Learn to Synchronize, Synchronize to Learn

Preprint · October 2020

CITATIONS
0

READS
298

3 authors:

Pietro Verzelli
University of Lugano
9 PUBLICATIONS **15** CITATIONS
SEE PROFILE

Cesare Alippi
Politecnico di Milano
293 PUBLICATIONS **4,780** CITATIONS
SEE PROFILE

Lorenzo Livi
University of Manitoba
92 PUBLICATIONS **1,073** CITATIONS
SEE PROFILE

Learn to Synchronize, Synchronize to Learn

Pietro Verzelli, Cesare Alippi, and Lorenzo Livi

***Abstract*—In recent years, the machine learning community has seen a continuous growing interest in research aimed at investigating dynamical aspects of both training procedures and perfected models. Of particular interest among recurrent neural networks, we have the Reservoir Computing (RC) paradigm for its conceptual simplicity and fast training scheme. Yet, the guiding principles under which RC operates are only partially understood. In this work, we study the properties behind learning dynamical systems with RC and propose a new guiding principle based on Generalized Synchronization (GS) granting its feasibility. We show that the well-known Echo State Property (ESP) implies and is implied by GS, so that theoretical results derived from the ESP still hold when GS does. However, by using GS one can profitably study the RC learning procedure by linking the reservoir dynamics with the readout training. Notably, this allows us to shed light on the interplay between the input encoding performed by the reservoir and the output produced by the readout optimized for the task at hand. In addition, we show that - as opposed to the ESP – satisfaction of the GS can be measured by means of the Mutual False Nearest Neighbors index, which makes effective to practitioners theoretical derivations.**

***Index Terms*—Reservoir Computing, Echo State Property, Dynamical Systems, Chaos Synchronization**

I. Introduction

In recent years, the scientific community has seen a raising interest in research aimed at coupling machine learning and dynamical systems. In fact, investigations have shown how the theory developed for dynamical systems was found to be useful to understand machine learning algorithms [23, 5, 45, 3]; the opposite holds as well, e.g. see [9, 39, 15, 50, 4].

Within machine learning the Reservoir Computing (RC) paradigm [51, 28] is particularly appealing due to its simplicity, cheap training mechanism and state-of-the-art results obtained in solving various tasks [27, 12, 54, 8]. RC was introduced independently by Jaeger [21] (who used the term Echo State Network), Maass, Natschläger, and Markram [30] (Liquid State Machine) and Tiňo and Dorffner [49] (Fractal Predicting Machine). In order to account for Neural Network implementation of RC we will use the term Reservoir Computing Network (RCN) in the sequel. The working principle of RC is based on creating a representation of the input sequence under investigation by feeding it to an untrained dynamical system, *the reservoir*, which should encode all relevant dynamics. The learning focuses on the *readout* function, which is trained to generate the desired output.

Pietro Verzelli is with the Faculty of Informatics, Università della Svizzera Italiana, Lugano, 69000, Switzerland. Referring author at: pietro.verzelli@usi.ch

Cesare Alippi is with the Faculty of Informatics, Università della Svizzera Italiana, Lugano, 69000, Switzerland and Department of Electronics, Information and bioengineering, Politecnico di Milano, Milan, 20133, Italy.

Lorenzo Livi is with the Departments of Computer Science and Mathematics, University of Manitoba, Winnipeg, MB R3T 2N2, Canada and Department of Computer Science, College of Engineering, Mathematics and Physical Sciences, University of Exeter, Exeter EX4 4QF, United Kingdom.

Some recent efforts have been devoted to understanding the learning mechanisms of RC and its capability to approximate dynamical systems. In particular, it was proven that RCN are a universal function approximators [18] and that their representations are rich enough to correctly embed dynamical systems in their state-space [20, 19]. Theoretical analysis of this learning principle lead to many results about their expressive power [16, 33, 34, 40, 52]. Moreover, interesting results can be derived when assuming linear dynamics [17, 32, 48, 53, 14]. Due to its simple training mechanism, the RC approach is also particularly appealing for neuromorphic computing and other hardware implementations; see [47] for a recent review on the topic.

The Echo State Property (ESP) was introduced in the seminal work by Jaeger [21] as a necessary tool for an effective computation. Basically, ESP consists in requiring that asymptotically the reservoir state depends only on the received input (i.e., the reservoir state *echoes* the input) and does not depend on the initial conditions of the reservoir. Notably, even though most theoretical results assume the ESP to hold [18, 20], the sufficient conditions are too restrictive [56] to be used in practical applications and the necessary conditions seem to suffice in most cases [57, 2]. In practice, some less restrictive criteria to verify satisfaction of the ESP have been proposed over time [56, 31, 10, 51] as well as a general formulation for the ESP accounting for multiple, stable responses to a driving input sequence [11]. Yet, the problem with the ESP verification lies on the fact that the ESP definition does not explicitly take into account the structure of the driving input, which is simply defined as a sequence of values in an admissible range. As a consequence, satisfaction of ESP cannot be verified but in simple cases for which the mathematics is amenable.

In order to verify the ESP, we propose a new method based on system synchronization. In recent years, the concept of synchronization was applied to the study of RC to obtain some interesting results [26, 55, 29]. The possibility of generalizing the concept of synchronization was first investigated by Afraimovich, Verichev, and Rabinovich [1] and Rulkov et al. [42], who introduced the term Generalized Synchronization (GS). Successively, different empirical methods for verifying the presence of GS from data have been introduced [37, 35]. A review on synchronization between dynamical systems was recently published in [7].

Even if the equivalence between ESP and GS is often taken for granted, to the best of our knowledge, this equivalence has never been properly formalized. The scope of this work is then two-fold. On one hand, we aim at properly characterizing such equivalence. On the other, we demonstrate that GS allows one to analyze RC in a novel way, shedding light on its working principles. In particular, we show that, under the assumption that there exists a dynamical system (called a source system)

generating the data, the following equivalence holds: requesting the ESP for the reservoir w.r.t. a driving input sequence is equivalent to requiring the GS between the reservoir and the (unknown) source system. This implies the existence of a stable *synchronization manifold* to which the system will converge to, and of a *synchronization function* mapping states of the source system onto states of the reservoir. By making use of these important implications, one can interpret the RC framework under a new light and obtain important results. More specifically:

- GS allows to explicitly link the dynamics introduced by the reservoir (listening phase) with the readout mapping (fitting phase), highlighting the relation between the encoding of the input in the reservoir dynamics and the performance of the RCN on the task at hand;
- we show that, for learning to be feasible, there must exist a function linking the dynamics of the source system with the ones of the reservoir. The synchronization function appears as a natural candidate for this role;
- moreover, we show that the presence of GS allows to exploit the ergodicity of the source system for estimating the error on the whole attractor underlying the source system.
- Finally, GS can be easily quantified for an RCN driven by an input sequence, thus allowing one to assess the degree to which GS holds in practice. For this we use an index, called the Mutual False Nearest Neighbors (MFNN), and empirically show that it is correlated with the RC performance on the tasks at hand.

The rest of the paper is organized as follows: in Sec. II we introduce the theoretical framework, discussing the task we aim at solving and how this can be done with RC. In Sec. III we present the concept of synchronization for dynamical system and formalize the similarities with the concept of ESP. Sec. IV is devoted to exploiting the GS to derive theoretical results concerning RC. In Sec. V we carry out simulations to validate the developed theory. Finally, we draw conclusions in Sec. VI. The paper contains three appendices located at the end of the manuscript.

II. Reservoir computing

In this section we introduce the RC framework, where a dynamical system, called a *reservoir*, is driven by an input sequence generated by an unknown source system and it is used to generate a desired output. We will adopt the terminology used in [28] and introduce the topic in the most abstract sense, disregarding the particular form of the source system or the reservoir.

A. The task

In our framework, we consider a discrete-time autonomous, noise-free system model described by:

$$\boldsymbol{s}(t+\tau) = \boldsymbol{g}(\boldsymbol{s}(t)) \tag{1}$$

where t indicates the discrete time, τ the *time increment*, $\boldsymbol{s}(t) \in \mathbb{R}^{d_s}$ denotes d_s-dimensional system *state*. We will refer to (1) as the *source system* because it is the source generating our data. We assume $\boldsymbol{g}$ to be differentiable and invertible, and that $\boldsymbol{s}(t)$ approaches and stays on a bounded attractor, $\mathcal{A}_s$. We are interested in the situation where we do not know $\boldsymbol{g}$ and do not have direct access to the source system states.

The source system (1) produces two outputs, $\boldsymbol{u}(t) \in \mathbb{R}^{d_u}$ and $\boldsymbol{y}(t) \in \mathbb{R}^{d_y}$, as follows:

$$\boldsymbol{u}(t) = \boldsymbol{h}(\boldsymbol{s}(t)) \tag{2a}$$

$$\boldsymbol{y}(t) = \boldsymbol{k}(\boldsymbol{s}(t)) \tag{2b}$$

We name $\boldsymbol{u}$ as the *measurements* (or *observables*), i.e. the available data to model the task. The vector-valued function $\boldsymbol{h}(\cdot)$ is introduced to account for the fact that a function of $\boldsymbol{s}$ is used to generate the data. We refer to $\boldsymbol{y}$ as the *targets*, i.e., the supervised information describing the task to be learned. The targets are generated via a vector-valued function $\boldsymbol{k}(\cdot)$. Both $\boldsymbol{h}(\cdot)$ and $\boldsymbol{k}(\cdot)$ are unknown. We assume that there is no measurement noise, as commonly done in the related literature [28, 28, 20, 19].

Both $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$ are accessible for $t < 0$, but for $t \geq 0$ only $\boldsymbol{u}(t)$ is available. Moreover, we call *training phase* the period $t < 0$, when both $\boldsymbol{u}(t)$ and $\boldsymbol{y}(t)$ are available. Our goal is then to use the continued knowledge about $\boldsymbol{u}$ to generate a valid prediction $\hat{\boldsymbol{y}}(t)$ of $\boldsymbol{y}(t)$, for $t \geq 0$. We call this phase the *predicting phase*.[1] Our goal is then to use the continued knowledge of $\boldsymbol{u}$ to generate a valid prediction of the target $\hat{\boldsymbol{y}}$ for $t > 0$. Figure 1 provides an example of the framework taken into account.

A typical instance of this problem is the forecasting task, say to predict the value $\boldsymbol{u}(t)$ will assume d times ahead, hence providing $\boldsymbol{y}(t) = \boldsymbol{u}(t+d)$. Another relevant task (called the *observer task* [28]) requires to estimate the state of the system having information about $\boldsymbol{u}(t)$ only, i.e. $\boldsymbol{y}(t) = \boldsymbol{s}(t)$. An example of the framework is provided in Fig. 1.

B. Training

For the training phase, we assume to have access to a (possibly infinite) series of measurements $\boldsymbol{u}(t)$ and a paired series of target values $\boldsymbol{y}(t)$. The goal of the training phase is to produce a function which generates an accurate prediction $\hat{\boldsymbol{y}}(t)$ of the target when reading $\boldsymbol{u}(t)$. The problem lies on the fact that the target values $\boldsymbol{y}$ depend on the full state of the source $\boldsymbol{s}$, while only the measurements $\boldsymbol{u}$ are accessible. So, one needs to be able to represent the full state of the source from the measurements only and then use it to estimate the target function. In the RC approach, these two parts are explicitly separated. To represent the full state, one uses a different dynamical system, *the reservoir*, which creates a meaningful representation of the source system $\boldsymbol{s}$ when driven by the measurements $\boldsymbol{u}$. We will call this part the *listening* phase. Then, a function must be used to compute the desired output from the reservoir states. This is done by estimating a *readout function*, which takes a reservoir state as input and produces an output. This phase is called the *fitting* phase.

[1] We choose this term – following [28] – to avoid the possible ambiguity between the testing and validation phases typically used in machine learning tasks, since this distinction is not well-defined in this context.

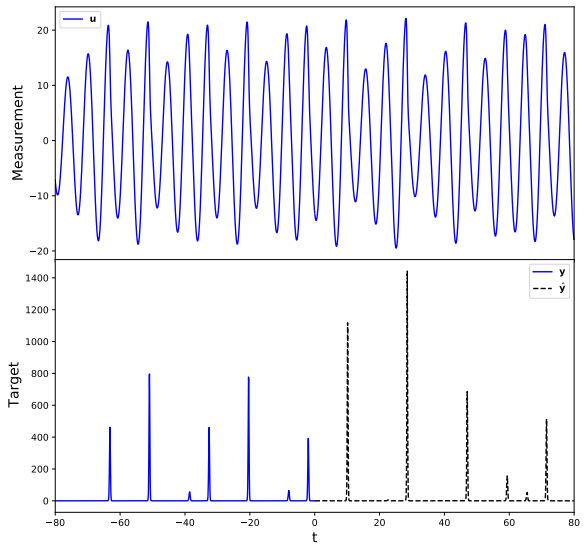

Fig. 1. An example of the problem under study, where both $\boldsymbol{u}$ and $\boldsymbol{y}$ are uni-dimensional. The input value $\boldsymbol{u}$ is always provided (top figure), while the target $\boldsymbol{y}$ is only accessible for $t < 0$ (bottom figure, blue solid line). The goal is to generate a prediction $\hat{\boldsymbol{y}}$ for $t > 0$. Here, the source system is the Rössler system (see Appendix B), the input is $u(t) = x(t)$ while the target is $y(t) = z^2(t)$, where $x(t)$ and $z(t)$ are two variables constituting the Rössler system.

a) Listening: In the *listening* phase, the training measurements are used as input to the reservoir, which is modelled as a discrete-time[2] deterministic dynamical system:

$$\boldsymbol{r}(t+\tau) = \boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{u}(t+\tau)) \tag{3}$$

Here $\boldsymbol{r}(t) \in \mathbb{R}^{d_r}$ is the *reservoir state*. We assume $\boldsymbol{f}$ to be a differentiable function controlling the reservoir evolution.

b) Fitting: Fitting consists in determining the so-called *readout function*, $\boldsymbol{\psi_\theta}$, which reads the reservoir state $\boldsymbol{r}(t)$ and estimates the output $\boldsymbol{y}(t)$. The parameters $\boldsymbol{\theta}$ are selected by a fitting procedure on the training data and the corresponding sequence of reservoir states $\boldsymbol{r}(t)$ are generated during the listening phase. Thus, the goal of the fitting phase is to determine a parameter configuration $\hat{\boldsymbol{\theta}}$ such that:

$$\hat{\boldsymbol{y}}(t) := \boldsymbol{\psi}_{\hat{\boldsymbol{\theta}}}(\boldsymbol{r}(t)) \approx \boldsymbol{y}(t) \tag{4}$$

From now on we will drop the $\hat{\boldsymbol{\theta}}$-notation and we will simply write $\boldsymbol{\psi}$ for $\boldsymbol{\psi}_{\hat{\boldsymbol{\theta}}}$.

It is important to point out that this is an offline learning procedure: the enriched input representation is first created through the reservoir states and only then the readout function is computed in one shot. Online learning procedures may be exploited as well, e.g. see [45, 26].

[2]In fact, the theory also applies to continuous-time models. In [27] they discuss the theory for discrete-time systems, but then use a continuous-time model in the experimental part

C. Predicting

After training is complete, the system will be used to predict new target values.

So, in this phase the reservoir continues to be driven by $\boldsymbol{u}(t)$ and the new output is simply $\hat{\boldsymbol{y}}(t)$. Being the readout time-independent, and in absence of an output feedback mechanism, the reservoir is subject to the *same* dynamics during training and predicting phases, since the network continues to be driven by $\boldsymbol{u}$.[3] In other words, the dynamical part of the system does not "perceive" in any way the change between the listening and the predicting phase: this is why the ESP was introduced in this setting.

A schematic representation of this approach is depicted in Fig. 2.

III. Synchronization and Echo-State Property

In this section, we will formalize the equivalence between ESP and GS.

A. Synchronization of identical systems

We start by recalling the concept of *sensitive dependence on initial conditions*. Consider two identical m-dimensional chaotic systems, say a and b, described by:

$$\boldsymbol{x}_a(t+\tau) = \boldsymbol{F}(\boldsymbol{x}_a(t)) \tag{5a}$$
$$\boldsymbol{x}_b(t+\tau) = \boldsymbol{F}(\boldsymbol{x}_b(t)) \tag{5b}$$

where the function $\boldsymbol{F}$ is the same for both systems. If the initial conditions differ even slightly, then the chaotic nature of the system will lead to exponential divergence: the two systems posses the same attractor but their motion will be uncorrelated over time.

In this context, an instance of chaos synchronization consists of designing a coupling between the two systems such that the two trajectories, $\boldsymbol{x}_a(t)$ and $\boldsymbol{x}_b(t)$, are identical. That is, if $\boldsymbol{x}_a(t) \approx \boldsymbol{x}_b(t)$ then $\|\boldsymbol{x}_a(t) - \boldsymbol{x}_b(t)\| \to 0$ as $t \to \infty$. A possible coupling for (5) might be:

$$\boldsymbol{x}_a(t+\tau) = \boldsymbol{F}(\boldsymbol{x}_a(t)) + \boldsymbol{c}_a\,(\boldsymbol{x}_a(t) - \boldsymbol{x}_b(t)) \tag{6a}$$
$$\boldsymbol{x}_b(t+\tau) = \boldsymbol{F}(\boldsymbol{x}_b(t)) + \boldsymbol{c}_b\,(\boldsymbol{x}_b(t) - \boldsymbol{x}_a(t)) \tag{6b}$$

The $\boldsymbol{c}_a = [c_{a,1}, c_{a,2}, \ldots c_{a,m}]$ and $\boldsymbol{c}_b = [c_{b,1}, c_{b,2}, \ldots c_{b,m}]$ are the *coupling constants*. If all the c_a's are null, we say that there is one-way coupling from a to b, since the state of a influences b but b does no influence a. If $c_{a,i} \neq 0$ and $c_{b,i} \neq 0$ for at least one i, we say that there is a two-way coupling.

System in (6) is a $2m$-dimensional dynamical system resulting from the coupling of the two original systems. Note that if synchronization is achieved, $\boldsymbol{x}_a(t) = \boldsymbol{x}_b(t)$ and the coupling terms are identically zero. Note that in this example we have chosen to couple only the first component of a to b and viceversa.

[3]We implicitly assume that $\boldsymbol{u}$ is characterized by the same dynamics in both phases, implying some form a stationarity of the source system. Otherwise, the learning would be unfeasible without proper adaptation mechanisms to changes in the driving input. Namely, we are assuming that the source system (1) has reached its attractor in the listening phase and that it will continue to stay on it.

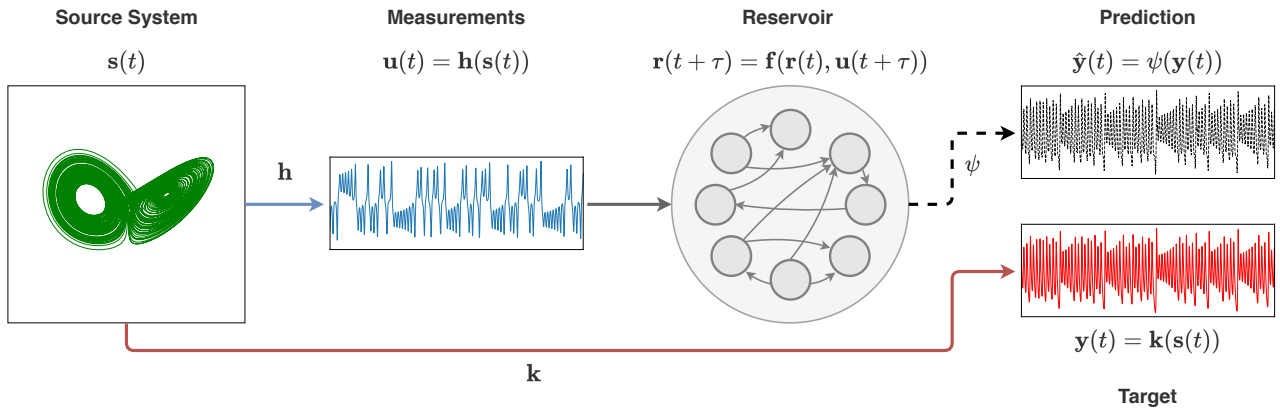

Fig. 2. Diagram representing the RC framework described in Section II. The source system $\boldsymbol{s}(t)$ evolves autonomously and generates the targets $\boldsymbol{y}(t)$ and the measurements $\boldsymbol{u}(t)$. The latter is coupled to the reservoir $\boldsymbol{r}(t)$ so that its dynamics are dependent on (i.e. driven by) $\boldsymbol{u}(t)$. The readout $\boldsymbol{\psi}$ is then trained to generate the prediction $\hat{\boldsymbol{y}}(t)$, which is an approximation of $\boldsymbol{y}(t)$.

In the $2m$-dimensional state-space of system (6), the synchronized state $\boldsymbol{x}_a = \boldsymbol{x}_b$ represents an m-dimensional invariant manifold. On this manifold, (6) reduces to (5).

B. Complete synchronization and asymptotic stability

Consider an m-dimensional system whose state $\boldsymbol{x}(t)$ evolves according to:

$$\boldsymbol{x}(t+\tau) = \boldsymbol{F}(\boldsymbol{x}(t)) \tag{7}$$

We assume it is possible to split the state variables in two groups, $\boldsymbol{x}(t) = [\boldsymbol{s}(t), \boldsymbol{r}(t)]$, where $\boldsymbol{s}$ is m_1-dimensional and $\boldsymbol{r}$ is m_2-dimensional, with $m = m_1 + m_2$, such that system (7) can be decomposed in two subsystems:

$$\boldsymbol{s}(t+\tau) = \boldsymbol{g}(\boldsymbol{s}(t)) \qquad \text{(Drive)} \tag{8a}$$

$$\boldsymbol{r}(t+\tau) = \boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{h}(\boldsymbol{s}(t+\tau))) \qquad \text{(Response)} \tag{8b}$$

We refer to (8a) as the *drive* and (8b) as the *response*. We denote the attractor of $\boldsymbol{s}$ (respectively, $\boldsymbol{r}$) as $\mathcal{A}_s$ ($\mathcal{A}_r$) and its basin of attraction as B_s (B_r).

We now introduce a *driven replica subsystem*:

$$\tilde{\boldsymbol{r}}(t+\tau) = \boldsymbol{f}(\tilde{\boldsymbol{r}}(t), \boldsymbol{h}(\boldsymbol{s}(t+\tau))) \tag{9}$$

Note that $\boldsymbol{f}$ is the same as in (8b). We then take the sequence of states $\boldsymbol{s}(t)$ from (8a) and use $\boldsymbol{h}(\boldsymbol{s})$ to feed the replica subsystem (9). The complete synchronization [36] between the response (8b) and its replica (9) is defined as the identity of the trajectories of $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$. In more formal terms, we are requiring the *asymptotic stability* of the response with respect to the replica subsystem [7, Sec. 3.6].

Definition 1 (Asymptotic stability)**.** A dynamical system is said to be *asymptotically stable* if, for any two copies $\boldsymbol{r}$ and $\tilde{\boldsymbol{r}}$ of the system driven by the same input $\boldsymbol{u}(t)$ and starting from different initial conditions in B_r, it holds that

$$\lim_{t\to\infty} \|\boldsymbol{r}(t, \boldsymbol{u}(t+\tau)) - \tilde{\boldsymbol{r}}(t, \boldsymbol{u}(t+\tau))\| = 0 \tag{10}$$

In our case, $\boldsymbol{u} = \boldsymbol{h}(\boldsymbol{s})$ and $\boldsymbol{s}(0) \in B_s$. The state of the full dynamical system is now constituted by (8) and (9), and thus it is $m_1 + 2m_2$ dimensional. The synchronized state $\hat{\boldsymbol{r}} = \boldsymbol{r}$ represents an $(m_1 + m_2)$-dimensional manifold embedded in the state-space of the full system.

C. Generalized Synchronization

So far, we have studied a form of synchronization in which the trajectories are identical or nearly identical. This requires the systems to be similar as well. We now describe a notion of synchronization, called *generalized synchronization*, in which the two systems under consideration are not necessarily similar to each other.

Consider again one-way coupled systems like in (8).

Definition 2 (Generalized synchronization)**.** The trajectories of the two systems $\boldsymbol{s}$ and $\boldsymbol{r}$ in (8) are said to be *synchronized in a generalized sense* if there exists a continuous *synchronization function* $\boldsymbol{\phi}$ such that:

$$\boldsymbol{r} = \boldsymbol{\phi}(\boldsymbol{s}), \tag{11}$$

and for which

$$\lim_{t\to\infty} \|\boldsymbol{r}(t) - \boldsymbol{\phi}(\boldsymbol{s}(t))\| = 0 \tag{12}$$

This means that the response state $\boldsymbol{r}$ is asymptotically given by the state of the driving system $\boldsymbol{s}$ and there exists a *synchronization manifold* $\mathcal{M} \subseteq \mathbb{R}^{d_r} \times \mathbb{R}^{d_s}$ (given by (11)) in the full state-space of the system. This manifold is invariant, in the sense that $\boldsymbol{r}(t) = \boldsymbol{\phi}(\boldsymbol{s}(t))$ implies $\boldsymbol{r}(t+\tau) = \boldsymbol{\phi}(\boldsymbol{s}(t+\tau))$. Moreover, (12) implies that such a manifold must be attracting. Since the relationship defined in (12) should hold on the attractor $\mathcal{A}_s$, which the drive system approaches asymptotically, it makes sense to write the attractor of the response system as $\mathcal{A}_r = \boldsymbol{\phi}(\mathcal{A}_s)$. We assume $\boldsymbol{\phi}$ to be smooth. The case in which the synchronization function exists but is complicated or even fractal is called *Weak Synchronization* [38]; this case is not taken into account in our paper.

In [22] the authors proved a necessary and sufficient condition for the GS between the driver $\boldsymbol{s}$ (through $\boldsymbol{u} = \boldsymbol{h}(\boldsymbol{s})$) and the response $\boldsymbol{r}$: GS occurs if and only if, for all initial

conditions $\boldsymbol{s}(t_0) \in B_s$, the response system is asymptotically stable (Def. 1). It is important to highlight the importance of this result: to assess the presence of GS between the drive and response systems, we can study the (complete) synchronization between identical copies of the response system when driven by the same input sequence (i.e., having the same drive). While such an approach might be hard to follow when considering physical systems, it poses no problem and can be straightforwardly implemented in the context we are interested in. Finally, it is worth noting that the GS framework includes complete synchronization as a special case, where the systems are identical and the synchronization function $\boldsymbol{\phi}$ is simply the identity.

D. Echo State Property

The definition of the Echo State Property (ESP) first appeared in the seminal work by Jaeger [21] and was then reformulated and re-visited in various works, e.g. see [56, 31, 18, 11]. The motivation behind the ESP is the following: for learning to be feasible, it is crucial that the current network state $\boldsymbol{r}(t)$ is uniquely determined by the input sequence $\{\boldsymbol{u}(t)\}$. Such a definition takes into account a specific input sequence, with values in a compact set $\mathcal{U}$; in practical applications, the input will always be bounded. Also the compactness of the reservoir state-space is required, but it is automatically guaranteed if one considers a bounded nonlinear activation functions (like $\tanh$).

Definition 3 (Compatibility)**.** We say that a state sequence $\{\boldsymbol{r}(t)\}$ is *compatible* with a bounded input sequence $\{\boldsymbol{u}(t)\}$ when:

$$\boldsymbol{r}(t+\tau) = \boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{u}(t+\tau))$$

The core idea behind the ESP boils down to requesting that all compatible state sequences will asymptotically converge to a unique trajectory under the same driving input.

Definition 4 (Echo State Property)**.** A driven dynamical system has the *Echo State Property* with respect to a driving input sequence $\{\boldsymbol{u}(t)\}$ if, for any pair of reservoir state sequences $\{\boldsymbol{r}_1(t)\}$, $\{\boldsymbol{r}_2(t)\}$ compatible with $\{\boldsymbol{u}(t)\}$, it holds that for all $\epsilon > 0$ there exists $s \in \mathbb{N}$ such that $\|\boldsymbol{r}_1(t) - \boldsymbol{r}_2(t)\| < \epsilon$ for all $t > s$, or equivalently:

$$\lim_{t\to\infty} \|\boldsymbol{r}_1(t) - \boldsymbol{r}_2(t)\| = 0$$

The above definition is a forward-oriented definition of the ESP; also backward-oriented definitions are possible, e.g. [56].

An important open problem consists in determining the class of inputs for which Def. 4 holds. Yet, Def. 4 does not take into account the nature of the driving input sequence: as pointed out in [56], such a definition for the ESP considers the input sequence as a range of admissible values, disregarding any relevant properties such as its (nonlinear) temporal correlation structure. This renders the task of assessing the ESP for a specific input sequence really hard [31, 56] and its validity is mostly taken as an assumption in theoretical studies.

E. Echo State Property as Generalized Synchronization

In the framework we are developing here, under the assumption that the input is given by (2a), it will be possible to re-frame the ESP in terms of GS. The analogy between the ESP in Def. 4 and the asymptotic stability (Def. 1) is evident. And since the implication of the theorem in [22] is true in both directions, this implies that – under the hypotheses discussed in Sec. II – requiring the ESP is the same as requiring the GS between the source system (1) and the reservoir (3).

Note that system (1) is autonomous, so that each trajectory is uniquely specified by the initial conditions. Therefore, we do not have to specify the asymptotic stability (and so, the GS) in terms of trajectories, but in terms of initial conditions generating them. That is, instead of studying individual trajectories, we can focus on studying the state of the systems: it is possible to define regions for which the asymptotic stability holds and regions for which it does not. Accordingly, by using GS, one partitions the state-space into regions where the ESP holds and regions where it does not hold. This basically solves the problem of defining ESP w.r.t. to the properties of the input: if, as in our case, the input signal $\boldsymbol{u}(t)$ is generated by a dynamical system like in Eq. (1), then GS (and so, ESP) can be defined in terms of initial conditions.

If the source system (1) has a unique attractor and its basin of attraction is the whole state space, it will be possible to assess the GS disregarding of initial conditions. In this work, we will only deal with this case.

IV. Implications on the training mechanism

In the scenario depicted above, one uses the reservoir $\boldsymbol{r}$ to create a representation of the input, which is finally processed by the readout $\boldsymbol{\psi}$. The goal is to generate a mapping from $\boldsymbol{s}$ to $\boldsymbol{y}$ and then to use such readout for generating $\hat{\boldsymbol{y}}(t)$ for values of $\boldsymbol{u}(t)$ which are not in the training set (i.e. for $t \leq 0$). Since $\boldsymbol{s}$ is unknown, what one really assumes is that it is possible to predict $\boldsymbol{y}$ from the knowledge of the whole history of $\boldsymbol{u}$. This is, in fact, an implication of *Takens embedding theorem* [46] and the feasibility of such a procedure was recently proved in the context of RC by Hart, Hook, and Dawes [20].

It is really important to emphasize the fact that we only consider the case where the fitting of the readout *does not* affect the reservoir dynamics in any way. The representation of the attractor of $\boldsymbol{s}$ into the reservoir states $\boldsymbol{r}$ by the use of the input sequence $\boldsymbol{u}$ is done in the listening phase, which is (in machine learning parlance) *unsupervised*. The fitting consists of trying to estimate the *static* function $\boldsymbol{k}$ mapping the state $\boldsymbol{s}$ to the desired output $\boldsymbol{y}$, i.e,

$$\hat{\boldsymbol{y}} := \boldsymbol{\psi}(\boldsymbol{r}) \approx \boldsymbol{k}(\boldsymbol{s}) = \boldsymbol{y} \tag{13}$$

A. Unsupervised system reconstruction during the listening phase

Let us consider the time interval $(t_s, 0)$ in which we assume that the GS has occurred (remember that we assume negative

times for the training phase, so $t_s < 0$). We consider the reservoir states generated in this interval,

$$\boldsymbol{R}_{(t_s,0)} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{r}(t_s) & \boldsymbol{r}(t_s+\tau) & \dots & \boldsymbol{r}(0) \\ | & | & | & | \end{bmatrix} = \\ = \begin{bmatrix} | & | & | & | \\ \boldsymbol{r}(t_s) & \boldsymbol{f}(\boldsymbol{r}(t_s),\boldsymbol{u}(t_s)) & \dots & \boldsymbol{f}(\boldsymbol{r}(-\tau),\boldsymbol{u}(-\tau)) \\ | & | & | & | \end{bmatrix} \tag{14}$$

GS guarantees that there exists a function mapping the source system states to the reservoir states and also its invariance. This means that

$$\boldsymbol{r}(t) = \boldsymbol{\phi}(\boldsymbol{s}(t)) \Rightarrow \boldsymbol{r}(t+\tau) = \boldsymbol{\phi}(\boldsymbol{s}(t+\tau)) = \boldsymbol{\phi}(\boldsymbol{g}(\boldsymbol{s}(t))) \tag{15}$$

so that (14) can be written as follows:

$$\boldsymbol{R}_{(t_s,0)} = \begin{bmatrix} | & | & | & | \\ \boldsymbol{\phi}(\boldsymbol{s}(t_s)) & \boldsymbol{\phi}(\boldsymbol{s}(t_s+\tau)) & \dots & \boldsymbol{\phi}(\boldsymbol{s}(0)) \\ | & | & | & | \end{bmatrix} \tag{16}$$

Note that $\boldsymbol{\phi}$ is a time-independent function that is the same for all $\boldsymbol{s}$. Since by assumption $d_r > d_s$, we can think of $\boldsymbol{\phi}(\boldsymbol{s})$ as an attempt to expand the source system state-space (which is unknown) into a higher-dimensional space, in the same fashion as the well-known reproducing kernel Hilbert space mechanism behind kernel methods [43]. Moreover note that this expansion $\boldsymbol{\phi}$ was not computed or estimated from data, but was "obtained" as a result of driving the reservoir with the input sequence under consideration: this means that the mapping is "informed" of the dynamics, since it was dynamically obtained. Accordingly, we can interpret (13) as follows:

$$\hat{\boldsymbol{y}} = \boldsymbol{\psi}(\boldsymbol{r}) = \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{s})) \approx \boldsymbol{k}(\boldsymbol{s}) = \boldsymbol{y} \tag{17}$$

To conclude, we suggest that the reservoir dynamics perform a sort of "nonlinear basis expansion" of the (unknown) attractor of $\boldsymbol{s}$. The use of the synchronization function $\boldsymbol{\phi}$ provides a sound theoretical framework to the fitting process, and the relation (11) can be seen a sound formulation of the "reservoir trick"; see [44].

We stress that, in the analysis above, the readout function is not involved and so the chosen training mechanism is irrelevant to our conclusions.

B. Learning feasibility

We define the concept of "feasible learning" as the situation where the readout is perfectly able to reconstruct the targets by using the reservoir states. More formally,

Definition 5 (Learning feasibility)**.** We say that the learning is *feasible* if there exists a readout $\boldsymbol{\psi}$ such that,

$$\boldsymbol{y}(t) = \boldsymbol{\psi}(\boldsymbol{r}(t)),\ \forall t \tag{18}$$

The following theorem proves that for the learning to be feasible, there must be a function mapping the trajectory of the source system into the trajectory of the reservoir. First we introduce some notation. Let us denote with $\mathcal{S} \subset \mathcal{A}_s$ the set containing all $\boldsymbol{s}(t)$, for all t. Analogously, we define $\mathcal{R} \subset \mathcal{A}_r$ as the set of all $\boldsymbol{r}(t)$, for all t. We define $\mathcal{Y}$ as the result of applying $\boldsymbol{k}$ to each point in $\mathcal{S}$, in short $\mathcal{Y} := \boldsymbol{k}(\mathcal{S})$.

Theorem 1. *For the learning to be feasible, there must be a function $\boldsymbol{\mathcal{F}}$ with the following property: each $\boldsymbol{r}$ such that $\boldsymbol{\psi}(\boldsymbol{r}) = \boldsymbol{y}$ can be written as:*

$$\boldsymbol{r} = \boldsymbol{\mathcal{F}}(\boldsymbol{s}), \tag{19}$$

where $\boldsymbol{s}$ is such that $\boldsymbol{k}(\boldsymbol{s}) = \boldsymbol{y}$.

Proof. Feasibility of learning corresponds to requiring $\boldsymbol{\psi}$ to be surjective when mapping $\mathcal{R}$ into $\mathcal{Y}$. The surjectivity of $\boldsymbol{k}$ is guaranteed by the way we constructed $\mathcal{Y}$. But since, in general, different source system states could result in the same target, $\boldsymbol{k}$ may not be an injective function. The same holds for $\boldsymbol{\psi}$. We define $\boldsymbol{\psi}^\dagger(\boldsymbol{y})$ as the function mapping each $\boldsymbol{y}$ onto an $\boldsymbol{r}$: if $\boldsymbol{\psi}$ is also injective, then $\boldsymbol{\psi}^\dagger$ corresponds to the inverse of $\boldsymbol{\psi}$, but in general it is not. These functions are called *right-inverse* since $\boldsymbol{\psi} \circ \boldsymbol{\psi}^\dagger$ is the identity but $\boldsymbol{\psi}^\dagger \circ \boldsymbol{\psi}$ is not. Since by definition $\boldsymbol{y}(t) = \boldsymbol{k}(\boldsymbol{s}(t))$, it will then be possible to construct the function $\boldsymbol{\mathcal{F}}$ as follows:

$$\boldsymbol{\mathcal{F}} = \boldsymbol{\psi}^\dagger \circ \boldsymbol{k} \tag{20}$$

Such a function maps all $\boldsymbol{s}$ into the corresponding targets $\boldsymbol{y}$ and inverts the readout function $\boldsymbol{\phi}$ so that it maps each target to a corresponding reservoir state $\boldsymbol{r}$.

So far, we have shown that (20) maps each $\boldsymbol{s}$ to an $\boldsymbol{r}$. We also need to make sure that each $\boldsymbol{r}$ can be written as $\boldsymbol{\mathcal{F}}(\boldsymbol{s})$. This is granted by our assumption (surjectivity of $\boldsymbol{\psi}$) which tells us that each $\boldsymbol{y}$ can be written as $\boldsymbol{\psi}(\boldsymbol{r})$. Then, using an argument analogous to the one above, we can associate each $\boldsymbol{y} \in \mathcal{Y}$ to an $\boldsymbol{s} \in \mathcal{S}$ by defining $\boldsymbol{k}^\dagger$. Again, this corresponds to the inverse of $\boldsymbol{k}$ only if $\boldsymbol{k}$ is also injective. Note that, in general, distinct $\boldsymbol{r}$'s might be associated to the same $\boldsymbol{s}$ (and *viceversa*). This shows that if learning is feasible, then $\boldsymbol{\mathcal{F}}$ must exist. □

The theorem also implies that if $\boldsymbol{\mathcal{F}}$ does not exist, then learning is not feasible. So, any successful training procedure must (i) develop an (implicit) mapping from $\mathcal{S}$ to $\mathcal{R}$ and (ii) find a suitable readout. Yet, the existence of $\boldsymbol{\mathcal{F}}$ *does not necessarily imply* the feasibility of learning: we have no guarantees that, in the presence of such a mapping, a readout solving the problem can be found. Moreover, the fact that learning is not feasible *does not necessarily imply* that $\boldsymbol{\mathcal{F}}$ does not exist: the problem might simply be that we are not able to conceive the right readout.

We now prove that, by requiring $\boldsymbol{\mathcal{F}}$ to be injective, we can always construct a readout which correctly solves the problem.

Theorem 2. *If there exists a function $\boldsymbol{\mathcal{F}}$ such that $\boldsymbol{r}(t) = \boldsymbol{\mathcal{F}}(\boldsymbol{s}(t))$ and $\boldsymbol{\mathcal{F}}$ is injective, then learning is feasible.*

Proof. The proof is similar to the one above. The injectivity of $\boldsymbol{\mathcal{F}}$ grants the existence of $\boldsymbol{\mathcal{F}}^*$, which is the *left-inverse* of $\boldsymbol{\mathcal{F}}$, i.e. the function for which $\boldsymbol{\mathcal{F}}^* \circ \boldsymbol{\mathcal{F}}$ is the identity (but $\boldsymbol{\mathcal{F}} \circ \boldsymbol{\mathcal{F}}^*$ is not). The readout function solving (18) then exists and it is given by:

$$\boldsymbol{\psi} = \boldsymbol{k} \circ \boldsymbol{\mathcal{F}}^* \tag{21}$$

□

The fact that $\boldsymbol{\mathcal{F}}$ is injective means that it always maps distinct $\boldsymbol{s}$ into distinct $\boldsymbol{r}$. Without it, $\boldsymbol{\mathcal{F}}$ may map two distinct $\boldsymbol{s}_1, \boldsymbol{s}_2$ into the same $\boldsymbol{r} = \boldsymbol{\mathcal{F}}(\boldsymbol{s}_1) = \boldsymbol{\mathcal{F}}(\boldsymbol{s}_2)$, as long as $\boldsymbol{k}(\boldsymbol{s}_1) = \boldsymbol{k}(\boldsymbol{s}_2) = \boldsymbol{y}$.

It is important to note that both $\boldsymbol{k}$ and $\boldsymbol{\mathcal{F}}$ are unknown in our problem setting, so that the theorem only guarantees the possibility of finding the right $\boldsymbol{\psi}$ but does not provide a constructive way of finding it. Therefore, when learning is unfeasible it is generally impossible to understand whether the problem is related to $\boldsymbol{\mathcal{F}}$, to $\boldsymbol{\psi}$, or even to the both of them.

Notably, this problem can be bypassed by considering the synchronization function $\boldsymbol{\phi}$ as a surrogate for $\boldsymbol{\mathcal{F}}$. As $\boldsymbol{\phi}$ is only related to the dynamical evolution of the reservoir (listening phase), we can discuss its existence and properties disregarding the readout. This shows the importance of $\boldsymbol{\phi}$ in the context of RC: it can be used to grant some sufficient conditions of the listening phase in an unsupervised way, so that one can then concentrate on finding the best readout, knowing that the error on the task at hand depends on it.

Finally, we point out that Theorem 2 formally proves that – as suggested in other works [27, 26] – the existence of an invertible[4] synchronization function is sufficient for the RC paradigm to work (provided that the readout is able to correctly approximate the target).

C. Error on the whole attractor

Since the readout $\boldsymbol{\psi}$ is generated after the listening phase, we have no guarantees that, in general, it will continue to correctly reproduce the target also in the predicting phase. In fact, by choosing a readout complex enough, it is always possible to map $\mathcal{R}$ into $\mathcal{Y}$, but this will not give us a useful way for predicting unseen values of $\boldsymbol{y}$. Some additional requirements are necessary. In particular, we need the relation in (18) to hold not only from $\mathcal{S}$ to $\mathcal{R}$, but also on the whole attractor, i.e. with an abuse of notation we need $\mathcal{A}_r = \boldsymbol{\psi}(\mathcal{A}_s)$. In the machine learning parlance, this can be described as follows: a single trajectory plays the role of a sample, while the attractor plays the role of the data-generating process. This becomes possible by assuming the attractor $\mathcal{A}_s$ to be ergodic [6]. In fact, ergodicity guarantees that a sufficiently long trajectory will be a "good sampling" of the whole attractor.

More in detail, let us define the loss function:

$$\mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) = \sqrt{\|\boldsymbol{y}(t) - \hat{\boldsymbol{y}}(t)\|} \tag{22}$$

called Root Mean Squared Error (RMSE).[5] The learning feasibility trivially implies that

$$\frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) = 0 \tag{23}$$

since $\mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) = 0, \forall t$. Note that time starts at $t = t_s$ because we want to remove transient effects.

By expanding $\boldsymbol{y}$ and $\hat{\boldsymbol{y}}$, we get:

$$\frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{k}(\boldsymbol{s}(t)), \boldsymbol{\psi}(\boldsymbol{r}(t))) = 0 \tag{24}$$

[4]If a function is invertible, it is also injective.

[5]Note that different choices can be made for $\mathcal{L}$ and the results do not depend on its particular form.

The existence of a function $\boldsymbol{\mathcal{F}}$ (Thm. 2) allows us to write $\boldsymbol{r}(t) = \boldsymbol{\mathcal{F}}(\boldsymbol{s}(t))$, so that our loss becomes

$$\mathcal{L}(\boldsymbol{k}(\boldsymbol{s}(t)), \boldsymbol{\psi}(\boldsymbol{r}(t))) = \mathcal{L}(\boldsymbol{k}(\boldsymbol{s}(t)), \boldsymbol{\psi}(\boldsymbol{\mathcal{F}}(\boldsymbol{s}(t)))) = \mathcal{L}(\boldsymbol{s}(t)) \tag{25}$$

where the last equality stresses the fact that $\mathcal{L}$ is a function of $\boldsymbol{s}(t)$ only (with an abuse of notation on the function $\mathcal{L}$). Taking the limit for $T \to \infty$, we can now exploit the ergodicity of $\mathcal{A}_s$ and obtain:

$$0 = \lim_{T\to\infty} \sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) = \underbrace{\lim_{T\to\infty} \sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{s}(t)) = \langle \mathcal{L}(\boldsymbol{s}) \rangle_{\mathcal{A}_s}}_{\text{ergodicity}} \tag{26}$$

In (26), $\langle \mathcal{L}(\boldsymbol{s}) \rangle_{\mathcal{A}_s}$ denotes the average loss over the whole attractor, $\mathcal{A}_s$. This means that, if learning is feasible for a single trajectory, then it will be feasible on the whole attractor of the source system.

Note that the crucial part of this approach is the dependence on $\boldsymbol{s}$ only, because only the source system attractor $\mathcal{A}_s$ is assumed to be ergodic.

D. Synchronization function

One would like to relax the definition of feasible learning (see Def. 5): in fact, in realistic situations the error is not exactly zero. This is formalized by assuming that $\mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) = \epsilon_t \geq 0, \forall t$, so that:

$$\mathcal{E}_T = \frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) \tag{27}$$

As proved in the previous section, if a mapping $\boldsymbol{\mathcal{F}}$ does not exist, then learning cannot be feasible according to Def. 5. But assuming GS to hold, we can make use of the synchronization function $\boldsymbol{\phi}$ and write:

$$\begin{aligned}\frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) &= \frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{k}(\boldsymbol{s}(t)), \boldsymbol{\psi}(\boldsymbol{\phi}(\boldsymbol{s}(t)))) = \\ &= \frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{s}(t))\end{aligned} \tag{28}$$

where, again, $\mathcal{L}$ depends only on $\boldsymbol{s}(t)$. In order to invoke ergodicity of the source system, we need to be sure that the above limit exists. An easy way for guaranteeing this consists of requiring the error to be bounded, i.e. to have $\mathcal{L}(\boldsymbol{y}(t), \hat{\boldsymbol{y}}(t)) = \epsilon_t < C$, where $C \geq 0$ is a constant. So, when $\mathcal{E} := \lim_{T\to\infty} \mathcal{E}_T$ exists and is finite, one can write:

$$\mathcal{E} = \lim_{T\to\infty} \mathcal{E}_T = \lim_{T\to\infty} \frac{1}{T}\sum_{t=t_s}^{T} \mathcal{L}(\boldsymbol{s}(t)) = \langle \mathcal{L}(\boldsymbol{s}(t)) \rangle_{\mathcal{A}_s} \tag{29}$$

The existence of the synchronization function guarantees that the error in a single trajectory is the same as the error in the whole attractor of the source system. This means that, by assuming GS, we can have some guarantees on the performance of our model even when learning is not feasible,

and this is due to the fact that $\mathcal{L}$ depends only on the source system states $\boldsymbol{s}$ when assuming GS. In fact, the definition of the synchronization function in Def. 2 has other implications for the training mechanisms. Making use of its smoothness along with the attractivity of the synchronization manifold, one can account not only for the error in the approximation, but also for the observational noise of the source system (see Appendix C for details).

We stress that the existence of GS is a property which only involves the source system (1) and its coupling with the reservoir (3) by mean of the measurements (2a), disregarding the particular task at hand. In fact, in the discussion above, we showed how using the synchronization function $\boldsymbol{\phi}$ one can, in some sense, decouple the learning task and separate the problem of finding a suitable readout from the problem of granting the existence of a mapping from the source system states, $\boldsymbol{s}$, to the reservoir states, $\boldsymbol{r}$.

V. Experimental results

A. The mutual false nearest neighbors

Identifying GS is hard due to the fact that the synchronization function (11) in unknown and may take any form. For this reason, in [42] a method to empirically assess the occurrence of GS from data was proposed under the name of MFNN. It is based on the fact that, under reasonable smoothness conditions for $\boldsymbol{\phi}$, 11 implies that two states that are close in state-space of the response system correspond to two close states in the state-space of the driving system. So, we are looking for a geometric connection between the two systems which preserves the neighbor-structure in state space.

Let us assume that we sample trajectories from a dynamical system at a fixed sampling rate, resulting in a series of discrete times $\{t_n\}$. The resulting measurements for the two systems will be $\{\boldsymbol{x}_n\}$ and $\{\boldsymbol{y}_n\}$, for the drive and the response respectively, where we used the notation $\boldsymbol{x}_n := \boldsymbol{x}(t_n)$ and $\boldsymbol{y}_n := \boldsymbol{y}(t_n)$. For each point $\boldsymbol{x}_n$ of the driving system, we seek the closest point from its neighbors, which we will call time index n_{NND}. Then, due to (11), the point $\boldsymbol{y}_n = \boldsymbol{\phi}(\boldsymbol{x}_n)$ will be close to $\boldsymbol{y}_{n_{\text{NND}}}$. If the distances between these pairs of points in state-space of both the drive and response systems are small, one can write:

$$\boldsymbol{y}_n - \boldsymbol{y}_{n_{\text{NND}}} = \boldsymbol{\phi}(\boldsymbol{x}_n) - \boldsymbol{\phi}(\boldsymbol{x}_{n_{\text{NND}}}) \approx \boldsymbol{D\phi}(\boldsymbol{x}_n)(\boldsymbol{x}_n - \boldsymbol{x}_{n_{\text{NND}}}) \tag{30}$$

where $\boldsymbol{D\phi}(\boldsymbol{x}_n)$ is the Jacobian of $\boldsymbol{\phi}$ evaluated at $\boldsymbol{x}_n$.

Now, we do a similar operation but in the response state space. We look for the closer point to $\boldsymbol{y}_n$ and we index it with n_{NNR}. Again, due to (11), it holds:

$$\boldsymbol{y}_n - \boldsymbol{y}_{n_{\text{NNR}}} = \boldsymbol{\phi}(\boldsymbol{x}_n) - \boldsymbol{\phi}(\boldsymbol{x}_{n_{\text{NNR}}}) \approx \boldsymbol{D\phi}(\boldsymbol{x}_n)(\boldsymbol{x}_n - \boldsymbol{x}_{n_{\text{NNR}}}) \tag{31}$$

So, due to (30) and (31) we have two different ways of evaluating $\boldsymbol{D\phi}(\boldsymbol{x}_n)$. This leads us to the definition of the MFNN as the following ratio:

$$\text{MFNN}(n) := \frac{\|\boldsymbol{y}_n - \boldsymbol{y}_{n_{\text{NND}}}\|}{\|\boldsymbol{x}_n - \boldsymbol{x}_{n_{\text{NND}}}\|} \frac{\|\boldsymbol{x}_n - \boldsymbol{x}_{n_{\text{NNR}}}\|}{\|\boldsymbol{y}_n - \boldsymbol{y}_{n_{\text{NNR}}}\|} \tag{32}$$

If the two systems are synchronized in a general sense, then $\text{MFNN}(n) \approx 1$. If the synchronization relation does not hold, then (32) should instead be of the order of (size of the attractor squared)/(distance between nearest neighbors squared) which is, in general, a large number.

Note that in this work we use the full knowledge of the source system to measure the GS by means of MFNN. Generally, one would not have such a knowledge: anyway the MFNN can be used also in this case, as showed in the paper where it was proposed [42], making use of the embedding theorem. For simplicity, we do not deal with this more complex case, since it would not be relevant for the discussion.

B. Reservoir computing networks

For simplicity, but without loss of generality, we will only deal with one-dimensional inputs $u(t) \in \mathbb{R}$. We use an RCN where the explicit form of the reservoir equation (3) reads:

$$\boldsymbol{r}(t+\tau) = \tanh\left(\boldsymbol{W}\boldsymbol{r}(t) + \boldsymbol{w}u(t+\tau) + \boldsymbol{b}\right) \tag{33}$$

$\boldsymbol{W} \in \mathbb{R}^{d_r \times d_r}$ is the *connectivity matrix*, which is an Erdos-Renyi matrix with average degree 6; d_r indicates the dimension of the reservoir. The non-null elements are drawn from a uniform distribution taking values in the interval $(-1, 1)$. $\boldsymbol{W}$ is re-scaled so that its *Spectral Radius (SR)* equals the user-defined hyper-parameter $\rho > 0$. The elements $\boldsymbol{w} \in \mathbb{R}^{d_r}$ of the *input vector* are drawn from a uniform distribution taking values in $(-\omega, \omega)$; we refer to ω as the *input scaling* hyper-parameter. $\boldsymbol{b} = [b, b, \ldots, b]$ is a constant bias term, which is useful to control the non-linearity of the system and to break the symmetry with respect to the origin [28]. Here, $\tanh$ stands for the hyperbolic-tangent function applied element-wise.

We use a linear readout, so that the predicted output is given by:

$$\hat{\boldsymbol{y}}(t) = \boldsymbol{\psi}(\boldsymbol{r}(t)) \equiv \boldsymbol{W}_{\text{out}}\boldsymbol{r}(t) \tag{34}$$

where $\boldsymbol{W}_{\text{out}}$ is a $d_y \times d_r$ matrix, called readout matrix; d_y is the output dimension. We train the readout using ridge-regression with regularization parameter λ, but more sophisticated, off-line optimization procedures can be designed as well [13, 44, 24].

C. Reservoir observer

We test the hypothesis that learning in RC can happen only when the network is synchronized with the source system (1). To do so, we adopt the framework named *reservoir observer* [28], which consists of setting $\boldsymbol{h}(\boldsymbol{s}) = s_1 = u$ and $\boldsymbol{y} = \boldsymbol{s}$. This means that the network is trained to reconstruct the full state of the source system by seeing only one component of it. Note that $\boldsymbol{k}$ in (2b) is the identity for this task, and (17) reduces to finding the left inverse of the synchronization function (11). Practically, when using a linear readout (as in fact we do here), one implicitly assumes that $\boldsymbol{k} \circ \boldsymbol{\phi}^*(\boldsymbol{r})$ in (17) can be expressed in linear form

$$\boldsymbol{\phi}^*(\boldsymbol{r}) = \boldsymbol{W}_{\text{out}}\boldsymbol{r} \tag{35}$$

implying that

$$\boldsymbol{\phi}(\boldsymbol{s}) = \boldsymbol{W}^*_{\text{out}}\boldsymbol{s} \tag{36}$$

where $\boldsymbol{W}^*_{\text{out}}$ is the left pseudo-inverse of the readout matrix.

TABLE I
DEFAULT HYPER–PARAMETERS USED IN ALL EXPERIMENTS, UNLESS DIFFERENTLY STATED.

ρ	1	T_{train}	$(-100, 0)$
ω	0.1	T_{test}	$(0, 80)$
d_r	300	τ	0.05
b	1	λ	10^{-6}

D. Results

As for the source system (1), we use the Lorenz model (see Appendix A for details). In the listening phase, we use t in the interval of $T_{\text{train}} = (-100, 0)$. We discarded the first $1/10$ of the data points for training, to account for transient effects in the synchronization process. The prediction phase was carried out for values of t in $T_{\text{test}} = (0, 80)$. The integration step was always set to $\tau = 0.05$. An example of this task is provided in Fig. 3.

For each hyper-parameter configuration, we repeated the experiment 10 times, generating a different realization of the source system (i.e. starting from distinct random initial conditions) and a different realization of the RCN (33). For each run, the MFNN between the driving system state $\boldsymbol{s}(t)$ and the reservoir $\boldsymbol{r}(t)$ was computed. As a performance measure for the prediction accuracy, we used the RMSE computed on the y and the z coordinate of the Lorenz system (since x is used as input). Following [42], we plot the inverse of the MFNN so that the higher the value, the more synchronized the systems are. Accordingly, we plot the inverse of the RMSE, which can be interpreted as a form of accuracy. Unless differently stated, all hyperparameters are the ones reported in Tab. I. All the plots refer to the *predicting phase*.

In Fig. 4, we show the RMSE and the MFNN index when the SR of the reservoir connectivity matrix varies in a suitable range. For smaller values of SR the reservoir dynamics are really simple and close to linear (since $\tanh(x) \approx x$ when x is small), so that the network it is not able to correctly represent the Lorenz attractor in its state. We see that the synchronization is weak ans the error is large. As the SR approaches 1 we see that the reservoir tends to become more synchronized with the source system state and this reflects in a smaller error. When the SR started growing, the reservoirs becomes more and more unstable and it gradually de-synchronizes with the source system, such that the reconstruction of the coordinates becomes less precise.

In Fig. 5 we repeated the experiment, but this time varying the input scaling ω and holding ρ fixed to its default value. We see that the two quantities still correlates, with both the accuracy and the synchronization decreasing as the input scaling grows.

To assess the generality of our findings, we performed additional simulations by changing the source system (1). To this end, we consider now the Rössler system as a source system (see Appendix B for details). Since the dynamics of the Rössler system are slower then the Lorenz ones, we set the integration step to $\tau = 0.5$, $T_{\text{train}} = (-200, 0)$ and $T_{\text{text}} = (0, 160)$. The remaining hyper-parameters are set as shown in Tab. I. Again, we use the x-coordinate as input and

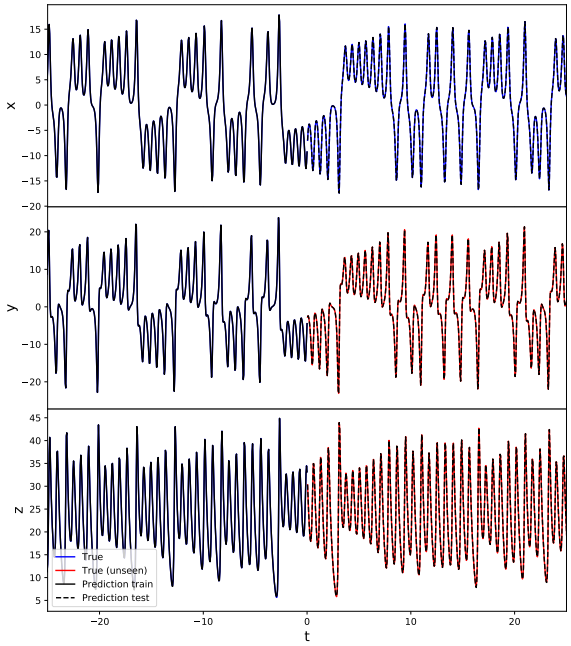

Fig. 3. An example of the observer task using the Lorenz system. The top panel show the measurement $\boldsymbol{u}$ (blue), which is always available. The middle and the bottom panels represents the targets $\boldsymbol{y}$: in the training phase they are available (blue) while in the predicting phase (red) they cannot be accessed anymore. The predicted targets $\hat{\boldsymbol{y}}$ (black dashed lines) are generated by means of the RCN described in Sec. V-B.

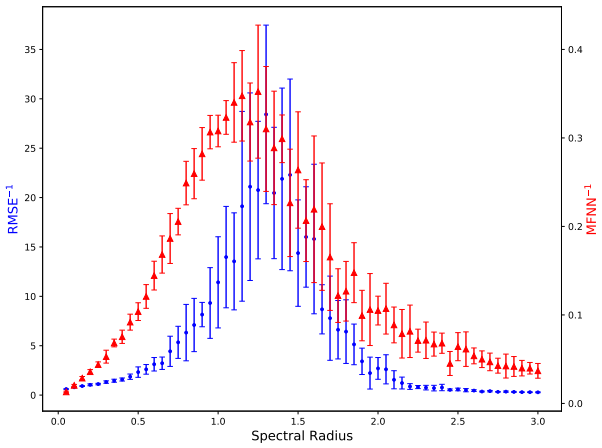

Fig. 4. Results for the reservoir observer when varying the SR of the connectivity matrix, when using the Lorenz system as a source. Blue dots account for the RMSE (left axis) while red triangles accounts for MFNN (right axis).

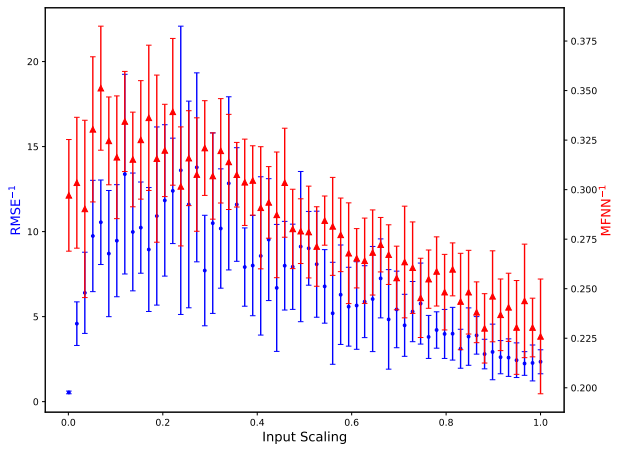

Fig. 5. Results for the reservoir observer on Lorenz system when varying the input scaling ω. when using the Lorenz system as a source.

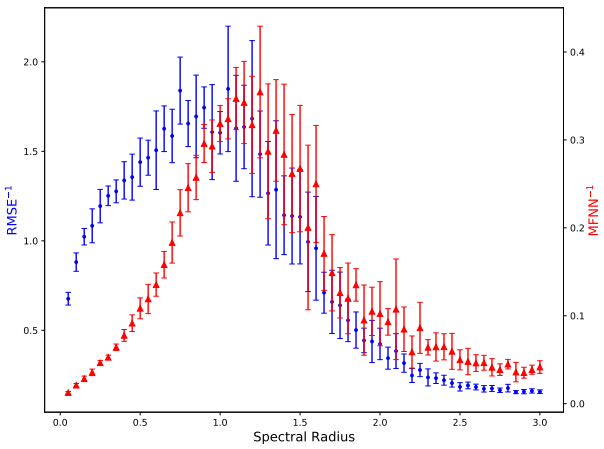

Fig. 7. Results for the forecasting task on Lorenz system when varying the SR.

the tasks consists of learning how to reproduce y and z. The results are displayed in Fig. 6 and look similar to the one obtained for the Lorenz system (Fig. 4).

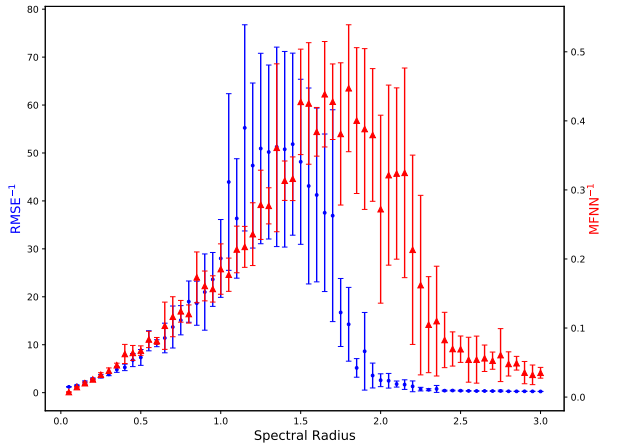

Fig. 6. Results for the reservoir observer when varying the SR, when using the Rössler system as a source. Blue dots account for the RMSE (left axis) while red triangles accounts for MFNN (right axis).

To show that GS plays an important role not only in the observer task, we also test our framework in a forecasting scenario. To this end, we use the x-coordinate of the Lorenz system as the input ($u(t) := x(t)$, but this time the target $\boldsymbol{y}$ was chosen to be $y(t) := u(t + 5\tau)$. This means that the RCN is required to correctly approximate the function $\boldsymbol{g}^5(\boldsymbol{s})$, which is highly nonlinear. The RMSE is computed between $y(t)$ and the network output $\hat{y}(t)$. Notably, the MFNN here is almost identical to the one in Fig. 4: both experiments use the same hyperparameters, the same source system and construct $\boldsymbol{u}$ in the same way, so that the only difference is the task they are trained to solve, which affects the readout and not on the dynamics. As in the other cases, we notice that the MFNN and the RMSE display a similar behavior.

These results confirm that the GS can be exploited to assess the quality of the source system representation encoded in the reservoir states: in order to correctly solve the task at hand, the reservoir and the the source system should be synchronized.

VI. Conclusions

In this work, we laid down the groundwork for establishing and analyzing working principles of RC within the theoretical framework of synchronization between dynamical systems. First, we made systematic the equivalence between the ESP and GS. Then, we showed that the presence of a synchronization function consents to formally consider the reservoir states as an unsupervised, high-dimensional representation of an unknown source system that generates the observed data. We showed that the feasibility of learning, defined as the possibility of perfectly solving the task, crucially depends of the existence of a function connecting the reservoir states with the source system states: the presence of GS implies the existence of a synchronization function playing an analogous role, which is found in an unsupervised way in RC. This formally proves that it is possible to solve the task at hand by firstly creating an unsupervised representation of the source system (listening phase) and then using a suitable readout to correctly represent the target (fitting phase), thus justifying the RC training principle in a formal way. Moreover, the presence of such a synchronization function allows one to make use of the ergodicity of the source system to grant some results on the generalization error for a given task. Finally, we made use of an index (MFNN) to quantify the degree of synchronization and experimentally validate our claims. Results show that the more the reservoir is synchronized with the source, the better the system approximates the target, hence stressing that synchronization is paramount and plays a fundamental role within the RC framework.

APPENDIX A
LORENZ SYSTEM

The Lorenz system [25] is a 3-dimensional dynamical system characterizing a simple model for atmospheric convection. Its equations read:

$$\begin{aligned} \dot{x} &= \sigma(y - x) \\ \dot{y} &= (\rho - z)x - y \\ \dot{z} &= xy - \beta z \end{aligned} \tag{37}$$

where $x = x(t)$, $y = y(t)$, $z = z(t)$ are the variables, σ, ρ and β are the the model parameters and the dot denotes the first-order derivative with respect to time t. In this work, we choose the commonly used values $\sigma = 10$, $\rho = 29$ and $\beta = 8/3$, for which the system is known to be a chaotic one and to have a strange attractor.

APPENDIX B
RÖSSLER SYSTEM

The Rössler system [41] is a 3-dimensional chaotic dynamical system defined as follows:

$$\begin{aligned} \dot{x} &= -y - z \\ \dot{y} &= x + ay \\ \dot{z} &= b + z(x - c) \end{aligned} \tag{38}$$

where $x = x(t)$, $y = y(t)$, $z = z(t)$ are the variables and a, b, and c are the model parameters, which in our paper are set to $a = 0.1$, $b = 0.1$, and $c = 14$. The dot denotes the first-order derivative with respect to time t.

APPENDIX C
MEASUREMENT NOISE

In many real situations the input is corrupted by some noise, so that instead of reading just $\boldsymbol{u}(t)$ one actually reads $\boldsymbol{u}(t) + \boldsymbol{\epsilon}(t)$, $\boldsymbol{\epsilon}(t)$ being i.i.d. noise. This lead to the following state-update for the reservoir:

$$\begin{aligned} \boldsymbol{r}(t+1) &= \boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{u}(t) + \boldsymbol{\epsilon}(t)) \\ &\approx \boldsymbol{f}(\boldsymbol{r}(t), \boldsymbol{u}(t)) + \boldsymbol{f}'(\boldsymbol{r}(t), \boldsymbol{u}(t))\boldsymbol{\epsilon}(t) \end{aligned} \tag{39}$$

This will affect the reservoir dynamics in general, but when the noise is small we can still hope that the trajectory will not be too far from the one generated without noise. That is, we assume it is possible to write each point as $\boldsymbol{r}(t) + \boldsymbol{\eta}(t)$. This will be in fact guaranteed by the GS, which requires the synchronization manifold not only to exist, but also to be attractive [22]. Note that $\boldsymbol{\eta}(t)$ is not i.i.d. anymore.

The synchronization problem (w.r.t. to the *true* system state) becomes:

$$\boldsymbol{s} = \boldsymbol{\phi}(\boldsymbol{r} + \boldsymbol{\eta}) \tag{40}$$

We can make use of the smoothness of $\boldsymbol{\phi}$ to write a first-order approximation of the source system state as follows:

$$\boldsymbol{s} \approx \boldsymbol{\phi}(\boldsymbol{r}) + \boldsymbol{\phi}'(\boldsymbol{r})\boldsymbol{\eta} \tag{41}$$

Such an approximation allows us to introduce a measure of *synchronization error* due to noise, which reads:

$$E := \|\boldsymbol{s} - \boldsymbol{\phi}(\boldsymbol{r})\| \approx \|\boldsymbol{\phi}'\boldsymbol{\eta}\| \leq \|\boldsymbol{\phi}'\|\|\boldsymbol{\eta}\| \tag{42}$$

For the observer task (see V-C), in the common case of a linear readout, $\boldsymbol{\phi}'$ is simply the pseudo-inverse of the readout matrix $\boldsymbol{W}^*_{\text{out}}$, whose singular values are the reciprocal of the singular values of $\boldsymbol{W}_{\text{out}}$. This implies the following bound on the synchronization error due to noise,

$$E \leq \|\boldsymbol{W}^*_{\text{out}}\|\|\boldsymbol{\eta}\| = \frac{\|\boldsymbol{\eta}\|}{\min_i \sigma_i(\boldsymbol{W}_{\text{out}})} \tag{43}$$

where $\sigma_i(\boldsymbol{W}_{\text{out}})$ denote the non-null singular values of $\boldsymbol{W}_{\text{out}}$.

REFERENCES

[1] VS Afraimovich, NN Verichev, and Ml I Rabinovich. "Stochastic synchronization of oscillation in dissipative systems". In: *Radiophysics and Quantum Electronics* 29.9 (1986), pp. 795–803.

[2] Sebastián Basterrech. "Empirical analysis of the necessary and sufficient conditions of the echo state property". In: *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2017, pp. 888–896.

[3] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult". In: *IEEE Transactions on Neural Networks* 5.2 (1994), pp. 157–166.

[4] Tyrus Berry, Dimitrios Giannakis, and John Harlim. "Bridging data science and dynamical systems theory". In: *arXiv preprint arXiv:2002.07928* (2020).

[5] F M Bianchi, Lorenzo Livi, and Cesare Alippi. "Investigating echo state networks dynamics by means of recurrence analysis". In: *IEEE Transactions on Neural Networks and Learning Systems* 29.2 (2018), pp. 427–439. DOI: 10.1109/TNNLS.2016.2630802.

[6] G D Birkhoff. "Proof of the ergodic theorem". In: *Proceedings of the National Academy of Sciences* 17.12 (1931), pp. 656–660.

[7] Stefano Boccaletti et al. "The synchronization of chaotic systems". In: *Physics Reports* 366.1-2 (2002), pp. 1–101.

[8] S Bompas, Bertrand Georgeot, and David Guéry-Odelin. "Accuracy of neural networks for the simulation of chaotic dynamics: precision of training data vs precision of the algorithm". In: *arXiv preprint arXiv:2008.04222* (2020).

[9] Jake Bouvrie and Boumediene Hamzi. "Kernel methods for the approximation of nonlinear systems". In: *SIAM Journal on Control and Optimization* 55.4 (2017), pp. 2460–2492. DOI: 10.1137/14096815X.

[10] Ken Caluwaerts et al. "The spectral radius remains a valid indicator of the echo state property for large reservoirs". In: *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2013, pp. 1–6.

[11] A Ceni et al. "The echo index and multistability in input-driven recurrent neural networks". In: *Physica D* 412 (2020). DOI: 10.1016/j.physd.2020.132609.

[12] Ashesh Chattopadhyay, Pedram Hassanzadeh, and Devika Subramanian. "Data-driven predictions of a multiscale Lorenz 96 chaotic system using machine-learning methods: reservoir computing, artificial neural network, and long short-term memory network". In: *Nonlinear Processes in Geophysics* 27.3 (2020), pp. 373–389.

[13] Claudio Gallicchio, Alessio Micheli, and Luca Pedrelli. "Deep reservoir computing: A critical experimental analysis". In: *Neurocomputing* 268 (2017), pp. 87–99.

[14] Surya Ganguli, Dongsung Huh, and Haim Sompolinsky. "Memory traces in dynamical systems". In: *Proceedings of the National Academy of Sciences* 105.48 (2008), pp. 18970–18975. DOI: 10.1073/pnas.0804451105.

[15] William Gilpin. "Deep learning of dynamical attractors from time series measurements". In: *arXiv preprint arXiv:2002.05909* (2020).

[16] L. Gonon, L. Grigoryeva, and J-P Ortega. "Memory and forecasting capacities of nonlinear recurrent networks". In: *Physica D: Nonlinear Phenomena* 414 (2020), p. 132721. ISSN: 0167-2789. DOI: 10.1016/j.physd.2020.132721.

[17] Alireza Goudarzi et al. "Memory and information processing in recurrent neural networks". In: *arXiv preprint arXiv:1604.06929* (2016).

[18] Lyudmila Grigoryeva and Juan-Pablo Ortega. "Echo state networks are universal". In: *Neural Networks* 108 (2018), pp. 495–508.

[19] Allen G Hart, James L Hook, and Jonathan HP Dawes. "Echo State Networks trained by Tikhonov least squares are L2(μ) approximators of ergodic dynamical systems". In: *arXiv preprint arXiv:2005.06967* (2020).

[20] Allen Hart, James Hook, and Jonathan Dawes. "Embedding and approximation theorems for echo state networks". In: *Neural Networks* 128 (2020), pp. 234–247. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2020.05.013.

[21] Herbert Jaeger. "The "echo state" approach to analysing and training recurrent neural networks-with an erratum note". In: *Bonn, Germany: German National Research Center for Information Technology GMD Technical Report* 148.34 (2001), p. 13.

[22] Ljupco Kocarev and Ulrich Parlitz. "Generalized synchronization, predictability, and equivalence of unidirectionally coupled dynamical systems". In: *Physical Review Letters* 76.11 (1996), p. 1816.

[23] Guan-Horng Liu and Evangelos A Theodorou. "Deep learning theory review: An optimal control and dynamical systems perspective". In: *arXiv preprint arXiv:1908.10920* (2019).

[24] Sigurd Løkse, Filippo Maria Bianchi, and Robert Jenssen. "Training echo state networks with regularization through dimensionality reduction". In: *Cognitive Computation* 9.3 (2017), pp. 364–378.

[25] E N Lorenz. "Deterministic nonperiodic flow". In: *Journal of the Atmospheric Sciences* 20.2 (1963), pp. 130–141.

[26] Zhixin Lu and Danielle S Bassett. "Invertible generalized synchronization: A putative mechanism for implicit learning in neural systems". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 30.6 (2020), p. 063133.

[27] Zhixin Lu, Brian R Hunt, and Edward Ott. "Attractor reconstruction by machine learning". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 28.6 (2018), p. 061104.

[28] Zhixin Lu et al. "Reservoir observers: Model-free inference of unmeasured variables in chaotic systems". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 27.4 (2017), p. 041102.

[29] Thomas Lymburn et al. "The reservoir's perspective on generalized synchronization". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 29.9 (2019), p. 093133.

[30] Wolfgang Maass, Thomas Natschläger, and Henry Markram. "Real-time computing without stable states: A new framework for neural computation based on perturbations". In: *Neural Computation* 14.11 (2002), pp. 2531–2560.

[31] Gandhi Manjunath and Herbert Jaeger. "Echo state property linked to an input: Exploring a fundamental characteristic of recurrent neural networks". In: *Neural Computation* 25.3 (2013), pp. 671–696.

[32] Sarah Marzen. "Difference between memory and prediction in linear recurrent networks". In: *Physical Review E* 96.3 (2017), p. 032308. DOI: 10.1103/PhysRevE.96.032308.

[33] Marc Massar and Serge Massar. "Mean-field theory of echo state networks". In: *Physical Review E* 87.4 (2013), p. 042809.

[34] Francesca Mastrogiuseppe and Srdjan Ostojic. "A geometrical analysis of global stability in trained feedback networks". In: *Neural computation* 31.6 (2019), pp. 1139–1182.

[35] Ulrich Parlitz. "Detecting generalized synchronization". In: *Nonlinear Theory and Its Applications, IEICE* 3.2 (2012), pp. 113–127.

[36] L M Pecora and T L Carroll. "Synchronization in chaotic systems". In: *Physical Review Letters* 64.8 (1990), p. 821.

[37] L M Pecora et al. "Fundamentals of synchronization in chaotic systems, concepts, and applications". In: *Chaos: An Interdisciplinary Journal of Nonlinear Science* 7.4 (1997), pp. 520–543.

[38] Kestutis Pyragas. "Weak and strong synchronization of chaos". In: *Physical Review E* 54.5 (1996), R4508.

[39] Di Qi and Andrew J Majda. "Using machine learning to predict extreme events in complex systems". In: *Proceedings of the National Academy of Sciences* 117.1 (2020), pp. 52–59.

[40] Alexander Rivkind and Omri Barak. "Local dynamics in trained recurrent neural networks". In: *Physical Review Letters* 118.25 (2017), p. 258101.

[41] O E Rössler. "An equation for continuous chaos". In: *Physics Letters A* 57.5 (1976), pp. 397–398.

[42] Nikolai F Rulkov et al. "Generalized synchronization of chaos in directionally coupled chaotic systems". In: *Physical Review E* 51.2 (1995), p. 980.

[43] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.

[44] Zhiwei Shi and Min Han. "Support vector echo-state machine for chaotic time-series prediction". In: *IEEE Transactions on Neural Networks* 18.2 (2007), pp. 359–372.

[45] David Sussillo and Larry F Abbott. "Generating coherent patterns of activity from chaotic neural networks". In: *Neuron* 63.4 (2009), pp. 544–557.

[46] Floris Takens. "Detecting strange attractors in turbulence". In: *Dynamical Systems and Turbulence*. Springer, 1981, pp. 366–381.

[47] Gouhei Tanaka et al. "Recent advances in physical reservoir computing: A review". In: *Neural Networks* 115 (2019), pp. 100–123. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2019.03.005.

[48] Peter Tiňo. "Dynamical Systems as Temporal Feature Spaces." In: *Journal of Machine Learning Research* 21.44 (2020), pp. 1–42.

[49] Peter Tiňo and Georg Dorffner. "Predicting the future of discrete sequences from fractal representations of the past". In: *Machine Learning* 45.2 (2001), pp. 187–217.

[50] Jonathan H Tu et al. "On dynamic mode decomposition: Theory and applications". In: *arXiv preprint arXiv:1312.0041* (2013).

[51] David Verstraeten et al. "An experimental unification of reservoir computing methods". In: *Neural Networks* 20.3 (2007), pp. 391–403.

[52] David Verstraeten et al. "Memory versus non-linearity in reservoirs". In: *The 2010 international joint conference on neural networks (IJCNN)*. IEEE. 2010, pp. 1–8.

[53] Pietro Verzelli et al. "Input representation in recurrent neural networks dynamics". In: *arXiv preprint arXiv:2003.10585* (2020).

[54] Pantelis R Vlachas et al. "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics". In: *Neural Networks* (2020).

[55] Tongfeng Weng et al. "Synchronization of chaotic systems and their machine-learning models". In: *Physical Review E* 99.4 (2019), p. 042203.

[56] I B Yildiz, Herbert Jaeger, and S J Kiebel. "Re-visiting the echo state property". In: *Neural Networks* 35 (2012), pp. 1–9.

[57] Bai Zhang, David J Miller, and Yue Wang. "Nonlinear system modeling with random matrices: echo state networks revisited". In: *IEEE Transactions on Neural Networks and Learning Systems* 23.1 (2011), pp. 175–182.

Pietro Verzelli received the B.Sc. in Physics from the University of Bologna in 2015 and the M.Sc. in Physics of Complex System from the University of Turin in 2017. Currently he is a Ph.D. student with the Faculty of Informatics at Universitá della Svizzera Italiana, in Lugano (Switzerland). His research focuses on the relationships between dynamical systems and machine learning, with a particular emphasis on recurrent neural networks and reservoir computing.

Cesare Alippi (F'06) received the degree in electronic engineering cum laude in 1990 and the PhD in 1995 from Politecnico di Milano, Italy. Currently, he is a Full Professor of information processing systems with Politecnico di Milano, Italy, and of Cyber-Phsical and embedded systems at the Universitá della Svizzera Italiana, Switzerland. He has been a visiting researcher at UCL (UK), MIT (USA), ESPCI (F), CASIA (RC), A*STAR (SIN). Alippi is an IEEE Fellow, Distinguished lecturer of the IEEE CIS, Member of the Board of Governors of INNS, Vice-President education of IEEE CIS, Associate editor (AE) of the IEEE Computational Intelligence Magazine, past AE of the IEEE-Trans. Instrumentation and Measurements, IEEE-Trans. Neural Networks, and member and chair of other IEEE committees. In 2004 he received the IEEE Instrumentation and Measurement Society Young Engineer Award; in 2013 he received the IBM Faculty Award. He was also awarded the 2016 IEEE TNNLS outstanding paper award and the 2016 INNS Gabor award. Among the others, Alippi was General chair of the International Joint Conference on Neural Networks (IJCNN) in 2012, Program chair in 2014, Co-Chair in 2011. He was General chair of the IEEE Symposium Series on Computational Intelligence 2014. Current research activity addresses adaptation and learning in non-stationary environments and Intelligence for embedded systems. Alippi holds 5 patents, has published in 2014 a monograph with Springer on "Intelligence for embedded systems" and (co)-authored more than 200 papers in international journals and conference proceedings.

Lorenzo Livi (M'14) received the B.Sc. degree and M.Sc. degree from the Department of Computer Science, Sapienza University of Rome, Italy, in 2007 and 2010, respectively, and the Ph.D. degree from the Department of Information Engineering, Electronics, and Telecommunications at Sapienza University of Rome, in 2014. He has been with the ICT industry during his studies. From January 2014 until April 2016, he was a Post Doctoral Fellow at Ryerson University, Toronto, Canada. From May 2016 until September 2016, he was a Post Doctoral Fellow at the Politecnico di Milano, Italy and Universita' della Svizzera Italiana, Lugano, Switzerland. Currently, he is an Assistant Professor jointly appointed with the Departments of Computer Science and Mathematics at the University of Manitoba, Canada. He is also a Lecturer (Assistant Professor) in Data Science at the Department of Computer Science, University of Exeter, United Kingdom. In November 2018, Dr. Livi was awarded the prestigious Tier 2 Canada Research Chair in Complex Data. He is an Associate Editor of the IEEE-TNNLS, Applied Soft Computing (Elsevier) and a regular reviewer for several international journals, including IEEE Transactions and Elsevier journals. His research interests include Machine Learning, Time Series Analysis and Complex Dynamical Systems, with focused applications in Systems Biology and Computational Neuroscience.