Parallel Machine Learning for Forecasting the Dynamics of Complex Networks

Keshav Srinivasan,[1] Nolan Coble,[1,2] Joy Hamlin,[3] Thomas Antonsen,[1] Edward Ott,[1] and Michelle Girvan[1]
[1]University of Maryland, College Park, Maryland 20742, USA
[2]SUNY Brockport, Brockport, New York 14420, USA
[3]Stony Brook University, Long Island, New York 11794, USA

(Received 27 August 2021; accepted 28 March 2022; published 20 April 2022)

Forecasting the dynamics of large, complex, sparse networks from previous time series data is important in a wide range of contexts. Here we present a machine learning scheme for this task using a parallel architecture that mimics the topology of the network of interest. We demonstrate the utility and scalability of our method implemented using reservoir computing on a chaotic network of oscillators. Two levels of prior knowledge are considered: (i) the network links are known, and (ii) the network links are unknown and inferred via a data-driven approach to approximately optimize prediction.

DOI: 10.1103/PhysRevLett.128.164101

Machine learning (ML) has played a vital role in recent scientific advances in many disciplines. A key problem in these contexts is time series prediction of a dynamical system for which a first-principles, knowledge-based description is unavailable [1]. By using ML in combination with measured time series data, one can hope to construct a faithful model of a system's dynamics and to then use this model to predict the future evolution of the system's state. Our aim in this Letter is to address this goal for large systems of interacting components with complex connectivity and dynamics—a system type of enormous technical and scientific interest in many fields, ranging, e.g., from neuroscience to power grids. However, straightforward application of the standard ML prediction schemes becomes problematic when applied to forecasting the dynamics of large networks. Leveraging the typical prevalence of sparse network topology in such systems, we introduce a parallel method for forecasting node dynamics. The key oft-occurring enabling system characteristic that our method requires is that the number of network nodes providing inputs to any node is small relative to the total number of nodes in the network. In our approach, we construct a ML architecture that mimics the topology of the network. Each node of the network to be predicted is assigned an individual small ML device and these individual ML devices are linked to each other based on the underlying connectivity of the network (either known *a priori* or inferred from the available time series data). We demonstrate and test this method by applying it to a network of Kuramoto oscillators [2,3] constructed to exhibit chaotic dynamics. Our method is motivated in part by previous work on parallel ML prediction of large spatiotemporally chaotic systems [4,5].

We consider two scenarios: (a) the connectivity of the network is known, and (b) the connectivity of the network is unknown *a priori*, yet may be approximately inferred from node time series data. Scenario (a) serves two purposes: first, as preparation for the more challenging situation presented by scenario (b) and, second, as a method applicable to cases where the connectivity is, in fact, known. The main conclusion of our Letter is that our proposed parallel ML scheme enables data-based network dynamics prediction in cases that would otherwise (i.e., without parallelization) be unattainable.

In order to demonstrate and test our approach, we consider the well-studied Kuramoto model of N network-coupled oscillators,

$$\dot{\theta}_i = \omega_i + K\sum_{j=1}^{N} A_{ij}\sin(\theta_j - \theta_i), \tag{1}$$

where θ_i is the phase angle of oscillator i, ω_i is the natural frequency of oscillator i when uncoupled, K is the coupling strength, and A_{ij} is the adjacency matrix that specifies the structure of the oscillator network ($A_{ij} = 1$, if there exists a network link from node j to node i with $i \neq j$, and $A_{ij} = 0$ otherwise). Here we consider an undirected ($A_{ij} = A_{ji}$), frequency assortative Kuramoto network [6]. By "frequency assortative" we mean that two nodes are more likely to be linked if their natural oscillation frequencies are numerically close. The resulting frequency assortative system has chaotic dynamics for certain choices of parameters [7], hence serving as a good example of complex network dynamics whose evolution is challenging to predict. Each node is taken to have the same number of connections (this number is called the node's degree). The oscillator natural frequencies ω_i are drawn from a uniform random distribution from $-\pi/2$ to $\pi/2$. The frequency assortative network (i.e., the set of matrix elements A_{ij}) is constructed by starting with N_0 unlinked nodes, each with its assigned frequency (ω_i for node i) and then successively adding links, as

 © 2022 American Physical Society

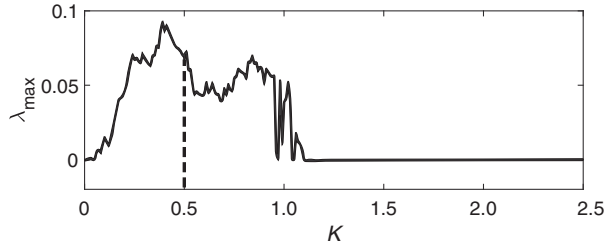

FIG. 1. Largest Lyapunov exponent as a function of the coupling constant K. The dashed line represents the chosen value of K for our studies.

follows. After randomly choosing a node i that still requires additional links, we next randomly pick another node j (not already connected to node i) that also still requires additional links, and then, with probability p_{ij}, we link nodes i and j, where

$$p_{ij} \propto \frac{\delta^\gamma}{\delta^\gamma + |\omega_i - \omega_j|^\gamma}. \tag{2}$$

We continue in this way to make links until all nodes have the desired degree d. Because of the form of p_{ij} [Eq. (2)], nodes with similar natural frequencies are connected with a higher probability (see Supplemental Material, Fig. 1 [8]). We use the global order parameter R as a metric to measure the dynamics of the oscillator network, where

$$R(t) = \frac{\sum_{i,j=1}^{N} A_{ij} e^{i\theta_j}}{N\langle d \rangle}. \tag{3}$$

For our frequency assortative network, with $N = 50$, a fixed nodal degree d of 3, $\delta = 0.8$, $\gamma = 5$, and $K = 0.5$ (our standard parameter set for most of out subsequent numerical experiments), we observe chaotic behavior, which is confirmed by the positive value of $\lambda_{\max}$, the largest Lyapunov exponent of the system (Fig. 1).

Background on nonparallel reservoir computing prediction.—In this Letter, we use reservoir computing (RC) [9,10] as our ML scheme, because of its demonstrated utility for time series prediction [1,11,12]. We consider a reservoir computer constructed with an artificial high-dimensional dynamical system, known as the reservoir, which is coupled to an input through an input layer, specified by a matrix $\mathbf{W}_{\text{in}}$, which maps the N_u-dimensional input vector $\mathbf{u}$ at discrete time t to the reservoir state variables, which are collectively expressed as the scalar components of the reservoir state vector $\mathbf{r}$. In our RC implementation, the reservoir is a network [not to be confused with the network, e.g., Eq. (1), whose state we desire to predict], and the kth component of the vector $\mathbf{r}$ is the scalar state of reservoir node k. The RC network is directed, sparse, and random with N_r nodes having average input degree, $\kappa = 3$. The RC adjacency matrix is denoted B, with matrix elements $B_{kk} = 0$ and B_{kl} for $k \neq l$ chosen randomly and uniformly from $[-\beta, \beta]$, where β is chosen to yield a maximum eigenvalue of B denoted ρ (known as the spectral radius). Each input to the reservoir is sent to N_r/N_{in} reservoir nodes, where N_{in} is the number of inputs to the RC (note that N_r is chosen to be an integer multiple of N_{in}). The input matrix $\mathbf{W}_{\text{in}}$ is then an $N_r \times N_{\text{in}}$-dimensional matrix. The elements of $\mathbf{W}_{\text{in}}$ are chosen so that every node in the reservoir receives exactly one input from $\mathbf{u}(t)$, while each input in $\mathbf{u}(t)$ is connected to N_r/N_{in} nodes in the reservoir network (see Supplemental Material for further discussion [8]). The nonzero elements are drawn from a uniform random distribution from $[-\sigma, \sigma]$, where σ is the input scaling. The reservoir state $\mathbf{r}(t)$ is taken to evolve according to

$$\mathbf{r}(t + \Delta t) = \alpha \mathbf{r}(t) + (1 - \alpha) \tanh[\mathbf{B}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{u}(t)], \tag{4}$$

where the tanh function is applied componentwise to its vector argument. Here α is the leak rate that controls the timescale of the reservoir nodes. The output of the system $\tilde{\mathbf{u}}$ is obtained through the $N_r \times N_u$-dimensional output layer matrix $\mathbf{W}_{\text{out}}$,

$$\tilde{\mathbf{u}}(t) = \mathbf{W}_{\text{out}}\mathbf{r}(t). \tag{5}$$

For the task of time series prediction, the reservoir computer is used in two different modes: a training mode and a prediction mode. In the training mode, the reservoir computing system, represented by Eqs. (4) and (5), is run for the time interval over which training data $u(t) = u(n\Delta t)(n = -n_t, (1 - n_t), (2 - n_t), \ldots, 0)$ are available, $\mathbf{r}(n\Delta t)$ is computed, and the output matrix $\mathbf{W}_{\text{out}}$ is adjusted ("trained") so that the output of the reservoir computer $\tilde{\mathbf{u}}(t)$ best approximates $\mathbf{u}(t)$. This is done through a ridge regression procedure, wherein we minimize the error summed over the training times $t = n\Delta t$ for n running from $1 - n_t$ to 0,

$$\min_{\mathbf{W}_{\text{out}}} \left\{ \sum [\|\mathbf{W}_{\text{out}}\mathbf{r}(t) - \mathbf{u}(t)\|^2] + \beta \text{Tr}(\mathbf{W}_{\text{out}}\mathbf{W}_{\text{out}}^T) \right\}. \tag{6}$$

Here β is the Tikhonov regularization parameter that is used to prevent overfitting. The quantities (N_r, ρ, σ, α, and β), referred to as hyperparameters of the reservoir computing setup, are collectively used to control the performance of system. In this Letter we chose the hyperparameters by a subsequent iterative process approximately maximizing the valid prediction time [see Eq. (8)] over the hyperparameters via a coarse grid search (see Supplemental Material, Sec. III [8]). In the prediction mode, the reservoir state now evolves autonomously in "closed-loop" mode; i.e., the output at time t now serves as the input at time $t + \Delta t$,

$$\mathbf{r}(t + \Delta t) = \tanh[\mathbf{B}\mathbf{r}(t) + \mathbf{W}_{\text{in}}\mathbf{W}_{\text{out}}\mathbf{r}(t)]. \tag{7}$$

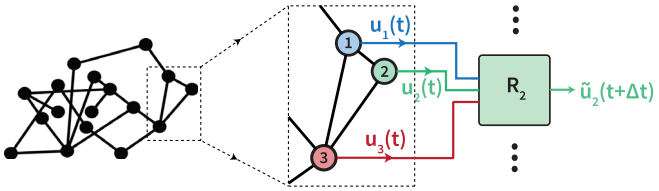

FIG. 2. A schematic diagram for the parallel network ML architecture. Here we show reservoir 2 (R_2), which receives input from its assigned node (node 2), plus inputs from nodes connected to node 2 (i.e., nodes 1 and 3). R_2 is then trained to predict its assigned node (node 2). This process is the same for each node in the network, such that the connectivity among the reservoirs mimics the network to be predicted.

This procedure generates a predicted time series $\hat{\mathbf{u}}(n\Delta t) = \mathbf{W}_{\text{out}}\mathbf{r}(t)$ that is assumed to approximate the true future evolution of the state of the system $\mathbf{u}(t)$ at a time $n\Delta t$ for $n > 0$ [we choose Δt small compared to the timescale for variation of u so that $u(n\Delta t)$ essentially specifies the continuous time function $\mathbf{u}(t)$].

Parallel ML scheme for network prediction.—In order to address the high computational complexity of predicting large networks, we introduce a parallel network RC architecture (see the schematic in Fig. 2; for code see Ref. [13]). Each node i in the predicted network is assigned its own reservoir R_i. The inputs to this reservoir are the signal of node i itself, as well as that of the nearest network neighbors of node i. The number of such neighbors is equal to the network degree. The reservoir R_i is then trained on these inputs to predict the signal of node i. Because each R_i predicts just one node, its size N_r can be relatively small. In addition, since our parallel scheme uses an interconnected network of independently trained reservoirs, we can efficiently parallelize our training process, making the system scalable to large networks.

Tests.—To compare the parallel RC scheme with the single RC approach, we use an $N_0 = 50$ node frequency assortative Kuramoto oscillator [$\delta = 0.8, \gamma = 5$, Eq. (2)] network with a coupling constant of $K = 0.5$ (see Supplemental Material, Sec. VI for results using a coupled Lorenz system [8]). We study the magnitude of the global order parameter $|R|$ [see Eq. (3)] which tells us about network-level activity (see Fig. 3) and the prediction of the evolution of individual node states (see Supplemental Material, Fig. 3 [8]), both of which show the same main qualitative behavior. For the purpose of forming inputs to the reservoir, we specify the state of the oscillator i as $[\sin\theta_i(t), \cos\theta_i(t)]$. The input matrix is generated as described above and in the Supplemental Material, Sec. II [8]. The degree of the test Kuramoto network is three links per node. This method works best for networks with low in-degrees. Large in-degrees and/or the addition of in-hubs can reduce predictive performance (see Supplemental Material, Secs. V and VI [8]). These effects may be mitigated in part by increasing the number of nodes N_r in the parallel reservoirs.

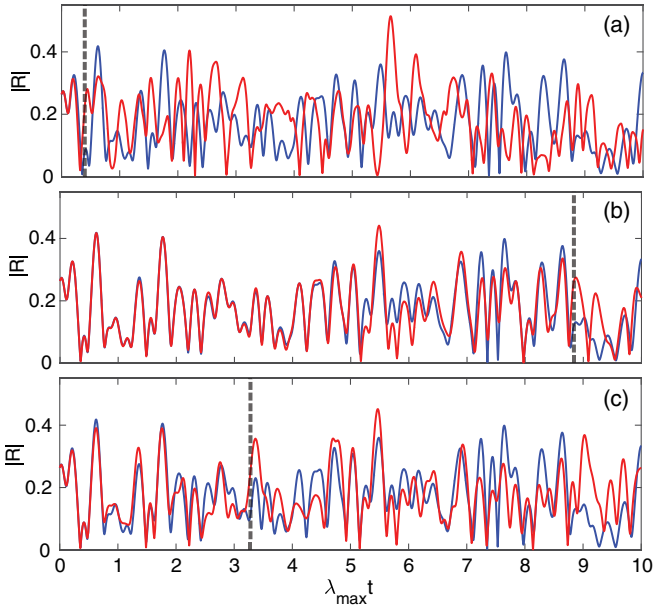

FIG. 3. Prediction of the order parameter in three different cases of ML implementations. The blue curves are the data (the truth) and red curves are predictions. The dotted lines in each plot denote the valid prediction time. (a) Single, nonparallel RC prediction using a large reservoir ($N_r = 10\,000$). (b) Parallel prediction with known network links, using 50 separate reservoirs each having modest size ($N_r = 200$). (c) Parallel prediction with unknown network links, using 50 separate reservoirs each having modest size ($N_r = 200$). The network structure is estimated by using transfer entropy as a metric to draw network edges.

Single nonparallel reservoir prediction.—The single reservoir computer prediction can fail as the size of the network we want to forecast increases. This is clearly demonstrated in Fig. 3(a), where the prediction breaks down in a fraction of a Lyapunov time $\lambda_{\max}t$. We quantify the duration of an accurate prediction by a metric that we call the "valid prediction time." This metric is defined as the amount of time elapsed before the normalized root-mean-squared prediction error $E(t)$ exceeds some chosen value f, $0 < f < 1$, for the first time, where

$$E(t) = \frac{\|\mathbf{u}(t) - \tilde{\mathbf{u}}(t)\|}{\langle\|\mathbf{u}(t)\|^2\rangle^{1/2}}. \tag{8}$$

The valid prediction time for $f = 0.1$ is marked in Fig. 3 by a vertical dotted lines. Even for the very large reservoir ($N_r = 10\,000$), close to the limit of our computer resources, that is used in Fig. 3(a), the system is still not able to predict past a fraction of a Lyapunov time.

Parallel scheme with known links.—In cases where the network structure is known *a priori*, such as in certain social networks, we can construct our parallel reservoir architecture by using the known network links. In the case of our Kuramoto oscillator network, we demonstrate our results on the 50 node network by using 50 separate parallel

reservoirs of relatively modest size [$N_r = 200$ as compared to $N_r = 10\,000$ for Fig. 3(a)], each having the same set of hyperparameters [see Fig. 3(b)]. The predictive performance of this architecture could potentially be enhanced by individually optimizing hyperparameters for each of the 50 reservoirs, but this would considerably increase both the time and computational resources required for this task. As seen from Fig. 3(b), our parallel scheme does exceedingly well for multiples of the Lyapunov time $\lambda_{\max} t$. This is particularly clear from a comparison of valid prediction time (vertical dashed lines) in Fig. 3(a) versus those in Fig. 3(b), the latter being $\gtrsim$10 times larger, while at the same time being much less computationally demanding (mainly due to the difference in N_r, $N_r = 10\,000$ for the nonparallel case versus $N_r = 200$ in the parallel case).

Parallel scheme with unknown links.—In many cases of interest, one may not have information about the underlying network structure. Using nodal time series data for finding links in networks, such as metabolic [14] and gene-regulatory networks [15], is an active area of current research. Many heuristic- [16] and statistics-based tools like conditional mutual information [17] and correlation [18], as well as recent nonlinear dynamics- [19] and machine-learning-based techniques [20], have been used for link inference and might give useful approximations of the underlying network structure. These methods could then potentially be used in our parallel network scheme.

Link inference methods, like the ones discussed above, typically use past nodal time series measurements to assign a score between each ordered pair of nodes in the network reflecting the likelihood of a directed link existing from the source to the target node. As we decrease this threshold, we draw more links and hence increase the average number of inferred neighbors for each node. Initially, decreasing the threshold, or in other words increasing the number of inferred neighbors, increases the number of true positive links and improves the predictive performance of our reservoir scheme. However, if the threshold is decreased too much, the number of false positives increases substantially and significantly degrades the predictions. Since our goal is prediction, we view the link inference threshold as an additional hyperparameter and choose it (along with the other hyperparameters), so as to optimize the valid prediction time. By this procedure, we effectively bootstrap our prediction process to determine the link threshold criterion. An example set of results for transfer entropy [21] for $N_r \approx 200$ (see Supplemental Material, Sec. II for details [8]) is shown in Fig. 3(c). Again, in marked contrast with the results in Fig. 3(a) for a large single RC ($N_r = 10\,000$), we obtain good predictions, e.g., a valid time between three and four Lyapunov times for $|R|$.

Dependence on the size of the predicted network.—Considering the case where the links are unknown, we test the performance of our parallel approach as a function of network size using two different link inference methods:

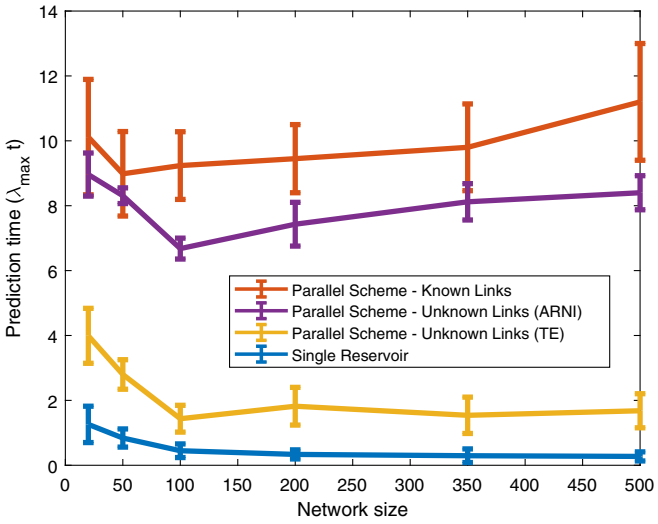

FIG. 4. Performance of the different reservoir computing methods as a function of the Kuramoto oscillator network size.

first, we use the frequently employed method based on transfer entropy [21], implemented via a MATLAB package [22] and, second, we use a method called Algorithm for Revealing Network Interactions (ARNI) [19] using code available online [23]. Figure 4 shows a plot of the valid prediction time as a function of the oscillator network size N_0. As we increase the size of the oscillator network, the prediction using a single reservoir ($N_r = 10\,000$) quickly degrades and becomes unable to capture the network dynamics at all. Since the parallel method assigns a reservoir to each oscillator in the network, for the case with known links, as expected, it maintains constant performance to within the estimated uncertainty of the valid times. In the case of link inference using transfer entropy (TE), the performance is far better than for the single reservoir, but not nearly as strong as for the case of link inference using ARNI. For a small oscillator network size of $N_0 = 10$, TE has a true positive rate of 100% and a false discovery rate of 37.5%. ARNI, on the other hand, has a true positive rate of 100% and a false discovery rate of 3.3%. When we increase the network size to $N_0 = 500$, TE has a true positive rate of 96% and a false discovery rate of 69.6%. ARNI gives us improved performance with a the true positive rate of 96% and a false discovery rate of only 7.3%. The parallel approach can tolerate imperfect link inference with a significant false discovery rate, as long as the true positive rate is high enough. The false negative inference of a link to node i deprives reservoir R_i of vital information needed for prediction of the state of node i. In contrast, reservoir R_i can compensate for a false positive link from node j to node i by learning, through its training, to ignore its time series input from node j.

We emphasize that we have not tried to determine an optimal link inference method and other methods may yield longer prediction times. We use transfer entropy as a

relatively standard link inference method with which to demonstrate our network parallel approach and ARNI as an appealing link inference choice for further investigation in other contexts.

Conclusion.—We are able to construct accurate, data-driven forecasts for the dynamics of large complex networks using a parallel ML architecture that reflects the topology of the network to be predicted. We demonstrate our approach using a chaotic network of oscillators, but we believe it should be widely applicable in a variety of contexts. In cases for which a nonparallel approach with comparable resources fails, our scheme is successful when the network links are either known or unknown *a priori*. The parallel nature makes our approach scalable for extremely large networks, creating potential applications to many fields.

This work was supported by the National Science Foundation under Grants No. PHY-1461089, No. DGE-1632976, and No. DMS-1813027.

[1] H. Jaeger and H. Haas, Science **304**, 78 (2004).
[2] Y. Kuramoto, in *International Symposium on Mathematical Problems in Theoretical Physics*, Lecture Notes in Physics (Springer, Berlin, Heidelberg, 1975), pp. 420–422.
[3] J. A. Acebrón, L. L. Bonilla, C. J. Pérez Vicente, F. Ritort, and R. Spigler, Rev. Mod. Phys. **77**, 137 (2005).
[4] J. Pathak, B. Hunt, M. Girvan, Z. Lu, and E. Ott, Phys. Rev. Lett. **120**, 024102 (2018).
[5] T. Arcomano, I. Szunyogh, J. Pathak, A. Wikner, B. R. Hunt, and E. Ott, Geophys. Res. Lett. **47**, e2020GL087776 (2020).
[6] J. G. Restrepo and E. Ott, Europhys. Lett. **107**, 60006 (2014).
[7] P. S. Skardal, J. G. Restrepo, and E. Ott, Phys. Rev. E **91**, 060902(R) (2015).
[8] See Supplemental Material at http://link.aps.org/supplemental/10.1103/PhysRevLett.128.164101 for more details regarding the Kuramoto oscillator network, reservoir computer implementation and parameters, node-level prediction, discussion on the effect of hubs, and parallel prediction of a coupled Lorenz system.
[9] H. Jaeger, GMD-German National Research Center for Information Technology Report No. 148, 2001.
[10] W. Maass, T. Natschläger, and H. Markram, Neural Comput. **14**, 2531 (2002).
[11] J. Dambre, D. Verstraeten, B. Schrauwen, and S. Massar, Sci. Rep. **2**, 514 (2012).
[12] D. Canaday, A. Griffith, and D. J. Gauthier, Chaos **28**, 123119 (2018).
[13] https://github.com/keshav224/NetworkReservoir.
[14] P. Holme and M. Huss, J. R. Soc. Interface **2**, 327 (2005).
[15] M. Banf and S. Y. Rhee, Sci. Rep. **7**, 41174 (2017).
[16] L. Lü and T. Zhou, Physica (Amsterdam) **390A**, 1150 (2011).
[17] F. Tan, Y. Xia, and B. Zhu, PLoS One **9**, e107056 (2014).
[18] S. Kumar and N. Deo, Phys. Rev. E **86**, 026101 (2012).
[19] J. Casadiego, M. Nitzan, S. Hallerberg, and M. Timme, Nat. Commun. **8**, 2192 (2017).
[20] A. Banerjee, J. Pathak, R. Roy, J. G. Restrepo, and E. Ott, Chaos **29**, 121104 (2019).
[21] T. Schreiber, Phys. Rev. Lett. **85**, 461 (2000).
[22] H. Peng, F. Long, and C. Ding, IEEE Trans. Pattern Anal. Mach. Intell. **27**, 1226 (2005).
[23] https://github.com/networkinference/ARNI.