reservoir computers (RCs): a dynamical system in the form of an ANN. RCs are particularly useful to study as they are known to confabulate in a well-defined way: when RCs are trained to reconstruct the dynamics of a given attractor, they sometimes construct an attractor that they were not trained to construct, a so-called ‘untrained attractor’ (UA). This paper sheds light on the role played by UAs when reconstruction fails and their influence when modelling transitions between reconstructed attractors. Based on our results, we conclude that UAs are an intrinsic feature of learning systems whose state spaces are bounded, and that this means of confabulation may be present in systems beyond RCs.

As artificial intelligence (AI) becomes increasingly integrated into modern society, the reliability and trustworthiness of AI-based systems is a growing concern. One of the main factors driving this concern is that these systems are prone to ‘confabulation’: generating false, but plausible, information. In this paper, we investigate the phenomenon of confabulation using a mathematically well-defined machine learning framework known as reservoir computing. Specifically, we consider training an artificial neural network in the form of a reservoir computer (RC) on tasks involving attractor reconstruction, a process which mimics how memories (i.e., attractors) are encoded in the brain. We investigate how RCs confabulate by generating attractors that they have not been trained to construct, so-called ‘untrained attractors’ (UAs). By analogy, these UAs are similar to false memories. We explore key questions such as, how and why do UAs arise in RCs? What are the most common types of UAs? What design features of RCs contribute to the generation of UAs? Beyond attractor reconstruction, how do RCs fill in gaps in their knowledge when generating transitions between attractors?

I. INTRODUCTION

The creation of false information by artificial neural networks (ANNs) to fill gaps in their memory has become a well-documented issue, particularly since the rise of generative artificial intelligence (AI) technologies such as OpenAI’s *ChatGPT*. For instance, when asked to perform a literature review on a particular topic, ChatGPT may provide information on a paper whose title appears to be genuine but the paper does not actually exist^{1,2}. This phenomenon is commonly referred to as a ‘hallucination’. In recognition of the scale of this problem,

Cambridge Dictionary chose ‘hallucinate’ as its 2023 word of the year³.

However, the use of hallucination to describe this phenomenon has sparked significant debate since hallucinations typically occur due to false sensory perceptions, such as seeing or hearing things that do not exist in reality. Many argue that ‘confabulation’ provides a more accurate analogy^{4–6}. This term originates from its use to describe the generation of certain types of false memories by humans, where gaps in memory are filled with false, yet usually plausible, information⁷. Confabulation often arises when normal neural activity has been disrupted, as seen in cases of Korsakoff’s syndrome⁸, Alzheimer’s disease⁹, and other memory-related neurological disorders. These disruptions are believed to prevent the brain from accurately encoding, storing, or retrieving information, thereby leading to false or distorted memories. More specifically, there are two types of confabulations in the psychology literature (see Gilboa *et al.*¹⁰ and references therein). “Momentary” (or “provoked”) confabulations are mild distortions of a pre-existing memory and might appear as embellishments or elaborations. “Fantastic” (or “spontaneous”) confabulations are less common and describe unlikely, sometimes bizarre, events.

While the abilities and applications of ANNs in AI-based technologies continue to advance at a remarkable pace, unfortunately the same cannot be said for our ability to explain how even the most elementary forms of ANNs learn, fail to learn, or confabulate. This lack of explainability presents a critical challenge for ensuring the reliability and trustworthiness of AI-based technologies, particularly as their adoption accelerates across scientific, industrial, and societal domains^{11,12}. In order to improve our understanding of how confabulations arise in relatively simple ANN designs, in this paper we choose to study the dynamics of confabulation in a ‘reservoir computer’ (RC)^{13–15}.

A RC is a dynamical system that is typically realised as an ANN with recurrent connections and can be trained to perform a wide variety of tasks, with applications ranging from control¹⁶ to weather prediction¹⁷, to name but a few. These

^{a)}Electronic mail: andrew_flynn@umail.ucc.ie

RC-based ANNs typically consist of randomly realised input and internal layers, and an output layer that is trained in order for the RC to perform a particular task. This philosophy of not training the input and internal network weights has inspired researchers to use a variety of physical mediums as the internal layer of a RC; we recommend Tanaka *et al.*¹⁸ and Nakajima¹⁹ for more information on the many interesting examples of ‘physical RCs’.

RCs also provide an ideal framework to systematically study how confabulations occur in ANNs as they are publicly-available, relatively inexpensive to train, and mathematically well-defined. RCs are known to exhibit a distinct form of confabulation: when trained to reconstruct the dynamics of a given attractor, the RC can construct attractors it was not trained to reconstruct, so-called ‘untrained attractors’ (UAs). UAs were first discovered by Flynn *et al.*²⁰ by initialising the trained RC (an autonomous dynamical system) from many different initial conditions and observing that certain initial conditions were within the basin of attraction of the reconstructed attractor and others were not. RCs also provide a valuable bridge between the brain and AI. A RC’s learning process mirrors certain aspects of how the brain learns new tasks²¹, the process of training a RC to reconstruct an attractor is also viewed in a similar vein to how the brain stores long-term episodic memories²². Thus, by extension, when the RC fails to accurately reconstruct an attractor it instead produces a false memory. The case of a poor reconstruction is analogous to a momentary confabulation, where “moments” or parts of the trajectory are inconsistent with the original attractor (e.g., see Fig. 2). An UA is an example of a fantastic confabulation, where the “complete” trajectory is inconsistent with the original attractor. Hence, we refer to this case as “complete confabulation” for the rest of the paper. As we will see, it mirrors how gaps in a human’s memory may be replaced with false information.

Taking into consideration the above associations between the brain, memory, and a RC’s learning process, we conduct experiments to investigate the dynamics of confabulation when (i) storing an individual memory/attractor, (ii) interpolating between related memories/attractors, and (iii) interpolating between unrelated memories/attractors (attractors from different systems that do not share the same nonlinearities). To be more specific, we study the dynamics of confabulation after training RCs to perform the following three tasks: (i) reconstruct the dynamics of the Lorenz attractor, (ii) reconstruct a period-1 and period-4 limit cycle at two different parameter settings, and (iii) reconstruct the Lorenz and Halvorsen attractor at two different parameter settings. For task (i) we train the RC presented in Lu *et al.*²³ and adapt this RC for tasks (ii) and (iii) by adding an additional ‘parameter-aware’ component using a simplified version of the schemes introduced by Kim *et al.*²⁴ and Kong *et al.*²⁵.

From these tasks we identify key parameter relationships that influence the likelihood of UAs appearing. Most importantly, we uncover the gap-filling mechanism that gives rise to UAs: the state of the RC has no choice but to approach an UA when it completely fails to reconstruct an attractor, since, like many other learning systems, the RC we work with is an

example of a bounded system, i.e. its state cannot approach infinity.

Another motivation behind these tasks is that they help us to understand how an ANN-based system generalises beyond its training data and what gap filling mechanisms it may use. While these tasks differ from what one would ordinarily associate with the capabilities of highly sophisticated networks of interest, such as the human brain or generative AI models, they are chosen to study some of the circumstances in which confabulations arise when storing and recalling a piece of information (task (i)), generalising between related pieces of information (task (ii)), and generalising between unrelated pieces of information (task (iii)).

When discussing confabulation mechanisms in the wider literature, researchers from both the psychology and machine learning communities argue that confabulations may arise when the brain or some learning system is ‘filling in the gaps’ in its memory in cases where an accurate solution is not found for a particular query. For instance, in the APA Dictionary of Psychology, confabulations are defined as “the falsification of memory in which gaps in recall are filled by fabrications that the individual accepts as fact”²⁶. In a machine learning context, Nejjar *et al.*²⁷ describe confabulations in terms of “...where an LLM [large language model] seemingly fills gaps in the information contained in the model with plausibly-sounding words”. However, to clearly identify and visualise such ‘gaps’ being filled on a network level is challenging due to the complexity of these network systems. In contrast, the value of our RC approach is that it is mathematically tractable and we are able to produce diagrams of intended, learned solutions with gaps that are naturally filled by alternative, sometimes unexpected, solutions. More specifically, our results illustrate that what we observe as confabulations is literally the presence of unexpected solutions (UAs) “filling in the gaps”.

The rest of the paper is organised as follows, in Sec. II we describe the experiments the conducted and the steps involved in training the RC to do so, in Secs. III–V we outline the results from training the RC to perform tasks (i)–(iii), and in Sec. VI we provide some final remarks.

II. SETUP OF NUMERICAL EXPERIMENTS

A. RC setup

The RC studied throughout this paper was presented by Lu *et al.*²³. There are two stages involved in training this RC to perform the tasks considered in this paper, a listening stage and a training stage. In both stages the RC, defined as the nonautonomous dynamical system in Eq. (1),

$$\dot{\mathbf{r}}(t) = \gamma[-\mathbf{r}(t) + \tanh(\mathbf{M}\mathbf{r}(t) + \sigma\mathbf{W}_{in}\mathbf{u}(t))], \quad (1)$$

$$\mathbf{r}(0) = \mathbf{0}^T, \quad (2)$$

is driven by an input $\mathbf{u}(t) \in \mathcal{A} \subset \mathbb{R}^D$, where \mathcal{A} is some attractor that we choose and D is the number of variables of the system that \mathcal{A} belongs to. We refer to this network driven by the input $\mathbf{u}(t)$ as the ‘open-loop RC’. The response to this

driving input is used for training purposes and is given by values of $\mathbf{r}(t) \in \mathbb{R}^N$, which describes the state of the open-loop RC at a given time t and N is the number of artificial neurons in the network. Values of $\mathbf{r}(t)$ are obtained by computing solutions of Eq. (1) using the 4th order Runge-Kutta method with time step τ . γ is a decay-rate parameter arising from the derivation of this RC from the discrete-time system proposed by Jaeger [13]. The hyperbolic tangent ‘activation function’ is a pointwise operation and is defined as $\tanh(\cdot) : \mathbb{R}^N \rightarrow \mathbb{R}^N$. The internal connections of the network are described by the adjacency matrix, $\mathbf{M} \in \mathbb{R}^{N \times N}$. The input strength parameter, σ , and the input matrix, $\mathbf{W}_{in} \in \mathbb{R}^{N \times D}$, when multiplied together represent the weight given to $\mathbf{u}(t)$, as it is fed into the open-loop RC. The steps involved in constructing \mathbf{M} and \mathbf{W}_{in} are outlined in Appendix A.

In our experiments, we vary the ‘spectral radius’ of \mathbf{M} , the maximum of the absolute values of the eigenvalues of a given matrix. We denote this quantity by $\rho \geq 0$. Thus, for \mathbf{M} to have a particular ρ , we rescale its elements accordingly. We show that ρ plays a crucial role in the confabulation dynamics of this RC design. ρ has also been a key parameter in our previous results on RCs overcoming the influence of UAs [20, 28–30] and features prominently in numerous research papers that focus on understanding the dynamics of how a RC learns.

B. Training RC to reconstruct an attractor

Let us first consider the case of training this RC to reconstruct the dynamics of a single attractor, $\mathcal{A} \subset \mathbb{R}^D$, when given access to a trajectory on \mathcal{A} described by a vector $\mathbf{u}(t) \in \mathcal{A}$.

In this case, the listening stage is the duration of time where Eq. (1) is driven by $\mathbf{u}(t)$ for $0 \leq t \leq t_{\text{listen}}$ where t_{listen} is chosen such that $\mathbf{r}(t)$ is determined by a history of driving inputs and is no longer dependent on its initial condition. The training stage is the time where Eq. (1) is driven by $\mathbf{u}(t)$ for $t_{\text{listen}} \leq t \leq t_{\text{train}}$.

Training consists of finding a ‘readout function/layer’ defined as $\hat{\psi}(\cdot) : \mathbb{R}^{2N} \rightarrow \mathbb{R}^D$ that replaces $\mathbf{u}(t)$ in Eq. (1) and is written as

$$\hat{\psi}(\mathbf{r}(t)) = \mathbf{W}_{out} \mathbf{q}(\mathbf{r}(t)), \quad (3)$$

where $\mathbf{W}_{out} \in \mathbb{R}^{D \times 2N}$ is the ‘readout matrix’ and $\mathbf{q}(\mathbf{r}(t)) \in \mathbb{R}^{2N}$ is given by

$$\mathbf{q}(\mathbf{r}(t)) = \begin{pmatrix} \mathbf{r}(t) \\ \mathbf{r}^2(t) \end{pmatrix}, \quad (4)$$

where $\mathbf{r}^2(t) = (r_1^2(t), r_2^2(t), \dots, r_N^2(t))^T$. From Eq. (4) it follows that Eq. (3) can be rewritten as

$$\hat{\psi}(\mathbf{r}(t)) = \mathbf{W}_{out}^{(1)} \mathbf{r}(t) + \mathbf{W}_{out}^{(2)} \mathbf{r}^2(t), \quad (5)$$

where $\mathbf{W}_{out}^{(1)}$ is the ‘linear readout matrix’, and $\mathbf{W}_{out}^{(2)}$ is the ‘square readout matrix’. The square readout matrix breaks the symmetry in Eq. (1) when replacing $\mathbf{u}(t)$ with $\hat{\psi}(\mathbf{r}(t))$ after the training stage and prevents the occurrence of ‘mirror-attractors’, which can impede the ability of the RC to reconstruct attractors [31, 32].

\mathbf{W}_{out} is determined by the ridge regression technique, which consists of minimising the expression

$$\frac{1}{t^* - l^*} \left[\sum_{i=l^*}^{t^*} \|\mathbf{W}_{out} \mathbf{q}(\mathbf{r}[i]) - \mathbf{u}[i]\|_2^2 + \beta \|\mathbf{W}_{out}\|_2^2 \right] \quad (6)$$

with respect to \mathbf{W}_{out} , where $\|\cdot\|_2$ is the Euclidean norm, $l^* = t_{\text{listen}}/\tau$ and $t^* = t_{\text{train}}/\tau$, $\mathbf{r}[i]$ and $\mathbf{u}[i]$ are discrete-time samples of the continuous-time variables $\mathbf{r}(t)$ and $\mathbf{u}(t)$ at discrete time $i = t/\tau$. The corresponding time series, $[\mathbf{r}[i]]_{i=l^*}^{t^*}$ and $[\mathbf{u}[i]]_{i=l^*}^{t^*}$, are constructed by sampling $\mathbf{r}(t)$ and $\mathbf{u}(t)$ at intervals of length τ (discrete-time intervals of length 1). β is the regularisation parameter and the purpose of the $\beta \|\mathbf{W}_{out}\|_2^2$ term in Eq. (6) is to modify the linear least-squares regression to reduce the magnitudes of elements in \mathbf{W}_{out} in order to discourage overfitting.

Minimising Eq. (6) involves computing its partial derivative with respect to \mathbf{W}_{out} and setting the resulting expression equal to 0. The \mathbf{W}_{out} that minimises Eq. (6) is given by

$$\mathbf{W}_{out} = \mathbf{Y} \mathbf{X}^T (\mathbf{X} \mathbf{X}^T + \beta \mathbf{I})^{-1}, \quad (7)$$

where

$$\mathbf{X} = [\mathbf{q}(\mathbf{r}[l^*]) \ \mathbf{q}(\mathbf{r}[l^*+1]) \ \cdots \ \mathbf{q}(\mathbf{r}[t^*])] \quad (8)$$

is the ‘response data matrix’,

$$\mathbf{Y} = [\mathbf{u}[l^*] \ \mathbf{u}[l^*+1] \ \cdots \ \mathbf{u}[t^*]] \quad (9)$$

is the ‘input data matrix’ and \mathbf{I} is the identity matrix.

After the training, $\mathbf{u}(t)$ in Eq. (1) is replaced by $\hat{\psi}(\mathbf{r}(t))$. In Eq. (10) we now define the ‘closed-loop’ RC as the following autonomous dynamical system,

$$\dot{\hat{\mathbf{r}}}(t) = \gamma [-\hat{\mathbf{r}}(t) + \tanh(\mathbf{M} \hat{\mathbf{r}}(t) + \sigma \mathbf{W}_{in} \mathbf{W}_{out} \mathbf{q}(\hat{\mathbf{r}}(t)))], \quad (10)$$

$$\hat{\mathbf{r}}(0) = \mathbf{r}(t_{\text{train}}), \quad (11)$$

where $\hat{\mathbf{r}}(t)$ denotes the state of the closed-loop RC at a given time t . To distinguish between the N -dimensional vectors, $\hat{\mathbf{r}}(t)$ and $\mathbf{r}(t)$, we define $\hat{\mathbf{r}}(t) \in \mathbb{S}$, where \mathbb{S} is referred to as the ‘RC’s state space’ and is used henceforth when discussing the dynamics of the closed-loop RC. By computing solutions of Eq. (10), predictions of $\mathbf{u}(t)$ for $t > t_{\text{train}}$, denoted as $\hat{\mathbf{u}}(t)$, are given by

$$\hat{\mathbf{u}}(t) = \hat{\psi}(\hat{\mathbf{r}}(t)). \quad (12)$$

Again, while both $\mathbf{u}(t)$ and $\hat{\mathbf{u}}(t)$ are D -dimensional vectors, to distinguish the dynamics of $\hat{\mathbf{u}}(t)$ from $\mathbf{u}(t)$, we define $\hat{\mathbf{u}}(t) \in \mathbb{P}$, where \mathbb{P} is referred to as the ‘projected state space’ and is used henceforth when discussing the dynamics of the closed-loop RC projected via $\hat{\psi}$.

We say that the closed-loop RC has achieved attractor reconstruction when the long-term dynamical characteristics of $\hat{\mathbf{u}}(t)$ are ‘indistinguishable’ from $\mathbf{u}(t)$, i.e. there exists an attractor $\mathcal{S} \subset \mathbb{S}$ such that when the state of the closed-loop RC approaches \mathcal{S} and is projected from \mathbb{S} to \mathbb{P} via $\hat{\psi}(\cdot)$, the dynamics of the ‘reconstructed attractor’, $\hat{\mathcal{A}} \subset \mathbb{P}$, resembles the

dynamics of \mathcal{A} in the long-term. By resembling the long-term dynamics it is meant that, for instance, \mathcal{A} and $\hat{\mathcal{A}}$ will have nearly identical Poincaré sections when computed for the same region of \mathbb{R}^D and \mathbb{P} as $t \rightarrow \infty$ for $t > t_{\text{train}}$. The method we use to determine this is specified in Appendix B.

C. Training a RC to reconstruct attractors at different parameter settings

We use a simplified version of the training procedure outlined in Kim *et al.*²⁴ and Kong *et al.*²⁵ to train a RC to reconstruct two different attractors, \mathcal{A}_1 and \mathcal{A}_2 , each at a different parameter settings of the RC. To do this, we adapt the $\tanh(\cdot)$ activation function in Eq. (1) to include a constant bias vector, $\mathbf{b} \in \mathbb{R}^N$, which is set to different values when reconstructing \mathcal{A}_1 or \mathcal{A}_2 . The resulting ‘parameter-aware’ open-loop RC is described as follows,

$$\dot{\mathbf{r}}(t) = \gamma[-\mathbf{r}(t) + \tanh(\mathbf{M}\mathbf{r}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{u}(t) + \mathbf{b})]. \quad (13)$$

The same initial condition specified in Eq. (2) and integration scheme are used to generate solutions of Eq. (13).

There are also two stages involved in training this RC, a listening and training stage. In both stages, we first drive the parameter-aware open-loop RC in Eq. (13) with input $\mathbf{u}_{(\mathcal{A}_1)}(t) \in \mathcal{A}_1$ for $0 < t \leq t_{\text{train}}$ with $\mathbf{b} = \mathbf{b}_1 = (b)\mathbf{1}^T$ and then repeat for $\mathbf{u}_{(\mathcal{A}_2)}(t) \in \mathcal{A}_2$ with $\mathbf{b} = \mathbf{b}_2 = -(b)\mathbf{1}^T$ for a given $b > 0$. This choice of setting $\mathbf{b}_1 = -\mathbf{b}_2$ is to further simplify the training procedure.

The responses of the open-loop RC to these driving inputs are denoted by $\mathbf{r}_{(\mathcal{A}_1)}(t)$ and $\mathbf{r}_{(\mathcal{A}_2)}(t)$. It is important to highlight that \mathbf{M} , \mathbf{W}_{in} , and all training parameters remain identical when generating $\mathbf{r}_{(\mathcal{A}_1)}(t)$ and $\mathbf{r}_{(\mathcal{A}_2)}(t)$ in this setup.

The same readout function design in Eq. (3) is used. To distinguish between the \mathbf{W}_{out} obtained from training the RC to reconstruct a single attractor and a collection of attractors, we denote the readout matrix used in the parameter-aware RC setup as $\mathbf{W}_{\text{out}}^C$ and the ridge regression approach for computing $\mathbf{W}_{\text{out}} \in \mathbb{R}^{D \times 2N}$ in Eq. (6) is modified to account for additional attractors. This consists of minimising the following with respect to $\mathbf{W}_{\text{out}}^C$,

$$\frac{1}{t^* - l^*} \left[\sum_{i=l^*}^{t^*} \|\mathbf{W}_{\text{out}}^C \mathbf{q}(\mathbf{r}_{(\mathcal{A}_1)}[i]) - \mathbf{u}_{(\mathcal{A}_1)}[i]\|_2^2 + \sum_{i=l^*}^{t^*} \|\mathbf{W}_{\text{out}}^C \mathbf{q}(\mathbf{r}_{(\mathcal{A}_2)}[i]) - \mathbf{u}_{(\mathcal{A}_2)}[i]\|_2^2 + \beta \|\mathbf{W}_{\text{out}}^C\|_2^2 \right], \quad (14)$$

where $\mathbf{r}_{(\mathcal{A}_1)}[i]$, $\mathbf{u}_{(\mathcal{A}_1)}[i]$, $\mathbf{r}_{(\mathcal{A}_2)}[i]$, and $\mathbf{u}_{(\mathcal{A}_2)}[i]$ are discrete-time samples of $\mathbf{r}_{(\mathcal{A}_1)}(t)$, $\mathbf{u}_{(\mathcal{A}_1)}(t)$, $\mathbf{r}_{(\mathcal{A}_2)}(t)$, and $\mathbf{u}_{(\mathcal{A}_2)}(t)$, constructed using the same convention outlined in Sec. II A. β plays the same role as discussed in Sec. II A. The $\mathbf{W}_{\text{out}}^C$ that minimises Eq. (14) is given by

$$\mathbf{W}_{\text{out}}^C = \mathbf{Y}_C \mathbf{X}_C^T (\mathbf{X}_C \mathbf{X}_C^T + \beta \mathbf{I})^{-1}. \quad (15)$$

$\mathbf{X}_C = [\mathbf{X}_{(\mathcal{A}_1)}, \mathbf{X}_{(\mathcal{A}_2)}]$ and $\mathbf{Y}_C = [\mathbf{Y}_{(\mathcal{A}_1)}, \mathbf{Y}_{(\mathcal{A}_2)}]$, where $\mathbf{X}_{(\mathcal{A}_1)}$ and $\mathbf{X}_{(\mathcal{A}_2)}$ are constructed as in Eq. (8) for the corresponding

$\mathbf{r}_{(\mathcal{A}_1)}[i]$ and $\mathbf{r}_{(\mathcal{A}_2)}[i]$ and similarly for $\mathbf{Y}_{(\mathcal{A}_1)}$ and $\mathbf{Y}_{(\mathcal{A}_2)}$ as in Eq. (9) for $\mathbf{u}_{(\mathcal{A}_1)}[i]$ and $\mathbf{u}_{(\mathcal{A}_2)}[i]$. In Eq. (15), \mathbf{I} is the identity matrix of the appropriate size.

The parameter-aware closed-loop RC is then described by the same autonomous continuous-time dynamical system as in Eq. (10) with the additional \mathbf{b} term included in the activation function and using the $\mathbf{W}_{\text{out}}^C$ obtained from Eq. (15). For completeness, the parameter-aware closed-loop RC setup we study is described as follows,

$$\dot{\hat{\mathbf{r}}}(t) = \gamma[-\hat{\mathbf{r}}(t) + \tanh(\mathbf{M}\hat{\mathbf{r}}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}^C \mathbf{q}(\hat{\mathbf{r}}(t)) + \mathbf{b})]. \quad (16)$$

To reconstruct the dynamics of either \mathcal{A}_1 or \mathcal{A}_2 , the \mathbf{b} term in the activation function needs to be chosen to correspond with the value used during the training and the parameter-aware closed-loop RC needs to be initialised with the corresponding $\hat{\mathbf{r}}(0) = \mathbf{r}_{(\mathcal{A}_1)}(t_{\text{train}})$ or $\mathbf{r}_{(\mathcal{A}_2)}(t_{\text{train}})$ (or some other initial condition that is in the basin of attraction of the corresponding \mathcal{S}_1 or \mathcal{S}_2).

The same steps outlined above can be extended in order to produce a parameter-aware closed-loop RC that reconstructs n attractors, $\{\mathcal{A}_1, \mathcal{A}_2, \dots, \mathcal{A}_n\}$, at parameter settings $\{\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n\}$, respectively. All that is required is to produce the corresponding $\mathbf{X}_C = [\mathbf{X}_{(\mathcal{A}_1)}, \mathbf{X}_{(\mathcal{A}_2)}, \dots, \mathbf{X}_{(\mathcal{A}_n)}]$ and $\mathbf{Y}_C = [\mathbf{Y}_{(\mathcal{A}_1)}, \mathbf{Y}_{(\mathcal{A}_2)}, \dots, \mathbf{Y}_{(\mathcal{A}_n)}]$ and solve for $\mathbf{W}_{\text{out}}^C$ as in Eq. (15).

It is also worth mentioning that if the same process is repeated with $\mathbf{b} = \mathbf{0}$, and the RC reconstructs each attractor in a co-existing fashion, then the RC is said to exhibit multifunctionality.

D. Defining untrained and generated attractors

Before proceeding any further, it is important to define what we consider to be an UA in terms of training a RC to reconstruct an attractor or to reconstruct attractors at different parameter settings via the steps outlined in Secs. II B and II C.

Definition 1: An attractor, \mathcal{Z} , is considered to be an ‘untrained attractor’ if (i) \mathcal{Z} was not present during training, and (ii) \mathcal{Z} does not originate from a continuous deformation/displacement or bifurcation of a given reconstructed attractor by varying certain parameters of the given reservoir computer.

This definition is consistent with our previous studies of UAs in the context of multifunctionality^{20,28–30}, as we referred to the attractors that the RC was explicitly trained to reconstruct as the ‘reconstructed attractors’, and the attractors that the RC was not explicitly trained to reconstruct as ‘untrained attractors’. However, when it comes to discussing the dynamics of a parameter-aware closed-loop RC, one must take further caution on how to refer to attractors that the RC was not explicitly trained to reconstruct that are also not UAs.

Let us consider the following, a parameter-aware RC is explicitly trained to reconstruct attractors at different parameter settings, in our case the values of \mathbf{b} . Thus, if the training is successful, then for each of the chosen values of \mathbf{b} , there exists a corresponding reconstructed attractor. However, if

the parameter-aware RC interpolates/extrapolates the dynamics of the reconstructed attractors between/beyond the chosen values of b , which involves the reconstructed attractors undergoing either a continuous deformation/displacement or a series of bifurcations, then it is fair to say that the RC was not explicitly trained to reconstruct such attractors. However, to call any of these attractors an UA would contradict Definition 1. To resolve this issue we introduce the following definition.

Definition 2: An attractor, \mathcal{G} , is considered to be a ‘generated attractor’ if (i) \mathcal{G} was not present during training, and (ii) \mathcal{G} originates from a continuous deformation/displacement or bifurcation of a given reconstructed attractor by varying certain parameters of the given reservoir computer.

We choose the term ‘generated’ to avoid having to refer to such an attractor as an interpolated/extrapolated attractor based on the value of b . To clarify, we only discuss generated attractors (GAs) in the context of parameter-aware RCs.

We would also like to remark that Definition 2 is designed to compliment Definition 1 in terms of their subtle differences and also to apply to attractors of all types. As a result, Definition 2 may seem relatively trivial in the context of fixed-points and limit cycles. However, we believe that the strength of Definition 2 is its generality.

We denote GAs by $\hat{\mathcal{A}}^*$ to emphasise its connection to the reconstructed attractor $\hat{\mathcal{A}}$. For completeness, when presenting our results, the attractor \mathcal{A}_i that we explicitly train the RC to reconstruct at parameter setting b_i is labelled by $\hat{\mathcal{A}}_i$. The corresponding family of GAs that exist at values of b not used during training are labelled as $\hat{\mathcal{A}}_{i,j}^*$. If, for instance, there is a family of GAs that connect a pair of reconstructed attractors, $\hat{\mathcal{A}}_i$ and $\hat{\mathcal{A}}_j$, then we refer to this family as $\hat{\mathcal{A}}_{i,j}^*$.

E. Description of tasks and experiments

In Secs. II E 1–II E 3, we describe how the training data is generated for each of the three tasks mentioned in Sec. I and outline the specifics of the experiments that are carried out in relation to each task. A schematic of the main aspects involved in each task is provided in Fig. 1.

1. Task (i) description

For task (i) we focus on exploring the dynamics of confabulation when training the open-loop RC in Eq. (1) to reconstruct the dynamics of the Lorenz attractor, \mathcal{L} , by finding a suitable \mathbf{W}_{out} matrix according to the steps outlined in Sec. II B. For training data, we obtain a trajectory on \mathcal{L} by generating solutions of the Lorenz system³³,

$$\begin{aligned}\dot{x}_1 &= 10(x_2 + x_1), \\ \dot{x}_2 &= x_1(28 - x_3) - x_2, \\ \dot{x}_3 &= x_1x_2 - \frac{8}{3}x_3,\end{aligned}\tag{17}$$

until $t = t_{predict}$ using the 4th order Runge-Kutta method with time step τ . For this task, we set $t_{listen} = 100$, $t_{train} =$

1.	The only attractor present in \mathbb{P} is $\hat{\mathcal{L}}$.
2.	The only attractor present in \mathbb{P} is $\tilde{\mathcal{L}}$.
3.	The only attractor(s) present in \mathbb{P} are UA(s).
4.	$\hat{\mathcal{L}}$ coexists with one or several UAs in \mathbb{P} .
5.	$\tilde{\mathcal{L}}$ coexists with one or several UAs in \mathbb{P} .

TABLE I: Attractors found in \mathbb{P} after training Eq. (1) to reconstruct the Lorenz attractor, \mathcal{L} .

$t_{listen} + 100$, $t_{predict} = t_{train} + 100$, $\sigma = 0.2$, $\beta = 0.001$, $\gamma = 10$, and consider several different values for ρ .

To assess how well the closed-loop RC reconstructs \mathcal{L} , we use the method outlined in Appendix B. From this we classify the output from a given initial condition as a ‘good’ reconstruction of \mathcal{L} , denoted by $\hat{\mathcal{L}}$, a ‘poor’ reconstruction of \mathcal{L} , denoted by $\tilde{\mathcal{L}}$. An example of good and poor reconstructions of \mathcal{L} , as determined through the steps outlined in Appendix B, is provided in Fig. 2. If there is no reconstruction of \mathcal{L} , i.e. there is no stable attractor located nearby \mathcal{L} with properties resembling \mathcal{L} , we then say that the state of the closed-loop RC approaches an UA, corresponding to complete confabulation. We find a wide variation of dynamics corresponding to $\tilde{\mathcal{L}}$ and later show examples of these attractors in Sec. III and Appendix D.

\mathbf{M} and \mathbf{W}_{in} are realised randomly in this RC setup, therefore, RCs with different randomly realised \mathbf{M} and \mathbf{W}_{in} may result in different training outcomes. Thus, as indicated in Fig. 1, for a given parameter setting of the RCs we study, UAs may be present for one realisation of \mathbf{M} and \mathbf{W}_{in} and not present in other realisations. From this, many questions arise, such as how often do UAs appear and what is the most common type of UA that appears when reconstructing \mathcal{L} ? However, exploring all realisations of \mathbf{M} and \mathbf{W}_{in} may not be so practical, depending on the size of the network and the degree of weight variation.

To gain some insight into the questions posed above, we consider training an ensemble of RCs with 50 different randomly realised versions of \mathbf{M} and one random realisation of \mathbf{W}_{in} for $N = 100$. Once these 50 realisations of \mathbf{M} have been generated, we rescale the weights of each \mathbf{M} to have a specific spectral radius, ρ . To stress this point further, we do not generate 50 new realisations of \mathbf{M} each time we change ρ . As indicated in Fig. 1, we initialise each of the closed-loop RCs from 100 different random initial conditions to determine whether there are any UAs present in each respective \mathbb{P} . We then classify the output of a given closed-loop RC (dynamics in \mathbb{P}) depending on the different attractors it approaches using the approach specified in Appendix B. Based on the resulting classifications, we identify five different scenarios on the type of dynamics seen in \mathbb{P} that we specify in Table I. In Sec. III we show that the frequency of these scenarios vary depending on the choice of ρ .

Building on this, we investigate some of the most common ways in which $\hat{\mathcal{L}}$ appears. We find that these mainly appear to involve saddle type bifurcations. Interestingly, another com-

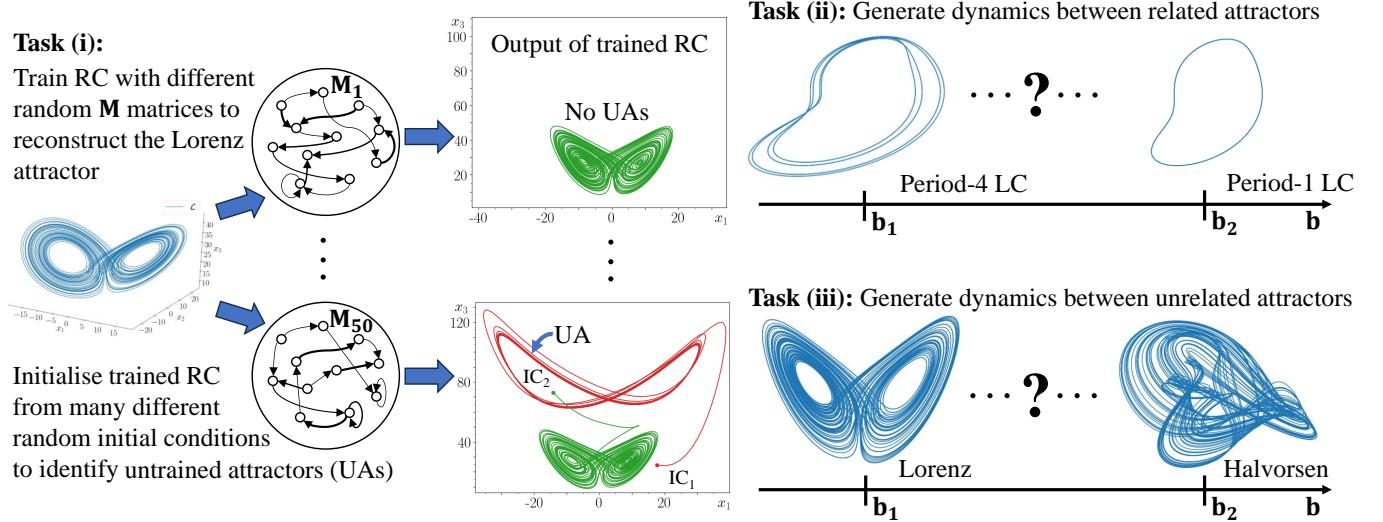
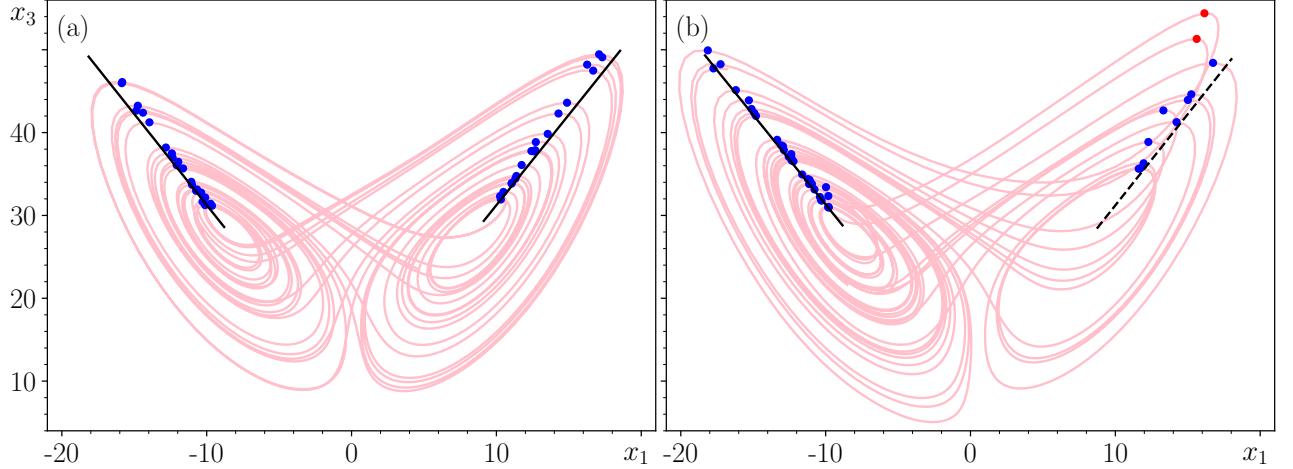


FIG. 1: Overview of tasks considered to explore confabulation dynamics in a RC.

FIG. 2: Illustration of decision process in distinguishing between a ‘good’ reconstruction of $\mathcal{L} (\hat{\mathcal{L}})$ in (a) and a ‘bad’ reconstruction of $\mathcal{L} (\tilde{\mathcal{L}})$ in (b). Blue points indicate local maxima that satisfy classification criteria, red point indicate local maxima that do not. Solid black lines indicate a good fit between \mathcal{L} and its reconstruction, dashed black lines indicate a poor fit.

mon case in which $\hat{\mathcal{L}}$ appears is through a period-doubling route to chaos. What is also interesting is that we do not find any cases where $\hat{\mathcal{L}}$ comes into existence through the well-known sequence of events that unfold by varying the constant term (28) in the \dot{x}_2 equation in Eq. (17). We discuss the ramifications of these results in greater detail in Sec. VI.

We would like to clarify that the reason why we chose to conduct our experiments with 50 different realisations of \mathbf{M} as opposed to 50 different realisations of both \mathbf{M} and \mathbf{W}_{in} was to simply only vary one component of the RC at a given time. From experiments not presented in the present paper, we found that different UAs can also appear for several different random realisations of \mathbf{W}_{in} and a given random \mathbf{M} . In future work we intend to investigate the relationship between confabulation dynamics and \mathbf{W}_{in} in much greater detail.

2. Task (ii) description

The purpose of task (ii) is to investigate the dynamics of confabulation that can arise after the parameter-aware RC has been trained to reconstruct two different, but *related attractors* at two different parameter settings. By ‘related’ we mean that there ordinarily exists a trivial sequence of transitions between these attractors. For task (ii) we consider training the parameter-aware open-loop RC in Eq. (13) to reconstruct a period-1 and period-4 limit cycle at two different parameter settings according to the steps outlined in Sec. II C.

For training data in task (ii), we choose to generate solutions of the system studied in Li *et al.*³⁴ to obtain a trajectory

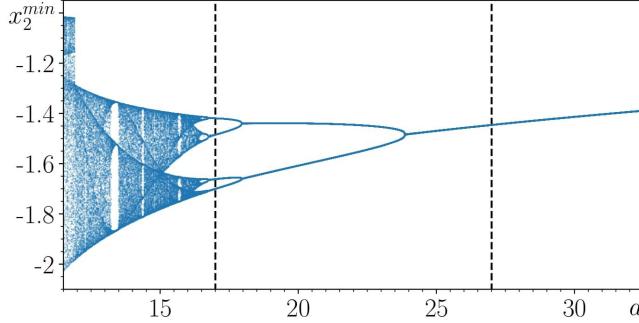


FIG. 3: Bifurcation diagram to be reconstructed in task (ii). Changes in local x_2 minima of attractor from Eq. (18) with respect to changes in a . Black lines indicate values of a used as training data.

on a period-1 and period-4 limit cycle,

$$\begin{aligned}\dot{x}_1 &= x_2(1+x_3), \\ \dot{x}_2 &= x_3(x_2-ax_1), \\ \dot{x}_3 &= 0.55x_3^2-x_2^2.\end{aligned}\quad (18)$$

For both limit cycles, we generate solutions of Eq. (18) up to $t = t_{predict}$ using the 4th order Runge-Kutta method with time step τ . For this task, we set $t_{listen} = 100$, $t_{train} = t_{listen} + 200$, and $t_{predict} = t_{train} + 200$. When $a = 17$ we obtain a period-4 limit cycle and when $a = 27$ we obtain a period-1 limit cycle, trajectories on these limit cycles are illustrated in the portion of Fig. 1 associated with task (ii).

When driving the parameter-aware open-loop RC setup in Eq. (13) with input from these limit cycles, we set $\mathbf{b} = \mathbf{b}_1 = (b)\mathbf{1}^T$ for the period-4 limit cycle, and $\mathbf{b} = \mathbf{b}_2 = -(b)\mathbf{1}^T$ for the period-1 limit cycle. Thus, through the convention used in Sec. II C, we denote the reconstructed period-4 limit cycle by $\hat{\mathcal{A}}_1$ and its related GAs by $\hat{\mathcal{A}}_1^*$. Similarly, we denote the reconstructed period-1 limit cycle by $\hat{\mathcal{A}}_2$ and its related GAs by $\hat{\mathcal{A}}_2^*$.

In Fig. 3 we show that for decreasing a , the period-1 limit cycle undergoes a sequence of period-doubling bifurcations. Therefore, if the parameter-aware closed-loop RC successfully interpolates the dynamics of Eq. (18) between the period-1 and period-4 limit cycles, then the RC should generate a period-2 limit cycle at some intermediate parameter setting between b and $-b$. In addition, the RC should generate a continuous transition between this period-2 limit cycle and the reconstructed period-1 and period-4 limit cycles, where each transition should occur due to a period-doubling bifurcation. Furthermore, the RC should continue to undergo successive period-doubling bifurcations beyond the period-4 limit cycle to produce chaotic dynamics. However, as our results show, it is not so straightforward for the RC to fill the gaps in its knowledge of Eq. (18) by producing GAs that mimic the dynamics shown in Fig. 3.

In our experiments, we investigate how different choices of the parameter b affects the ability of the parameter-aware closed-loop RC in Eq. (16) to reconstruct the period-1 and

period-4 limit cycles at the respective \mathbf{b}_1 and \mathbf{b}_2 settings of \mathbf{b} . We find empirically that by setting $\rho = 1.2$, $\sigma = 1.6$, $\beta = 0.01$, and $\gamma = 10$, we obtain reasonable reconstruction of the period-1 and period-4 limit-cycles for $b = \{\pm 0.2, \pm 0.3, \pm 0.4\}$. We find that reconstruction worsens for larger and smaller b , particularly for the period-4 limit cycle. After training, we track the changes in the reconstructed attractors in b -space for $b \in [0.42, -0.42]$. To do this we increase/decrease the value of b used during training by a small amount and initialise Eq. (16) at this new b value with a point on the attractor from the previous b value and continue this process until we can no longer track the subsequent GAs. We map the points on the bifurcation diagram in Fig. 3 to the set of b values specified above for the case of $b = \pm 0.2$. Thus, for $b = \pm 0.3$ and $b = \pm 0.4$, the resulting bifurcation diagrams from the RC correspond to rescalings of the original bifurcation diagram shown in Fig. 3.

Another reason why we chose to train the RC using limit cycles from Eq. (18) is because there is another limit cycle that undergoes period-doubling route to chaos for the same values of a , with both attractors merging to form one larger attractor when $a \approx 12$ (evidenced by the sudden growth in the range of the x_2 local minima in Fig. 3). This other limit cycle is a symmetric copy of the limit cycle we study, it exists due to the following symmetry relation of Eq. (18), $(x_1, x_2, x_3) \rightarrow (-x_1, -x_2, x_3)$. In some of the parameter settings we explore, we find that the parameter-aware closed-loop RC generates an attractor with similar properties to this other limit cycle.

3. Task (iii) description

In contrast to task (ii), the purpose of task (iii) is to investigate the dynamics of confabulation that can arise after the parameter-aware RC has been trained to reconstruct two different *unrelated attractors* at two different parameter settings. By ‘unrelated’ we mean that there exists no pre-defined sequence of transitions or continuous deformation from one attractor to the other. For task (iii) we focus on exploring the dynamics of confabulation when training the parameter-aware RC in Eq. (13) to reconstruct the Lorenz attractor, \mathcal{L} , and the Halvorsen attractor, \mathcal{H} , at two different parameter settings according to the steps outlined in Sec. II C.

For training data, we obtain a trajectory on \mathcal{L} according to the steps specified in Sec. II E 1, we obtain a trajectory on \mathcal{H} with the same procedure by generating solutions of the following system,

$$\begin{aligned}\dot{x}_1 &= -1.3x_1 - 4(x_2 + x_3) - x_2^2, \\ \dot{x}_2 &= -1.3x_2 - 4(x_3 + x_1) - x_3^2, \\ \dot{x}_3 &= -1.3x_3 - 4(x_1 + x_2) - x_1^2,\end{aligned}\quad (19)$$

until $t = t_{predict}$ using the 4th order Runge-Kutta method with time step τ . For this task, we set $t_{listen} = 100$, $t_{train} = t_{listen} + 200$, and $t_{predict} = t_{train} + 200$.

When training the parameter-aware RC setup in Eq. (13) to reconstruct \mathcal{L} , we set $\mathbf{b} = \mathbf{b}_1 = (b)\mathbf{1}^T$. For \mathcal{H} we set $\mathbf{b} = \mathbf{b}_2 = -(b)\mathbf{1}^T$. We denote the reconstructed \mathcal{L} at parameter setting b_1 by $\hat{\mathcal{L}}$, and its corresponding GAs by $\hat{\mathcal{L}}^*$. Similarly,

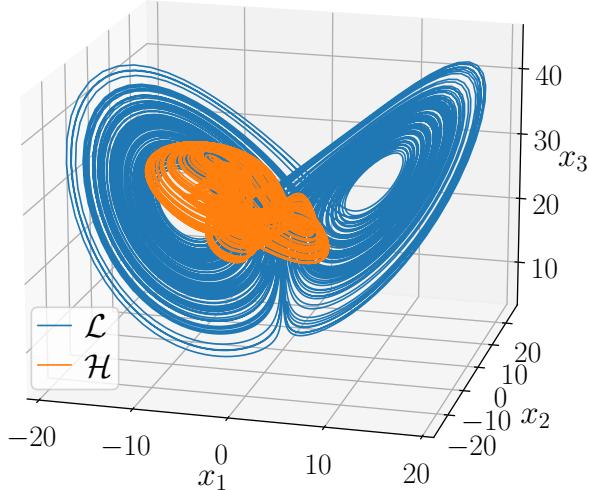


FIG. 4: Training data for task (iii).

we denote the reconstruction of \mathcal{H} at parameter setting b_2 by $\hat{\mathcal{H}}$, and its corresponding GAs by $\hat{\mathcal{H}}^*$. Similar to task (ii), we investigate how different choices of the parameter b impacts the ability of the parameter-aware closed-loop RC to reconstruct these attractors for $b = b_1$ and b_2 .

In our investigations we concentrate on a set of RC training parameters for which the parameter-aware closed-loop RC reconstructs \mathcal{L} and \mathcal{H} reasonably well for $b = b_1$ and b_2 for different choices of the constant b . We find that by setting $\rho = 1.2$, $\sigma = 0.2$, $\beta = 0.1$, and $\gamma = 10$, we obtain reasonable reconstruction of \mathcal{L} and \mathcal{H} for $b = \{\pm 0.2, \pm 0.3, \pm 0.4\}$. In order to find such a set of training parameters, we had to manually move the training data corresponding to \mathcal{H} to overlap with \mathcal{L} , the location of the attractors in the (x_1, x_2, x_3) -plane is shown in Fig. 4. Similar to task (ii), we also find that reconstruction of both attractors worsens for larger and smaller b .

After training, we track the changes in the reconstructed attractors in b -space for $b \in [0.42, -0.42]$ using the same procedure mentioned in Sec. II E 2. From this we uncover the different ways in which the RC interpolates and extrapolates beyond its training dataset. We display these results in diagrams that show the changes in the local minima of the x_1 variable in \mathbb{P} for changes in b . We choose to illustrate these results in terms of the x_1 variable for clarity of presentation as the dynamics of both reconstructed attractors and their families of GAs are confined to the most similar range of values. From this we identify the following three different ways in which, depending on the choice of b , the parameter-aware closed-loop RC fills in gaps in its memory by constructing:

1. An UA that coexists with $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$ at different values of b .
2. A region of bistability between $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$ for a small range of b .
3. A continuous transition between $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$.

F. Summary of related work

As mentioned during Sec. II A, in our previous investigations of multifunctionality in a RC, we illustrated that in many cases changing the spectral radius, ρ , of the internal connections strongly influences the prevalence of UAs^{20,28–30}. We show that this can also depend on the random realisation of \mathbf{M} from the results of task (i) in this paper. Interestingly, in studies of multistability in input driven RCs^{35,36}, UAs also appeared, with the authors referring to these as ‘spurious attractors’. Upon further investigation, we found that the same term has been used by researchers studying behaviours of Hopfield networks that were not learned by the network, like in Robins and McCallum³⁷, however these studied have mainly focused on trained and untrained attractors in the form of fixed points.

While the presence of UAs in RC systems are being increasingly studied and acknowledged^{22,38–41}, and have recently been detected in a physical RC system⁴², it is quite likely that UAs have lurked in the background of many researchers dealings with RCs but have simply gone unnoticed. For instance, in one of the most influential papers on reservoir computing in recent times, Lu *et al.*²³, the authors present an interesting result in their Fig. 5 where they show that for a relatively small change in the input scaling parameter, σ , there is an abrupt transition in the RC’s dynamics from a period-2 limit cycle to the chaotic Lorenz attractor. Based on the results presented in the present paper, it is not too far-fetched to speculate that the limit cycle is an UA that coexists with the reconstructed Lorenz attractor. Other examples of UAs potentially hiding in plain sight can be seen in de Jong *et al.*⁴³.

The connection between tasks (i)–(iii) and the process of storing and recalling a memory has been established by Lu and Bassett²¹ and Kong *et al.*²² who relate different processes of how the brain stores multiple memories simultaneously to results in reservoir computing. More specifically, Kong *et al.* relate ‘context-addressable memory’ to how a multi-functional RC stores different attractors in different regions of state space, and relate ‘location-addressable memory’ to how a parameter-aware RC stores different attractors at different parameter settings. Another motivation behind tasks (ii) and (iii) is to build on the above by relating how a parameter-aware RC generalises beyond its training data by filling in gaps between these attractors to how the brain generalises between memories and also to studies of generalisation in the wider machine learning research community. These tasks are analogous to studies of how generative AI models interpolate in embedding or latent spaces (semantic spaces learned during training where similar concepts cluster together). The results we present may provide some further insight towards the dynamics exhibited by these models before the ‘gropking transition’ occurs, when a model’s ability to generalise beyond its training data changes suddenly from a very poor to a perfect generalisation⁴⁴. In the context of our RC set-up, poor generalisations correspond to UAs and GAs that do not resemble the dynamics they are supposed to. Task (ii) enables us to identify factors that influence the generative properties of the parameter-aware RC by testing whether the transition from one attractor to another is representative of how transi-

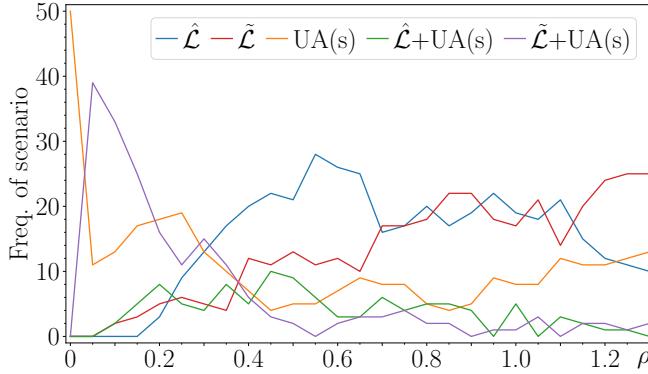


FIG. 5: Frequency of each scenario specified in Table I for a given ρ when training Eq. (1) to reconstruct \mathcal{L} using 50 different random realisations of \mathbf{M} .

tions between these attractors naturally occur, which shares many similarities with the work presented in Struski *et al.*⁴⁵. On the other hand, task (iii) provides us with a means to study these generative properties in a situation when there is no reference for how these transitions should occur. Moreover, tasks (ii) and (iii) represent studies of how different memories are linked together in a neural network via the dynamics of GAs and UAs, which in some sense relates to the ability of the hippocampus to form relationships between different memories in decision making⁴⁶. It has also been reported that related and unrelated events can become linked in memory when they happen close together⁴⁷. In the case of tasks (ii) and (iii), two events occurring within a brief window of time is similar to storing the attractors close together in b -space, and events becoming linked in memory is similar to there being a continuous transition from one attractor to the other.

III. TASK (I) RESULTS

A. Frequency of scenarios in Table I across 50 different M

Fig. 5 illustrates the frequency of each of the scenarios specified in Table I for a given spectral radius, ρ , across 50 different random realisations of the internal connectivity matrix, \mathbf{M} , whose elements have been scaled such that each \mathbf{M} has the same ρ .

Fig. 5 shows that for $\rho = 0$, the only attractor present in each \mathbb{P} is an UA. By increasing ρ to 0.05, there is a dramatic change in \mathbb{P} as the case of a poor reconstruction of \mathcal{L} coexisting with UAs becomes the dominant scenario. For $\rho = 0.1$ we see the first occurrences of RCs that are able to reconstruct \mathcal{L} , albeit poorly, without any UAs in \mathbb{P} . This trend continues for increasing ρ and when $\rho = 0.2$ we see the first instances of RCs that are able to provide a good reconstruction of \mathcal{L} without any UAs present in \mathbb{P} . By increasing ρ further we see this becomes the dominant scenario for $0.3 \lesssim \rho \lesssim 0.7$. Within this range of ρ values, UAs are found for a decreasing number of \mathbf{M} . For $0.7 \lesssim \rho \lesssim 1.1$, Fig. 5 shows that there is a relatively equal likelihood that the RC will provide either a

good or bad reconstruction of \mathcal{L} , and that it is much less likely for these reconstructions to coexist with UAs. However, there is a steady increase in the number of RCs that fail to provide any reconstruction of \mathcal{L} within this range of ρ values, thus, resulting in an increase in the UA only scenario. These trends continue for $\rho > 1.1$ with poor reconstructions of \mathcal{L} becoming the dominant scenario.

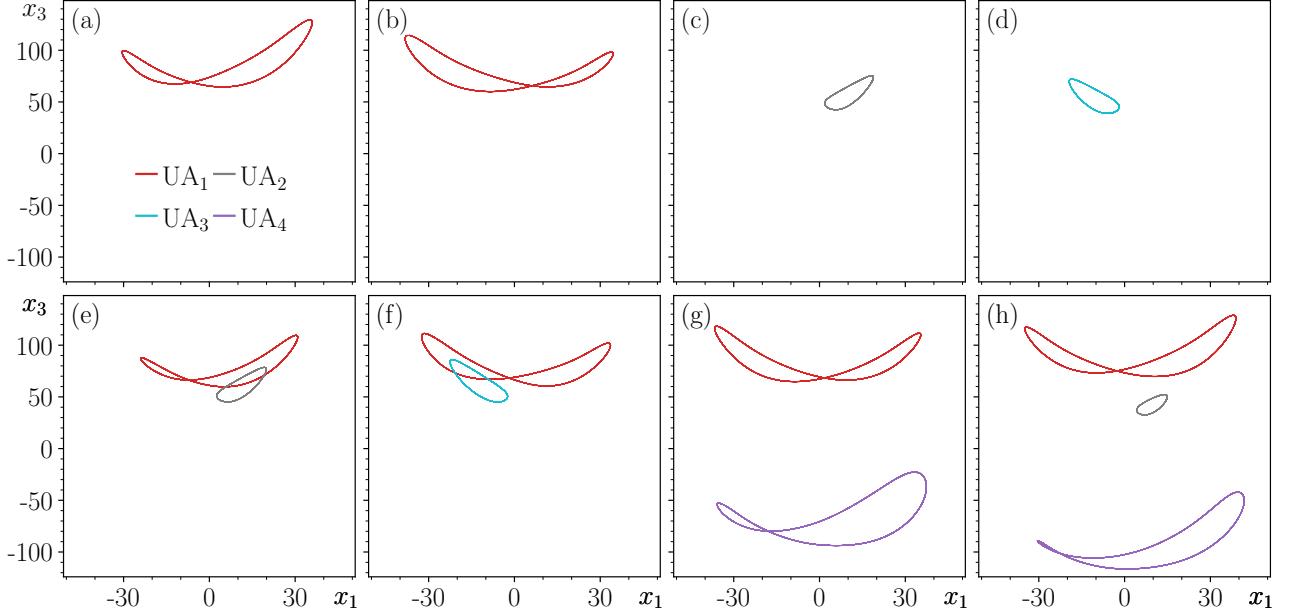
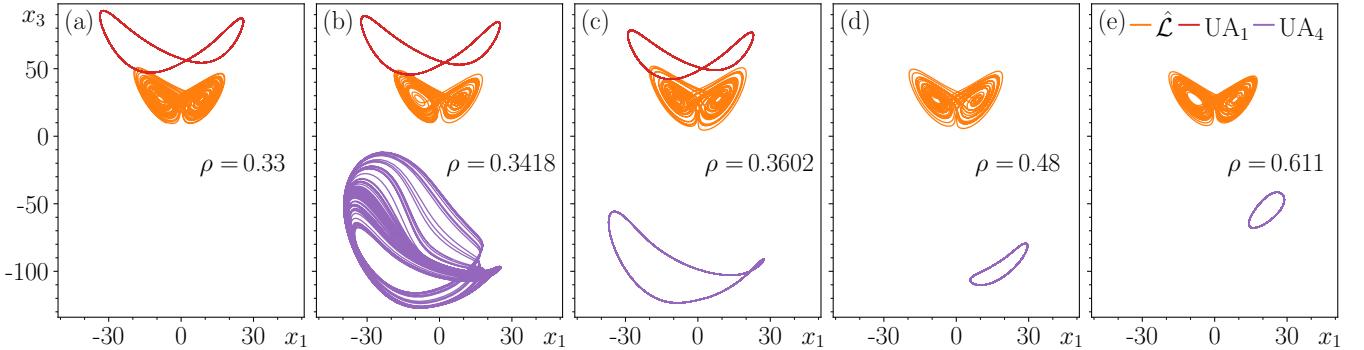
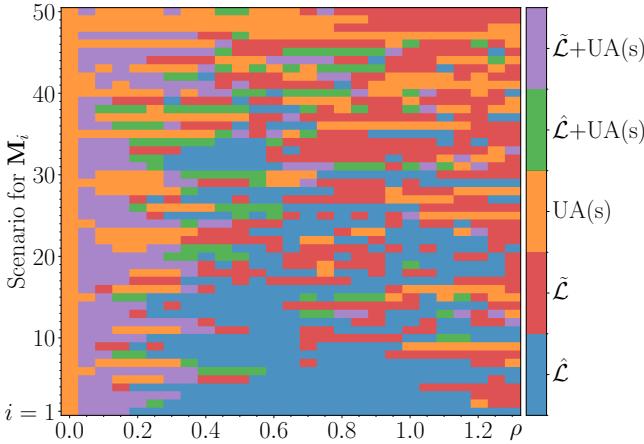
When generating the results shown in Fig. 5 we encounter a variety of UAs. As an example, in Fig. 6 we illustrate the UAs that most frequently appear when setting $\rho = 0.15$. We label each of these UAs according to the region of \mathbb{P} that they occupy. We omit any reconstructions of \mathcal{L} from Fig. 6 in order to focus solely on the UAs that appear. Panels (a) and (b) illustrate the different behaviours of the most commonly found UA, denoted by UA_1 . The difference between these illustrations is whether the maximum x_3 value occurs for either positive or negative x_1 . The next most frequently found UAs are shown in panels (c) and (d), denoted by UA_2 and UA_3 . We also find cases where UA_1 coexists with either UA_2 or UA_3 , like in panels (e) and (f). Panel (g) shows that another UA, denoted by UA_4 , can also coexist with UA_1 . Less commonly, we find instances where three UAs coexist, like in panel (h).

Interestingly, for one of the random \mathbf{M} realisations in our ensemble, we show in Fig. 7 that while UA_4 is a limit cycle for large ρ , panels (e), (d), (c), and (b) show that by decreasing ρ this limit cycle becomes chaotic before becoming unstable in panel (a). On the other hand, for increasing ρ , panels (a)–(c) show that UA_1 moves closer and closer to $\hat{\mathcal{L}}$ until it becomes unstable and no longer directly observable. From this perspective, this behaviour indicates that for increasing ρ , UA_1 either collides with a nearby saddle or with the basin boundary of $\hat{\mathcal{L}}$.

In Fig. 8 we re-present the results of Fig. 5 from the point of view of each \mathbf{M} in order to illustrate the changes that occur in each of the corresponding \mathbb{P} when changing ρ . We label each \mathbf{M} by \mathbf{M}_i , from $i = 1$ to 50, in order of the number of ρ values that the corresponding closed-loop RC in Eq. (10) provides a good reconstruction of \mathcal{L} , with $i = 1$ being the most amount of ρ values. Fig. 8 provides us with additional information, such as: some random realisations of \mathbf{M} quickly overcome the influence of UAs, like \mathbf{M}_1 and \mathbf{M}_4 , whereas others require relatively large ρ , like \mathbf{M}_{43} needing $\rho > 0.9$.

B. UAs filling in the gaps when reconstruction of \mathcal{L} fails

Fig. 8 shows that when $\rho = 0.05$ the majority of the different \mathbf{M} matrices are able to provide a poor reconstruction of \mathcal{L} , denoted by $\tilde{\mathcal{L}}$, which coexists with at least one UA. In this subsection, we illustrate an interesting route in which the RC reconstructs \mathcal{L} that involves period-doubling bifurcations. In Appendix C we illustrate the most common way that the RC reconstructs \mathcal{L} when increasing ρ from 0. In Appendix D we provide additional examples of interesting ways in which the RC reconstructs \mathcal{L} .

FIG. 6: Examples of UAs that frequently appear for different random realisations of \mathbf{M} when $\rho = 0.15$.FIG. 7: Example where UA_1 becomes unstable for increasing ρ and UA_4 becomes chaotic and then unstable for decreasing ρ .FIG. 8: Resulting scenario for each of the 50 different \mathbf{M} used when training Eq. (1) to reconstruct \mathcal{L} for a given ρ .

1. Reconstruction of \mathcal{L} via period-doubling route to chaos

In Fig. 9, we show that by increasing the parameter ρ of \mathbf{M}_{28} from 0, the first time the RC reconstructs a Lorenz-like attractor, it reconstructs \mathcal{L} as a limit cycle. In the inset panel we see that this limit cycle undergoes a period-doubling route to chaos before becoming unstable. To provide further insight, in Fig. 10 we illustrate the dynamics of this limit cycle as it undergoes these bifurcations at different ρ values. As indicated by the arrow in Fig. 10 (e), after this reconstruction becomes unstable the state of the RC then approaches UA_1 as this is the only stable state in \mathbb{P} for these ρ values. Prior to becoming unstable, this reconstruction grows in its resemblance to \mathcal{L} , which is why our algorithm classifies this as a poor reconstruction of \mathcal{L} rather than an UA. At larger ρ values, we see this poor reconstruction of \mathcal{L} reappear and disappear within different ranges of ρ values for $\rho < 0.1$ in a similar way to the events shown in Fig. 21 in Appendix C. Within this range of ρ

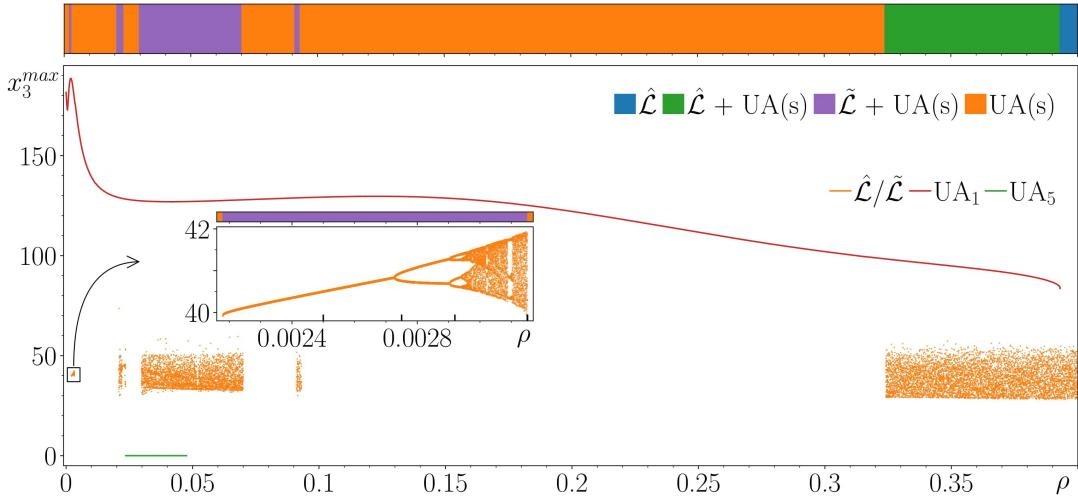


FIG. 9: Period-doubling route for reconstruction of \mathcal{L} for \mathbf{M}_{28} . Plotting changes in local maxima of x_3 in \mathbb{P} for $x_1 > 0$ for changes in ρ for all attractors found for $\rho \in [0, 0.4]$. Inset plot's inward ticks indicate ρ values used to generate Fig. 10. Coloured bars above bifurcation diagram indicate the algorithm's classification of the RC's output for a given ρ .

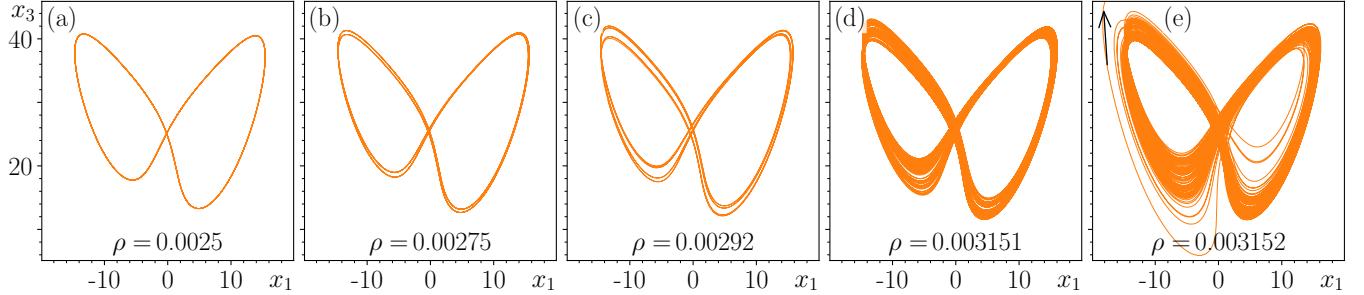


FIG. 10: Snapshots of period-doubling route for reconstruction of \mathcal{L} shown in Fig. 9 at specified values of ρ in panels (a)–(e).

values we also find another UA, a fixed point that we denote by UA_5 , which the state of the RC approaches when reconstruction fails for $0.025 \lesssim \rho \lesssim 0.03$. For $0.1 < \rho < 0.325$ we find that the reconstruction exists as a transient and, depending on the value of ρ , this transient activity can last for significantly long durations of time. In order to determine whether the RC provides a stable reconstruction of \mathcal{L} within this large range of ρ values, we had to increase $t_{predict}$ to 30,000. The RC achieves a good reconstruction of \mathcal{L} for $\rho = 0.325$ and maintains this for much larger ρ values.

Above all, Fig. 9 demonstrates the benefit of performing this type of analysis of tracking the changes in these attractors. It provides a strong visual aid to show that when the RC fails to provide any reconstruction of \mathcal{L} , there is already an UA waiting for the RC's state to approach. Furthermore, this is not the only time we see period-doubling bifurcations playing a role in how the RC reconstructs \mathcal{L} . In Appendix D we illustrate that period-doubling bifurcations of another Lorenz-like period-1 limit cycle and also a saddle-focus homoclinic orbit result in the reconstruction of \mathcal{L} . We would also like to mention that we find no sequence of events where $\hat{\mathcal{L}}$ comes into existence through the well-known sequence of events that

unfold when varying the constant term in the \dot{x}_2 equation in Eq. (17)⁴⁸.

IV. TASK (II) RESULTS

A. Reconstructed attractors for $b = \pm 0.4, \pm 0.3, \pm 0.2$

We first show in Fig. 11 that the closed-loop parameter-aware RC (Eq. (16)) achieves a reasonably good reconstruction of the period-1 and period-4 limit cycles for each choice of b used during the training. The parameter b is used to change the distance in parameter-space (b -space) between these attractors. The other training parameters are specified in Sec. II E 2.

It is clear from Fig. 11 that there is an improvement in the RC's ability to reconstruct both limit cycles at their respective b values when the trained limit cycles are further apart in b -space. Panels (a) and (c) show period-4 limit cycles, while panel (e) reveals a similar appearing limit cycle that is actually period-2. Nonetheless, Figs. 12 (a)–(b) shows there is a limit to this: if the limit cycles are too far apart, like for $b = \pm 0.5$,

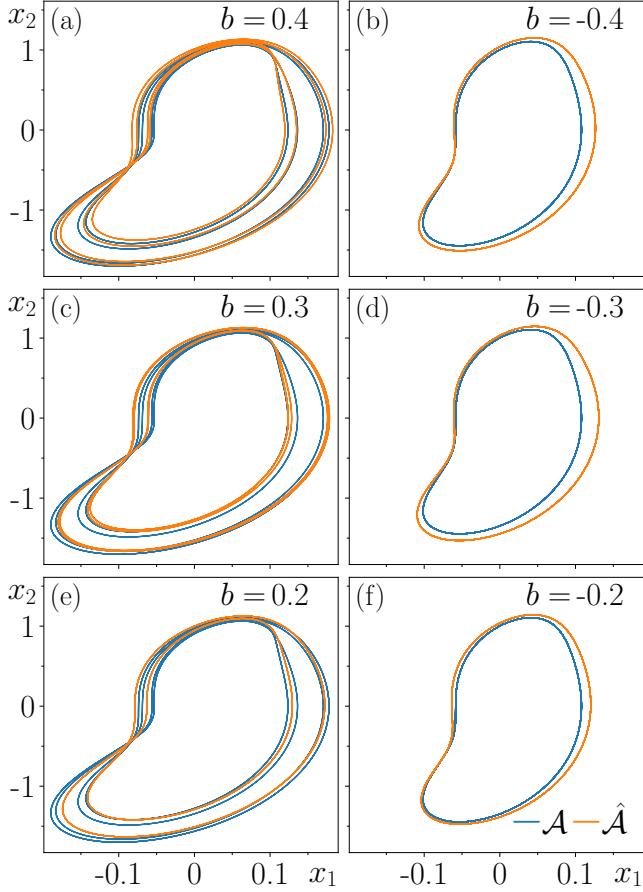


FIG. 11: Reconstructed attractors (in orange) when parameter-aware RC is trained on task (ii) for $b = \pm 0.4$ in (a)–(b), $b = \pm 0.3$ in (c)–(d), and $b = \pm 0.2$ in (e)–(f).

then there is a decline in the RC’s ability to reconstruct both limit cycles at their respective b values. The RC fails to reconstruct the period-4 limit cycle and instead produces a chaotic attractor. Further, panels (c)–(d) show what can happen when the limit cycles are too close together in parameter space. For $b = \pm 0.1$ neither limit cycles are reconstructed properly and the RC instead produces period-2 limit cycles at both values of b .

B. Analysis of GAs and UAs for $b = \pm 0.4, \pm 0.3, \pm 0.2$

In each panel of Fig. 13 we show how the closed-loop parameter-aware RC in Eq. (16) generates the dynamics between and beyond the reconstructed limit cycles shown in Fig. 11 by tracking the changes in the dynamics of all attractors present in the RC’s prediction state space, \mathbb{P} , for $b \in [0.42, -0.42]$. In Fig. 14 we show the attractors generated by the RC in each panel of Fig. 13 for $b = 0$. We discuss each of the corresponding panels of Figs. 13 and 14 for values of b under the following headings.

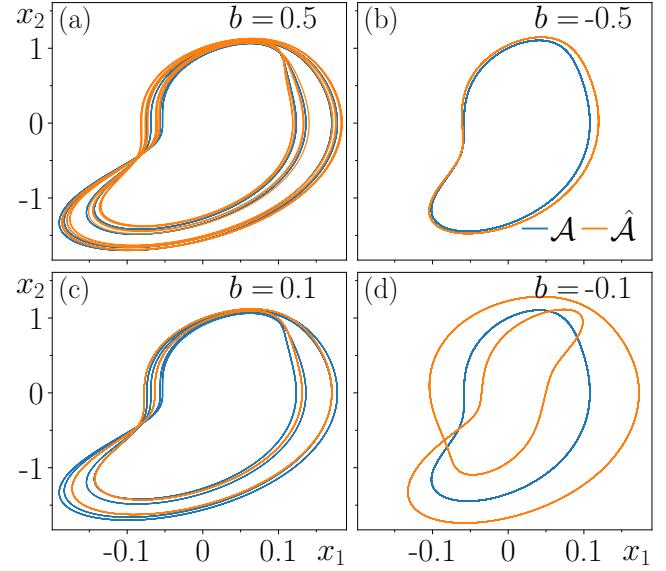


FIG. 12: Reconstructed attractors (in orange) when parameter-aware RC is trained to reconstruct both limit cycles from task (ii) for $b = \pm 0.5$ in (a)–(b) and $b = \pm 0.1$ in (c)–(d).

1. Case $b = \pm 0.4$

We first discuss the results for $b = \pm 0.4$ shown in Fig. 13 (a). The most prominent feature here is the continuous transition between the trained period-1 and period-4 limit cycles. This continuous transition is consistent with the period-doubling bifurcation that occurs in Eq. (18). However, the RC’s period-doubling bifurcation does not occur nearby the corresponding bifurcation in Eq. (18). Also, the generated period-2 limit cycle bears little resemblance to the limit cycle from Eq. (18), as evidenced by the difference in the local minima of the x_2 variable. Fig. 14 (a) further illustrates this difference between the RC’s generated dynamics for $b = 0$, generating a period-1 limit cycle instead of a period-2 limit cycle.

For $b > 0.4$, Fig. 13 (a) shows that the period-4 limit cycle undergoes a period-doubling route to chaos, and that the resultant chaotic attractor becomes unstable for $b \approx 0.415$. To fill in the gaps, Eq. (16) generates an UA, that we denote by UA_1 (which has nothing to do with the UA_1 studied in task (i), nor do any of the UAs mentioned in this section). From further inspection, UA_1 is a period-1 limit cycle and its dynamics are consistent with the dynamics of the limit cycle that coexists with the limit cycle from Eq. (18) that we use for training, which to some extent indicates that this RC learns properties of Eq. (18) without being explicitly trained to. This phenomenon is described in Röhm *et al.*⁴⁹.

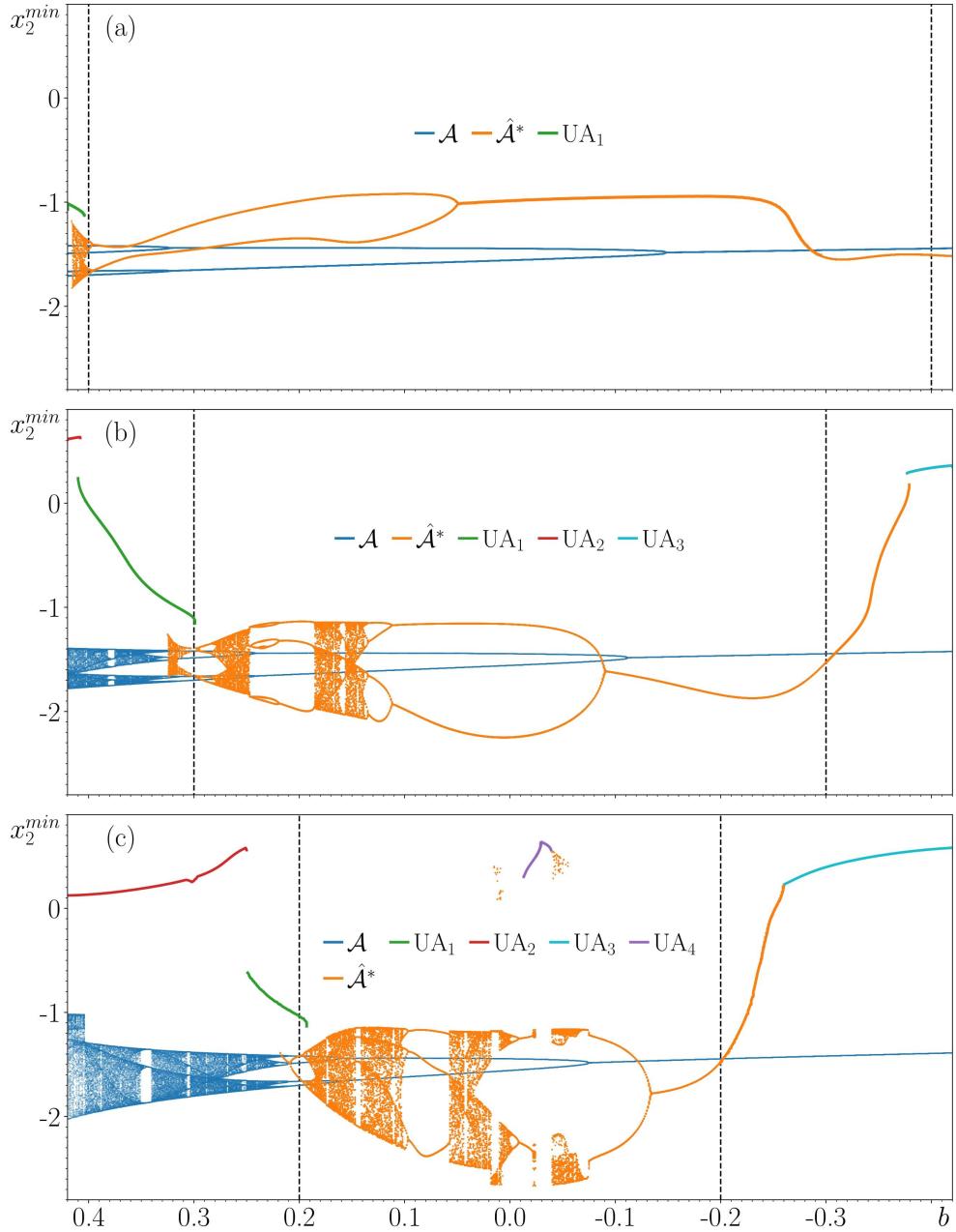


FIG. 13: Resultant bifurcation diagram from training parameter-aware RC to perform task (ii) with $b = \pm 0.4$ in (a), $b = \pm 0.3$ in (b), or $b = \pm 0.2$ in (c). Legend indicates the attractors that are tracked.

2. Case $b = \pm 0.3$

The most prominent feature in Fig. 13 (b) is that the continuous transition between the trained period-1 and period-4 limit cycles is not consistent with the period-doubling bifurcation that occurs in Eq. (18). Instead of undergoing a period-doubling bifurcation from the reconstructed period-1 and period-4 limit cycles to a GA in the form of a period-2 limit cycle, the RC generates infinite sequences of period-doubling and halving bifurcations to and from generated chaotic attractors. Fig. 14 (b) illustrates the period-2 limit

cycle generated by the RC for $b = 0$, which occupies a much larger region of the state space than the corresponding dynamics from Eq. (18).

Similar to Fig. 13 (a), Fig. 13 (b) shows that for $b > 0.3$, the reconstructed period-4 limit cycle undergoes a period-doubling route to chaos, and the resultant chaotic attractor becomes unstable for $b \approx 0.325$. UA₁ fills in the resulting gap. However, the changes in the dynamics of UA₁ can no longer be tracked for $b \gtrsim 0.41$. The RC fills in the gap with another UA, which for this range of b values is a fixed point that we denote by UA₂. Note, there is a small range of b values where

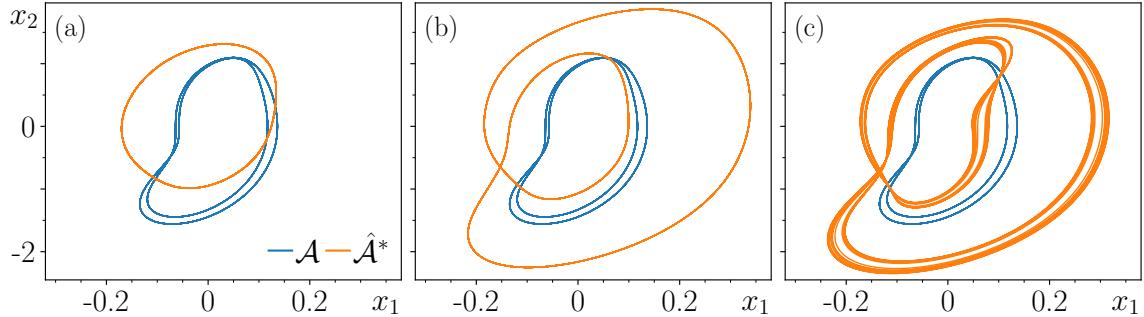


FIG. 14: Examples of GAs (in orange) for $b = 0$ when trained for $b = \pm 0.4$ in (a), $b = \pm 0.3$ in (b), and $b = \pm 0.2$ in (c).

UA_1 and UA_2 coexist, specifically, $b \in [0.408, 0.410]$.

Fig. 13 (b) shows that GAs of the period-1 limit cycle can no longer be tracked for $b < -0.379$. The RC fills in the resulting gap with UA_3 , which is a fixed point for the b values shown here and coexists with the GAs of the reconstructed period-1 limit cycle for $b \in [-0.377, -0.379]$.

3. Case $b = \pm 0.2$

Fig. 13 (c) shows there is no continuous transition between the reconstructed period-1 and period-4 limit cycles when training for $b = \pm 0.2$. The attractors that are generated from the reconstructed period-1 and period-4 limit cycles become chaotic and subsequently unstable for decreasing/increasing b . No GAs exist for $b \in (-0.024, -0.04)$ and UA_4 , which is a fixed point, fills in this gap. UA_4 coexists with the generated dynamics from the reconstructed period-4 limit cycle for $b \in [-0.013, -0.024]$, and coexists with the generated dynamics from the reconstructed period-1 limit cycle for a much smaller range of b values. Fig. 14 (c) illustrates the dynamics of the chaotic attractor shown in Fig. 13 (c) for $b = 0$.

Similar to Fig. 13 (b), Fig. 13 (c) shows that UA_1 and UA_2 fill in the gaps from $b > 0.2$, and likewise UA_3 fills in the gaps for $b < -0.2$. The small dip in the branch of points describing the changes in the dynamics of UA_2 corresponds to a bifurcation from a fixed point to a limit cycle for decreasing b .

C. Does adding more data help?

A natural question arises from the results presented so far on task (ii), what would happen if the parameter-aware RC was also trained to reconstruct the period-2 limit cycle from Eq. (18)? Does this improve the RCs ability to generate transitions, i.e. period-doubling bifurcations, between the period-1, 2, and 4 limit cycles? Would the resultant dynamics be more consistent with the dynamics of Eq. (18)? To shed light on these questions, in this subsection we illustrate our results from training the parameter-aware RC to reconstruct the period-1 and period-4 limit cycles for $b = (\pm 0.2)\mathbf{1}^T$ as before and (i) to also reconstruct the intermediary period-2 limit cycle for $b = (0)\mathbf{1}^T$, and (ii) reconstruct the period-2 limit cycle

for $b = (0)\mathbf{1}^T$ as well as reconstruct the corresponding intermediary period-1 and period-2 limit cycles for $b = (\pm 0.1)\mathbf{1}^T$. We do so using the extended training method mentioned in the second last paragraph of Sec. II C.

In both cases we found that the closed-loop parameter-aware RC was unable to successfully reconstruct the limit cycles at their respective b values with the set of RC training parameters specified in Sec. II E 2 that were used so far in task (ii). Thus, to follow the same rationale as before, we find a new set of RC training parameters that provides reasonably good reconstruction of all limit cycles for the values of b that were used during the training. This allows us to concentrate our discussion on how well the RC fills in gaps in its memory for memories (i.e. attractors) that are stored correctly. For the first case of three limit cycles we achieved this by setting $\rho = 1.4$, $\sigma = 1.3$, without changing the other parameters. For the second case of five limit cycles we achieved this by setting $\rho = 1.3$, $\sigma = 1.0$, $\beta = 0.1$, and the other parameters unchanged. We illustrate in Figs. 15 and 17 that for these new choices of training parameters, the closed-loop parameter-aware RC achieves a good reconstruction of each limit cycle. This is further exemplified in Figs. 16 and 18 by the overlap between the blue and orange data points at the values of b used during the training, as indicated by the vertical dashed black lines.

In Fig. 16 we examine how the closed-loop parameter-aware RC in Eq. (16) generates the dynamics between and beyond the three reconstructed limit cycles shown in Fig. 15. More specifically, Fig. 16 shows the result of tracking the changes in the dynamics of all attractors generated by Eq. (16) for $b \in [0.42, -0.42]$. While Fig. 15 shows that the RC achieves a good reconstruction of each limit cycle at their respective b values, the same cannot be said for how the RC generates the dynamics between these limit cycles. While the transition from the period-1 to period-2 limit cycle is relatively consistent with the dynamics of Eq. (18), there is no continuous transition between the period-2 and period-4 limit cycles. The period-2 and period-4 limit cycles both become chaotic when tracking the changes in the dynamics of these attractors for increasing b from 0 and decreasing b from 0.2, respectively. The chaotic attractor that is born from the reconstructed attractors reaches a point where it can no longer be tracked and thus becomes unstable for $b \in (0.12, 0.06)$. Like

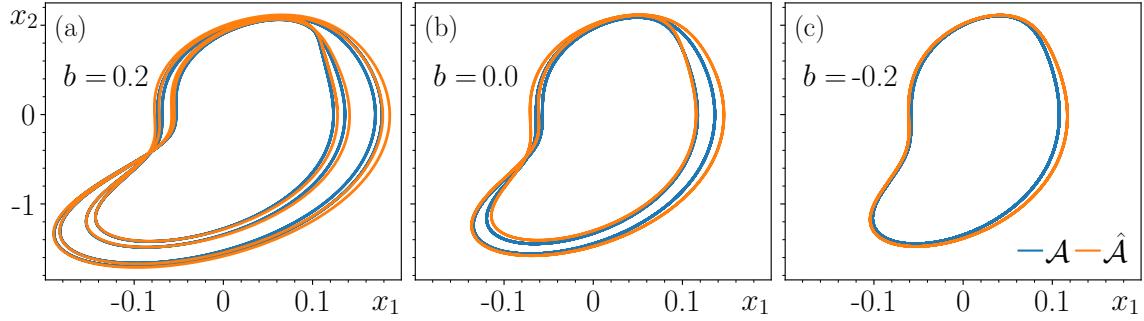


FIG. 15: Examples of reconstructed attractors (in orange) when trained for $b = 0.2, 0.0, -0.2$.

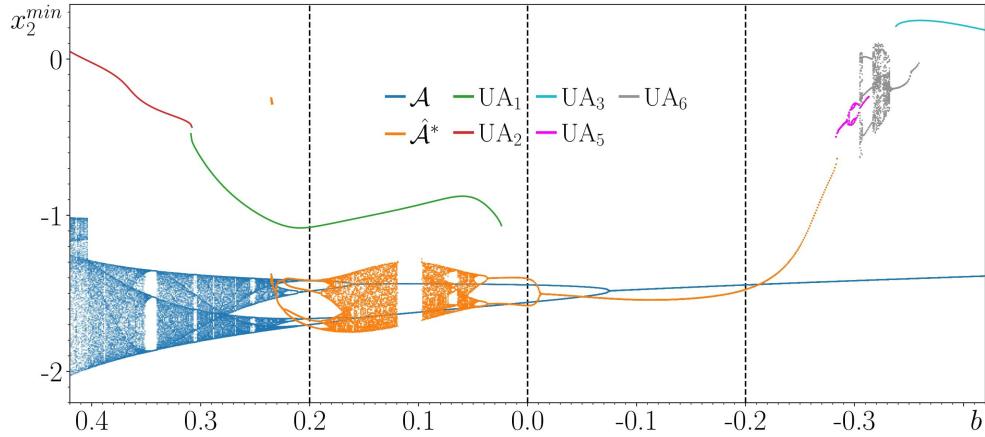


FIG. 16: Resultant bifurcation diagram from training parameter-aware RC to reconstruct period-1 limit cycle at $b = -0.2$, period-2 limit cycle at $b = 0.0$, and period-4 limit cycle at $b = 0.2$. Legend indicates the attractors that are tracked.

before, UA_1 fills this gap, while UA_1 and UA_2 fill in the gap for $b > 0.235$. A different scenario from Fig. 13 (c) emerges in Fig. 16 for $b < -0.2$. In addition to UA_3 , there are two further UAs required to fill in the gaps: the limit cycle UA_5 and the mostly chaotic dynamics of UA_6 .

In Fig. 18 we apply the same procedure to examine how Eq. (16) generates the dynamics between and beyond the five reconstructed limit cycles shown in Fig. 17. Similarly, while Fig. 17 shows that the RC achieves a good reconstruction of each limit cycle at their respective b values, there is a slightly better agreement between the attractors that the RC generates and the dynamics of Eq. (18) than in Fig. 15, aside for the branch of GAs between $b = 0.2$ and 0.1 which shows the RC's dynamics become chaotic nearby $b = 0.18$. Furthermore, there is still no continuous transition between all reconstructed attractors. The periodic and chaotic dynamics of UA_7 fills the gap in this case, coexisting with the corresponding GAs for a small range of b values. Just as in many previous cases, UA_1 and UA_3 fill in the gaps for $b > 0.25$ and $b < -0.27$.

In summary, the main message from Figs. 16 and 18 is that even if you try to account for gaps in the system's memory then complete confabulations may still appear. However, by taking greater measures to account for these gaps, complete confabulations may become less prevalent and there may be a

greater resemblance between the generated dynamics and the dynamics of the system where the training data comes from.

In the next section, we investigate how the RC fills in gaps in its memory when there exists no natural or simple way to account for gaps in its memory.

V. TASK (III) RESULTS

A. Reconstructed attractors for $b = \pm 0.4, \pm 0.3, \pm 0.2, \pm 0.1$

We first show in Fig. 19 that the closed-loop parameter-aware RC in Eq. (16) achieves a reasonably good reconstruction of the Lorenz (\mathcal{L}) and Halvorsen (\mathcal{H}) chaotic attractors for $b = \pm 0.2, \pm 0.3$, and ± 0.4 , and relatively poor reconstruction for $b = \pm 0.1$, in particular for \mathcal{L} . Similar to Fig. 11, Fig. 19 shows that as the attractors are moved further apart in b -space then reconstruction of both attractors improves. However, there is also a limit to this as well, comparing Fig. 19 (b) to (c) we see that reconstruction of \mathcal{H} begins to worsen as b is increased from ± 0.3 to ± 0.4 .

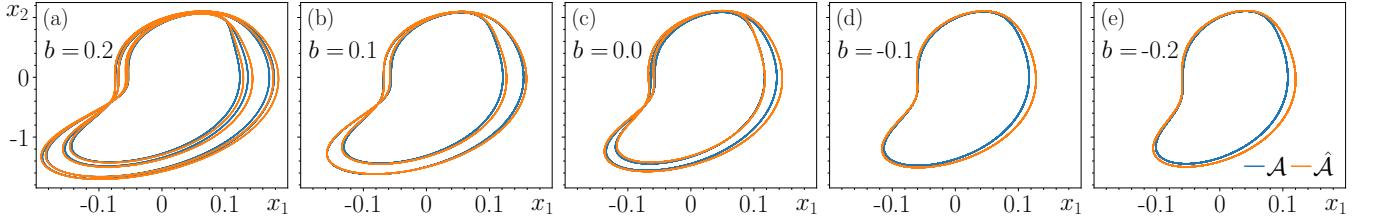


FIG. 17: Examples of reconstructed attractors (in orange) when trained for $b = 0.2, 0.1, 0.0, -0.1, -0.2$.

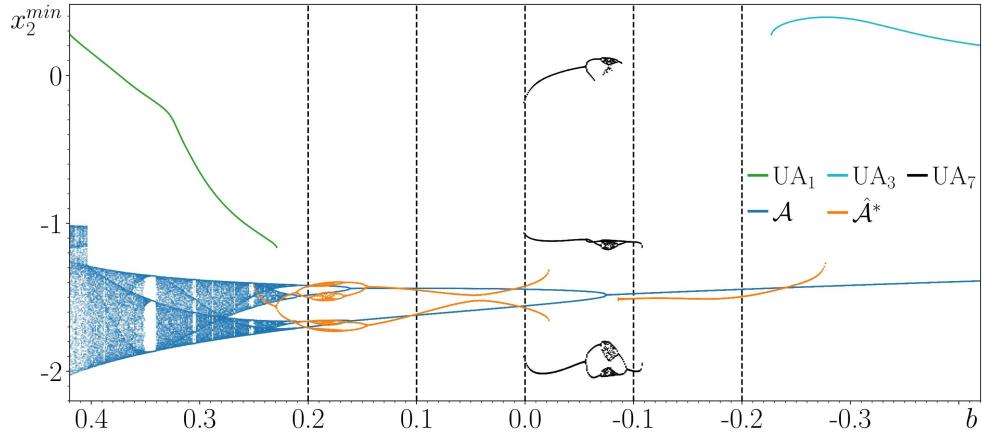


FIG. 18: Resultant bifurcation diagram from training parameter-aware RC to reconstruct period-1 limit cycle at $b = -0.2$ and -0.1 , period-2 limit cycle at $b = 0.0$ and 0.1 , and period-4 limit cycle at $b = 0.2$. Legend indicates the attractors that are tracked.

B. Analysis of GAs and UAs for $b = \pm 0.4, \pm 0.3, \pm 0.2, \pm 0.1$

In each panel of Fig. 20 we show the result of tracking the changes in the dynamics of all attractors generated by Eq. (16) for $b \in [0.42, -0.42]$. We do this in order to examine how the closed-loop parameter-aware RC in Eq. (16) generates the dynamics between and beyond the reconstructed chaotic attractors, \mathcal{L} and \mathcal{H} , shown in Fig. 19. We discuss each of the corresponding panels of Figs. 19 and 20 for values of b under the following headings.

1. Case $b = \pm 0.4$

We first discuss the results for $b = \pm 0.4$ shown in Fig. 20 (a). The most prominent feature here is that there is no continuous transition between the two attractors. To fill the gap between the attractors, the RC creates UA₁ (of no connection to the previously mentioned UA₁s), a fixed point which coexists for a wide range of b values with the attractors that are generated from the reconstruction of \mathcal{L} and \mathcal{H} , namely $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$. Despite Figs. 19 (a) and (b) illustrating that the RC achieves a reasonably good reconstruction of \mathcal{L} and \mathcal{H} , Fig. 20 (a) shows that both attractors alternate between chaotic and periodic dynamics at several b values when tracking the changes in the dynamics of $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$ for decreasing and increasing b . Both attractors can no longer be tracked for

$b \in [0.09, -0.01]$ and UA₁ fill this gap.

2. Case $b = \pm 0.3$ and $b = \pm 0.2$

In Figs. 20 (b) and (c) we see that while both \mathcal{L}^* and \mathcal{H}^* become unstable at certain b values, there is no need for the RC to create an UA as there is a small region of bistability between the GAs. Furthermore, despite that Figs. 19 (c)–(d) and (e)–(f) show that the RC achieves a reasonably good reconstruction of \mathcal{L} and \mathcal{H} at the respective b values, Figs. 20 (b) and (c) show that when tracking the changes in the dynamics of $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$ for decreasing and increasing b , the dynamics of both GAs also alternate between chaotic and periodic dynamics at several b values like in Fig. 20 (a).

In Fig. 20 (b), we see that when tracking the changes in the dynamics of both attractors from $b = \pm 0.3$, $\hat{\mathcal{L}}^*$ can no longer be tracked for $b < 0.2$ and $\hat{\mathcal{H}}^*$ can no longer be tracked for $b > 0.21$. Thus, the region of bistability between $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$ exists for $0.2 \leq b \leq 0.21$. When tracking $\hat{\mathcal{L}}^*$ for decreasing b from $b = 0.3$, we find that these GAs are deformations of the original reconstructed attractor $\hat{\mathcal{L}}$. On the other hand, when tracking $\hat{\mathcal{H}}^*$ for increasing b from $b = -0.3$ we find that the dynamics of the GAs share much less similarities to \mathcal{H} . When tracking $\hat{\mathcal{L}}^*$ for increasing b from $b = 0.3$ and $\hat{\mathcal{H}}^*$ for decreasing b from $b = -0.3$, both GAs become period-1 limit cycles.

In Fig. 20 (c), we see that when tracking the changes in

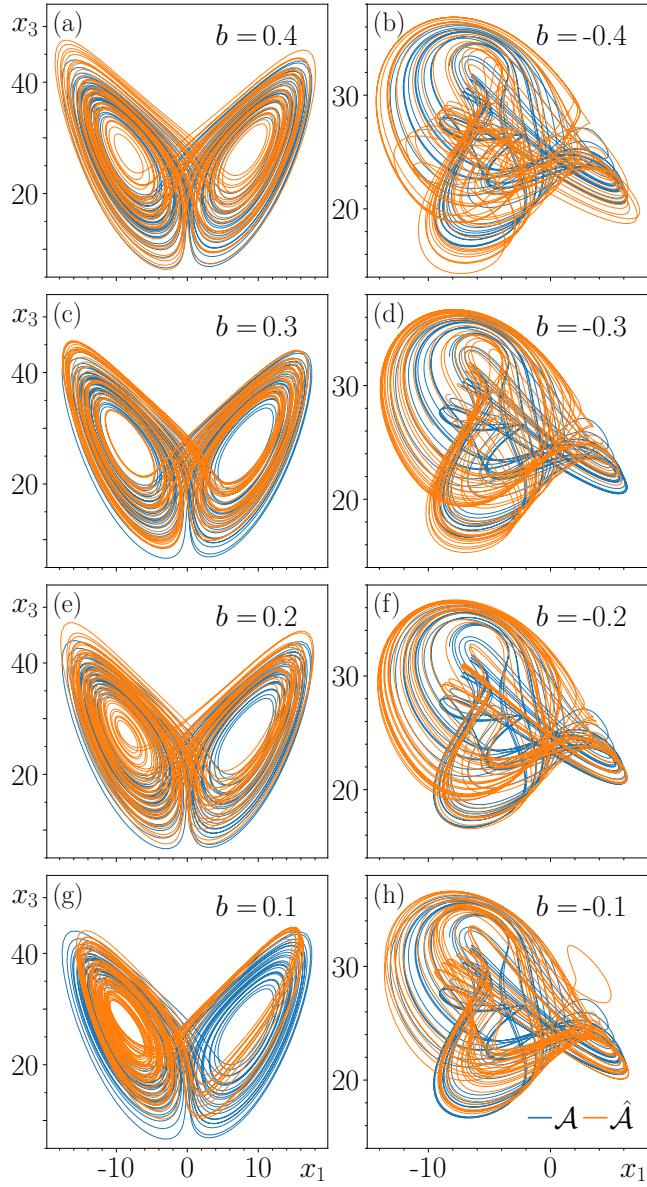


FIG. 19: Reconstructed attractors (in orange) when parameter-aware RC is trained on task (iii) for $b = \pm 0.4$ in (a)–(b), $b = \pm 0.3$ in (c)–(d), $b = \pm 0.2$ in (e)–(f), and $b = \pm 0.1$ in (g)–(h).

the dynamics of both attractors from $b = \pm 0.2$, $\hat{\mathcal{L}}^*$ can no longer be tracked from $b < -0.04$ and $\hat{\mathcal{H}}^*$ can no longer be tracked for $b > -0.038$. Thus, the region of bistability between $\hat{\mathcal{L}}^*$ and $\hat{\mathcal{H}}^*$ exists for $-0.04 \leq b \leq -0.038$. When tracking $\hat{\mathcal{L}}^*$ for decreasing b from $b = 0.2$ we find that the deformations of the original reconstructed attractor $\hat{\mathcal{L}}$ occur for a much shorter range of b values. Instead, $\hat{\mathcal{L}}^*$ is a limit cycle for $b \in (0.11, 0.02)$ and fixed point for $b \in (0.02, -0.038]$. When tracking $\hat{\mathcal{H}}^*$ for increasing b from $b = -0.2$, $\hat{\mathcal{H}}^*$ becomes a limit cycle for $b \in [-0.06, -0.038]$. When tracking $\hat{\mathcal{L}}^*$ for increasing b from $b = 0.2$, at $b = 0.35$ this GAs undergoes a continuous transition from a chaotic attractor to a fixed

point. When tracking the changes in $\hat{\mathcal{H}}^*$ for decreasing b from $b = -0.2$, the GAs alternate between regions of chaotic and periodic behaviour. Note that the different branches of x_1^{min} nearby $b = -0.315$ do not correspond to a region of bistability between two limit cycles, instead there is one limit cycle that has two local minima at different b values.

3. Case $b = \pm 0.1$

The most interesting result from task (iii) is illustrated in Fig. 20 (d) which shows that the closed-loop parameter-aware RC in Eq. (16) creates a continuous transition between attractors from two different systems. Thus, using the terminology developed in Sec. II D, since there is a continuous transition then the generated dynamics between both attractors aligns with our definition of $\hat{\mathcal{A}}_{i,j}^*$. However, we find it more convenient in the plot legend to refer to the information presented here as ‘ $\hat{\mathcal{L}}^*$ ’ and ‘ $\hat{\mathcal{H}}^*$ ’, since there is a continuous transition between all GAs in Fig. 20 (d).

When tracking the changes in $\hat{\mathcal{L}}^*$ for decreasing b from $b = 0.1$, we find that $\hat{\mathcal{L}}^*$ undergoes a series of bifurcations to become a period-1 limit cycle near $b = 0.06$ and undergoes a further bifurcation to a fixed point near $b = 0.01$. When tracking the changes in $\hat{\mathcal{H}}^*$ for increasing b from $b = -0.1$, $\hat{\mathcal{H}}^*$ undergoes what appears to be a more abrupt transition from chaotic behaviour to the same fixed point that is found when tracking $\hat{\mathcal{L}}^*$. We study these transitions in greater detail in Appendix E.

Similar to Fig. 20 (c), Fig. 20 (d) shows that when tracking $\hat{\mathcal{L}}^*$ for increasing b from $b = 0.1$, at $b = 0.22$ this GA undergoes a continuous transition from a chaotic attractor to a fixed point. Likewise, when tracking $\hat{\mathcal{H}}^*$ for decreasing b from $b = -0.1$, the GAs alternate between regions of chaotic and periodic dynamics. While this continuous transition emerges for decreasing b , Figs. 19 shows that the RC’s reconstruction of \mathcal{L} and \mathcal{H} worsens for decreasing b . From further experiments not shown in this paper, we find this trend continues for decreasing b .

VI. CONCLUSION

The primary motivation of this paper was to develop a foundational understanding of the reasons why and how ANNs confabulate and generate false information to fill in gaps in their memory. Our broader objective was to use these insights to inform the study of analogous processes in other learning systems, including the brain. As a first step, we study how confabulations arise in a relatively simple ANN design, a dynamical system in the form of a RC.

Previous work mentioned in Sec. II F has shown that when RCs are trained to reconstruct the dynamics of a given attractor, the RC can construct an attractor it was not trained to reconstruct, a so-called ‘untrained attractor’ (UA). Building on the work of Lu and Bassett²¹ and Kong *et al.*²², we interpret the phenomenon of UAs through a memory-theoretic lens, associating trained attractors with encoded memories and UAs

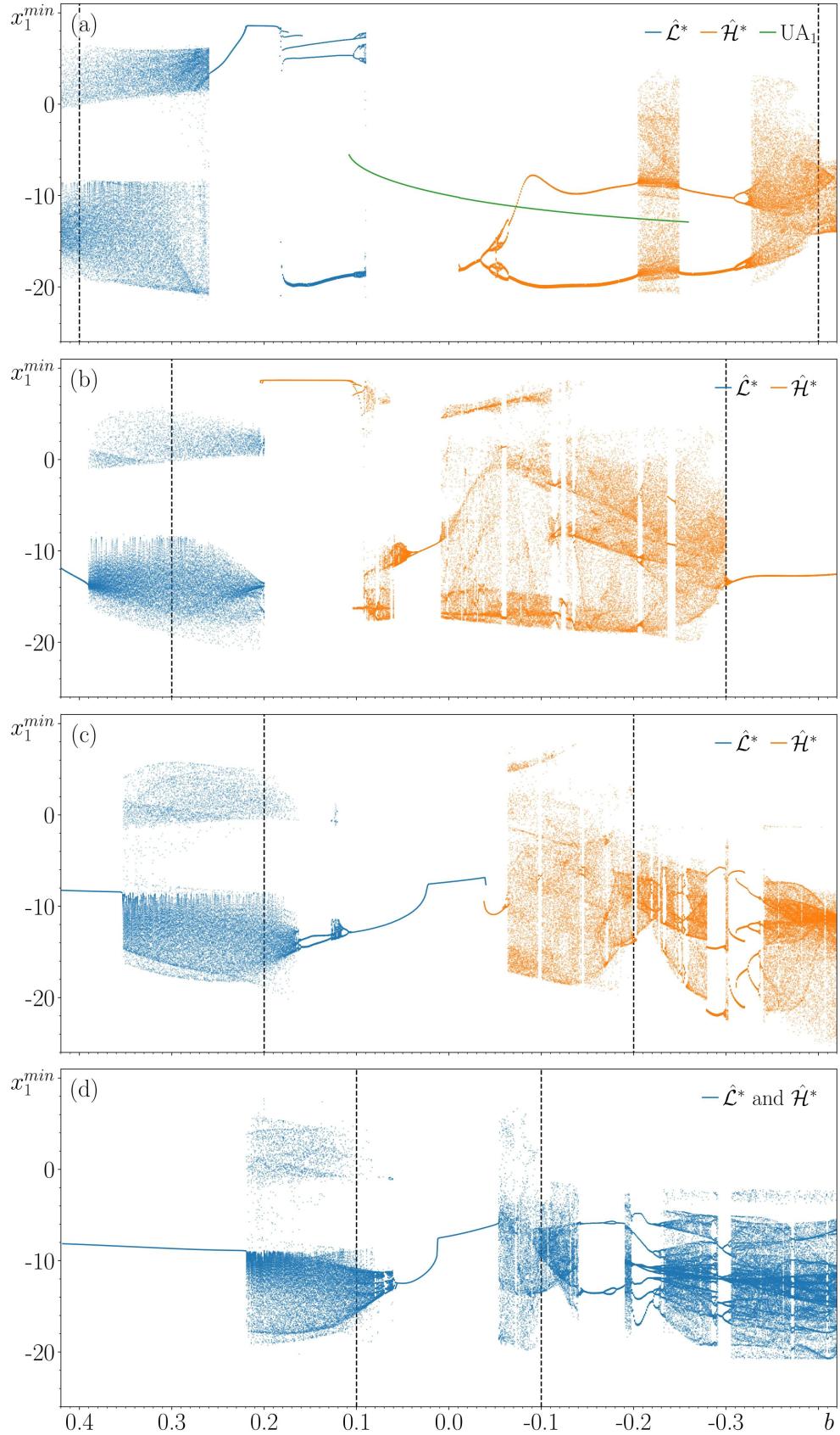


FIG. 20: Resultant bifurcation diagram from training parameter-aware RC to perform task (iii) with $b = \pm 0.4, \pm 0.3, \pm 0.2$, and ± 0.1 . Legend indicates the attractors that are tracked.

with *complete confabulations*.

We designed three tasks to explore the reasons why and how complete confabulations in the form of UAs emerge. In task (i) (Sec. III) we studied the UAs that appear when a RC is trained to store and recall an individual attractor/memory. In tasks (ii) and (iii), (Sects. IV and V), we studied the UAs that appear when a RC is trained to generalise between related and unrelated attractors/memories. The common result across all tasks is that the appearance of UAs consistently coincided with failure modes, either in recall or generalisation, appearing whenever the RC failed to encode or sensibly generalise beyond its training data. This aligns with the known association between the emergence of confabulations and a decline in the brain's ability to store and/or recall memories^{8,9}.

Several additional key findings emerge from our experiments. First, confabulations are not necessarily a system error but are an intrinsic feature of how learning systems behave. For instance, the results presented in Sec. III show that confabulations can appear as soon as the system is trained and disappear as the system improves its performance, in these cases the UAs become unstable and thus their presence is no longer directly visible. Second, our results challenge the assumption that confabulations can be mitigated solely through training the system with additional data. As shown in Sec. IV, increasing the amount of training data does not guarantee that the system provides a more sensible generalisation between learned states and UAs may still appear. Furthermore, while it has been shown that a reasonably good agreement can be achieved between the bifurcation diagrams of parameter-aware RCs and certain systems, like in Kim *et al.*²⁴, Kong *et al.*²⁵, Kong *et al.*⁵⁰, Köglmayr and Räth⁵¹ and even when using experimental data from systems like in Panahi *et al.*⁵², here we show that the amount of attractors used and the spacing between the parameter values where the attractors are stored (the choices of b) are two factors that influence the level of agreement. Third, our findings showcase the explanatory power of applying concepts from dynamical systems theory to better understand how machines learn by analysing the stability and bifurcation structure of trained and untrained states in ANNs. From this analysis we show that UAs may become stable or unstable through collisions with transients or reconstructions of attractors as certain training parameters are varied. To augment this analysis, our next step is to extend current continuation software, such as AUTO⁵³, in order to study the unstable dynamics of trained, untrained, and generated states and how these interact with their stable counterparts.

While the results presented in this paper are based on ‘toy examples’ (synthetic data, not real-world data), in the future we intend to conduct follow-up research to the work presented here using real-world data. However, we would like to take the opportunity to highlight investigations of confabulation in learning systems that include a real-world component. For instance, in terms of confabulation in physical RCs, in Terajima *et al.*⁴² the authors conduct an in-depth study of UAs that appear in a multifunctional physical RC in the form of a soft tensegrity robot. More specifically, the authors trained the robot to reconstruct attractors that correspond to two different motions (rapid crawl and shake). However, after fur-

ther investigation of the robots state space, the authors found that the robot has five additional UAs present in its state space which correspond to completely different motions that it was not trained to exhibit. In terms of confabulation in large language models (LLMs), in Castilho *et al.*⁵⁴ the authors present an interesting study of how different versions of ChatGPT confabulate when used to translate sentences from the English to Irish language (Gaeilge). Interestingly, the authors identify that these LLMs confabulate in a variety of ways by replacing unknown words with made-up words that, for instance, do or do not follow plausible translation rules, or disregard conventions of the Irish alphabet (which does not include certain letters from the English alphabet like ‘v’). In terms of confabulation in the human brain, in El Haj and Larøi⁵⁵ the authors present a study of complete confabulations in patients with Alzheimer’s disease, concluding that complete confabulations are correlated with patients that have spatial and temporal disorientation and that momentary confabulations occur more often than complete confabulations which, interestingly and perhaps coincidentally, corresponds with our findings in Fig. 5.

A. Complete confabulation conjecture

Our results suggest that complete confabulations in the form of UAs are not a random occurrence but a by-product of the system being *bounded* (i.e. a system whose state cannot approach infinity). In the RC we study, this boundedness property is induced by the $\tanh(\cdot)$ activation function as it confines the elements of the RC’s state to be valued between -1 and 1 . Thus, with both the decay term and bounded activation function the state of the system $r(t)$ cannot grow to infinity and the state space of the RC is bounded. As a result, when reconstruction fails, the system does not become unstable with its state tending towards infinity. Instead, since the system cannot approach infinity, it must approach an alternative attractor, which is the UA. In contrast, when reconstruction fails using a so-called ‘next-generation RC’, the system may tend to infinity because there is no guarantee of boundedness in the design of a next-generation RC. Examples of this occurrence can be found in Fig. 1 of Flynn *et al.*⁵⁶.

Here we suggest that the boundedness property of a system may serve as a unifying principle for the intrinsic existence of complete confabulations in artificial and biological learning systems:

- **The complete confabulation conjecture:** any bounded learning system has the capacity to create complete confabulations.

Many learning systems share this property of boundedness. The state of most deep neural networks will be bounded due to the use of common activation functions that are either bounded or asymptotically linear⁵⁷, such that any bounded input will generate a bounded output. Similarly, biological neurons in the brain are subject to physiological constraints and can only respond to stimuli within a finite dynamical range⁵⁸. In our future research we will test the validity of the complete

confabulation conjecture across different types of RCs, other machine learning paradigms, different tasks, and levels of biological realism.

ACKNOWLEDGMENTS

This work is a continuation of work that started during A.F.'s time as a PhD student which was funded by the Irish Research Council Enterprise Partnership Scheme (Grant No. EP-SPG/2017/301) and features elements of work from J.O'H.'s M.Sc. thesis. We would like to thank Andreas Amann for his influential conversations and input when discussing the contents of this paper as it has evolved over the past few years. A.F. would also like to thank Hiroshi Kokubu, Masato Hara, Ryo Terajima, and Yuichiro Terasaki for their influential conversations when discussing the concept of untrained and generated attractors at the recent DEDS conference in Kyoto.

Appendix A: RC design and training parameters

In this paper, \mathbf{M} is constructed with an Erdős-Renyi topology where each of the non-zero elements are then replaced with a random number between -1 and 1 , the matrix is subsequently scaled to a specific spectral radius, ρ . To elaborate, \mathbf{M} is designed such that each element is chosen independently to be nonzero with probability P (i.e. sparsity = P or degree = N/P) and these nonzero elements are chosen uniformly from $(-1, 1)$. This random sparse matrix is rescaled such that the spectral radius, which is the maximum of the absolute values of its eigenvalues, is ρ . Similarly, the input matrix, \mathbf{W}_{in} , is designed such that each row has only one nonzero randomly assigned element, chosen uniformly from $(-1, 1)$.

The numerical experiments in task (ii) and (iii) are performed using the same \mathbf{M} and \mathbf{W}_{in} . From further analysis and results which are not shown here, we remark that there are relatively small quantitative changes to our results in task (ii) and (iii) when using different initialisations of \mathbf{M} and \mathbf{W}_{in} however the main characteristics of our results remain similar.

Appendix B: Classifying output of closed-loop RC in Task (i)

The algorithm used to classify the output of the closed-loop RC (Eq. (10)) after it was trained to reconstruct the chaotic Lorenz attractor, \mathcal{L} , according to the steps outlined in Sec. II E 1 involves the following three main components: The algorithm checks, if after a time t_{trans} ,

- C1: The output remains stationary (fixed point) or periodic (limit cycle).
- C2: The output remains in a subspace of \mathbb{P} that we associate with \mathcal{L} and denote by $\mathbb{P}_{\mathcal{L}}$, i.e., whether the output remains within a range of x_1 , x_2 , and x_3 values.
- C3: The location of the local maxima of the x_3 variable on both wings of the reconstructed \mathcal{L} is less than a distance

α away from the lines fitted through the corresponding local maxima on both wings of \mathcal{L} .

Thus our algorithm has eight parameters, from empirical testing we set these as, $t_{trans} = t_{train} + 70$, $\alpha = 3.75$, $\mathbb{P}_{\mathcal{L}} = \{(x_1, x_2, x_3) | -22 \leq x_1 \leq 22, -32 \leq x_2 \leq 32, 0 \leq x_3 \leq 55\}$.

When C1 is satisfied, we preliminarily classify the output as an ‘untrained attractor’ (UA). By preliminarily classify we mean that we later manually inspect whether this classification is consistent with our definition of an UA as specified in Sec. II D. If C1 is not satisfied, the output is classified as chaotic/quasi-periodic and we then check whether C2 or C3 are satisfied.

We classify the output as a ‘good reconstruction of \mathcal{L} ’, that we denote by $\hat{\mathcal{L}}$ if C2 and C3 are satisfied. A reconstructed attractor that satisfies this criteria is shown in Fig. 2 (a). This shows that all local maxima of the reconstructed attractor (blue points) are sufficiently close to the line fitted through the corresponding local maxima values of \mathcal{L} , indicated by the solid black line.

We classify the output of the closed-loop RC as a ‘poor reconstruction of \mathcal{L} ’, that we denote by $\tilde{\mathcal{L}}$ if C2 is satisfied but C3 is not. A reconstructed attractor which falls into this category is shown in Fig. 2 (b). This shows that some of the local maxima of the reconstructed attractor (red points) are too far away from the line fitted through the corresponding local maxima values of \mathcal{L} , indicated by the dashed black line.

If C2 is not satisfied and C3 is, or if C1, C2, or C3 are not satisfied then we also manually inspect the output to decide whether the output is more consistent with $\tilde{\mathcal{L}}$ or a UA.

Appendix C: Most common reconstruction route for \mathcal{L}

In Fig. 21 we show the most common way that the RC provides a reconstruction of \mathcal{L} when increasing ρ from 0 for one of the matrices \mathbf{M}_i from our ensemble. In each panel of Fig. 21, the black dotted curves correspond to the output of the closed-loop RC (Eq. (10)) when initialised with the last point of the training, $r(t_{train})$. The same colour scheme from Fig. 6 is used to illustrate different UAs that we find. In panel (a) we show that while the closed-loop RC is unable to provide any reconstruction of \mathcal{L} , the brief transient behaviour of the RC's state approaching UA_1 resembles Lorenz-like behaviour. In panels (b)–(e) we see that before approaching UA_1 or UA_2 , the duration of this transient activity increases and the transient also grows in its resemblance to \mathcal{L} . In panel (f) we no longer see this transient behaviour as the RC is able to provide a poor reconstruction of \mathcal{L} . However, this reconstruction has some noticeable features on the right-hand wing ($x_1 > 0$ portion of \mathbb{P}) that are not consistent with \mathcal{L} which is why our algorithm classifies this attractor as a poor reconstruction of \mathcal{L} . In panels (g)–(i) we see that as ρ increases, UA_2 shrinks in size. Interestingly, in panel (j), after UA_2 becomes unstable, this activity on the right-hand wing of the reconstructed \mathcal{L} is no longer visible, and this attractor is classified as a good reconstruction of \mathcal{L} . The RC maintains its good reconstruction of \mathcal{L} for larger values of ρ . Without too much speculation, one reason for this could be that since UA_2 is located near \mathcal{L}

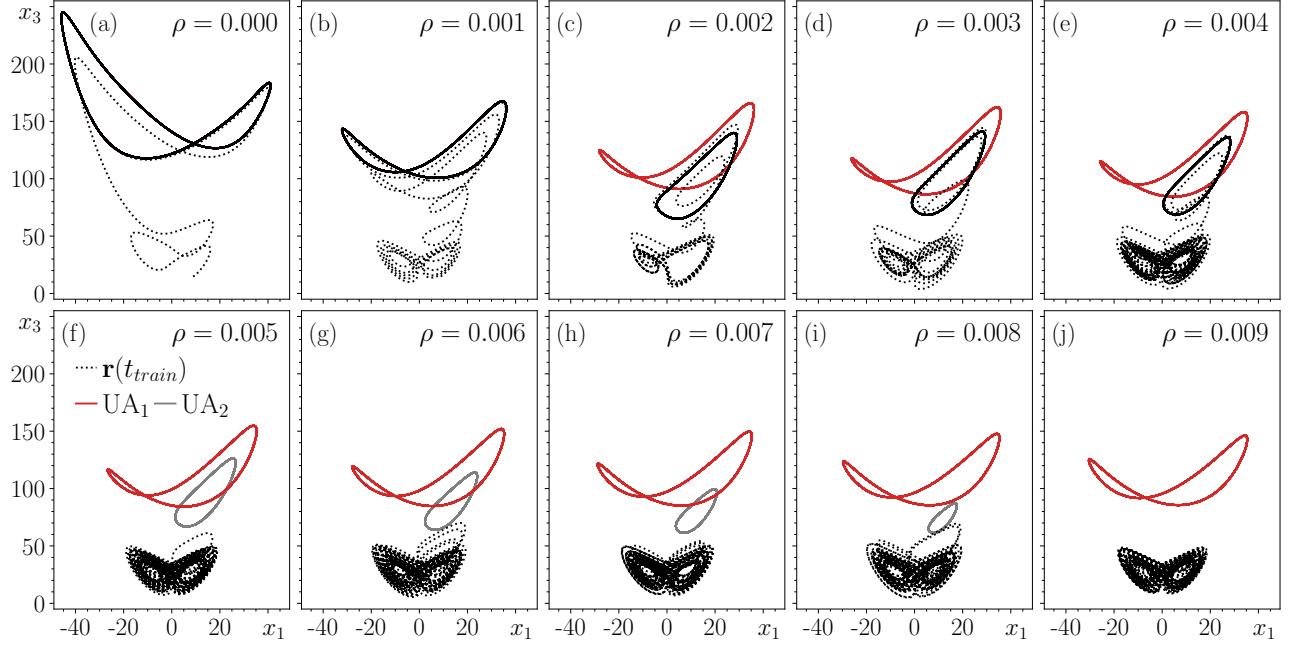


FIG. 21: Lorenz transient becoming stable for increasing ρ shown in the (x_1, x_3) projection of \mathbb{P} . Each black dashed curve corresponds to the trajectory taken by Eq. (10) when initialised with Eq. (11). Curves of different colours correspond to UAs that are present in \mathbb{P} for the specified ρ values.

in this projection, it may also be in \mathbb{S} , and therefore the saddle which separates these states may play a role in preventing the RC from achieving a good reconstruction of \mathcal{L} , and after UA_2 becomes unstable this enables the RC to improve its reconstruction.

In summary, Fig. 21 shows a sequence of events where we see a transition from an unstable reconstruction of \mathcal{L} to a stable reconstruction of \mathcal{L} with a stable UA filling in the gaps. From further investigation we find transitions from a stable UA and unstable reconstruction of \mathcal{L} to an unstable UA and stable reconstruction of \mathcal{L} . We also find switching dynamics via transitions from a stable UA and unstable reconstruction of \mathcal{L} to an unstable UA and unstable reconstruction of \mathcal{L} before another transition to an unstable UA and stable reconstruction of \mathcal{L} . As shown in Flynn and Amann³⁰, when two attractors become unstable in quick succession of one another and there are no other attractors present in \mathbb{P} , then the state of the RC will switch between transients of both attractors in finite time and remain in either transient for different durations of time.

Appendix D: Additional reconstruction routes for \mathcal{L}

In this Appendix we illustrate some less common routes which we find the RC takes to reconstruct \mathcal{L} in task (i). The routes of reconstruction we illustrate here correspond to the classification of the RC's output at different ρ values for \mathbf{M}_{19} and \mathbf{M}_{27} as shown in Fig. 8. We use the same method that was used in Fig. 9 to track the changes in the dynamics of the reconstructed attractor. For both \mathbf{M}_{19} and \mathbf{M}_{27} we encounter no UAs for the values of ρ where the tracking is performed.

More specifically, we plot x_3^{max} (local maxima of the x_3 variable) and the algorithm's classification at specified values of ρ in Figs. 22 and 24. In Figs. 23 and 25 we illustrate the RC's reconstruction of \mathcal{L} at key points indicated by the inward ticks in Figs. 22 and 24.

a. Reconstruction routes found from \mathbf{M}_{19}

The coloured bars and bifurcation diagram in Fig. 22 show that while the RC's reconstruction of \mathcal{L} is reasonably good for $\rho = 0.6$, the reconstruction worsens before it improves for increasing ρ from 0.6 to 0.87. Within this range of ρ values the poor reconstructions of \mathcal{L} , $\tilde{\mathcal{L}}$, undergo several changes between chaotic and periodic dynamics. Fig. 25 highlights some of these changes.

The first major change in the RC's reconstruction of \mathcal{L} occurs nearby $\rho = 0.625$, the transition from $\hat{\mathcal{L}}$ to $\tilde{\mathcal{L}}$ as shown in Figs. 25 (a) and (b) involves a break in the symmetry of the reconstructed attractor. However what is more interesting about the dynamics of $\tilde{\mathcal{L}}$ for this \mathbf{M} , and the changes it undergoes for increasing ρ as shown in panels (b) to (i), is that it provides a much clearer indication of the mechanisms that give rise to this interesting break in the symmetry of the RC's dynamics. From looking at these changes in the dynamics of $\tilde{\mathcal{L}}$, the prolonged winding motion nearby the point $(x_1, x_3) = (8, 20)$ followed by the much larger orbit strongly resembles a saddle-focus homoclinic orbit, indicating that a Shilnikov bifurcation occurs. Figs. 22 and 25 show that this attractor exhibits periodic dynamics for a wide range of ρ values and also undergoes a period-doubling route to chaos nearby $\rho = 0.675$. While

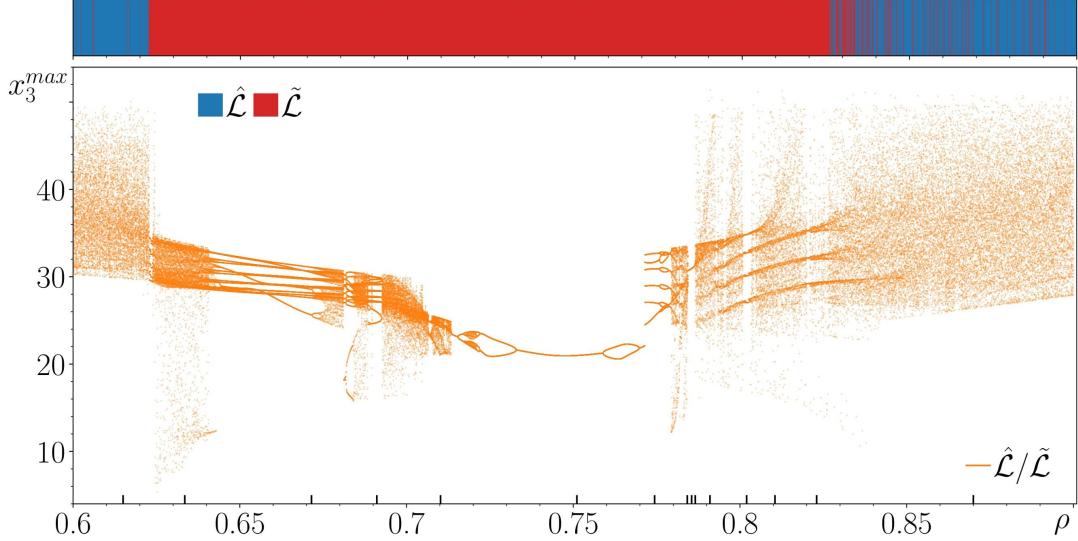


FIG. 22: Route for reconstruction of \mathcal{L} for \mathbf{M}_{19} . Plotting changes in local maxima of x_3 in \mathbb{P} for $x_1 > 0$ for changes in ρ for all attractors found for $\rho \in [0.6, 0.9]$. Inward ticks indicate ρ values used to generate Fig. 23. Coloured bars above bifurcation diagram indicate the algorithm's classification of the RC's output for a given ρ .

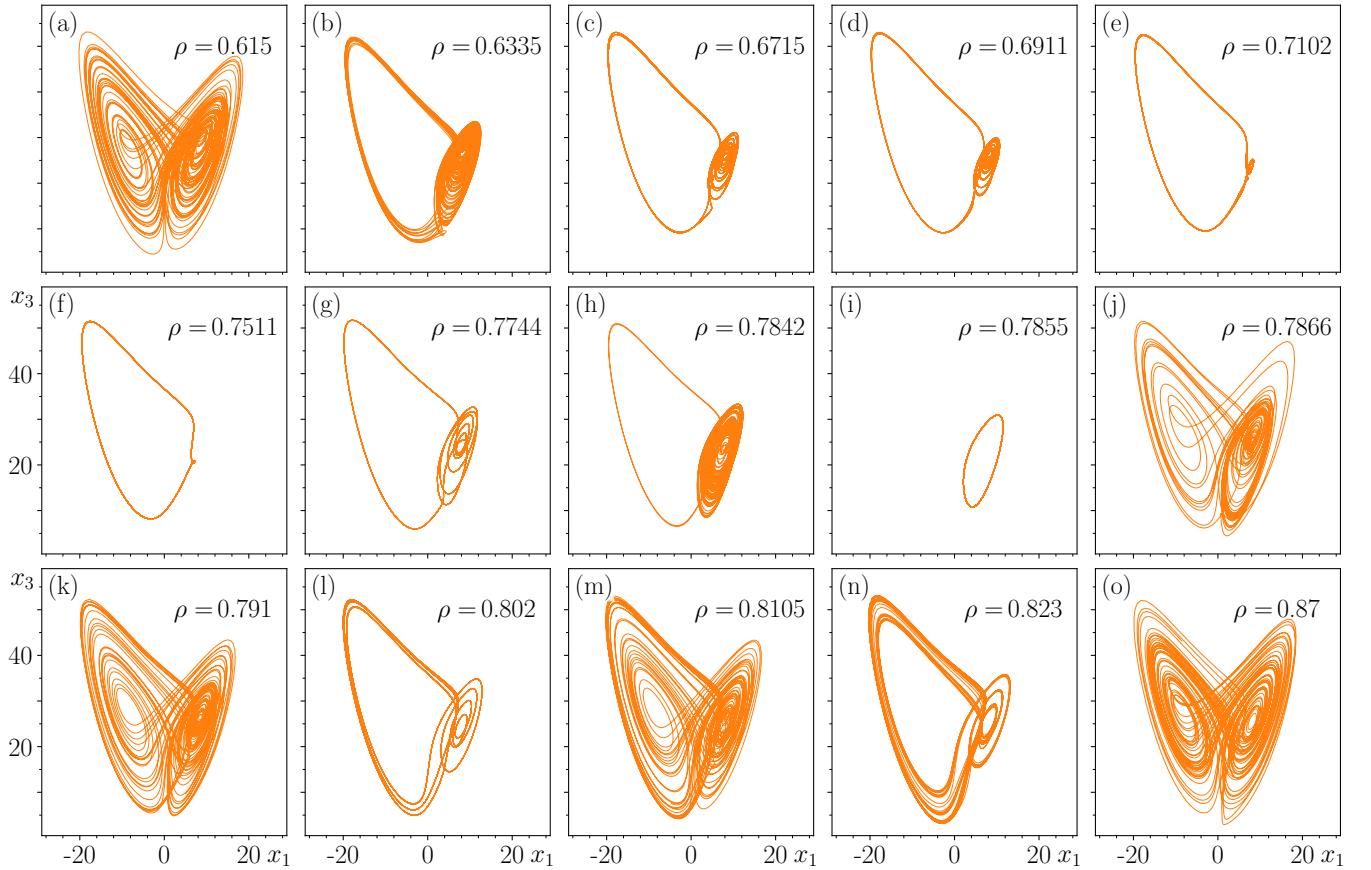


FIG. 23: Snapshots of the different routes for reconstruction of \mathcal{L} shown in Fig. 22 at specified values of ρ in panels (a)–(e).

this attractor alternates between chaotic and periodic dynamics, what is most interesting here is that the chaotic dynamics are largely confined to the portion of the trajectory that revolves around the saddle-focus and while that the homoclinic orbit portion of the trajectory can also be chaotic (like in (b)) it is mainly periodic (like in (c)–(h)) depending on the value of ρ . In other words, in (b) there are multiple routes in and out of the saddle-focus portion of the trajectory on the attractor but in (c)–(h) there is typically only one or two routes in and out. For a small range of ρ values nearby 0.785 we see in Fig. 22 (i) that there is no homoclinic orbit component to $\tilde{\mathcal{L}}$. The growth of the saddle-focus portion of the attractor from (f)–(h) indicates that the transition from (h) to (i) may be due to a collision with a saddle. Increasing ρ further we find that the reconstruction alternates between a more Lorenz-like chaotic attractor and the saddle-focus homoclinic orbit which also alternates between chaotic and periodic dynamics as shown in Figs. 22 (j)–(o). For $\rho > 0.825$ we see that reasonably good reconstructions of \mathcal{L} , like in Fig. 22 (o), become more and more prevalent.

b. Reconstruction routes found from M_{27}

The coloured bars and bifurcation diagram in Fig. 22 show that the RC's reconstruction of \mathcal{L} improves by increasing ρ , however there is by no means a gradual improvement from the poor reconstruction, $\tilde{\mathcal{L}}$, to the good reconstruction, $\hat{\mathcal{L}}$, as $\tilde{\mathcal{L}}$ undergoes a wide-range of changes from $\rho = 0.6$ to 1.25.

The first of the major changes in the dynamics of $\tilde{\mathcal{L}}$ occurs around $\rho = 0.66$ where the Lorenz-like chaotic attractor in Fig. 23 (a) becomes a limit cycle in (b). What is interesting about this limit cycle is that there is a clear break in the symmetry of the RC's dynamics, two round trips of the right-hand wing occur before a single trip around the left-hand wing. These periodic dynamics do not persist for many ρ values as $\tilde{\mathcal{L}}$ becomes chaotic again near $\rho = 0.67$. However, it is after the return to chaos that we see break in the symmetry of the RC's dynamics becomes much more pronounced, the range of values taken by x_3^{\max} for the right-hand wing of $\tilde{\mathcal{L}}$ increases as ρ is increased to 0.8633. These changes in $\tilde{\mathcal{L}}$ are further highlighted by Figs. 23 (c)–(e), it is also worth noting in (e) that we indirectly observe the presence of a saddle point given the sharp bend in the trajectory nearby $(x_1, x_3) = (0, 0)$ thereby indicating that the attractor is increasing in volume and approaching a saddle point. This may be responsible for the return to periodic dynamics in (f) and (g) that we see as ρ is increased further. In contrast to the limit cycle shown previously in (b), we see in (f) and (g) there are more round trips on the left-hand wing. $\tilde{\mathcal{L}}$ alternates between chaotic and periodic dynamics for increasing ρ and the range of values taken by x_3^{\max} for the right-hand wing of $\tilde{\mathcal{L}}$ also starts to decrease. The first instances of good reconstructions of \mathcal{L} , namely $\hat{\mathcal{L}}$, emerge near $\rho = 1.215$ and the RC maintains a reasonably good reconstruction of \mathcal{L} for $\rho > 1.24$.

Appendix E: Examining the continuous transition between $\hat{\mathcal{L}}$ and $\hat{\mathcal{H}}$

To more clearly illustrate that in Fig. 20 (d) the transition in $\hat{\mathcal{H}}^*$ from chaotic to the fixed point dynamics near $b = -0.05$ happens via a bifurcation, in Fig. 26 we re-present Fig. 20 (d) from the perspective of the changes in the local minima of the x_3 -variable. Also, to help further illustrate this bifurcation and the bifurcation from the limit cycle near $b = 0.055$, in Figs. 27 and 28, we present snapshots of the RC's dynamics at different b values before and after these bifurcations take place.

When looking at Figs. 26 and 27 together we see that there is no direct transition from chaotic attractor to fixed point, but rather the chaotic attractor that exists for $b = -0.056$ transitions to a limit cycle for $b = -0.055$. Prior to this transition, we see in Fig. 27 (c) that significant portions of the trajectory on this chaotic attractor are concentrated around where the limit cycle subsequently emerges. What is common among the limit cycle and chaotic attractor is that the trajectory around each attractor appears to contain a point nearby $(x_1, x_3) = (-6, 26)$ and the state of the system spends a relatively large duration of time in the vicinity of this point. This suggests these attractors are of the saddle-focus homoclinic orbit type. Furthermore, it is this point (the saddle-focus) which becomes the stable fixed point for $b = -0.054$ shown in Fig. 27 (a).

In Fig. 28 we take a closer look at the transition in $\hat{\mathcal{L}}^*$ from Lorenz-like dynamics to the limit cycle near $b = 0.06$ in Fig. 26. In Fig. 28 (a) we see that the state of the RC spends much larger durations of time in the left-hand wing of the attractor. The sharp corner in the trajectory nearby $(x_1, x_3) = (-1, 12)$ indicates there is a saddle nearby, as flow converges towards this point along the x_3 direction and diverges away from this point along the x_1 direction. When decreasing b from 0.062 in Fig. 28 (a) to 0.06 in Fig. 28 (b), it appears that the Lorenz-like attractor and saddle collide, which results in the right-hand wing disappearing and subsequently results in the period-4 limit cycle seen in Fig. 28 (b). We choose to show the change in the dynamics of this limit cycle from $b = 0.06$ in Fig. 28 (b) to $b = 0.04$ in Fig. 28 (b) in order to compliment the dynamics shown in Fig. 26, that the limit cycle continues to shrink until it becomes point-like and subsequently becomes the fixed point shown explicitly in Fig. 27 (a), which suggests that a Hopf bifurcation is responsible for this transition.

While we do not see regions of bistability between the GAs from the reconstructions of the different limit cycles in task (ii), the results presented in Figs. 20 (d) and 26 indicates that such a scenario is also possible. This leads us to make the following, more general, statement. If the RC fails to interpolate between reconstructed attractors via a sequence of continuous transitions then two options are possible: there is either a region of bistability between GAs that arise from the reconstructed attractors, or there is an UA that coexists with GAs or reconstructed attractors at different parameter settings.

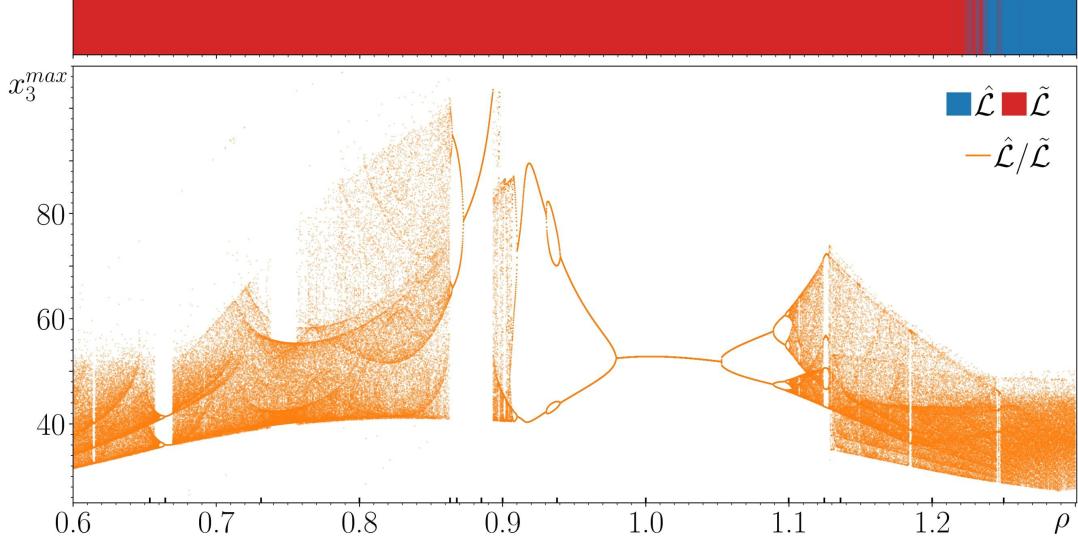


FIG. 24: Period-doubling route for reconstruction of \mathcal{L} for \mathbf{M}_{27} . Plotting changes in local maxima of x_3 in \mathbb{P} for $x_1 > 0$ for changes in ρ for all attractors found for $\rho \in [0.6, 1.3]$. Inward ticks indicate ρ values used to generate Fig. 25. Coloured bars above bifurcation diagram indicate the algorithm's classification of the RC's output for a given ρ .

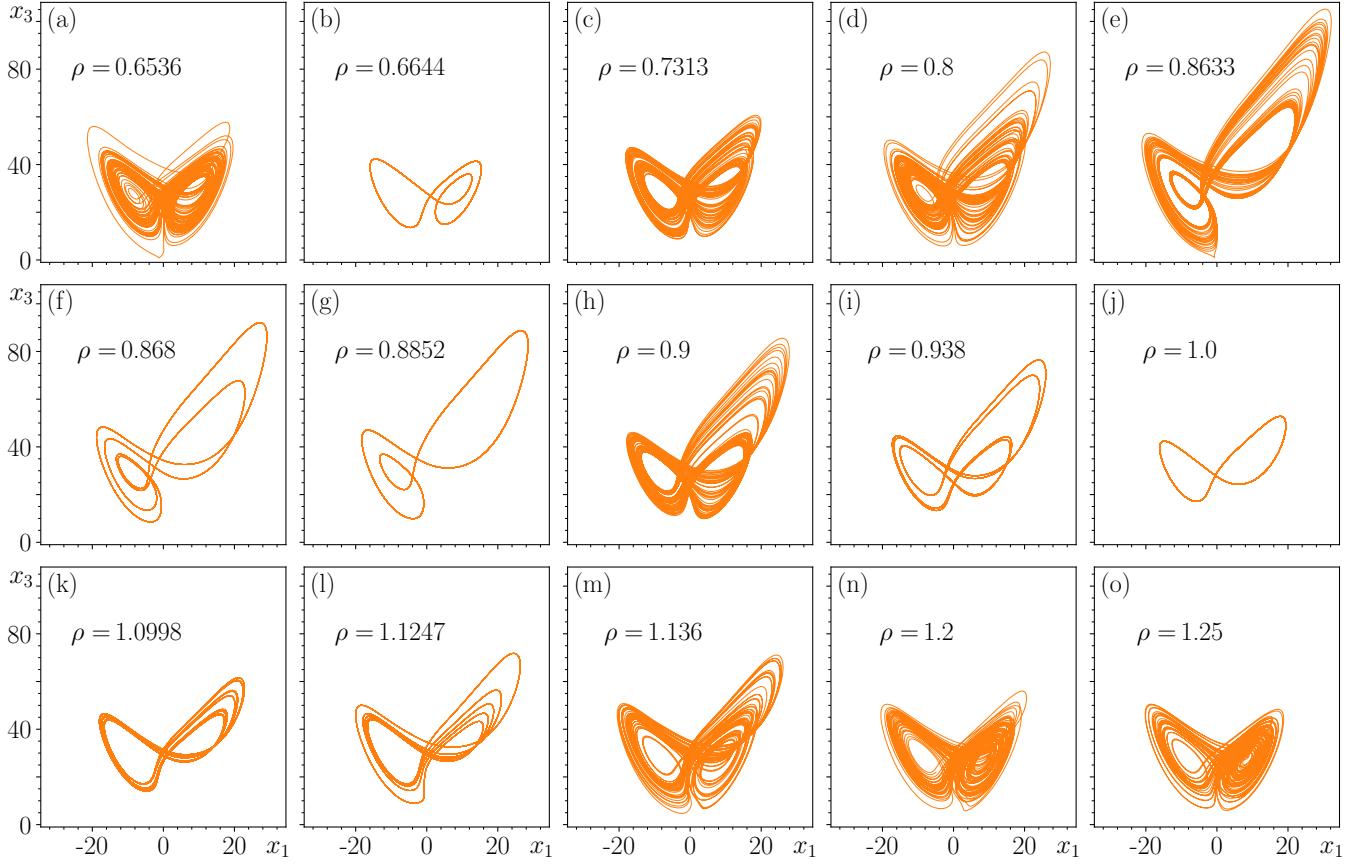


FIG. 25: Snapshots of the different routes for reconstruction of \mathcal{L} shown in Fig. 24 at specified values of ρ in panels (a)–(e). Main types of transitions: homoclinic orbit of saddle focus - period-doubling, interior crisis describes rapid growth of chaotic attractor/limit cycle

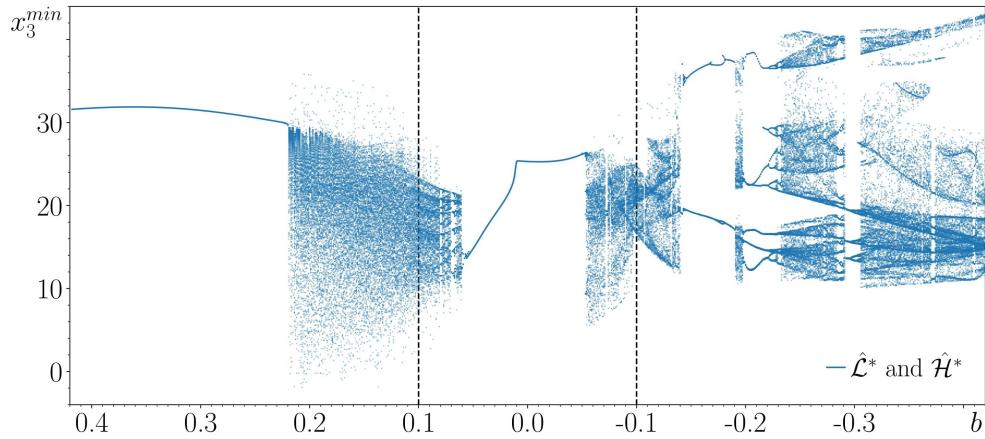


FIG. 26: Tracking changes with respect to b in the local minima of the x_3 variable of the GAs that form after training the parameter-aware RC to perform task (iii) with $b = \pm 0.1$.

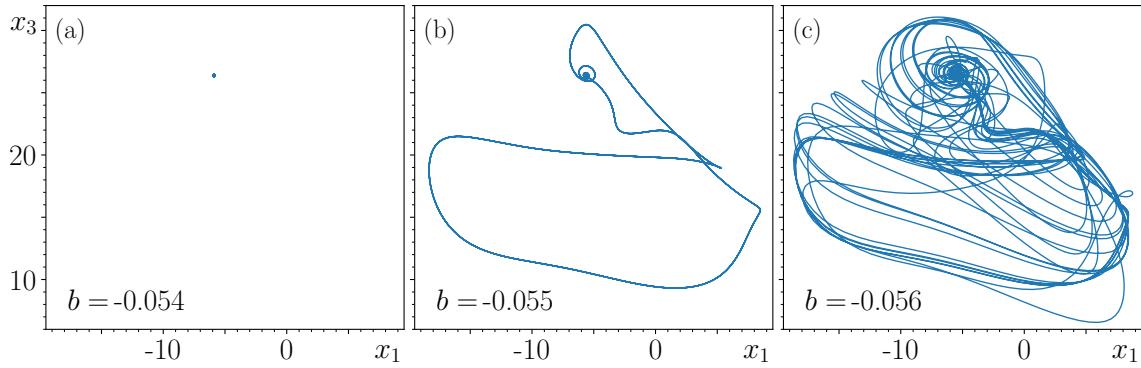


FIG. 27: Illustrating transition from $\hat{\mathcal{H}}^*$ to fixed point for decreasing b as shown in Figs. 19 (d) and 26.

DATA AVAILABILITY

The data that support the findings of this study are available from the corresponding author upon reasonable request.

- ¹R. Emsley, *Schizophrenia* **9**, 52 (2023).
- ²W. H. Walters and E. I. Wilder, *Scientific Reports* **13**, 14045 (2023).
- ³Cambridge Dictionary, *Word of the Year 2023* (2023).
- ⁴A. McGowan, Y. Gui, M. Dobbs, S. Shuster, M. Cotter, A. Selloni, M. Goodman, A. Srivastava, G. A. Cecchi, and C. M. Corcoran, *Psychiatry Research* **326**, 115334 (2023).
- ⁵B. Edwards, *Ars Technica* (2023).
- ⁶A. Seth, *Conscious artificial intelligence and biological naturalism* (2024).
- ⁷A. Schnider, *Nature Reviews Neuroscience* **4**, 662 (2003).
- ⁸N. J. Arts, S. J. Walvoort, and R. P. Kessels, *Neuropsychiatric disease and treatment* , 2875 (2017).
- ⁹M. El Haj and F. Larøi, *Psychiatry and Clinical Neurosciences* **71**, 61 (2017).
- ¹⁰A. Gilboa, M. Moscovitch, A. Baddeley, M. Kopelman, and B. Wilson, *Handbook of memory disorders* **2**, 315 (2002).
- ¹¹Y. Bengio, G. Hinton, A. Yao, D. Song, P. Abbeel, T. Darrell, Y. N. Harari, Y.-Q. Zhang, L. Xue, S. Shalev-Shwartz, *et al.*, *Science* **384**, 842 (2024).
- ¹²L. Weidinger, J. Uesato, M. Rauh, C. Griffin, P.-S. Huang, J. Mellor, A. Glaese, M. Cheng, B. Balle, A. Kasirzadeh, *et al.*, in *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency* (2022) pp. 214–229.
- ¹³H. Jaeger, Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148** (2001).
- ¹⁴W. Maass, T. Natschläger, and H. Markram, *Neural computation* **14**, 2531 (2002).
- ¹⁵K. Nakajima and I. Fischer, *Reservoir Computing* (Springer, 2021).
- ¹⁶Z.-M. Zhai, M. Moradi, L.-W. Kong, B. Glaz, M. Haile, and Y.-C. Lai, *Nature communications* **14**, 5698 (2023).
- ¹⁷T. Arcomano, I. Szunyogh, A. Wikner, B. R. Hunt, and E. Ott, *Geophysical Research Letters* **50**, e2022GL102649 (2023).
- ¹⁸G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, *Neural Networks* **115**, 100 (2019).
- ¹⁹K. Nakajima, *Japanese Journal of Applied Physics* **59**, 060501 (2020).
- ²⁰A. Flynn, V. A. Tsachouridis, and A. Amann, *Chaos* **31**, 013125 (2021).
- ²¹Z. Lu and D. S. Bassett, *Chaos* **30**, 063133 (2020).
- ²²L.-W. Kong, G. A. Brewer, and Y.-C. Lai, *Nature Communications* **15**, 4840 (2024).
- ²³Z. Lu, B. R. Hunt, and E. Ott, *Chaos* **28**, 061104 (2018).
- ²⁴J. Z. Kim, Z. Lu, E. Nozari, G. J. Pappas, and D. S. Bassett, *Nature Machine Intelligence* **3**, 316 (2021).
- ²⁵L.-W. Kong, H.-W. Fan, C. Grebogi, and Y.-C. Lai, *Physical Review Research* **3**, 013090 (2021).
- ²⁶APA Dictionary of Psychology, *Definition of Confabulation* (2018).
- ²⁷M. Nejjar, L. Zacharias, F. Stiehle, and I. Weber, *Journal of Software: Evolution and Process* **37**, e2723 (2025).
- ²⁸A. Flynn, V. A. Tsachouridis, and A. Amann, *Chaos* **33** (2023).
- ²⁹J. Morra, A. Flynn, A. Amann, and M. Daley, in *2023 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (IEEE, 2023) pp. 4961–4966.

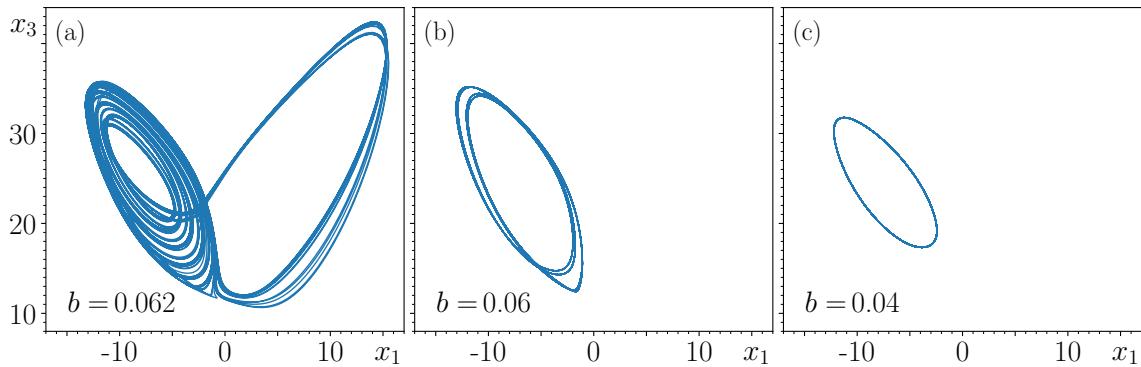


FIG. 28: Illustrating transition from $\hat{\mathcal{L}}^*$ to limit cycle for decreasing b as shown in Figs. 19 (d) and 26.

- ³⁰A. Flynn and A. Amann, *Frontiers in Network Physiology* **4**, 1451812 (2024).
³¹J. Herteux and C. Räth, *Chaos* **30**, 123142 (2020).
³²A. Flynn, J. Herteux, V. A. Tsachouridis, C. Räth, and A. Amann, *Chaos* **31**, 073122 (2021).
³³E. N. Lorenz, *Journal of the Atmospheric Sciences* **20**, 130 (1963).
³⁴C. Li, J. C. Sprott, and H. Xing, *International Journal of Bifurcation and Chaos* **26**, 1650233 (2016).
³⁵A. Ceni, P. Ashwin, and L. Livi, *Cognitive Computation* **12**, 330 (2020).
³⁶M. Fadera and P. Ashwin, *Physica D: Nonlinear Phenomena* **470**, 134358 (2024).
³⁷A. V. Robins and S. J. McCallum, *Neural Networks* **17**, 313 (2004).
³⁸L. Pecora and T. Carroll, arXiv preprint arXiv:2409.04571 (2024).
³⁹Y. Terasaki and K. Nakajima, arXiv preprint arXiv:2405.08825 (2024).
⁴⁰T. Kabayama, Y. Kuniyoshi, K. Aihara, and K. Nakajima, *Phys. Rev. E* **111**, 034207 (2025).
⁴¹H. Tomioka, K. Inoue, Y. Kuniyoshi, and K. Nakajima, arXiv preprint arXiv:2503.05601 (2025).
⁴²R. Terajima, K. Inoue, K. Nakajima, and Y. Kuniyoshi, arXiv preprint arXiv:2507.21496.
⁴³T. G. de Jong, H. Notsu, and K. Nakajima, arXiv preprint arXiv:2502.04818 (2025).
⁴⁴Z. Liu, O. Kitouni, N. S. Nolte, E. Michaud, M. Tegmark, and M. Williams, *Advances in Neural Information Processing Systems* **35**, 34651 (2022).
⁴⁵L. Struski, M. Sadowski, T. Danel, J. Tabor, and I. T. Podolak, *IEEE Transactions on Neural Networks and Learning Systems* (2023).
⁴⁶D. Zeithamova, M. L. Schlichting, and A. R. Preston, *Frontiers in human neuroscience* **6**, 70 (2012).
⁴⁷M. G. Uitvlugt and M. K. Healey, *Psychological science* **30**, 92 (2019).
⁴⁸R. Barrio, A. Shilnikov, and L. Shilnikov, *International Journal of Bifurcation and Chaos* **22**, 1230016 (2012).
⁴⁹A. Röhm, D. J. Gauthier, and I. Fischer, *Chaos* **31**, 103127 (2021).
⁵⁰L.-W. Kong, Y. Weng, B. Glaz, M. Haile, and Y.-C. Lai, *Chaos* **33**, 033111 (2023).
⁵¹D. Köglmayr and C. Räth, *Scientific Reports* **14**, 507 (2024).
⁵²S. Panahi, L.-W. Kong, M. Moradi, Z.-M. Zhai, B. Glaz, M. Haile, and Y.-C. Lai, *Physical Review Research* **6**, 043194 (2024).
⁵³E. Doedel, A. Champneys, F. Dercole, T. Fairgrieve, Y. Kuznetsov, B. Oldeman, R. Paffenroth, B. Sandstede, X. Wang, and C. Zhang, *Auto-07p: Continuation and bifurcation software for ordinary differential equations* (2007).
⁵⁴S. Castilho, Z. Fitzsimmons, C. Holton, and A. M. Donagh, arXiv preprint arXiv:2504.07680 (2025).
⁵⁵M. El Haj and F. Larøi, *Psychiatry and Clinical Neurosciences* **71**, 61 (2017).
⁵⁶A. Flynn, O. Heilmann, D. Köglmayr, V. A. Tsachouridis, C. Räth, and A. Amann, in *2022 International Joint Conference on Neural Networks (IJCNN)* (2022) pp. 1–8.
⁵⁷T. Szandalà, in *Bio-inspired neurocomputing* (Springer, 2020) pp. 203–224.
⁵⁸P. Miller and J. Cannon, *Biological cybernetics* **113**, 47 (2019).