Learning Using Thinned Networks: A Crowdsourcing Phenomena in Reservoir Computing

J Jamieson,[1, a)] D. J. Passey,[2, b)] B. Webb,[1, c)] and J. Wilkes[1, d)]
[1)] *Brigham Young University*
[2)] *University of North Carolina at Chapel Hill*

(Dated: 15 April 2022)

Abstract: Reservoir computers rely on an internal network to predict the future state(s) of dynamical processes. To understand how a reservoir's accuracy depends on this network we study how varying the network's topology affects the reservoir's ability to predict the dynamics on the Lorenz attractor. We find that thinned networks, in which a large majority of the network's edges have been removed, statistically outperform standard networks typically used in reservoir computing. This finding is similar in a number of way to phenomena observed in crowdsourcing. Specifically, in crowdsourcing, responses gathered from a crowd improve the more independent the individuals act and the more diverse the individuals are. In the same way thinned networks create a more diverse response from the nodes that make up the reservoir's internal network allowing for a more accurate reconstruction of the Lorenz attractor. This leads us to propose an answer to a long-standing question in reservoir computing regarding how to choose a reservoir's spectral radius where the optimal spectral radius depends on how thinned the reservoir's internal network is. We also show how the oscillations in a reservoir's response to input is determined by the reservoir's spectral radius when using thinned networks.

Reservoir computers are one of the few machine learning models to use an arbitrary internal network. As such, reservoirs have the potential to reveal connections between learning problems and network structure in ways that state of the art feed-forward and recurrent neural cannot. In practice it is typical in reservoir computing to use a network that has a real-world like topology. We test several different complex network topologies and investigate how the associated reservoir's ability to learn changes as edges are removed from the network. We find evidence that extremely sparse networks are better suited for learning the dynamics of the Lorenz attractor. We also find that reservoirs have features similar to those found in the distributed problem solving method of crowdsourcing. We offer some analysis of these sparse reservoir computers as an explanation for why this occurs.

I. INTRODUCTION

A reservoir computer is a machine learning model that can be used to predict the future state(s) of time-dependent processes. Applications of reservoir computers include the modeling of dynamical systems, pattern classification, modeling robotics controllers, event predictions, and speech recognition[1].

To train a reservoir, data, in the form of an input-signal, is fed into the reservoir creating a response from each of the reservoir's internal nodes. These nodes, which are linked together to form the reservoir's internal network, respond both according to the input-signal and their interactions they have with other nodes. Reservoir computers are one of the few machine learning models that use an arbitrary internal network to process in-coming information. In practice, these networks are often chosen to have a *complex network* topology, i.e. a topology that mimics the structure of real-world networks[2–7]. In some instances it has been shown that a real-world structure improves the performance of reservoir computers. For example, Watts-Strogatz type networks have been shown to improve prediction accuracy for certain types of reservoirs including echo state networks and liquid state machines[4,5].

In this paper our goal is to understand to what extent a reservoir's internal topology is responsible for its ability to (i) accurately create short-term forecasts and (ii) capture the long-term dynamical properties of a given input-signal. To do this we consider the Lorenz system, specifically the dynamics on the Lorenz attractor. The particular reservoirs we study are descendants of echo state networks[8] and liquid state machines[9], and are of the ODE variety considered by Lu[6] et al.

The study of (i) and (ii) is known as *attractor reconstruction*, which has been studied at least since the 1980s using a wide variety of machine learning models[10–13]. This includes neural networks and deep learning reservoir models. In these studies reservoir computers were found to be competitive with deep learning methods, even outperforming RNNs and LSTMs on specific tasks including the reconstruction of dynamical systems[14–16]. In such tasks reservoirs have been shown to be able to learn and reproduce characteristic features of chaotic systems including their first return maps, Lyapunov exponents, etc.[6,17,18].

To understand how a reservoir's performance depends on the structure of its internal network we study how varying the topology of this network affects the reservoir's ability to make short and long-term predictions on the Lorenz attractor. The networks we consider are of two types: The first are *complex network models* which include Erdos-Renyi, Watts-Strogatz, Barabasi-Albert, Random Digraph, and Random Geometric

[a)] Electronic mail: joebobjamieson@gmail.com
[b)] Electronic mail: djpassey@unc.edu
[c)] Electronic mail: bwebb@math.byu.edu
[d)] Electronic mail: joseph.benson.wilkes@gmail.com

graphs. The second are *artificial network topologies* that we consider to determine whether real-world topologies are, in fact, better suited to learn the Lorenz attractor. We also consider how randomly removing a fixed percentage of the edges from these networks affects reservoir accuracy. We refer to such networks as *thinned networks*.

A major challenge in this endeavor of analyzing the effect network topology has on reservoir performance is the many possible parameter combinations to consider. Reservoir computers have *intrinsic parameters*, which influence how a reservoir processes incoming-data, and *topological parameters* that describe the structure of the reservoir's internal network. The sheer number of possible network structures combined with the reservoir's parameters presents a significant challenge in determining which reservoirs are effective at predicting the future states of time-dependent processes.

In order to understand which topologies are most effective we analyze an extensive data set consisting of over one-hundred million numerical experiments. Using this data set we investigate how a reservoir's (a) internal network topology and (b) intrinsic reservoir parameters effect a reservoir's ability to learn and predict the chaotic dynamics of the Lorenz attractor. What our experiments suggest is that thinned networks perform at least as well as the diverse real-world models and artificial network topologies we consider. In fact, networks that are *extremely thinned*, in which 90% or more of the edges are removed, statistically outperform the large majority of other reservoirs we consider. We note that this is in direct contrast to the way internal networks are typically chosen in the literature[2–7] as these "thinned" networks are fractured to the point that most nodes are isolated. This idea that sparse networks can be effective in reservoir computing has been suggested before[19,20]. However, we take a much more detailed look at this phenomena and specifically its connections to research in the social sciences.

The result that independently acting nodes can be highly accurate in learning and recreating dynamical processes leads us to phenomena often associated with crowdsourcing. Crowdsourcing can be thought of as a means of leveraging the collective intelligence, abilities, and/or resources of a group of individuals[21–23]. In gathering information from a crowd, i.e. its *responses*, a number of conditions have been observed that often improve the quality of a group's response. First, the responses improve the more individuals act *independently* and in a *decentralized* manner within the group[21]. Second, *diversity* among individuals is desired as too much consensus can lead to limited types of responses[24–26]. The idea is that a high level of agreement within a group weakens the group's ability to make accurate judgements and collective decisions[27].

Thinned reservoir computers appear to have features that are in line with those observed in crowdsourcing. If the reservoir's internal network is extremely thinned, i.e. there are very few connections between reservoir nodes, the response of the reservoir nodes to a given input-signal will typically have no influence each other. In this sense the responses are independent. In our experiments we find that this results in more *diverse* responses from the reservoir nodes and, similar to crowdscourcing, this increase in diversity directly corresponds to an improvement in reservoir accuracy, specifically in our ability to recreate the Lorenz attractor (see Section VI). Thus, *independence* and *decentralization* appear to play a similar role in both crowdsourcing and reservoir computers by allowing for greater diversity in the crowd and reservoir's response, respectively.

To understand how independence and response diversity are linked in reservoir computers we consider how the reservoir's spectral radius can be used in conjunction with thinning to tune diversity in the reservoir's response. We find that if the network is at least modestly connected, increasing its spectral radius has a homogenizing effect on the reservoir's response vector, reducing response diversity and prediction accuracy. However, if the network is thinned to where the network consists of mostly isolated nodes, increasing its spectral radius appears to increase this diversity, thereby improving reservoir accuracy (see Section VI).

Related to this phenomena is a long-standing question in reservoir computing regarding how to choose the spectral radius of a reservoir computer. For instance, in discrete-time a reservoir's optimal performance is thought to occur near "criticality" or near a spectral radius of one[28–30]. Our results help to at least partially answer how to choose a spectral radius. For highly connected networks we find that picking a smaller spectral radius will typically improve reservoir performance as this decreases dependence between nodes and increases response diversity. For thinned networks choosing a larger, often much larger, spectral radius will have a similar effect of increasing response diversity and improving reservoir performance.

As thinned networks have a relatively simple structure it is possible to analytically study the dynamics of the associated reservoir as it relates to the reservoir's spectral radius. We find that the spectral radius of a thinned reservoir determines the scale at which isolated nodes respond to oscillations in a given input-signal. That is, by tuning the spectral radius we can effectively filter out smaller oscillations in the reservoir's response (see Theorems 1 and 2 in Section VII). We note that this interpretation of spectral radius is quite different from that found in other studies of reservoir dynamics where a reservoir's spectral radius is often chosen to guarantee that differences in initial conditions will disappear over time.[29,31]

We note that the notion of a thinned network, i.e. a network consisting mainly of isolated vertices, is not well studied in network science. To understand this class of networks and their relationship to other more studied networks we make the following distinction. We note that there is a difference between the reservoir's internal network before and after training. In the reservoirs we consider, the reservoir's initial internal network is used to process an input-signal by allowing one node's response to effect another node's response. As such, we refer to this network as the reservoir's *processing network*. Once the reservoir is trained its internal network is updated and consists of the original processing network together with a scaled version of the network used to aggregate the reservoir node responses to input (see Section II). We refer to this network in the trained reservoir as the reservoir's *aggregate network*.

In our numerical experiments we find that even when we use an extremely thinned processing network in which nearly all nodes are isolated, the associated aggregate network is much more complex (see Figures 4 and 6 from Examples 1 and 2, respectively). This may suggest that the complex structure of real-world networks may not be due to the need to process information but rather the need to link and to store this information using some form of information aggregation. That is, structures that link or aggregate information are potentially not the same as those that process or learn from information (see Example 3). If true it is possible that in reservoir computing we are confusing the usefulness of processing networks with aggregate networks when we choose a reservoir's initial internal network to have a real-world like structure.

The paper is organized as follows. In Section II we describe the specific reservoir computing model we consider. In Section III we describe how this model can be used to reconstruct the Lorenz attractor both in terms of short-term forcasts and long-term dynamical properties. In Section IV we introduce the network models we consider and our method of thinning these networks. We then describe how effective both thinned and nonthinned networks are at creating accurate short-term forecasts on the Lorenz attractor. In Section VI we describe how these results are related to crowdsourcing. Specifically, we describe how thinning, spectral radius, response diversity, and reservoir accuracy are related to phenomena observed in aggregating group responses. In Section VII we analytically investigate the properties of thinned reservoirs specifically describing the effect spectral radius has on the response of thinned networks. We conclude in Section VIII with some comments and open questions.

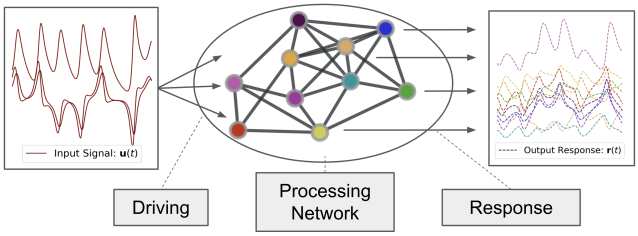


FIG. 1. Reservoir Processing: A reservoir computer is *trained* by driving the nodes of its processing network by an input-signal $\mathbf{u}(t) \in \mathbb{R}^m$ over the time interval $t \in [0,T]$. The *response* $\mathbf{r}(t) \in \mathbb{R}^n$ of the nodes to the input-signal over the same time interval is given by the solution to Equation (1).

II. RESERVOIR COMPUTING

A reservoir computer is a machine learning model that is used to learn and make predictions regarding time-dependent data[32]. Given an input-signal $\mathbf{u}(t) \in \mathbb{R}^m$ for $t \in [0,T]$ the goal is to use a reservoir to accurately predict future values of this series, i.e. predict $\mathbf{u}(t)$ for $t > T$. This is done in a sequence of three steps which we refer to as *processing*, *aggregation*, and *prediction*. We choose to use this terminology as it directly relates to concepts found in crowdscourcing (see Section VI).

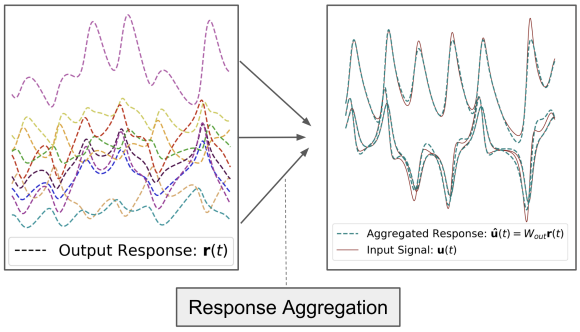


FIG. 2. Response Aggregation: The response vector $\mathbf{r}(t) \in \mathbb{R}^n$ is aggregated by determining the matrix $\mathbf{W}_{\text{out}} \in \mathbb{R}^{m\times n}$ that minimizes the sum in Equation (2). The result is the aggregated response vector $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t) \in \mathbb{R}^m$ that best approximates the input-signal $\mathbf{u}(t)$ for $t \in [0,T]$.

In the processing step the nodes of a network are driven by the input-signal $\mathbf{u}(t)$ over the time interval $t \in [0,T]$. In reservoir computing this network, which we refer to the as a *processing network*, can be any network. The network is given by the matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$ where $\mathbf{A}_{ij}$ represents the weight of the network connection from node j to node i. Here $\mathbf{A}$ is scaled for convenience so that it has a spectral radius of 1.

The state of the nodes within the processing network evolve according to the differential equation

$$\frac{d}{dt}\mathbf{r}(t) = \gamma[-\mathbf{r}(t) + \tanh\big(\rho\mathbf{A}\mathbf{r}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{u}(t)\big)] \qquad (1)$$

for $t \in [0,T]$ where $\mathbf{r}(t) \in \mathbb{R}^n$ is the vector representing the state of the nodes in the reservoir at time t. The matrix $\mathbf{W}_{\text{in}} \in \mathbb{R}^{n\times m}$ in Equation (1) is a fixed linear mapping that sends a linear combination of the m-dimensional training data $\mathbf{u}(t)$ to each of the n reservoir nodes. As $\mathbf{W}_{\text{in}}$ allows each node to observe some combination of the training signal we refer to it as the reservoir's *observation matrix*. Similar to other studies, we choose each entry of the observation matrix uniformly at random with $[\mathbf{W}_{\text{in}}]_{ij} \sim U(-0.5, 0.5)$. The function $\tanh(\cdot)$ in Equation (1) is applied element-wise and γ, ρ, and σ are nonnegative scalar parameters of the reservoir. The parameter ρ is referred to as the *spectral radius* of the reservoir since the matrix $\rho\mathbf{A} \in \mathbb{R}^{n\times n}$ in Equation (1) has spectral radius ρ.

As the input-signal $\mathbf{u}(t)$ appears in Equation (1) the nodes of the processing network depend on, or are *driven* by, this time-series. Their *response* to this input signal is given by the solution $\mathbf{r}(t)$ to this differential equation for $t \in [0,T]$ (see Figure 1).

The second phase in creating a reservoir is aggregating the network responses. Aggregation is done by first discretizing the time interval $[0,T]$ into equal time-steps each of length τ. The result is the discrete time-sequence $\{\tau_\ell\}_{\ell=0}^{L}$ where $\tau_\ell = \ell\cdot\tau$ and $L\cdot\tau = T$. From this discetization we determine the matrix

$$\mathbf{W}_{\text{out}} = \underset{\mathbf{W}\in\mathbb{R}^{m\times n}}{\operatorname{argmin}} \left[\sum_{\ell=0}^{L} \|\mathbf{W}\mathbf{r}(t_\ell) - \mathbf{u}(t_\ell)\|_2^2 + \alpha\|\mathbf{W}\|_2^2\right] \qquad (2)$$

that minimizes the sum on the right. The parameter $\alpha > 0$ specifies the amount of regularization in this minimization process to prevent overfitting.

This process results in the mapping $\mathbf{W}_{\text{out}} \in \mathbb{R}^{m\times n}$ which we use to aggregate the network responses into the vector $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t) \in \mathbb{R}^m$ (see Figure 2). In practice, α is typically small so that these *aggregated responses* can be thought of as a proxy for the input-signal, i.e. $\hat{\mathbf{u}}(t) \approx \mathbf{u}(t)$ for $t \in [0,T]$.

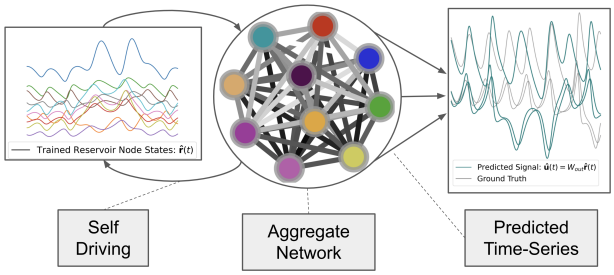


FIG. 3. Reservoir Prediction: Replacing the input-signal $\mathbf{u}(t)$ in Equation (1) by the aggregate response vector $\hat{\mathbf{u}}(t)$ results in the trained reservoir given by Equation (3). This procedure updates the reservoir's original processing network resulting in the new aggregate network. Once the input-signal is removed the trained reservoir drives itself giving the output $\hat{\mathbf{r}}(t) \in \mathbb{R}^m$ and the predicted time-series $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ that can be compared to the true time-series.

The prediction step begins by replacing the training data $\mathbf{u}(t)$ by $\hat{\mathbf{u}}(t)$ in Equation (1). This results in the autonomous differential equation

$$\frac{d}{dt}\hat{\mathbf{r}}(t) = \gamma[-\hat{\mathbf{r}}(t) + \tanh\big([\rho\mathbf{A} + \sigma\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}]\hat{\mathbf{r}}(t)\big)\,], \qquad (3)$$

which removes the system's dependency on the training data. Because of this we refer to Equation (3) as the *trained system*. This system has the new internal network given by the matrix $\hat{\mathbf{A}} = \rho\mathbf{A} + \sigma\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}} \in \mathbb{R}^{n\times n}$, which we call the system's *aggregate network* as it combines the original adjacency matrix $\mathbf{A}$ with a scaled version of the matrix $\mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}$ that aggregates the responses $\mathbf{r}(t)$.

To predict the future values of $\mathbf{u}(t)$ we initialize the trained system at the value $\hat{\mathbf{r}}(T) = \mathbf{r}(T)$ at the end of the training period. This produces the time series $\hat{\mathbf{r}}(t)$ for $t \in (T,\infty)$ from which we create the *predicted time-series* $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t) \in \mathbb{R}^m$ for all future times (see Figure 3).

In the following section we analyze the extent to which reservoir computers can learn the dynamics of the Lorenz system, specifically the dynamics on its attractor. Our main focus in what follows is to understand what type(s) of internal processing network are best suited for this task.

III. RECONSTRUCTION OF THE LORENZ ATTRACTOR

Here we consider a reservoir's ability to (i) accurately create short-term forecasts on a chaotic attractor and (ii) capture the long-term dynamic properties of the attractor. The specific system we consider is the Lorenz system given by

$$\begin{aligned} dx/dt &= 10(y-x) \\ dy/dt &= x(28-z)-y \\ dz/dt &= xy - 8z/3 \end{aligned}$$

where the system's parameters are chosen to yield chaotic dynamics[33].

A. Short-Term Forecasts on the Lorenz Attractor

We can think of the predicted time-series $\hat{\mathbf{u}}(t)$ as a *forecast* of the future state of the input-signal $\mathbf{u}(t)$. Forecasts on the Lorenz attracator are typically accurate for only a short amount of time owning to the chaotic nature of the attractor. This prediction time is defined as follows.

Definition 1. ***(Valid Prediction Time)*** *Given the input-signal $\mathbf{u}(t)$ for $t \in [0,T]$ and tolerance $\varepsilon > 0$ the vector $\hat{\mathbf{u}}(t)$ has a valid prediction time (VPT) of $t_* = T_* - T$ where $T_* > T$ is the first time the inequality*

$$\|\hat{\mathbf{u}}(t) - \mathbf{u}(t)\|_2 < \varepsilon \qquad (4)$$

does not hold. That is, the T_* is the first time following T that the difference in Equation (4) *exceeds the tolerance ε.*

A valid prediction time gives a measure of how well a trained reservoir can recreate the dynamics of a single trajectory $\mathbf{u}(t)$ on the Lorenz attractor.

Example 1. ***(Standard Processing Network)*** *Let $\mathbf{u}(t) \in \mathbb{R}^3$ for $t \in [0,40]$ be the finite trajectory on the Lorenz attractor shown in grey in Figure 4 (bottom center). Here we choose our processing network to be an Erdos-Renyi graph with $n = 50$ nodes with the probability $p = 0.08$ of an edge forming between any two nodes (see Section IV for more details). The resulting network with weighted adjacency matrix $\mathbf{A} \in \mathbb{R}^{50\times 50}$ is shown in Figure 4 (top left). For $\gamma = 5$, $\rho = 5$, and $\sigma = 0.05$ the resulting response vector $\mathbf{r}(t) \in \mathbb{R}^{50}$ for $t \in [0,40]$ is shown in Figure 4 (top center) where each colored response is associated with the node of the same color. Computing $\mathbf{W}_{out}$ using $\alpha = 0.0001$ yields the aggregate network with adjacency matrix $\hat{\mathbf{A}}$ shown in Figure 4 (bottom left). From this we compute the aggregate response vector $\hat{\mathbf{u}}(t)$ shown together with the true trajectory $\mathbf{u}(t)$ for $t \in [40,45]$ in Figure 4 (bottom center).*

The agreement between the actual and predicted trajectories in this example is marginal at best. In particular, setting $\varepsilon = 5$ the inequality $\|\hat{\mathbf{u}}(t) - \mathbf{u}(t)\|_2 \leq \varepsilon$ holds only so long as $t \in [40, 40.8]$ meaning that, for the tolerance $\varepsilon = 5$, we have a valid prediction time of length $t_* = 0.8$.

Although a valid prediction time of $t_* = 0.8$ may seem short when compared with the reservoir's training time of $T = 40$ the fact that the chaotic dynamics on the Lorenz attractor can be approximated by a reservoir is perhaps itself surprising, especially for those unfamiliar with reservoir computing. In the following section we briefly consider the extent to which reservoir computers can also recreate the long-term dynamical properties of the Lorenz attractor.

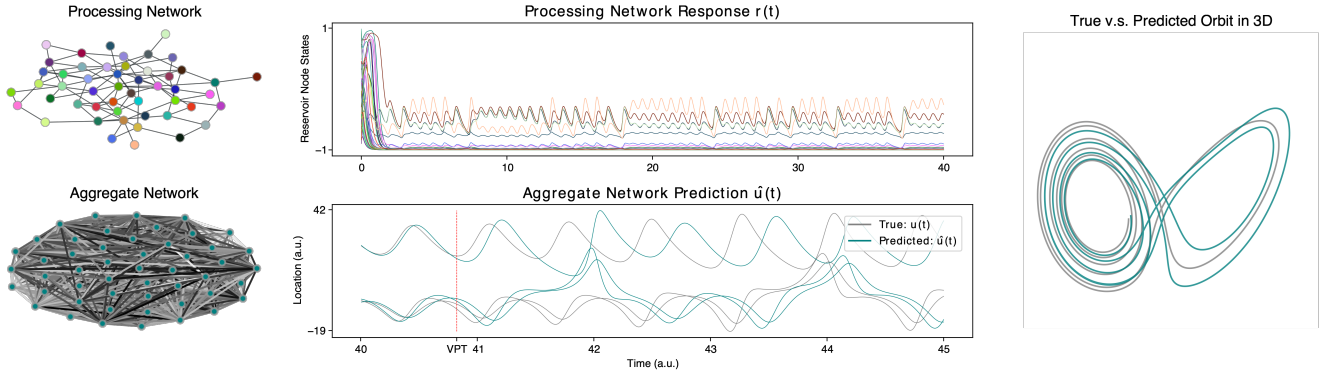


FIG. 4. Top Left: An Erdos-Renyi processing network is used to predict future values of the finite trajectory $\mathbf{u}(t)$ for $t \in [0,40]$ on the Lorenz attractor. Top Center: The resulting response vector $\mathbf{r}(t)$ for $t \in [0,40]$ is shown where each response color corresponds to the node of the same color. Bottom Left: The associated aggregate network is shown where edge color indicates the magnitude of the edge weight with lighter colors corresponding to larger magnitudes. Bottom Center: A comparison between the true trajectory $\mathbf{u}(t)$ and the predicted trajectory $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ for $t \in [40,45]$ is shown in grey and blue, respectively. The three curves indicate the x, y, and z-coordinates of each trajectory. The dashed line indicates the valid prediction time (VPT) $t_* = 0.8$, which is the first time at which the inequality $\|\hat{\mathbf{u}}(t) - \mathbf{u}(t)\| < \varepsilon$ is exceeded for the tolerance $\varepsilon = 5$. Right: The true and predicted trajectories $\mathbf{u}(t)$ and $\hat{\mathbf{u}}(t)$ are shown together for $t \in [0,45]$ in $\mathbb{R}^3$.

B. Long-Term Properties of the Lorenz Attractor

The short-term predictions $\hat{\mathbf{u}}(t)$ of the previous section are often compared with generating short-term *weather* forecasts. Here we also consider the extent to which reservoir computers can replicate the long-term or global features of the Lorenz attractor. This can be compared with replicating not the *weather* but the *climate* on the attractor. Taking our lead from (Lu 2018) we consider the first-return map $Z^n_{\max}$ of the z-coordinate's n^{th} local maxima on the Lorenz attractor[6]. Shown numerically in Figure 5 is this map with $Z^n_{\max}$ plotted against $Z^{n+1}_{\max}$. In the figure the same first-return map is shown in red for a reservoir computer trained on the Lorenz attractor using an Erdos-Renyi graph with $n = 500$ nodes. Here the trained reservoir's return map only loosely approximates that of the Lorenz attractor. That is, the reservoir does not do particularly well in replicating the long-term dynamical properties of the attractor including Lyapunov exponents, etc.

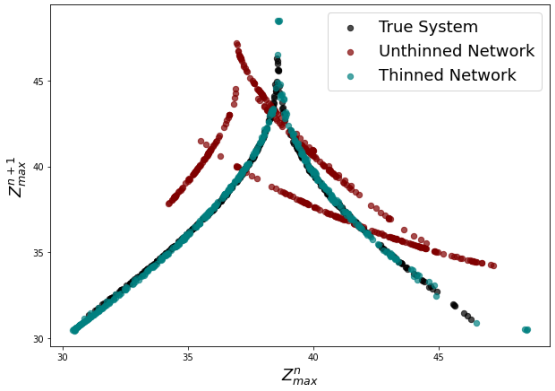


FIG. 5. The first-return map $Z^n_{\max}$ of the z-coordinate's n^{th} local maxima on the Lorenz attractor is plotted versus $Z^{n+1}_{\max}$ for $n \in [0,300]$. The same is done for two trained reservoirs. Shown in red is the first-return map for a reservoir with a standard Erdos-Renyi processing network with five-hundred nodes. In blue is the first-return map for a thinned version of this reservoir with $p_{\text{thin}} = 0.95$. The reservoir's hyper-parameters in both the thinned and unthinned cases are $\rho = 25, \sigma = 0.05, \gamma = 5$, and $\alpha = 0.0001$.

IV. THINNED PROCESSING NETWORKS AND RESERVOIR PREDICTIONS

The primary question we consider in this paper is how our choice of processing network effects a reservoir's ability to forecast trajectories on the Lorenz attractor. In practice the networks chosen to be a reservoir's processing network are networks that have a real-world like topology.[2–7] The motivation behind this choice is likely based on the observation that many natural and technological networks are known to process large amounts of information.

A. Thinned Processing Networks

Not only do we consider how effective different networks are as processing networks but also how thinned versions of these networks perform as processing networks in reservoir computers. Thinning a network is done in two steps: Once a graph is created we build a directed version of the graph if it is undirected. This is done by replacing each of the graph's undirected edges with two directed edges; one edge pointing from the first to the second node, the other pointing from the second to the first node. Once the graph is directed we then remove a fixed percentage $p_{\text{thin}} \in [0,1]$ of these directed edges uniformly and independently at random.

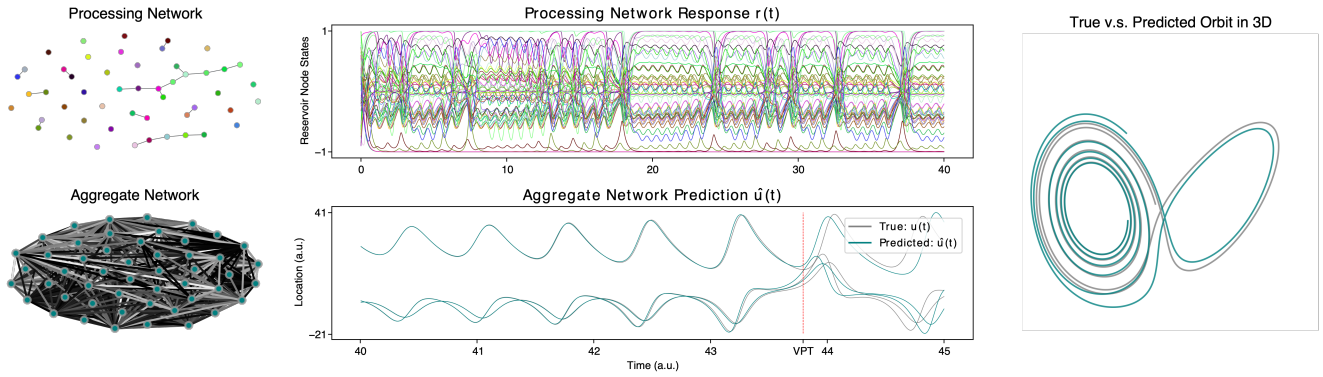


FIG. 6. Top Left: A thinned version of the Erdos-Renyi processing network in Figure 4 with $p_{\text{thin}} = 0.9$ is used to predict future values of the same finite trajectory $\mathbf{u}(t)$ as in Figure 4. Edges are directed but drawn without arrows for simplicity. Top Center: The resulting response vector $\mathbf{r}(t)$ for $t \in [0,40]$ is shown where each response color corresponds to the node of the same color. Bottom Left: The associated aggregate network is shown where edge colors indicate the strength of the connection, with lower strength shown in black and higher strength in lighter colors. Bottom Center: A comparison between the true trajectory $\mathbf{u}(t)$ and the predicted trajectory $\hat{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\hat{\mathbf{r}}(t)$ for $t \in [40,45]$ is shown in grey and blue, respectively. The three curves indicate the x, y, and z-coordinates of each trajectory. The dashed line indicates the valid prediction time (VPT) of $t_* = 3.75$, which is much larger than the valid prediction time shown in Figure 4 for the same tolerance. Right: The true and predicted trajectories $\mathbf{u}(t)$ and $\hat{\mathbf{u}}(t)$ are shown together for $t \in [0,45]$ in $\mathbb{R}^3$.

This random removal of edges, otherwise known as *bond percolation*, is known to fracture networks in specific ways[34]. In the following example we consider the effect thinning has on the accuracy of reservoir the processing network described in Example 1.

Example 2. ***(Thinned Processing Network)*** *Consider the finite trajectory $\mathbf{u}(t) \in \mathbb{R}^3$ for $t \in [0,40]$ on the Lorenz attractor from Example 1. Here, instead of using the original Erdos-Renyi processing network (cf. Figure 4 (top left)) we use a thinned version of the network with $p_{thin} = 0.9$ in which ninety percent of the edges are removed (see Figure 6 (top left)). Using the same parameters as in Example 1 the resulting response vector $\mathbf{r}(t)$, shown in Figure 6 (top center), is different in a number of ways from the response vector in Example 1. (These difference are discussed in Section VI.) Computing $\mathbf{W}_{out}$ gives us the aggregate network shown in Figure 6 (bottom left) from which we compute the aggregate response vector $\hat{\mathbf{u}}(t)$ which is shown together with the true trajectory $\mathbf{u}(t)$ for $t \in [40,45]$ (see Figure 4 (bottom center and right)).*

Here the agreement between the actual and predicted trajectories are much better than in Example 1. In particular, for the same tolerance $\varepsilon = 5$ the inequality $\|\hat{\mathbf{u}}(t) - \mathbf{u}(t)\|_2 \leq \varepsilon$ now holds for $t \in [40, 43.75]$ so that the VPT is $t_* = 3.75$, which is an increase in our short-term forecast in Example 1 by roughly a factor of five.

The difference in valid prediction times between Example 1 and Example 2 is somewhat striking especially if we consider the fact that the processing network in the latter is thinned to the point that it may seem unreasonable to refer to it as a *network*. However, this processing network gives us a much better short-term forecast in this specific case. Similarly, the reconstruction of long-term properties can also improve as the reservoir's processing network is thinned. This is demonstrated in Figure 5 where the return-map associated with a thinned version of the original reservoir more closely matches the return map of the Lorenz system than the original unthinned reservoir.

In the following section we analyze whether this happens in general. That is, we analyze whether thinning typically leads to better short-term forecasts on the Lorenz attractor.

V. THINNED PROCESSING NETWORKS AND RESERVOIR PREDICTIONS

In this section our specific goal is to investigate the extent to which thinning a reservoir's processing network effects the reservoir's ability to make short-term forecasts.

A. Network Models and Numerical Setup

We consider five different network models used to generate graphs with features observed in real-world networks. These are Erdos-Renyi, directed Erdos-Renyi (also known as random digraphs), random geometric graphs, Watts-Strogatz, and Barabasi-Albert type graphs[35–38].

(i) **Erdos-Renyi Model (ER)**: The Erdos-Renyi model is a two-parameter model with parameters $n \in \mathbb{N}$ and $p \in [0,1]$, where n is the number of network nodes and p is the probability that there is an *undirected* edge between nodes i and j for any pair of distinct nodes (see, for instance, Figure 4 (top left)).

(ii) **Random Digraph Model (RD)**: The random digraph model is the Erdos-Renyi model in which $n \in \mathbb{N}$ is the number of network nodes and $p \in [0,1]$ is the probability that a *directed* edge is placed from node i to node j in the network for any pair of possibly non-distinct nodes i and j. This is

the topology that is commonly used in the study of reservoir computers (see, for instance, Lu 2018).[6]

(iii) **Random Geometric Graph Model (RG)**: The random geometric graph model is also a two-parameter model with parameters $n \in \mathbb{N}$ and $r \geq 0$. A network is generated by placing n nodes uniformly at random in the unit square and connecting nodes that are within a distance r by an undirected edge.

(iv) **Watts-Strogatz Model (WS)**: The Watts-Strogatz model or *small-world network model* uses three parameters; $n \in \mathbb{N}$, $k \in \mathbb{N}$, and $q \in [0,1]$. Networks are formed by starting with an undirected cycle on n nodes with each node attached to its k nearest neighbors. For each node these edges are randomly reassigned with probability q to attach to another node where this other node is chosen uniformly at random.

(v) **Barabasi-Albert Model (BA)**: The Barabasi-Albert model produces a network by preferentially attaching new nodes to the existing network. The model has two parameters, $n \in \mathbb{N}$ and $m \in \mathbb{N}$, where n is the number of nodes and m is the number of new undirected edges attached to each new node when it enters the network.

To compare the effectiveness of different processing networks we consider both the real-world topologies (i)-(v) and a number of *artificial* network topologies. These artificial topologies include (vi) chain networks (CN), (vii) loop networks (LN), (viii) identity networks (IN), and (ix) empty networks (EN). A *chain network* consists of a single undirected cycle. In contrast, a *loop network* consists of a single directed cycle. An *identity network* is a network whose associated adjacency matrix is the identity matrix $\mathbf{A} = \mathbf{I} \in \mathbb{R}^{n\times n}$, and an *empty network* is a network whose associated adjacency matrix is the zero matrix $\mathbf{A} = \mathbf{0} \in \mathbb{R}^{n\times n}$. (We note that the zero matrix $\mathbf{A} = \mathbf{0}$ cannot be scaled to have spectral radius 1.) We consider these artificial networks to contrast our results with the real-network topologies typically chosen in reservoir computing. Our goal with thinning these network topologies is to understand how fracturing a processing network, either real-world like or not, effects a reservoir's ability to reconstruct the Lorenz attractor.

The major complication is that there are many possible parameter combinations to consider. To make this process tractable we limit these combinations by considering two types of parameters; the *topological parameters* that describe the specifics of the network models (i)-(ix) and the *reservoir parameters* found in Equations (1) and (2).

The topological parameters we consider are size n, thinning parameter p_{thin}, average degree c, attachment number m in Barabasi-Albert graphs; number of nearest neighbors k and the attachment parameter q in Watts-Strogatz graphs. The *average degree c* is the average number of directed or undirected edges attached to a node in a graph. In our experiments on Erdos-Renyi, random digraphs, and random geometric graphs we let c range over a fixed set of values (see below). In Erdos-Renyi graphs and random digraphs fixing an average degree c determines the edge probability p and radius size r for a given size n, respectively. Hence, we do not directly consider the parameters p and r in our numerical experiments.

For our topological parameters we consider the size and thinning parameters

$$\begin{aligned} n &= \{500, 1500, 2500\}; \text{ and} \\ p_{\text{thin}} &= \{0, .1, .2, \ldots, .7, .8, .82, .84, \ldots, .98, 1\}. \end{aligned} \tag{5}$$

The model specific parameters we consider are

$$\begin{aligned} &\text{(i) } \textit{Erdos-Renyi} : c = \{.5, 1, 2, 3, 4\}. \\ &\text{(ii) } \textit{Random Digraph} : c = \{.5, 1, 2, 3, 4\}. \\ &\text{(iii) } \textit{Random Geometric} : c = \{.5, 1, 2, 3, 4\}. \\ &\text{(iv) } \textit{Barabasi-Albert} : \ m = 1 \text{ (BA1) } \textit{and } m = 2 \text{ (BA2)}. \\ &\text{(v) } \textit{Watts-Strogatz} : \ k = 2 \text{ (WS2) } \textit{and } k = 4 \text{ (WS4)} \\ &\quad \textit{for } q = \{.01, .05, .1\}. \end{aligned} \tag{6}$$

We note that the artificial networks (vi)-(ix) are determined by the single parameter n and are therefore determined by the choice of parameters in (5).

The reservoir parameters are found in Equations (1) and (2) and are the parameters γ, ρ, σ and α. In our experiments, for each choice of topological parameters we select a set of reservoir parameters taken uniformly over the discrete hypercube

$$\begin{aligned} \gamma &\in \{0.1, 0.5, 1, 2, 5, 10, 25, 50\} \\ \rho &\in \{0.1, 0.9, 1, 1.1, 2, 5, 10, 25, 50\} \\ \sigma &\in \{10^{-3}, 50^{-3}, 10^{-2}, 50^{-2}, 0.14, 0.4, 0.7, 1, 10\} \\ \alpha &\in \{10^{-8}, 10^{-6}, 10^{-4}, 10^{-2}, 1\}. \end{aligned} \tag{7}$$

Each experiment consists of first fixing a set of topological and reservoir parameters then determining an input-signal $\mathbf{u}(t)$. This second step is done by choosing an initial random point near the Lorenz attractor. To remove any transient dynamics we run the point's trajectory forward in time for $t \in [-40, 0]$ starting at time $t = -40$ until it is sufficiently close to be considered *on* the attractor. Using this as our starting point we generated the time-series $\mathbf{u}(t)$ for $t \in [0, 40]$ using *lsoda* from the FORTRAN library *odepack* (via a python wrapper *scipy*).

To compute the matrix $\mathbf{W}_{out}$ in Equation (2) we use the discrete time-step $\tau = .001$ with $L = 4 \cdot 10^4$. As mentioned, the observation matrix $\mathbf{W}_{\text{in}}$ in each experiment is generated such that each entry of the matrix is chosen uniformly at random from the interval $(-.5, .5)$. The tolerance is fixed at $\varepsilon = 5$. The python libraries and code that we utilized to run our experiments as well as the data that supports our findings can be found via the references[39,40].

B. Results

In what follows we analyze the results of our approximately 10^8 individual experiments. In this section we specifically consider the effect thinning has on reservoir accuracy, i.e. on valid-prediction times. Later in Section VI we give a more detailed analysis of the interplay that thinning and spectral radius parameters have on valid-prediction times.

To analyze the effect thinning has on reservoir accuracy, in this section we first remove the experiments that resulted

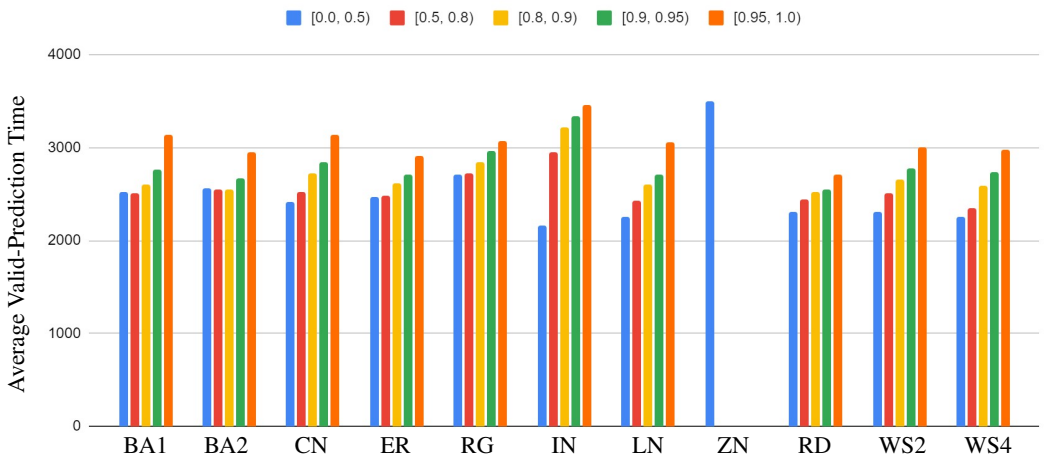


FIG. 7. For each of the network topologies BA1, BA2,..., WS4 we show the average valid-prediction times for increasing levels of the thinning parameter $p_{\text{thin}} \in [0,1]$. For each topology the top 50% of all prediction times for p_{thin} in the intervals $[0,0.5)$, $[0.5,0.8)$, $[0.8,0.9)$, $[0.9,0.95)$, $[0.95,1.0)$ are shown in blue, red, yellow, green, and orange; respectively. In the case of the zero network (ZN) we note that, as the network has no edges, p_{thin} has no effect on outcome so the average valid-prediction time over the top 50% of experiments for this model is shown in blue.

in the lower 50% of our valid-prediction times. We do so to focus on those parameters that are potentially better suited to forecast the dynamics on the Lorenz attractor. We note that because of the nonlinear interaction reservoir and topological parameters have on the valid-prediction times we do not have a precise picture of which parameter combinations are best suited for creating forecasts of the system. However, in some cases, specifically in the case of the thinning parameter p_{thin}, some clear patterns do emerge (cf. Figures 7 and 9).

To illustrate the effect of thinning we split the top 50% of our results into subsets corresponding to the eleven different topologies we consider. We then break the data corresponding to each of these respective topologies into categories determined by the extent to which each topology has been thinned. For each category we determine the average valid prediction times. The intervals for p_{thin} we use are $[0,0.5)$, $[0.5,0.8)$, $[0.8,0.9)$, $[0.9,0.95)$, $[0.95,1.0)$. The results are shown in Figure 7.

With few exceptions the trend for every topology that we consider is a strict monotonic increase in average valid-prediction times as p_{thin} increases. In fact, in every case except for the zero network (ZN) where thinning has no effect, the highest average accuracy is achieved for $p_{\text{thin}} \in [0.95,1.0)$. The highest average accuracy is found in the zero network, which may appear to suggest that the inclusion of any edges in a reservoir's processing network has a detrimental effect on valid prediction times. We note that this does not appear to be the case in general as we later discuss in Section VI. However, what we find is that increasing p_{thin} does increases valid-prediction times over many different topological and reservoir parameters.

This does suggest that thinned network are better suited, at least on average, to learn the dynamics of the Lorenz attractor when compared with reservoirs that use the standard real-world network topologies (i)-(v) or the more unconventional artificial topologies (vi)-(ix). Additionally, the trends seen in Figure 7 also holds for the top 25% and top 10% of our experiments suggesting that reservoirs that are especially good at creating forecasts still benefit from having a thinned processing network.

It is worth reiterating that for our highest accuracy, when a reservoir network is thinned by removing over ninety-five percent of its edges, the result is a processing network in which a large fraction of the network's vertices are isolated. Any nontrivial connected components of the network are also typically small in size (cf. Figure 6). This observation that independently acting nodes can be highly accurate in learning and recreating chaotic dynamics leads us to a phenomena in social science regarding crowdsourcing.

In the following section we consider how the reservoir accuracy seen in Figure 7 and crowdsourcing are potentially related.

VI. CONNECTIONS TO CROWDSOURCING

Crowdsourcing can be thought of as a means of leveraging the collective intelligence, abilities, and/or resources of a group of individuals[21–23]. Often the term is used to describe a distributed problem-solving model in which a task, typically performed by an individual, is given to members of a large group or crowd. Examples include medical diagnosis, family linking, fraud detection, astronomical classification, etc.[41]

Similar to reservoir computing, gaining useful information from a crowd requires that we collect individual *responses*. Once the group's responses to a query or outcome to a task are collected we then have the problem, as in reservoir computing, of how to *aggregate* the responses. The method of aggregating responses from a crowd takes different forms but once this information is compiled it can be used by the group to make decisions and predictions.

In gathering information from a crowd, i.e. its responses, a number of conditions have been observed that improve

the quality of a group's response. First, responses improve the more individuals act *independently* in a *decentralized* manner[21]. One reason for this is the less influence the group exerts on individuals the less individual mistakes are correlated. Second, *diversity* among the individuals' responses is also desired as too much concensus leads to limited types of responses[24–26] and inaccurate decisions[27]. Similar to reservoir computing, in crowdsourcing there is also a need for a mechanism that *aggregates* the information gained from the crowd so that individual responses can be turned into collective decisions and predictions.

These conditions of independence, decentralization, diversity, and a method of aggregation are illustrated in the following example.

Example 3. ***(Google's Crowdsourcing of the WWW)*** *An example of basing predictions on aggregated information is Google's PageRank algorithm, which forms the foundation of the system Google uses to match user queries with webpages. The PageRank algorithm views a hyperlink in the World Wide Web (WWW) from page A to page B as a vote from page A for page B.*[42] *In the context of crowdsourcing, the crowd in this example are the webpages that comprise the WWW, or more precisely their owners. The responses are the pages these webpages cite by placing a hyperlink on the page directing users to the another webpage.*

The process of how these decisions are made is thought to be decentralized, at least at the level of a websites, with the owner(s) of each website independently making their own decision of which sites/pages to link to. Hence, the processing network in this example of a "crowd of webpages" looks to some degree like the thinned networks described in Section IV. The wide range of websites with their specialized function potentially ensure that hyperlinks are chosen in diverse ways.

The aggregation of these hyperlinks results in what we think of as the WWW. Google uses this aggregated data to predict which webpages are most relevant for particular user queries. The PageRank algorithm, which carries out this task, uses the global structure of the aggregate network, the WWW, to make its predictions. It is worth emphasizing, this algorithm does not use the original processing network of webpages or their owners.

Reservoir computers appear to use a similar crowdsourcing scheme. For a reservoir computer the nodes in the associated processing network act as the "crowd". If the processing network is extremely thinned, the response $\mathbf{r}(t)$ of the individual nodes to the training data $\mathbf{u}(t)$ is both *independent* and *decentralized* as there are few to no connections between nodes (see Equation (1)). As the numerical experiments in Section V suggest, similar to the individuals in crowds, reducing dependence between the nodes in the processing network allows reservoir computers to more accurately recreate the Lorenz attractor. That is, it appears that independence and decentralization play a similar role in both crowdsourcing and reservoir computing.

In the following section we consider how reducing dependence between nodes increases *diversity* within the reservoir response vector $\mathbf{r}(t)$. In particular, we consider how the reservoir's spectral radius ρ can be used in conjunction with the thinning parameter p_{thin} to tune this diversity.

A. Thinning, Spectral Radius, and Response Diversity

The processing network used to train a reservoir has the weighted adjacency matrix $\rho\mathbf{A} \in \mathbb{R}^{n\times n}$ which is constructed to have the spectral radius $\rho \geq 0$. As the nodes of this network are driven according to the differential equation

$$d\mathbf{r}(t)/dt = \gamma[-\mathbf{r}(t) + \tanh\big(\rho\mathbf{A}\mathbf{r}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{u}(t)\big)\,]$$

the spectral radius parameter scales the effect of the interactions between reservoir nodes relative to the combined driving signal $\sigma\mathbf{W}_{\text{in}}\mathbf{u}(t) \in \mathbb{R}^n$ the nodes receive. Thus, if $\mathbf{A}_{ij} \neq 0$, the larger the spectral radius ρ the more influence node j has on the response of node i, i.e. the more *i depends* on j.

In social experiments dealing with group dynamics a consistent observation is that the more individuals depend on the group to form their responses the more homogeneous the group's responses become. That is, the more influence the group has on its members, i.e. the more *dependent* and *centralized* the decision making is, the more uniform the group's responses[24,27]. This loss in response diversity means there are fewer responses from which to build an accurate solution or make an accurate decision.

This loss of diversity can also be seen in reservoir computing. For instance, the processing network used in Example 1 is highly connected consisting of a single connected component where nodes have an average degree $c = 4$ (see Figure 4 (top left)). The spectral radius in this example is $\rho = 5$ making the dependence between adjacent nodes relatively high within the network. The resulting responses $\mathbf{r}(t)$ for this choice of spectral radius and network topology are fairly homogeneous consisting of very few distinct types of outputs (see Figure 4 (top center)). In Example 2 the same reservoir is used with the exception that the processing network is thinned by removing $p_{\text{thin}} = .90$ of the network's edges. With few interactions, few nodes have an influence on the response of other nodes. The result is a much more diverse set of responses from which to create the dynamics of the Lorenz attractor (see Figure 6 (top center)) potentially offering an explanation for the increase in VPT between Examples 1 and 2 and in the numerical experiments described in Section V.

Decreasing interactions between nodes in a processing network to increase diversity can be done in at least two ways. The first is by thinning the reservoir's processing network as in Example 2. The second is by decreasing the spectral radius of the processing network in the case that the network is highly connected. These two strategies are compared in the following example.

Example 4. ***(Thinning, Spectral Radius, and Diversity)*** *Consider the Erdos-Renyi graph with $n = 50$ nodes and average degree $c = 4$ shown in Figure 8 (top left). The way in which the nodes of this graph respond to the input-signal $\mathbf{u}(t)$ for $t \in [0,40]$ depends both on the thinning parameter p_{thin} and the reservoir's spectral radius ρ.*

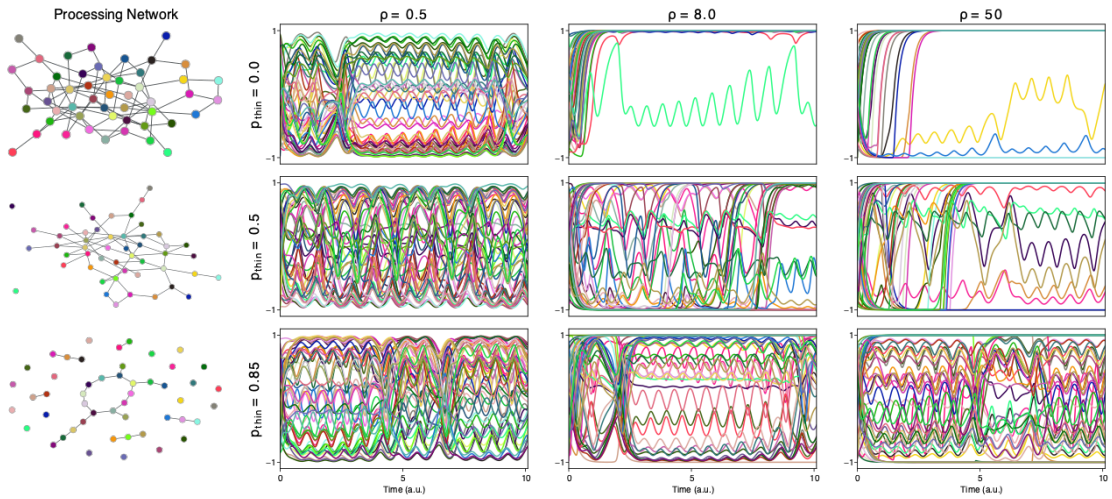


FIG. 8. First Row: The response vector $\mathbf{r}(t)$ of the unthinned Erdos-Renyi graph (left) to the finite trajectory $\mathbf{u}(t)$ with $t \in [0,10]$ are shown left to right, respectively, for $\rho = 0.5, 8, 50$. Second Row: The response vector $\mathbf{r}(t)$ to $\mathbf{u}(t)$ of the thinned graph with $p_{\text{thin}} = 0.5$ (left) are shown for $\rho = 0.5, 8, 50$. Third Row: The response vector $\mathbf{r}(t)$ of the thinned graph with $p_{\text{thin}} = 0.85$ (left) are shown for $\rho = 0.5, 8, 50$. In the first and second row the responses become increasingly homogeneous (synchronized) as ρ increases and network influences become stronger. In the third row where the graph has few interactions increasing ρ give what appears to be an increasingly diverse response.

For the case $p_{thin} = 0$, in which the graph is unthinned, we consider progressively larger spectral radii. For $\rho = 0.5$ the network response $\boldsymbol{r}(t)$ appear to be relatively diverse, presumably because a small spectral radius allows nodes to function almost independently with only weak interactions. As the spectral radius increases to $\rho = 8$ then to $\rho = 50$ the network nodes between much more dependent and their responses become less diverse (see Figure 8 (first row)). In fact, for $\rho = 50$, the responses become nearly synchronous so that reconstructing the Lorenz attractor with any reasonable accuracy is not possible.

Setting $p_{thin} = 0.5$ and removing half of the (directed) edges results in the graph shown in Figure 8 (middle left). Again as the spectral radius increases from $\rho = 0.5$ to $\rho = 8$ to $\rho = 50$ the network response $\boldsymbol{r}(t)$ become increasingly homogeneous (see Figure 8 (second row)). As in the case of the unthinned network this decrease in diversity, although less extreme, appears to be related to the fact that the response of a large majority of nodes in this processing network are influenced by other nodes.

Increasing the thinning parameter to $p_{thin} = .85$ and removing a large majority of the network's (directed) edges results in the graph in Figure 8 (bottom left). In this graph only a small fraction of the nodes are not isolated and those linked to other nodes form only relatively small clusters. In this case increasing ρ in the same manner leads to what appears to be less synchronization within the response vector $\mathbf{r}(t)$ and greater diversity for large ρ (see Figure 8 (third row)). As before, this is likely a consequence of the fact the there is little to no interaction between nodes. As such, these nodes can respond independently to the unique variant of the input-signal they receive.

This suggests that increasing a network's spectral radius has two different effects on a network's response depending on how thinned the network is. If the network is more or less connected, increasing ρ has a homogenizing effect on the response vector. However, if the network consists of mostly isolated nodes increasing ρ appears to increase the diversity of the network's response vector. This makes sense in that an increase in spectral radius of a thinned network does not increase node-to-node influence for the large majority of the network. What it does not explain is why diversity and accuracy in prediction both increase in this case. (The question as to why extremely thinned networks benefit from a high spectral radius is considered analytically in Section VII.)

In general, what we find for nearly every topology is that, similar to crowdsourcing, too much influence between nodes leads to poor results. For instance, in Figure 9 the effects of varying both ρ and p_{thin} is shown on average VPT-values over all our experiments using BA2, RG, and WS4 type processing networks. In each, for small ρ-values there is little change in average VPT-values as p_{thin} increases, although for $p_{\text{thin}} \approx 1$ there is a modest increase as the network becomes increasingly like the zero network. For large ρ-values the change is much more dramatic where we see a change from the lowest to the highest VPT-values as p_{thin} increases in both the WS2 and BA2 models (cf. Example 4). In fact, we find that this trend holds for all the real-world models (i)-(v) we consider where a combination of large spectral radius ρ and a large p_{thin} is typically superior to other parameter combinations.

This suggests an answer to long standing question in reservoir computing on how to pick the spectral radius of a reservoir computer[6,28–31]. Similar to what is described in the previous section for increasing response diversity, for highly connected processing networks our experiments suggest that picking a smaller spectral radius will typically improve the reservoir's valid prediction time. For thinned networks choosing a larger spectral radius will have a similar effect of im-

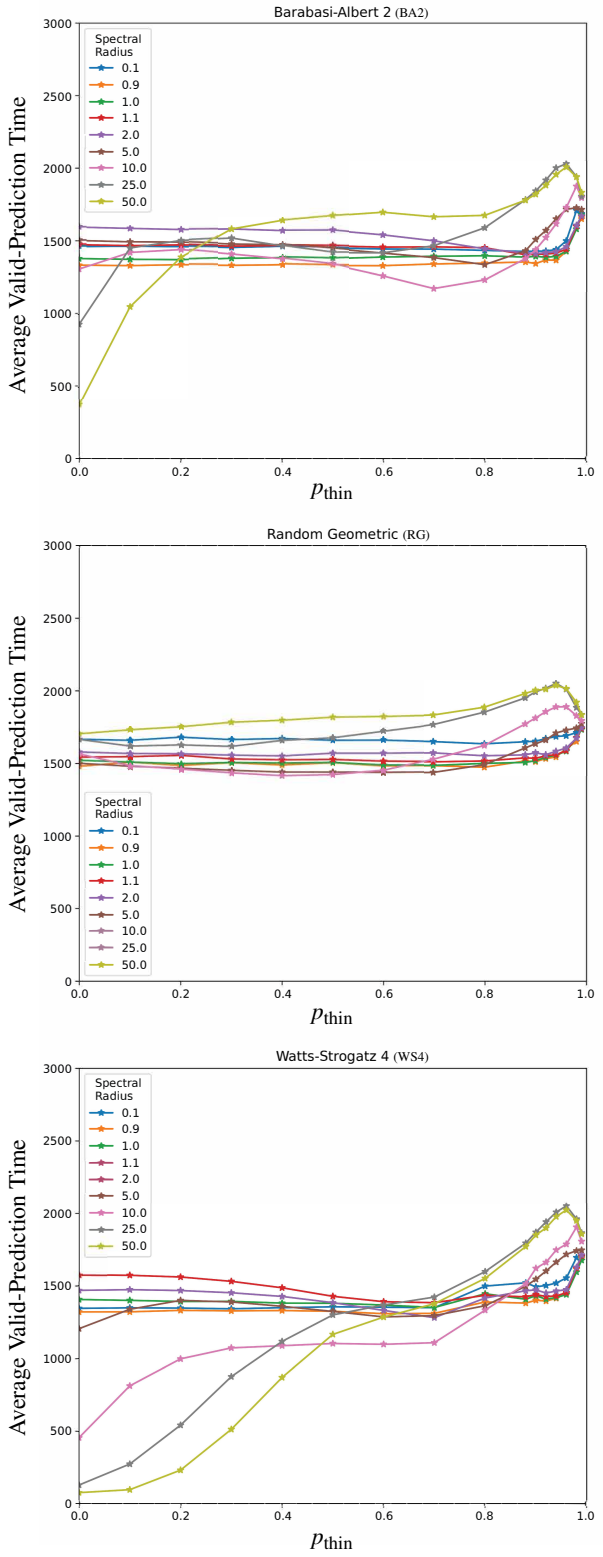


FIG. 9. The effects of varying the parameters p_{thin} and spectral radius ρ on the average VPTs are shown top to bottom for the BA2, RG, and WS4 topologies, respectively. Each value of the spectral radius $\rho = \{0.1, 0.9, \ldots, 10, 25, 50\}$ is indicated by a distinct color shown in each panel. The differing values of p_{thin} considered are given in Equation 5.

proving reservoir accuracy.

An interesting feature found in Figure 9 is that the maximal effectiveness of a reservoir peaks when we remove between 90% and 100% of the network edges. This suggests that right before a reservoir's internal network network fractures into purely isolated vertices, becoming the zero network, there is an optimal number of small connected components that maximize reservoir performance. Currently it is an open question as to what the structure of these components are. Determining this structure may give us insight into what could be thought of as an optimal group size for *cooperative learning* in reservoirs.

VII. ANALYSIS OF THINNED NETWORK DYNAMICS

In this section our goal is to develop an analytic understanding of the experimental results of Section V by analyzing the dynamical system underlying the untrained reservoir computer. The generic untrained reservoir ODE,

$$\frac{d}{dt}\mathbf{r}(t) = \gamma[-\mathbf{r}(t) + \tanh\big(\rho\mathbf{A}\mathbf{r}(t) + \sigma\mathbf{W}_{\text{in}}\mathbf{u}(t)\big)\,] \qquad (8)$$

resists analysis because of its nonlinearity and its dependence on an arbitrary processing matrix $\mathbf{A}$, observation matrix $\mathbf{W}_{\text{in}}$, and arbitrary training data $\mathbf{u}(t)$. However, thinned networks have the advantage over unthinned networks in that they have a much simplified structure of interactions. In particular, our experimental results prompt us to study the special case of the identity network where $\mathbf{A} = \mathbf{I}$. The reason is that for large p_{thin} values a large fraction of the nodes in any processing network will be isolated and the response of an isolated node is the same whether it is in a thinned network or in the identity network. Aside from this the identity network itself has a relatively high accuracy compared to the other models we consider (see Figure 7). This suggests that the mechanism(s) required for effective reservoir computing are present when this processing network is used.

The use of the processing matrix $\mathbf{A} = \mathbf{I}$ creates an uncoupled ODE in which the i^{th} component has the form

$$\frac{dr_i}{dt} = \gamma\big[-r_i(t) + \tanh\big(\rho r_i(t) + \sigma\mathbf{w}_i^T\mathbf{u}(t)\big)\big] \qquad (9)$$

where $\mathbf{w}_i$ is the ith row of $\mathbf{W}_{\text{in}}$. The analysis of this ODE reduces to analyzing the 1-*d input-dependent* system

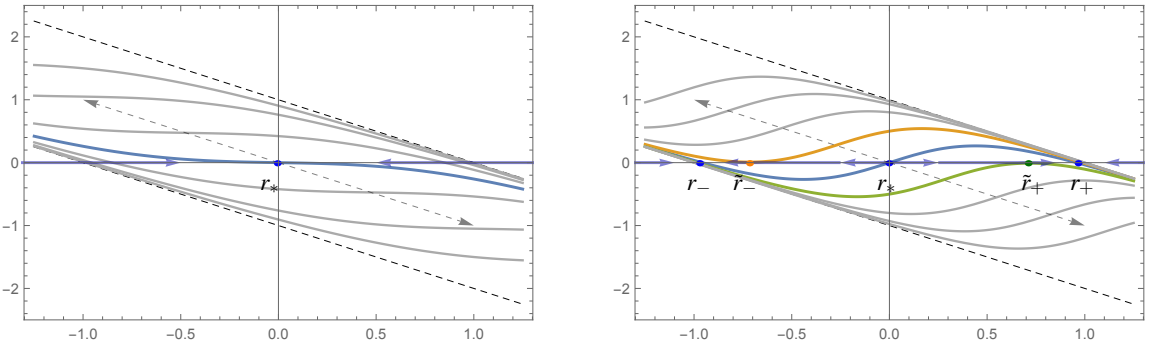
$$dr/dt = \gamma[-r + \tanh(\rho r + \beta(t))], \qquad (10)$$

where function $\beta(t) = \sigma\mathbf{w}_i^T\mathbf{u}(t)$ is the *input function*. We call the related differential equation

$$dr/dt = \gamma[-r + \tanh(\rho r + \beta)], \qquad (11)$$

where $\beta \in \mathbb{R}$, the *constant-input version* of Equation (10).

We note that different values of ρ result in different behaviors of the constant-input system. Specifically, there is a bifurcation in the system's dynamics at $\rho = 1$. To examine this bifurcation we let $f(r;\beta) = \gamma[-r + \tanh(\rho r + \beta)]$ be our



$\rho \leq 1$ $\qquad$ $\rho > 1$

FIG. 10. Left: The family of curves $f(r;\beta) = \gamma[-r + \tanh(\rho r + \beta)]$ for $\rho \leq 1$ is shown where the curve $f(r;0)$ is shown in blue. Blue arrows indicate the system's increase or decrease towards the globally attracting fixed point r_*. Right: The family of curves $f(r;\beta)$ for $\rho > 1$ is shown. The case $\beta = \beta_-$, $\beta = 0$, and $\beta = \beta_+$ are shown in green, blue, and orange, respectively. Blue arrows indicate the direction of increase or decrease in the system $dr/dt = f(r;0)$ towards the fixed points r_- and r_+. Gray arrows indicate the translation of the function's graph as β is varied.

family of constant-input functions. For $\rho < 1$, this family is shown in Figure 10 (left) where the blue curve is the function $f(r,0)$. Each curve $f(r;\beta)$ is a translate of $f(r,0)$ in the direction $(1,\gamma) \in \mathbb{R}^2$ indicated by the dashed grey arrows. Those above and to the left are the curves $f(r,\beta)$ for $\beta > 0$ while the curves down and to the right are the curves $f(r,\beta)$ for $\beta < 0$.

Theorem 1. ***(Small Spectral Radius)*** *For $\rho \leq 1$ the constant-input system $dr/dt = f(r;\beta)$ in Equation (11) has a globally attracting fixed point $r_* = r_*(\beta) \in (-1,1)$ where $r_*(\beta)$ is a monotonically increasing function of β.*

Proof. For $\rho \leq 1$ the fixed point $r_* = r_*(\beta)$ is the unique solution to the equation $r_* = \tanh(\rho r_* + \beta)$ for all $\beta \in \mathbb{R}$ which is globally attracting since $f(r;\beta)$ is a monotonically decreasing function of r. Moreover, from the equation $r_* = \tanh(\rho r_* + \beta)$, as $\tanh(\rho r)$ is an increasing function of r then $r_*(\beta)$ is an increasing function of β. Last, since $\gamma(-r-1) < f(r;\beta) < \gamma(-r+1)$ then $r_*(\beta) \in (-1,1)$. □

Based on Theorem 1, if $\rho < 1$ the input-dependent differential equation $dr/dt = f(r,\beta(t))$ in Equation (10) has the input-dependent point $r_*(t) = r_*(\beta(t)) \in (-1,1)$ that attracts all orbits $\phi(t) = \phi(t,r_0)$ of the system for any initial condition $r_0 \in \mathbb{R}$. Specifically, any orbit is moving towards or is at $r_*(t)$ for all time $t \in \mathbb{R}$. As such, the trajectory of the input-dependent system *mimics* the input function $\beta(t)$ since an increase (decrease) in $\beta(t)$ causes an increase (decrease) in $r_*(t)$. In this system large input values drive $\phi(t)$ towards $r_*(t) \approx 1$ and large negative values drive the trajectory towards $r_*(t) \approx -1$. The parameter γ dictates how quickly the trajectory *responds* to changes in $\beta(t)$ with larger γ-values making the trajectory more reactive to changes.

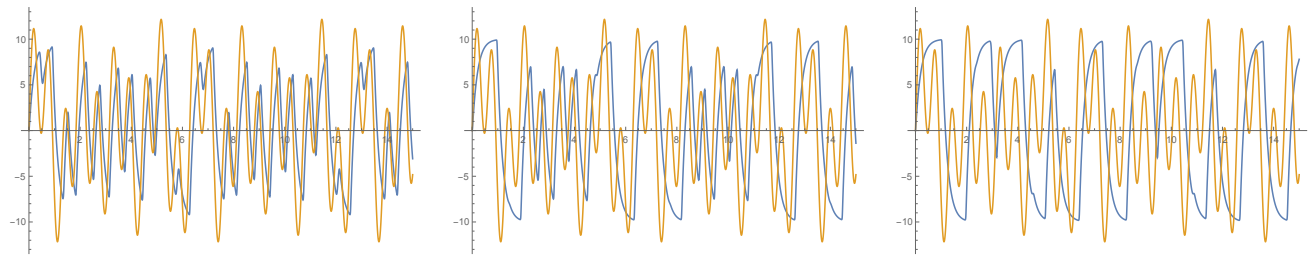
Suppose now that $\rho > 1$ in Equation (11). As ρ passes through $\rho = 1$ the system $dr/dt = f(r;0)$ experiences a pitchfork bifurcation in which the single fixed point $r_*(0)$ bifurcates into the three fixed points $r_-(0) < r_*(0) < r_+(0)$. As before, positive (negative) β-values translate the function $f(r;\beta)$ up (down) and to the right (left) in the direction $(\gamma,1) \in \mathbb{R}^2$ (see Figure 10 (right)). Thus, there are parameters $\beta_+ > 0$ and $\beta_- < 0$ at which the system has a saddle-node bifurcation where $r_*(\beta_+) = r_-(\beta_+)$ and $r_*(\beta_-) = r_+(\beta_-)$, respectively. These are the orange and green curves shown in Figure 1 (right). Here we call $\tilde{r}_-$ the r-value where $r_*(\beta_+) = r_-(\beta_+)$ and $\tilde{r}_+$ the r-value where $r_*(\beta_-) = r_+(\beta_-)$.

Theorem 2. ***(Large Spectral Radius)*** *For $\rho > 1$ the constant-input system $dr/dt = f(r;\beta)$ has the following properties:*
(i) If $\beta \in (\beta_-,\beta_+)$ the system has three fixed points $r_-(\beta) \in (-1,\tilde{r}_-)$, $r_*(\beta) \in (\tilde{r}_-,\tilde{r}_+)$, and $r_+(\beta) \in (\tilde{r}_+,1)$ where $r_-(\beta)$ and $r_+(\beta)$ have the basin of attractions $(-\infty,r_*(\beta))$ and $(r_*(\beta),\infty)$, respectively. The point $r_*(\beta)$ is repelling;
(ii) If $\beta < \beta_-$ then the system has the single globally attracting fixed point $r_-(\beta) \in (-1,\tilde{r}_-)$, which is a monotonically decreasing function of β;
(iii) If $\beta > \beta_+$ then the system has the single globally attracting fixed point $r_+(\beta) \in (\tilde{r}_+,1)$, which is a monotonically increasing function of β; and
(iv) $\beta_+ < \rho - 1$ and $\beta_- > -\rho + 1$.

Proof. As $r_-(\beta) < r_*(\beta) < r_+(\beta)$ are the three distinct solutions to $f(r;\beta) = 0$ for $\beta \in (\beta_-,\beta_+)$ and the inequality $\gamma(-r-1) < f(r;\beta) < \gamma(-r+1)$ holds for $\beta \in (-\infty,\infty)$ then $r_-(\beta), r_*(\beta), r_+(\beta) \in (-1,1)$ of $\beta \in (\beta_-,\beta_+)$. Since $r_-(\beta)$ is a decreasing function of $\beta \in (-\infty,\beta_+)$ and $r_-(\beta_+) = \tilde{r}_-$ then $r_-(\beta) \in (-1,\tilde{r}_-)$ for $\beta \in (-\infty,\beta_+)$. Similarly, as $r_+(\beta)$ is an increasing function of $\beta \in (\beta_-,\infty)$ and $r_+(\beta_-) = \tilde{r}_+$ then $r_+(\beta) \in (\tilde{r}_+,1)$ for $\beta \in (\beta_-,\infty)$. Since $r_*(\beta_+) = \tilde{r}_-$, $r_*(\beta_-) = \tilde{r}_+$, and $r_*(\beta)$ is a decreasing function of $\beta \in (\beta_-,\beta_+)$ then $r_*(\beta) \in (\tilde{r}_-,\tilde{r}_+)$.

For $\beta \in (\beta_-,\beta_+)$ the function $f(r,\beta)$ is positive on the intervals $(-\infty,r_-(\beta))$ and $(r_*(\beta),r_+(\beta))$ and negative on the intervals $(r_-(\beta),r_+(\beta))$ and $(r_+(\beta),\infty)$. Thus the basin of attraction of $r_-(\beta)$ and $r_+(\beta)$ are $(-\infty,r_*(\beta))$ and $(r_*(\beta),\infty)$, respectively and the point $\tilde{r}_*(\beta)$ is repelling.

For $\beta < \beta_-$ the function $f(r;\beta)$ is positive on $(-\infty,r_-(\beta))$ and negative on $(r_-(\beta),\infty)$. Thus, $r_-(\beta) \in (-1,\tilde{r}_-)$ is globally attracting. Similarly, for $\beta > \beta_+$ the function $f(r;\beta)$ is positive on $(-\infty,r_+(\beta))$ and negative on $(r_+(\beta),\infty)$ so that $r_+(\beta) \in (\tilde{r}_+,1)$ is globally attracting.



$\rho = 0.1$ $\rho = 5$ $\rho = 10$

FIG. 11. Shown is the periodic input function $\beta(t) = 2\sin(3t) + 5\sin(4t) + 7\sin(10t)$ in yellow and the scaled solution to the input-dependent equation $dr/dt = \gamma[-r + \tanh(\rho r + \beta(t))]$ with $\gamma = 5$ in blue for the values $\rho = 0.1, 5, 10$. The blue curves are scaled by a factor of 10 to have approximately the same amplitude as $\beta(t)$ for ease of comparison.

To show that $\beta_+ < \rho - 1$ and $\beta_- > -\rho + 1$ note that the piecewise linear family of functions

$$\ell(r;\beta) = \begin{cases} \gamma(-r-1) & r < \frac{-1-\beta}{\rho} \\ \gamma[-r + \rho r + \beta] & \frac{-1-\beta}{\rho} \le r \le \frac{-1+\beta}{\rho} \\ \gamma(-r-1) & r > \frac{1-\beta}{\rho} \end{cases}$$

has the property that
(a) $\ell(r;\beta) < f(r;\beta)$ for $r < -\beta/\rho$; and
(b) $\ell(r;\beta) > f(r;\beta)$ for $r > -\beta/\rho$.
Hence, when $\beta = \rho - 1$ part (a) implies that

$$0 \le \ell(r;\beta) < f(r;\beta) \text{ for } r > -\beta/\rho.$$

Since $f(r;\beta) \neq 0$ for $r > \beta/\rho$ then $r_-(\beta)$ does not exist at this β-value. Given that $r_-(\beta)$ exists for all $\beta \in (-\infty, \beta_+)$ then $\beta_+ < \beta = \rho - 1$ implying $\beta_+ < \rho - 1$. Similarly, when $\beta = -\rho + 1$ then using (b) we have that $\beta_- > \beta = -\rho + 1$ implying $\beta_- > \rho + 1$. □

Based on Theorem 2, if $\rho > 1$ the trajectory of the input-dependent system $dr/dt = f(r;\beta(t))$ also mimics the input function $\beta(t)$. Large input-values where $\beta(t) > \beta_+$ drive the system towards $r_+ = r_+(\beta(t)) \approx 1$ and large negative input-values where $\beta(t) < \beta_-$ drive the system towards $r_- = r_-(\beta(t)) \approx -1$.

The major difference from the case $\rho \le 1$ in Theorem 1 is that if $\rho > 1$, orbits can be much less sensitive to changes in inputs for intermediate values of $\beta(t) \in (\beta_-, \beta_+)$. In this range the repelling point $r_* = r_*(\beta(t))$ pushes the system's orbit $\phi(t) = \phi(t, r_0)$ towards either r_+ or r_- depending on which side of r_* the orbit is on. If $\phi(t) < r_*$ then $\phi(t)$ continues moving towards $r_- < 0$ although $\beta(t)$ may be positive. However, once $\beta(t)$ crosses to become greater than $\beta_+ < \rho - 1$ then r_* disappears and $\phi(t)$ moves towards r_+.

Roughly speaking, this implies that the system only recreates oscillations of the input-signal $\beta(t) = \sigma \mathbf{w}_i^T \mathbf{u}(t)$ that are *large enough*, i.e. approximately the size of ρ or larger. Hence, using the parameter ρ we can tune the scale of the oscillations we want to pick up from the input-signal. This is illustrated in Figure 11. Note that this result holds for any node in any processing network that is not influenced by other nodes in the network. We note that this gives us a very different notion of the role of ρ in a thinned reservoir when compared to its standard interpretation in the literature where ρ is used as a measure for how contractive/expansive the system is.

In practice, ρ is often chosen to be approximately 1, which is referred to as being near *criticality*, and is considered to be important in making accurate reservoir predictions[6,31]. In contrast, for a large spectral radius, i.e. $\rho \gg 1$, our data indicates that reservoir computers are more accurate at attractor reconstruction if the reservoir's processing network is quite sparse (cf. Figure 9).

Based on our second theorem, for these large ρ-values the network nodes only recreate the larger oscillations they are driven by (see Figure 11). Currently, it is an open question as to why a large spectral radius is so effective since at these parameter values the reservoir responses only capture the larger coarse-grained features of the input-signal. Possibly this has to do with removing the potential of overfitting while retaining the coarser statistics of the system's trajectories. A large spectral radius may also increase accuracy by being more discerning about where true oscillations occur in the data given that each node is driven by a linear combination $\beta(t) = \sigma \mathbf{w}_i^T \mathbf{u}(t)$ of the input-signal. However, the reason for this improved accuracy is currently unknown.

VIII. DISCUSSION AND CONCLUSION

In this paper we consider the question of what type of internal network structure is best suited to reconstruct the Lorenz attractor. We note that as nearly every machine learning method and real-world system that processes information has an underlying network structure this is related to the much broader topic of understanding the relationship between an algorithm's or system's underlying topology and its ability to learn from data.

One of the key findings presented in this paper is that thinning a reservoir can improve the reservoir's prediction accuracy. This improvement can be seen over many different network topologies and parameter combinations. We find that some of the most accurate reservoirs are created by removing nearly all network edges. The result is a near complete fracturing of the network into mostly isolated nodes. In this sense the resulting reservoir is quite different from those typically used in reservoir computing that have a real-world like

topology and are relatively connected.

Another phenomena we observe is that thinning reduces dependence between nodes in reservoir computers, which appears to increase *diversity* in the reservoir nodes' response to input. We find that the reservoir's spectral radius can be used in conjunction with thinning to tune this diversity and that this diversity in response has a positive effect on reservoir accuracy. This helps to partially answer a long-standing question regarding how to choose the spectral radius of a reservoir where choosing a spectral radius $\rho < 1$ for unthinned networks and $\rho > 1$ for thinned networks appears to improve performance. This phenomena reinforces the connection between reservoir computing and crowdsourcing where, in the latter, group influence is known to homogenize individual response and weaken the group's ability to make collective decisions, e.g. make accurate forecasts.

One of the features we find consistent over all network types we consider, aside from the identity and zero networks, is that there appears to be an optimal component size in reservoir computing for large spectral radius ρ (see Figure 9). This suggests there is an optimal scale of what might be called *node cooperation* at which dynamical processes are learned via reservoirs. Understanding this scale has potential implications to developing learning/training strategies in reservoir computing.

Aside from improving accuracy, thinned networks also have the advantage over traditional reservoirs of being computationally inexpensive to create and train. In fact, the fractured nature of the underlying network structure means that training the reservoir can be done largely in parallel before the responses are aggregated so that scaling the size of the such reservoirs is much less of an issue compared to traditional reservoirs.

As thinning progressively simplifies the structure of a reservoir we are also able to rigorously analyze the dynamic response of thinned networks to a large class of input-signals. In this direction, our analysis suggest that the spectral radius of a thinned reservoir determines the scale at which isolated nodes respond to oscillations in the input-signal (see Theorems 1 and 2). This interpretation of spectral radius is quite different than that found in other studies of reservoir dynamics and suggests a new framework for understanding the parameters used in reservoir computing.

Last, the idea that simplifying structure can improve both learning and prediction accuracy leads us to introduce the distinction between processing and aggregation networks, both found in reservoir computers. This distinction has implications to machine learning and network science as it divides the work required to learn and make predictions into (i) processing information using an underlying *network of responders* and (ii) aggregating the responses. This distinction suggests the possibility that the structure of some, if not many, real-world networks, e.g. the WWW, Internet, and brain are the result of both (i) the need to process information and (ii) the need to aggregate, synthesize, and store this information. This notion of processing and aggregation may be a useful way, for instance, to analyze the growth of real-world networks that are known to process information.

REFERENCES

[1] B. Schrauwen, D. Verstraeten, and J. M. V. Campenhout, "An overview of reservoir computing: theory, applications and implementations," in *ESANN* (2007).

[2] F. Damicelli, C. Hilgetag, and A. Goulas, "Brain connectivity meets reservoir computing," (2021).

[3] N. Rodriguez, E. Izquierdo, and Y.-Y. Ahn, "Optimal modularity and memory capacity of neural reservoirs," Network Neuroscience **3**, 551–566 (2019), https://direct.mit.edu/netn/article-pdf/3/2/551/1092682/netn_a_00082.pdf.

[4] L. Manevitz and H. Hazan, "Stability and topology in reservoir computing," in *Advances in Soft Computing*, edited by G. Sidorov, A. Hernández Aguirre, and C. A. Reyes García (Springer Berlin Heidelberg, Berlin, Heidelberg, 2010) pp. 245–256.

[5] Y. Kawai, J. Park, and M. Asada, "A small-world topology enhances the echo state property and signal propagation in reservoir computing," Neural Networks **112**, 15–23 (2019).

[6] Z. Lu, B. R. Hunt, and E. Ott, "Attractor reconstruction by machine learning," Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 061104 (2018), https://doi.org/10.1063/1.5039508.

[7] D. Canaday, A. Pomerance, and D. Gauthier, "Model-free control of dynamical systems with deep reservoir computing," (2020).

[8] H. Jaeger, "The" echo state" approach to analysing and training recurrent neural networks-with an erratum note'," Bonn, Germany: German National Research Center for Information Technology GMD Technical Report **148** (2001).

[9] M. H. Maass W, Natschläger T, "Real-time computing without stable states: a new framework for neural computation based on perturbations." **14** (2002).

[10] N. H. Packard, J. P. Crutchfield, J. D. Farmer, and R. S. Shaw, "Geometry from a time series," Phys. Rev. Lett. **45**, 712–716 (1980).

[11] F. Takens, "Detecting strange attractors in turbulence. dynamical systems and turbulence," (1981) pp. 366–381.

[12] T. Sauer, J. A. Yorke, and M. Casdagli, "Embedology," Journal of Statistical Physics **65**, 579–616 (1991).

[13] R. Yu, S. Zheng, and Y. Liu, "Learning chaotic dynamics using tensor recurrent neural networks," (2017).

[14] K. Aihara, T. Takabe, and M. Toyoda, "Chaotic neural networks," Physics Letters A **144**, 333 – 340 (1990).

[15] M. Sangiorgio and F. Dercole, "Robustness of lstm neural networks for multi-step forecasting of chaotic time series," Chaos, Solitons Fractals **139**, 110045 (2020).

[16] P. Vlachas, J. Pathak, B. Hunt, T. Sapsis, M. Girvan, E. Ott, and P. Koumoutsakos, "Backpropagation algorithms and reservoir computing in recurrent neural networks for the forecasting of complex spatiotemporal dynamics," Neural Networks **126**, 191 – 217 (2020).

[17] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, "Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach," Phys. Rev. Lett. **120**, 024102 (2018).

[18] J. Pathak, A. Wikner, R. Fussell, S. Chandra, B. R. Hunt, M. Girvan, and E. Ott, "Hybrid forecasting of chaotic processes: Using machine learning in conjunction with a knowledge-based model," Chaos: An Interdisciplinary Journal of Nonlinear Science **28**, 041101 (2018), https://doi.org/10.1063/1.5028373.

[19] D. J. Passey, *Growing Complex Networks for Better Learning of Chaotic Dynamical Systems*, Master's thesis (2020).

[20] A. Griffith, A. Pomerance, and D. Gauthier, "Forecasting chaotic systems with very low connectivity reservoir computers," Chaos: An Interdisciplinary Journal of Nonlinear Science **29**, 123108 (2019).

[21] J. Surowiecki, *The Wisdom of Crowds: Why the Many Are Smarter Than the Few and How Collective Wisdom Shapes Business, Economies, Societies and Nations* (Doubleday, 2004).

[22] A. Ghezzi, D. Gabelloni, A. Martini, and A. Natalicchio, "Crowdsourcing: A review and suggestions for future research," Wiley-Blackwell: International Journal of Management Reviews (2018).

[23] N. L. Kerr and R. S. Tindale, "Group performance and decision making," Annual Review of Psychology **55**, 623–655 (2004), pMID: 14744229.

[24] E. Apfelbaum, K. Phillips, and J. Richeson, "Rethinking the baseline in

diversity research: Should we be explaining the effects of homogeneity?" Perspectives on Psychological Science **9**, 235–244 (2014).

[25]H. Rauhut and J. Lorenz, "The wisdom of crowds in one mind: How individuals can simulate the knowledge of diverse societies to reach better decisions," Journal of Mathematical Psychology **55**, 191–197 (2011).

[26]*The Difference: How the Power of Diversity Creates Better Groups, Firms, Schools, and Societies (New Edition)* (Princeton University Press, 2007).

[27]J. Lorenz, H. Rauhut, F. Schweitzer, and D. Helbing, "How social influence can undermine the wisdom of crowd effect," Proceedings of the National Academy of Sciences (2011), 10.1073/pnas.1008636108, https://www.pnas.org/content/early/2011/05/10/1008636108.full.pdf.

[28]D. Verstraeten, B. Schrauwen, M. D'Haene, and D. Stroobandt, "An experimental unification of reservoir computing methods," Neural Networks **20**, 391–403 (2007), echo State Networks and Liquid State Machines.

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

[30]R. Pascanu and H. Jaeger, "A neurodynamical model for working memory," Neural Networks **24**, 199–207 (2011).

[31]K. Caluwaerts, F. Wyffels, S. Dieleman, and B. Schrauwen, "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)* (2013) pp. 1–6.

[32]G. Tanaka, T. Yamane, J. B. Héroux, R. Nakane, N. Kanazawa, S. Takeda, H. Numata, D. Nakano, and A. Hirose, "Recent advances in physical reservoir computing: A review," Neural Networks **115**, 100 – 123 (2019).

[33]N. Lorenz, "Deterministic nonperiodic flow," J. Atmospheric Science **20**, 130–141 (1962).

[34]V. Beffara and V. Sidoravicius, "Percolation theory," in *Encyclopedia of Mathematical Physics*, edited by J.-P. Françoise, G. L. Naber, and T. S. Tsun (Academic Press, Oxford, 2006) pp. 21 – 28.

[35]A.-L. Barabasi and R. Albert, "Emergence of scaling in random networks," Science **286**, 509–512 (1999).

[36]D. J. Watts and S. H. Strogatz, "Collective dynamics of 'small-world'networks," Nature **393**, 440–442 (1998).

[37]P. Erdös and A. Rényi, "On the strength of connectedness of a random graph," Acta Mathematica Acad. Sci. Hungar. **12**, 261–267 (1961).

[38]M. Penrose, *Random Geometric Graphs* (Oxford University Press, 2003).

[39]D. Passey, J. Wilkes, and J. Jamieson, "ChaosReservoir," `https://github.com/djpasseyjr/ChaosReservoir`.

[40]D. Passey and Q. Leishman, "Rescomp," `https://github.com/djpasseyjr/Rescomp`.

[41]K. Wazny, "Crowdsourcing ten years in: A review." Journal of global health **7 2**, 020602 (2017).

[42]S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine," Comput. Netw. ISDN Syst. **30**, 107–117 (1998).

[43].