Exploring the limits of multifunctionality across different reservoir computers

Andrew Flynn*
School of Mathematical Sciences
University College Cork
Cork, Ireland
andrew_flynn@umail.ucc.ie

Oliver Heilmann
Department of Physics
Ludwig-Maximilians University
Munich, Germany
o.heilmann@campus.lmu.de

Daniel Köglmayr
Department of Physics
Ludwig-Maximilians University
Munich, Germany
d.koeglmayr@physik.uni-muenchen.de

Vassilios A. Tsachouridis
Collins Aerospace
Cork, Ireland
vassilios.tsachouridis@collins.com

Christoph Räth
Institut für KI-Sicherheit
Deutsches Zentrum für Luft-und Raumfahrt
Wessling, Germany
christoph.raeth@dlr.de

Andreas Amann
School of Mathematical Sciences
University College Cork
Cork, Ireland
a.amann@ucc.ie

***Abstract*—Multifunctional neural networks are capable of performing more than one task without changing any network connections. In this paper we explore the performance of a continuous-time, leaky-integrator, and next-generation 'reservoir computer' (RC), when trained on tasks which test the limits of multifunctionality. In the first task we train each RC to reconstruct a coexistence of chaotic attractors from different dynamical systems. By moving the data describing these attractors closer together, we find that the extent to which each RC can reconstruct both attractors diminishes as they begin to overlap in state space. In order to provide a greater understanding of this inhibiting effect, in the second task we train each RC to reconstruct a coexistence of two circular orbits which differ only in the direction of rotation. We examine the critical effects that certain parameters can have in each RC to achieve multifunctionality in this extreme case of completely overlapping training data.**

***Index Terms*—Reservoir Computing; Multifunctionality; Floquet analysis**

I. Introduction

Advancements in machine learning oftentimes arise from a 'two-way street' between neuroscientific observation and mathematical representation. By following this approach, 'multifunctional reservoir computing' [1] has emerged as a means of training an artificial neural network to perform more than one task without the need to switch between different network configurations unlike other approaches such as modular neural networks [2], or conceptors [3].

'Multifunctionality' describes the ability of a single neural network to perform a multitude of mutually exclusive tasks, in other words, they possess a form of multistability. There are many examples of multifunctional neural networks in nature, for further reading we suggest [4].

The types of multistabilities that multifunctional reservoir computers (RCs) have been trained to produce range from reconstructing the dynamics of chaotic attractors which already coexist, to creating a coexistence of chaotic attractors from two different dynamical systems [1], [5]. Furthermore, in [6], it was shown that even when there are overlapping regions between these attractors, a RC can also achieve multifunctionality. However, this required increasing the spectral radius of the RCs internal connections from the case where there was no overlap to achieve multifunctionality.

As no multistable autonomous dynamical system will ordinarily have coexisting attractors which share common regions of state space, this brings into question what are the essential conditions for a RC to achieve multifunctionality in this scenario. In order to gain a greater understanding of the limits of multifunctionality in the case of overlapping training data, in this paper we examine the critical effects that certain parameters can have in the training of different types of RCs. Furthermore, by exploring these limitations we are able to come closer to establishing the extent of the dynamical functionality a given RC can reconstruct.

In our numerical experiments we investigate the behaviour of three RC setups, a continuous-time RC (CT-RC) [7], a leaky-integrator RC (LI-RC) [8], and the recently introduced 'next generation' RC (NG-RC) [9]. Each of these RCs are trained using a ridge regression approach.

These experiments consist of training a RC to achieve multifunctionality by reconstructing a coexistence of chaotic attractors from the Lorenz and Halvorsen systems like the example used in [5]. In this task we examine the performance of each RC setup as the data describing each attractor are moved closer together. For a given set of training parameters we find that each RC can reconstruct both attractors reasonably well when they are well separated in state space. However, for the same set of training parameters, the closer the attractors are to one another, the poorer each RC performs.

To place further emphasis on the influence of these training parameters, in the second task we study an extreme case

*Corresponding author
AF is funded by the Irish Research Council Enterprise Partnership Scheme (Grant No. EPSPG/2017/301).

of overlapping training data. In this experiment each RC is trained to reconstruct a coexistence of trajectories on two completely overlapping circular orbits rotating in opposite directions. In order for the CT-RC and LI-RC to achieve multifunctionality in this scenario, these RCs are required to convert this overlapping driving input data into attractors with two distinct basins of attraction that can be projected to resemble the desired coexistence. While for the NG-RC, we find that it achieves multifunctionality in a different way due to its design. Instead of generating trajectories on limit cycles, the trained NG-RC is a linear system which produces centers.

Overall, our results indicate that the LI-RC outperforms the CT and NG-RC on these tasks. Despite arguing that memory is a key element for the RCs to achieve multifunctionality in the case of overlapping training data, we find that a commonly used 'memory capacity' metric [10] is insufficient in distinguishing whether different random realisations of the RCs can give rise to multifunctionality. Instead, we find that a Floquet analysis provides a more rigorous assessment.

The rest of the paper is organised as follows: In Sec. II we outline the steps involved in training each RC setup to achieve multifunctionality. In Sec. III we describe the numerical experiments in greater detail and the analysis tools used to assess the performance each RC. In Sec. IV we discuss our results and in Sec. V we provide some concluding remarks.

II. Multifunctional Reservoir Computing

Since the introduction of echo-state networks (ESNs) [11], liquid-state machines (LSMs) [12], and their unification under the umbrella term of reservoir computing [13], there have been many variations of RCs in terms of mathematical formulation and topology [14]–[18].

In this paper we examine the performance of CT, LI, and NG RC setups on tasks requiring multifunctionality. We now outline the method of using each RC for time series prediction.

A. Continuous-time RC

We use the CT-RC setup introduced in [7]. During the training stage, the CT-RC is driven by an input signal, $\mathbf{u}(t)$, and its response is described by,

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

$\mathbf{r}_{CT}(t) \in \mathbb{S} \subset \mathbb{R}^N$ is the state of the CT-RC in state space, $\mathbb{S}$, at a given time t, N is the number of neurons, and $\mathbf{r}_{CT}(0) = \mathbf{0}$. γ is a time-scale parameter. $\mathbf{M} \in \mathbb{R}^{N\times N}$ is the adjacency matrix. The input strength parameter, σ, and the input matrix, $\mathbf{W}_{in} \in \mathbb{R}^{N\times D}$, assign the weight given to the D-dimensional input, $\mathbf{u}(t) \in \mathbb{R}^D$, as it's projected into the reservoir. $\mathbf{M}$ and $\mathbf{W}_{in}$ are randomly initialised and design details are provided in the Appendix. The relationship between multifunctionality and the spectral radius, ρ, of $\mathbf{M}$ is assessed in Sec. IV.

Solutions of Eq. (1) are generated using the 4^{th} order Runge-Kutta method with time step $\tau = 0.01$. From $t = t_{listen}$ to $t = t_{train}$, we store the CT-RCs state in $\mathbf{X} = [\mathbf{q}(\mathbf{r}_{CT}(t_{listen})) \ldots \mathbf{q}(\mathbf{r}_{CT}(t_{train}))]$ for $\mathbf{q}(\mathbf{r}(t)) = (\mathbf{r}(t)\,\mathbf{r}^2(t))^T$, and the input in $\mathbf{Y} = [\mathbf{u}(t_{listen}) \ldots \mathbf{u}(t_{train})]$. In order to train the RC we compute a readout layer, $\psi(\mathbf{r}(t)) = \mathbf{W}_{out}\,\mathbf{q}(\mathbf{r}(t))$ using the following ridge regression formula,

$$\mathbf{W}_{out} = \mathbf{Y}\mathbf{X}^T\left(\mathbf{X}\mathbf{X}^T + \beta\,\mathbf{I}\right)^{-1}. \quad (2)$$

β is the regularisation parameter, and $\mathbf{I}$ is the identity matrix of the appropriate dimension. Note that the RCs response from $t = 0$ until $t = t_{listen}$ is not included in the training in order to remove any dependence the RC has on its initialisation.

After training, the 'predicting RC' evolves according to,

$$\dot{\hat{\mathbf{r}}}_{CT}(t) = \boldsymbol{\gamma}[-\hat{\mathbf{r}}_{CT}(t) + \tanh(\,\mathbf{M}\,\hat{\mathbf{r}}_{CT}(t) + \sigma\mathbf{W}_{in}\mathbf{W}_{out}\,\mathbf{q}(\hat{\mathbf{r}}_{CT}(t))\,)], \quad (3)$$

and $\hat{\mathbf{r}}_{CT}(0) = \mathbf{r}_{CT}(t_{train})$.

B. Leaky integrator RC

The LI-RC was introduced in [8], its response to a driving input signal, $\mathbf{u}[i]$, during the training stage is given by,

$$\mathbf{r}_{LI}[i+1] = (1-\alpha)\mathbf{r}_{LI}[i] + \alpha\tanh(\mathbf{M}\mathbf{r}_{LI}[i] + \sigma\mathbf{W}_{in}\mathbf{u}[i]), \quad (4)$$

where $\mathbf{r}_{LI}[i]$ describes the state of the LI-RC in discrete-time with terms equivalently defined as the CT-RC in Sec. II-A. The difference between the CT-RCs and LI-RCs is the role of the leaky-integrator parameter, $\alpha \in [0, 1]$. For $\alpha = 1$, the influence of the RCs previous state appears only in $\tanh(\cdot)$.

The LI-RC is trained with the same approach as the CT-RC for the corresponding solutions between i_{listen} and i_{train}. The predicting LI-RC is written as,

$$\hat{\mathbf{r}}_{LI}[i+1] = (1-\alpha)\hat{\mathbf{r}}_{LI}[i] + \alpha\tanh(\,\mathbf{M}\hat{\mathbf{r}}_{LI}[i] + \sigma\mathbf{W}_{in}\mathbf{W}_{out}\mathbf{q}(\hat{\mathbf{r}}_{LI}[i])\,), \quad (5)$$

and $\hat{\mathbf{r}}_{LI}[0] = \mathbf{r}_{LI}[i_{train}]$.

Note, dividing Eq. (5) by τ and taking the limit as $\tau \to 0$ gives Eq. (3) and implies that $\gamma = \alpha\tau$ and $t = i\tau$. In Sec. IV we compare the influence that different values of α and γ have on the LI and CT RCs.

C. Next generation RC

The NG-RC we use was introduced in [9]. In this setup, the input data are transformed with a polynomial multiplication dictionary, $\mathbf{P}^{[O]}$, into a higher dimensional state space consisting of the unique polynomials of orders, O. For example, transforming a two-dimensional input data point, $\mathbf{u}[i] = (u_1[i], u_2[i])^T$, with $\mathbf{P}$ for $O = 1, 2$ is written as,

$$\mathbf{P}^{[\mathbf{1},\mathbf{2}]}(\mathbf{u}[i]) = \left(u_1[i]\ u_2[i]\ u_1^2[i]\ u_2^2[i]\ u_1[i]u_2[i]\right)^T. \quad (6)$$

A time shift expansion function, $\mathbf{L}_k^s$, of the input data is used in [9] to distinguish the NG-RC from nonlinear vector autoregression algorithms [19]. The k value defines the number of past data points that the current data point is concatenated with, and the s value denotes how far these points are separated in time. Following the example of a two-dimensional input data point, we write this time shift as,

$$\mathbf{L}_2^1(\mathbf{u}[i]) = \left(u_1[i]\ u_2[i]\ u_1[i-1]\ u_2[i-1]\right)^T. \quad (7)$$

The NG-RCs response to $\mathbf{u}[i]$ during training is written as,

$$\mathbf{r}_{NG}[i+1] = \mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{u}[i])). \tag{8}$$

Therefore, in this example we write $\mathbf{r}_{NG}[i+1] \in \mathbb{R}^{14}$ as,

$$(u_1[i]\ u_2[i]\ u_1[i-1] \dots u_1[i]u_2[i-1]\ u_2[i]u_2[i-1])^T. \tag{9}$$

In this setup, the reservoir state vector, $\mathbf{r}_{NG}[i]$, is projected using a readout matrix, $\mathbf{W}_{out}$, to resemble, $\Delta\mathbf{u}[i] = \mathbf{u}[i] - \mathbf{u}[i-1]$, for $i > i_{train}$. $\mathbf{W}_{out}$ is found using Eq. 2, with the corresponding $\mathbf{X}$ and $\mathbf{Y}$ constructed as follows.

The input training data $\mathbf{Y} = [\mathbf{u}[i_{warm}], \dots, \mathbf{u}[i_{train}]]$ is transformed to the state matrix $\mathbf{X} = \mathbf{q}\left(\mathbf{P}^{[O]}(\mathbf{L}_k^s(\mathbf{Y}))\right)$. Note, a warm up time of $i_{warm} = ks$ is needed, where entries of the state matrix at time $i < i_{warm}$ are not defined. The output target matrix used in Eq. 2 for this setup is written as, $\mathbf{Y}' = \mathbf{Y}[i] - \mathbf{Y}[i-1]$. The trained NG-RC evolves according to,

$$\hat{\mathbf{r}}_{NG}[i+1] = \mathbf{P}^{[O]}(\mathbf{L}_k^s(\hat{\mathbf{u}}_{NG}[i-1] + \mathbf{W}_{out}\mathbf{q}\left(\hat{\mathbf{r}}_{NG}[i]\right))), \tag{10}$$

for $\hat{\mathbf{r}}_{NG}[0] = \mathbf{r}_{NG}[i_{train}]$ and $\hat{\mathbf{u}}_{NG}[i-1] = \mathbf{u}[i_{train}] + \sum_{i_{train}}^{i-1} \mathbf{W}_{out}\mathbf{q}\left(\hat{r}_{NG}[i]\right)$ estimates $\mathbf{u}[i]$ for $i > i_{train}$.

As the NG-RC requires tuning only O, k, s, and β, the issues which relate to improper random initialisations of $\mathbf{M}$ and $\mathbf{W}_{in}$ do not arise. On the other hand we see in Sec. IV that there are limitations to what can be achieved with the NG-RC because of its design in comparison to CT and LI-RCs.

D. Training each RC to achieve multifunctionality

The same steps are used in training each RC to achieve multifunctionality and are outlined as follows.

For multifunctionality, we require the same $\mathbf{W}_{out}$ to hold for $\psi(\hat{\mathbf{r}}_{\mathcal{S}_1}(t)) \approx \mathbf{u}_{\mathcal{P}_1}(t)$ and $\psi(\hat{\mathbf{r}}_{\mathcal{S}_2}(t)) \approx \mathbf{u}_{\mathcal{P}_2}(t)$ for $t > t_{train}$ in the CT-RC and $i > i_{train}$ in the LI and NG-RCs . $\hat{\mathbf{r}}_{\mathcal{S}_1}$ and $\hat{\mathbf{r}}_{\mathcal{S}_2}$ describe the state of each RC on the coexisting attractors, $\mathcal{S}_1$ and $\mathcal{S}_2$, which are the RC's representation of the time series described by $\mathbf{u}_{\mathcal{P}_1}$ and $\mathbf{u}_{\mathcal{P}_2}$ that each RC is required to reconstruct a coexistence of.

To do this, we generate each RCs response to $\mathbf{u}_{\mathcal{P}_1}$ and store it in $\mathbf{X}_{\mathcal{S}_1}$. The same process is repeated for $\mathbf{u}_{\mathcal{P}_2}$ to obtain $\mathbf{X}_{\mathcal{S}_2}$. These RC training data matrices are concatenated as, $\mathbf{X}_C = (\mathbf{X}_{\mathcal{S}_1}, \mathbf{X}_{\mathcal{S}_2})$, and similarly for the corresponding $\mathbf{Y}_{\mathcal{P}_1}$ and $\mathbf{Y}_{\mathcal{P}_2}$ to obtain $\mathbf{Y}_C$. $\mathbf{W}_{out}$ is calculated using Eq. 2 for $\mathbf{X} = \mathbf{X}_C$ and $\mathbf{Y} = \mathbf{Y}_C$.

As all RCs are trained using Eq. 2, in Sec. IV we compare the performance of each RC for different values of β.

III. Numerical experiments and analysis tools

In this section we outline the specifics of each numerical experiment used to test the limits of multifunctionality.

A. Coexisting chaotic attractors

We train each RC to provide a coexistence of the chaotic Lorenz attractor, $\mathcal{L}$, described by,

$$\begin{aligned} \dot{x} &= 10(y-x), \\ \dot{y} &= x(28-z) - y, \\ \dot{z} &= xy - \frac{8}{3}z + x, \end{aligned} \tag{11}$$

and the chaotic Halvorsen attractor, $\mathcal{H}$, described by,

$$\begin{aligned} \dot{x} &= -1.3x - 4y - 4z - y^2, \\ \dot{y} &= -1.3y - 4z - 4x - z^2, \\ \dot{z} &= -1.3z - 4x - 4y - x^2. \end{aligned} \tag{12}$$

This pairing of attractors was studied in [5] when investigating the relationship between symmetry and 'mirror attractors'.

The training data is obtained by generating solutions of Eqs. 11-12 using the 4^{th} order Runge-Kutta method with time step $\tau = 0.01$. Trajectories on $\mathcal{L}$ and $\mathcal{H}$ are normalised such that the furthest point away from the origin on each attractor is less than 1. In our numerical experiments we move both normalised attractor data sets equidistantly in opposite directions along the z-axis with shift parameter, δz.

B. The 'seeing double' problem

For the second task we consider training each RC to reconstruct trajectories on two completely overlapping circular orbits rotating in opposite directions. We call this paradigmatic multifunctionality task, the 'seeing double' problem.

We generate the respective input sequences for the RC with,

$$\mathbf{u}(t) = \begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = \begin{pmatrix} c_x \cos(t) \\ c_y \sin(t) \end{pmatrix}. \tag{13}$$

We use Eq. 13 to construct a time-series which resembles a trajectory around a circle of radius $c = |c_x| = |c_y|$ and centered at $(0,0)$. Two sets, $\mathcal{C}_A$ and $\mathcal{C}_B$, are produced with Eq. 13 and the corresponding input time-series are denoted by $\mathbf{u}_{\mathcal{C}_A}$ and $\mathbf{u}_{\mathcal{C}_B}$. To create $\mathcal{C}_A$ we set $c_x = c_y = 5$ and for $\mathcal{C}_B$ we set $c_x = -5$ and $c_y = 5$.

C. Analysis tools

In Sec. IV we use the following analysis tools to assess performance of the RCs in the task described in Sec. III-B. As the data points describing both circles are effectively the same, the RC needs to have a sufficient memory of its previous state in order to remain on the correct circular orbit. To measure a given RCs memory we make use of the memory capacity metric introduced in [10].

1) Roundness: To determine whether a given RC achieves multifunctionality, we examine the predictions of the trained RC with an error metric called the 'roundness'. For both cycles, we first determine if the prediction of a given cycle is indeed a periodic, and then if it is rotating in the correct direction. Following this we compute the roundness as the difference between the radius of the largest and smallest circle needed to enclose and inscribe the predicted cycle. If the maximum roundness of both roundness values is less than a threshold value $= 0.5$ (determined from empirical testing), then we say the RC has achieved multifunctionality.

2) Memory capacity: We use the 'short-term memory' (STM) [10] to assess if, in this sense, a given RCs memory is critical to achieving multifunctionality.

The STM characterises the capability of a RC to remember inputs from the past. It is measured by training the RC to fit an input signal, $\mu[n]$, at time n to its time shifted signal, $\mu[n-j]$,

with each point in μ chosen from a uniformly random i.i.d of real numbers in $[-1, 1]$. Instead of 'closing the loop' after training, the RC is driven with the input signal μ. The STM is then calculated as the square of the correlation between the reservoir output $\mathbf{W}^j_{out} r[n]$ and the true values given by $\mu[n-j]$ summed over all j,

$$\text{STM} = \sum\nolimits_j cor^2\Big(\mu[n-j], \mathbf{W}^j_{out} r[n]\Big). \tag{14}$$

W^j_{out} depends on j since for every j the training process needs to be repeated. Note that, $\mathbf{M}$, is shared by the RC used during prediction and the RC used to measure the STM while the input and output matrices are different.

3) Floquet multipliers: Given the periodic nature of the seeing double problem, by computing the Floquet multipliers of the RC we can determine whether or not a given configuration of the RC can support the coexistence of both $\mathcal{C}_A$ and $\mathcal{C}_B$ after the training. The Floquet multipliers are the eigenvalues of the 'monodromy matrix', $\boldsymbol{Q}$, which is the solution of,

$$\dot{\boldsymbol{Q}}(t) = \boldsymbol{J}(t)\boldsymbol{Q}(t), \quad \boldsymbol{Q}(0) = \mathbf{I}, \tag{15}$$

after one period, T, of the RCs response to a given $\mathbf{u}(t)$ for one period, T, during the training. Here $\boldsymbol{J}(t)$ is the Jacobian matrix of the predicting RC and $\mathbf{I}$ is the identity matrix.

For a given driving input signal, if the absolute value of any Floquet multiplier, λ_i, is > 1 then the limit cycle in $\mathbb{S}$ is unstable or, if the absolute value of the largest Floquet multiplier, λ_1, is 1 and all other λ_i's have absolute value < 1 then a given limit cycle is stable.

IV. Results

The training parameters used to generate all the results shown in this section are given in Tables I-III in the Appendix.

A. Reconstructing Lorenz and Halvorsen

In this section we discuss the results shown in Fig. 1 where each RC is trained on the task described in Sec. III-A.

In Fig. 1 we see that all RCs reconstruct a coexistence of $\mathcal{L}$ and $\mathcal{H}$ for $0.75 \leq \delta \leq 1.25$. However, there are some noticeable differences in the performance of each RC as the attractors are brought closer together and further apart. For the set of training parameters specified in the Appendix, no RC is capable of reconstructing a coexistence of $\mathcal{L}$ and $\mathcal{H}$ for all δz's and in particular for $\delta z = 0$. The NG-RC is unable to reconstruct both attractors in the case of overlapping data, the CT and LI-RC are able to provide reasonable reconstruction of both $\mathcal{L}$ and $\mathcal{H}$ for $\delta z = 0.25$ where large parts of the attractors are already overlapping. This relationship between multifunctionality and overlap is not trivial as, for instance, the CT-RC fails to achieve multifunctionality at times when there is less of an overlap between the attractors , i.e., for $\delta = 0.5$ the CT-RC only reconstructs $\mathcal{L}$ but for $\delta = 0.25$ both are reconstructed. For the CT-RC and NG-RC we see that as $\mathcal{L}$ and $\mathcal{H}$ are moved away from one another, both RCs fail to achieve multifunctionality with the CT-RCs predictions decaying to limit cycles and the NG-RC becoming unstable.

As a further comment, in our experiments we found that the CT and NG-RCs achieve multifunctionality for $\delta > 1.5$ with different training parameters. However, by keeping the parameters fixed this provides better insight towards the complex relationship between multifunctionality and overlapping training data. In the modes of failure, the main difference between the NG-RC and the 'traditional' CT and LI-RCs is that the state of the NG-RC tends to infinity whereas the CT and LI-RCs prediction decay to some other attractor.

B. Solving the seeing double problem

In this section we compare the performance of each RC on the seeing double problem and assess the effects that different training parameters have on multifunctionality.

1) LI-RC: ρ and memory capacity: In Fig. 2a we illustrate the number of times out of 100 random realisations of $\mathbf{M}$ and $\mathbf{W}_{in}$ that a given LI-RC gave rise to multifunctionality for different values of ρ as determined by the roundness analysis procedure described in Sec. III-C1. We see that for $\alpha = 0.05$ these LI-RCs outperform the LI-RCs with $\alpha = 1$ resulting in not only a broader range of ρ values where multifunctionality was achieved but also a greater probability of success.

In order to determine if the RCs memory as measured with the STM approach discussed in Sec. III-C2 plays an important role in a given RCs ability to achieve multifunctionality, in Figs. 2b-2d we compute the STM for ten LI-RCs which achieve multifunctionality and for ten LI-RCs that do not at different ρ values chosen from Fig. 2a. While there is no significant difference in the STM amongst successful or unsuccessful realisations of the LI-RCs, there is a significant difference in the STM for $\alpha = 1$ compared to LI-RCs with $\alpha = 0.05$. Given that the LI-RC performed much better for $\alpha = 0.05$ as shown in Fig. 2a, we see here in Figs. 2b-2d that there is no correlation between multifunctionality and STM.

2) CT-RC: ρ and Floquet multipliers: In this section we examine the behaviour of the CT-RC for different values of ρ and determine that the Floquet multipliers, as discussed in Sec. III-C3 is a more suitable analysis tool than the STM to distinguish between whether a given realisation of the CT-RC can given rise to multifunctionality in this scenario.

Similarly to Fig. 2a, in Fig. 3a we illustrate the number of times out of 100 that a given random realisation of the CT-RC gave rise to multifunctionality for different values of ρ.

In contrast to the LI-RCs with $\alpha = 0.05$, these CT-RCs with $\gamma = 5$ are able to achieve multifunctionality with a success ratio $> 50\%$ for a smaller range of ρ values but still performs much better than the LI-RCs with $\alpha = 1$. Furthermore, from Figs. 3b-3d we see that there is a direct correlation between whether a given CT-RC gave rise to multifunctionality and its Floquet multipliers for different values of ρ.

3) CT & LI RCs: γ, α, and β: In this section we restrict our analysis to $\rho = 1.4$ where we found reasonable performance for both the CT and LI-RCs in order to determine the influence that different values of α, γ, and β, have on multifunctionality.

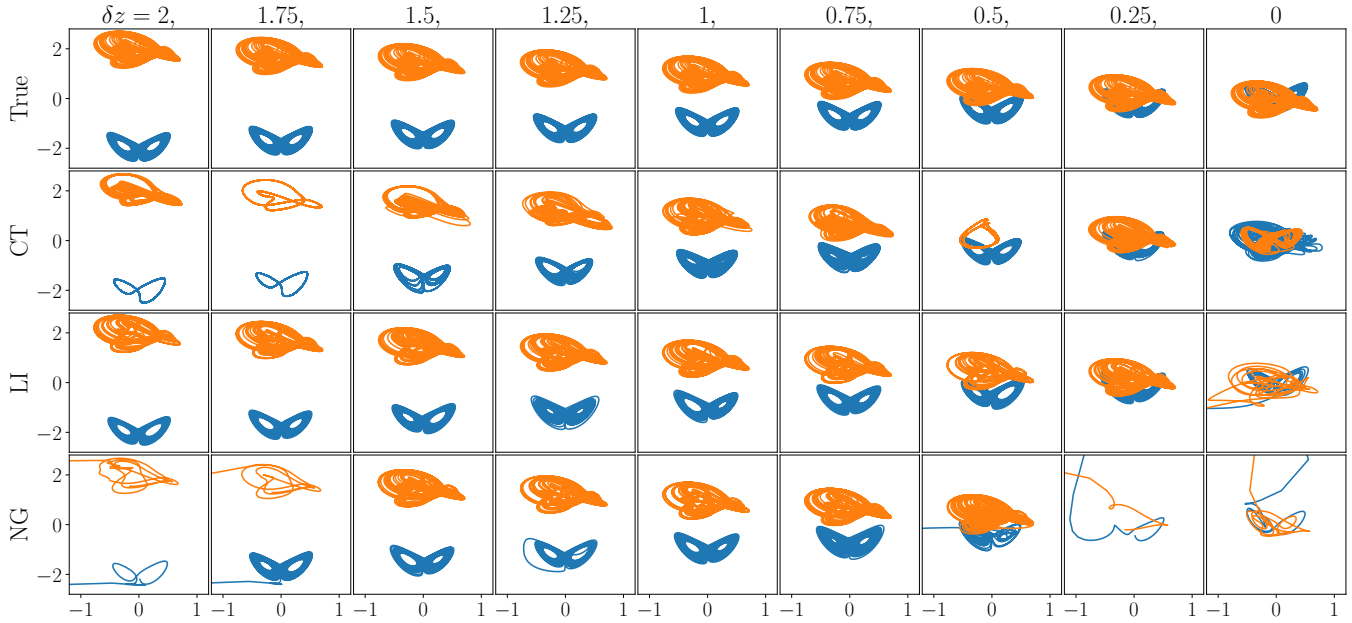
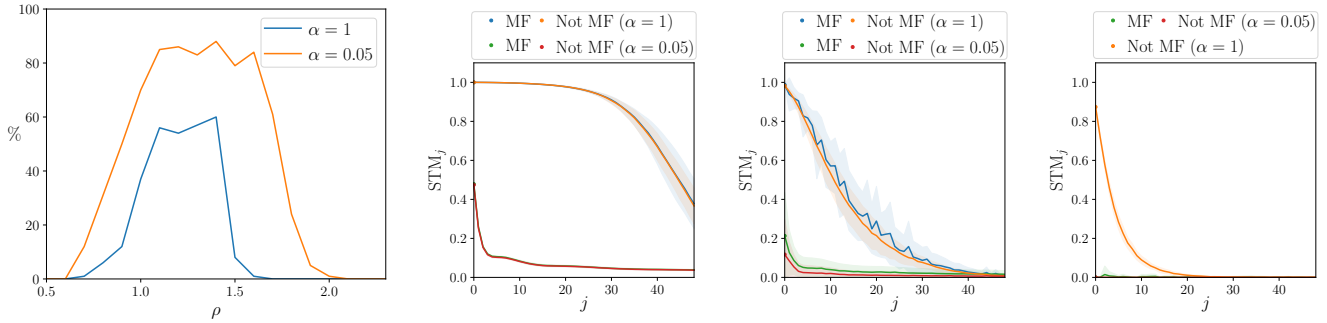
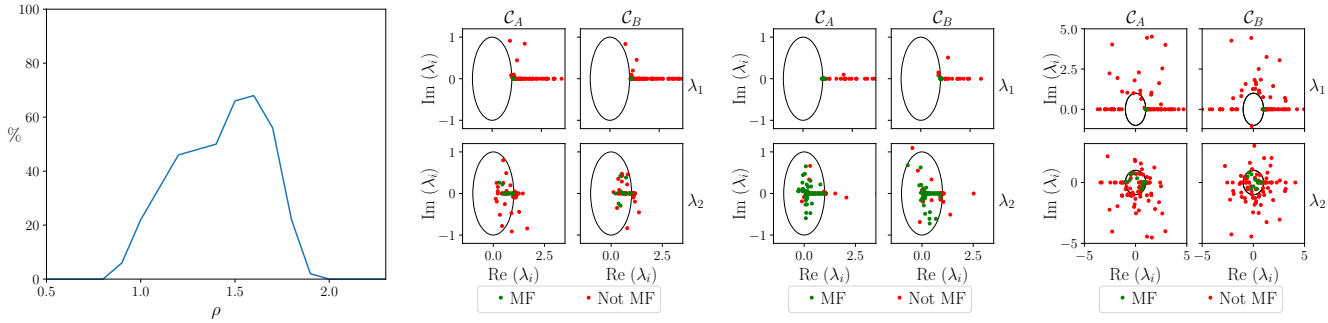


Fig. 1: Comparison of CT (continuous-time), LI (leaky-integrator), NG (next-generation) RCs performance when trained to provide of a coexistence of the Lorenz attractor (blue) and the Halvorsen attractor (orange) when separated by a factor of δz away from the origin in the (x, z)-plane.



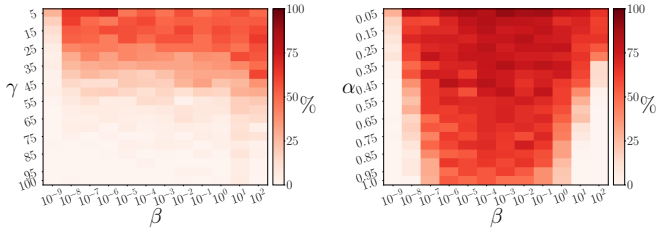
(a) ρ vs. Success (b) $\rho = 1.0$ (c) $\rho = 1.4$ (d) $\rho = 1.8$

Fig. 2: LI-RC: (a) ρ vs. number of times out of 100 that a given random realisation of $\mathbf{M}$ and $\mathbf{W}_{in}$ gave rise to multifunctionality. (b)-(d) Short-term memory capacity for RCs that fail and succeed to solve the seeing double problem with error bars given by the standard deviation. j is the time shift used in calculating the STM. MF describes when multifunctionality was achieved.



(a) ρ vs. Success (b) $\rho = 1.0$ (c) $\rho = 1.4$ (d) $\rho = 1.8$

Fig. 3: CT-RC: (a) ρ vs. number of times out of 100 that a given random realisation of $\mathbf{M}$ and $\mathbf{W}_{in}$ gave rise to multifunctionality. (b)-(d) Corresponding two largest Floquet multipliers, λ_1 and λ_2, for both $\mathcal{C}_A$ and $\mathcal{C}_B$ for these $\mathbf{M}$ and $\mathbf{W}_{in}$ at specified ρ. MF describes when multifunctionality was achieved.



(a) CT-RC (b) LI-RC

Fig. 4: Success rate in the (β, γ)-plane and the (β, α)-plane for 100 random realisations of the CT and LI-RCs.

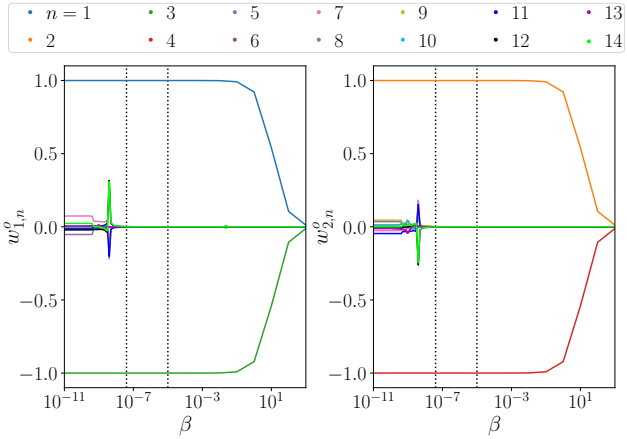


Fig. 5: β vs. $w^o_{i,j}$, the elements of $\mathbf{W}_{out}$ for the NG-RC setup

In Fig. 4 we plot the number of times out of 100 that a given pair of (β, α) or (β, γ) values give rise to multifunctionality using the error analysis technique as described in Sec. III-C1.

In Figs. 4a and 4b we see that the limit of $\alpha = \gamma\tau$ for $\tau \to 0$ also holds for only small values of α and γ. Furthermore, in Fig. 4a we see no significant improvement in the performance of the CT-RC for different β and γ values. In particular, for $\gamma > 40$ the majority of random realisations of the CT-RCs will not give rise to multifunctionality. Whereas for the LI-RC in Fig. 4b, its best performance occurs for the same $\alpha \approx 0.05$. However there is a much wider range of α and β values where over 70% of the tested LI-RCs achieve multifunctionality.

What is common between both Figs. 4a and 4b is the role of β. If β is too small then both RCs have a success rate $< 5\%$ and while larger β values prevent overfitting, if β is too large this also prevents the RCs from learning the correct dynamics.

4) NG-RC: In this section we discuss the results of training the NG-RC on the seeing double task with $O = [1, 2], k = 2, s = 1$. Note the square readout function, $\mathbf{q}(\cdot)$, is not used to obtain $\mathbf{W}_{out}$. As a result, the trained $\mathbf{W}_{out}$ is $\in \mathbb{R}^{2\times 14}$, and in Fig. 5 we plot each element, $w^o_{m,n}$, vs. β.

As indicated by the vertical dotted lines in Fig. 5 we find that the NG-RC only achieves multifunctionality within this range of β values. Here there are four $\mathbf{W}_{out}$ elements, $w^o_{1,1} = 0.99989995, w^o_{1,3} = -0.99999995, w^o_{2,2} = 0.99990005$, and $w^o_{2,4} = -1.00000005$ which are $\not\approx 0$. For $\beta < 4 \times 10^8$, the

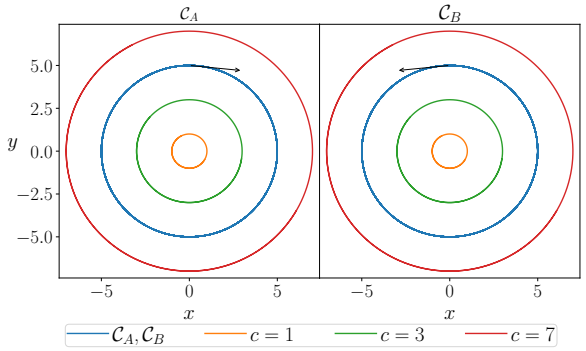


Fig. 6: Trained NG-RC output dynamics

state of the NG-RC tends to infinity, and for $\beta > 10^5$ the system tends to the fixed point at the origin.

Given the architecture of the NG-RC we are able to write the learned equations as,

$$x(t+1) = x(t) + \Delta x(t) = (1 + w^o_{1,1})x(t) + w^o_{1,3}x(t-1),$$
$$y(t+1) = y(t) + \Delta y(t) = (1 + w^o_{2,2})y(t) + w^o_{2,4}y(t-1).$$

Here we see that these governing equations are uncoupled and linear, like the driving input described in Eq. 13. As a consequence, the NG-RC has not reconstructed limit cycles but has produced a set of equations that can mimic the input training data. Furthermore, in Fig. 6 we show that by initialising the trained NG-RC with two consecutive points on a circle it produces circular trajectories of any given radius.

It's also important to note that despite, $w^o_{1,1} \approx 1, w^o_{1,3} \approx -1, w^o_{2,2} \approx 1$, and $w^o_{2,4} \approx -1$, the NG-RC becomes unstable if the values 1 and -1 are used for the weights.

V. CONCLUSIONS

In this paper we investigate some of the limits of multifunctionality in CT, LI, and NG RCs when overlapping training data is introduced in different tasks.

As illustrated in Fig. 1, we find that for a given set of training parameters, each RC can reconstruct a coexistence of the chaotic Lorenz and Halvorsen attractors when the data is sufficiently separated in state space, however if the attractors are too far apart then we see the CT and NG RCs begin to fail. For the same set of training parameters, the NG-RC is unable to reconstruct a coexistence of these attractors once they begin to overlap while the CT and LI RCs are able to achieve multifunctionality when the attractors share mutual regions of state space up to a certain extent.

In order to further explore the limits of multifunctionality, we investigate the performance of each RC when trained to solve the seeing double problem. As shown in Figs. 2a and 3a, it is clear that ρ, the parameter associated with memory in the CT and LI RCs, plays an important role in whether a given RC can achieve multifunctionality. To measure the RCs memory we use the STM metric as described in Sec. III-C2, and in Figs. 2b-2d, we identify the shortcomings of the STM metric as a means to successfully capture the role of memory

in this sense. However, by using the Floquet analysis described in Sec. III-C3 we are able to identify in Figs. 3b-3d that the effect of 'closing the loop' is of greater significance to whether a given RC can achieve multifunctionality. Furthermore, we find that despite choosing a NG-RC with polynomial terms, the trained NG-RC results in a uncoupled set of linear equations which can generate circular trajectories when initialised with two consecutive on any given circle.

The defining characteristic of the reservoir computing approach to machine learning is the need to train only a suitable readout layer, $\mathbf{W}_{out}$, to solve a given problem. However, it has only recently been discovered that a given $\mathbf{W}_{out}$ can enable a RC to perform more than one task. In this context, other than the results regarding multifunctionality, RCs have been trained to infer unseen attractors, learn global bifurcation structures and anticipate synchronisation [20]–[23].

Multifunctionality opens up new application areas for RCs, for instance, in producing data-driven models of real world phenomenon where multistability is thought to play a role, like in the epileptic brain [24]. However, many questions remain, in particular, how much dynamical functionality a single RC can be trained to exhibit. In future work we aim to address this with further RC designs in other paradigmatic scenarios given the insight gained through the seeing double problem.

APPENDIX

$\mathbf{M} \in \mathbb{R}^{N\times N}$ has a sparse Erdös–Renyi network where each element is chosen independently with probability p from a uniform distribution of $(-1, 1)$. After initialisation, the spectral radius, ρ, of $\mathbf{M}$ is tuned in order to provide the network with a sufficient amount of memory. $\mathbf{W}_{in} \in \mathbb{R}^{N\times D}$ is designed such that each row has only one nonzero randomly assigned element, chosen uniformly from $(-1, 1)$.

REFERENCES

[1] A. Flynn, V. A. Tsachouridis, and A. Amann, "Multifunctionality in a reservoir computer," *Chaos*, **31**, 1, 2021.

[2] D. M. Wolpert and M. Kawato, "Multiple paired forward and inverse models for motor control," *Neural networks*, **11**, 7-8, 1998.

[3] H. Jaeger, "Using conceptors to manage neural long-term memories for temporal patterns," *The Journal of Machine Learning Research*, **18**, 1, 2017.

[4] K. L. Briggman and W. Kristan Jr, "Multifunctional pattern-generating circuits," *Annu. Rev. Neurosci.*, **31**, 2008.

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

[6] A. Flynn, J. Herteux, V. A. Tsachouridis, C. Räth, and A. Amann, "Symmetry kills the square in a multifunctional reservoir computer," *Chaos*, **31**, 7, 2021.

[7] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," *Chaos*, **28**, 6, 2018.

[8] H. Jaeger, M. Lukoševičius, D. Popovici, and U. Siewert, "Optimization and applications of echo state networks with leaky-integrator neurons," *Neural networks*, **20**, 3, 2007.

[9] D. J. Gauthier, E. Bollt, A. Griffith, and W. A. Barbosa, "Next generation reservoir computing," *Nature Communications*, **12**, 1, 2021.

[10] H. Jaeger, "Short term memory in echo state networks," *GMD-German National Research Institute for Computer Science*, 2002.

[11] H. Jaeger, "The 'echo state' approach to analysing and training recurrent neural networks-with an erratum note," *German National Research Center for Information Technology*, **148**, 01, 2001.

Fig	N	p	ρ	σ	γ	β	t_l	t_t
1	1000	0.05	1.6	5	7	10^2	100	200
3	500	0.05	Fig	0.2	5	10^{-2}	$6T$	$15T$
4a	500	0.05	1.4	0.2	Fig	Fig	$6T$	$15T$

TABLE I: CT-RC training parameters in the specified figures. $t_l = t_{listen}$, $t_t = t_{train}$, $T =$ period.

Fig	N	p	ρ	σ	α	β	t_l	t_t
1	1000	0.012	0.9	1.2	0.2	10^{-3}	100	200
2	500	0.05	Fig	0.2	Fig	10^{-2}	$6T$	$15T$
4b	500	0.05	1.4	0.2	Fig	Fig	$6T$	$15T$

TABLE II: LI-RC training parameters in the specified figures. $t_l = t_{listen}$, $t_t = t_{train}$, $T =$ period.

Fig	O	k	s	β	t_w	t_t
1	$1, 2, 3, 4, 5$	3	2	3×10^{-5}	6	200
5-6	$1, 2$	2	1	Fig	2	$15T$

TABLE III: NG-RC training parameters in the specified figures. $t_w = t_{warm}$, $t_t = t_{train}$, $T =$ period.

[12] W. Maass, T. Natschläger, and H. Markram, "Real-time computing without stable states: A new framework for neural computation based on perturbations," *Neural computation*, **14**, 11, 2002.

[13] D. Verstraeten, B. Schrauwen, M. d'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," *Neural networks*, **20**, 3, 2007.

[14] L. Appeltant, M. C. Soriano, G. Van der Sande, J. Danckaert, S. Massar, J. Dambre, B. Schrauwen, C. R. Mirasso, and I. Fischer, "Information processing using a single dynamical node as complex system," *Nature communications*, **2**, 1, 2011.

[15] C. Cuchiero, L. Gonon, L. Grigoryeva, J.-P. Ortega, and J. Teichmann, "Discrete-time signatures and randomness in reservoir computing," *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[16] T. L. Carroll and L. M. Pecora, "Network structure effects in reservoir computers," *Chaos*, **29**, 8, 2019.

[17] C. Gallicchio, A. Micheli, and L. Pedrelli, "Deep reservoir computing: a critical experimental analysis," *Neurocomputing*, **268**, 2017.

[18] K. Nakajima, "Physical reservoir computing—an introductory perspective," *Japanese Journal of Applied Physics*, **59**, 6, 2020.

[19] E. Bollt, "On explaining the surprising success of reservoir computing forecaster of chaos? the universal machine learning dynamical system with contrast to var and dmd," *Chaos*, **31**, 1, 2021.

[20] A. Röhm, D. J. Gauthier, and I. Fischer, "Model-free inference of unseen attractors: Reconstructing phase space features from a single noisy trajectory using reservoir computing," *Chaos*, **31**, 10, 2021.

[21] J. Z. Kim, Z. Lu, E. Nozari, G. J. Pappas, and D. S. Bassett, "Teaching recurrent neural networks to infer global temporal structure from local examples," *Nature Machine Intelligence*, **3**, 4, 2021.

[22] H. Fan, L.-W. Kong, Y.-C. Lai, and X. Wang, "Anticipating synchronization with machine learning," *Physical Review Research*, **3**, 2, 2021.

[23] M. Goldmann, C. R. Mirasso, I. Fischer, and M. C. Soriano, "Inferring untrained complex dynamics of delay systems using an adapted echo state network," *arXiv preprint arXiv:2111.03706*, 2021.

[24] Suffczynski, P., Kalitzin, S., and da Silva, F. L. (2004) Dynamics of non-convulsive epileptic phenomena modeled by a bistable neuronal network. *Neuroscience,* **126**(2), 467–484.