RESEARCH

A General Deep Learning Framework for Network Reconstruction

Zhang Zhang[1], Yi Zhao[3], Jing Liu[1], Shuo Wang[2], Ruyue Xin[1] and Jiang Zhang[1*]

[*]Correspondence:
zhangjiang@bnu.edu.cn
[1]School of Systems Science,
Beijing Normal University, 100875
Beijing, People's Republic of China
Full list of author information is
available at the end of the article

Abstract

Recovering latent network structure and dynamics from observed time series data are important tasks in network science, and host a wealth of potential applications. In this work, we introduce Gumbel Graph Network (GGN), a model-free, data-driven deep learning framework to accomplish network reconstruction and dynamics simulation. Our model consists of two jointly trained parts: a network generator that generating a discrete network with the Gumbel Softmax technique; and a dynamics learner that utilizing the generated network and one-step trajectory value to predict the states in future steps. We evaluate GGN on Kuramoto, Coupled Map Lattice, and Boolean networks, which exhibit continuous, discrete, and binary dynamics, respectively. Our results show that GGN can be trained to accurately recover the network structure and predict future states regardless of the types of dynamics, and outperforms competing network reconstruction methods.

Keywords: Network reconstruction; Dynamics learning; Graph network

1 Introduction

Network is a powerful approach for modeling a wide range of phenomena in real-world systems, where the elements are regarded as nodes and the interactions as edges[1, 29, 37]. One particular interest in the field of network science is the interplay between the network topology and its dynamics[5]. Much attention has been paid on how collective dynamics on networks are determined by topology of graph. However, the inverse problems, i.e., inferring network topology based on the observed network dynamics, or designing network structure for given dynamical features, is also significant. For example, inferring gene regulatory networks from expression data can help us to identify the major genes and reveal the functional properties of genetic networks[14] ; in the study of climate changes, network reconstruction may help us to reveal the atmospheric teleconnection patterns and understand their underlying mechanisms[6]; it can also find applications in reconstructing epidemic spreading processes in social networks, which is essential to identifying the source and preventing further spreading[36]. We also note that in the field of automated machine learning (AutoML)[13, 33], various approaches are proposed to search for the optimal neural network architecture, which can be viewed as a directed graph consisting of neural network modules. Hence, a network reconstruction method may help to discover the optimal neural network architecture based on the training data. In a word, network reconstruction is pivotal to a wide span of applications.

A considerable amount of methods have been proposed for reconstructing network from time series data. One class of them is based on the method of statistical inference such as Granger causality[8, 34], and correlation measurements[2, 11, 38]. These methods, however, can usually discover functional connectivity and may fail to reveal structural connection [12]. This means that in the reconstructed system, strongly correlated areas in function need to be also directly connected in structure. Nevertheless this requirement is seldom satisfied in many real-world systems like brain [32] and climate systems [6]. Another class of methods were developed for reconstructing structural connections directly under certain assumptions. For example, methods such as driving response[39] or compressed sensing[36, 41–43] either require the functional form of the differential equations, or the target specific dynamics, or the sparsity of time series data. Although a model-free framework presented by Casadiego et al.[9] do not have these limitations, it can only be applied to dynamical systems with continuous variables so that the derivatives can be calculated. Thus, a general framework for reconstructing network topology and learning dynamics from the time series data of various types of dynamics, including continuous, discrete and binary ones, is necessary.

Recently, deep Learning has gained success in many areas such as image classification [23] and speech recognition [16]. Can we apply this state-of-the-art technique on network reconstruction problem? This is possible because Graph network framework [4] have enabled deep learning techniques applied on graph structures successfully by mapping graph-structured data onto Euclidean space with update and aggregation functions [49]. With a wealth of different avenues available, GN can be tailored to perform various tasks, such as node or graph classification [40, 48], graph generation [7, 10, 27, 46], and spatial-temporal forecasting [17, 26, 45, 47]. Recently, the topic of recovering interactions and predicting physical dynamics under given interaction networks has attracted much attention. A most used approach is introduced by Battaglia et al. [3], representing particles as nodes and interactions as edges, then reconstruct the trajectories in a inference process on the given graph. However, most of the works in this field have focused on physical reasoning task while few dedicate to solving the inverse problem of network science: revealing network topology from observed dynamics. Some related works [30, 31] attempted to infer implicit interaction of the system to help with the state prediction via observation. But they did not specify the implicit interaction as the network topology of the system, therefore the network reconstruction task remains ignored. Of all literature as we known, only NRI (Neural Relational Inference) model[22] is working on this goal. Nevertheless, only a few continuous dynamics such as spring model and Kuramoto model are studied, and discrete processes were never considered. So in the rest of this article, we will take NRI as one of our baselines and will be compared against our own model.

In this work we introduce Gumbel Graph Network (GGN), a model-free, data-driven method that can simultaneously reconstruct network topology and perform dynamics prediction from time series data of node states. It is able to attain high accuracy on both tasks under various dynamical systems, as well as multiple types of network topology. We first introduce our architecture which is called Gumbel Graph Networks in Section 2 and then give a brief overview of our experiments on three typical dynamics in Section 3. In Section 4, we show our results. Finally, some concluding remarks and discussions are given in Section 5.

2 GGN Architecture

2.1 Gumbel Graph Network

The goal of our Gumbel Graph Network is to reconstruct the interaction graph and simulate the dynamics from the observational data of N interacting objects.

Typically, we assume that the system dynamics that we are interested can be described by a differential equation $dX/dt = \psi(X^t, A)$ or the discrete iteration $X^t = \psi(X^{t-1}, A)$, where $X^t = (X_1^t, ..., X_N^t)$ denotes the states of N objects at time t, and X_i is the state of the object i. h is the dynamical function that we will simulate, and A is the adjacency matrix of an unweighted directed graph that we will reconstructed, and $A_{ij} = 1$ if the edge (i, j) from vertex i to vertex j is connected, while $A_{ij} = 0$ if it's disconnected. In the calculation, the data used by GGN model consists of a sequence of time series data, $X = (X^t, ..., X^{t+P})$, where P is the number of prediction steps. In the training process, we feed one step trajectory value: X^t as its input , and their succeeding states, namely $(X^{t+1}, ..., X^{t+P})$ as the targets.

Thus, our algorithm aims to learn the network structure (Specifically, the adjacency matrix) and the dynamical model ψ simultaneously in an unsupervised way.

In order to achieve that, our model consists of two jointly trained parts: a network generator that generates a discrete network with Gumbel Softmax trick[18]; and a dynamics learner that utilizes the network generated by the generator and one-step trajectory value to predict the value in the next one or multiple steps. We alter the network generator for S_n steps and the dynamics learner for S_d steps. The S_n and S_d are two hyper-parameters which taking different values for different models. The objective of our model is to minimize the errors between the predictions and the observed time series data. As for loss function, when the time series to be predicted is a discrete sequence with finite countable symbols, the cross-entropy objective function is adopted otherwise the mean square errors are used. Figure 1 outlines the basic structure of our framework. And Algorithm 1 gives the pseudocode of how Network Generator and Dynamics Learner work and be trained.

2.2 Network Generator

One of the difficulties for reconstructing a network from the data is the discreteness of the graph data, such that the back-propagation technique, which is widely used in differential functions, cannot be applied.

To conquer this problem, we apply Gumbel-softmax trick to reconstruct the adjacency matrix of the network directly. This technique simulates the sampling process from a discrete distribution by a continuous function such that the distributions generated from the sampling processes in real or simulation are identical. In this way, the simulated process allows for back-propagation because it is differentiable.

Suppose we are going to reconstruct a network with size N, and the adjacency matrix is $\{a_{ij}\}_{N\times N}$. Where, a_{ij}s are real values generated by the gumbel-softmax function with the components of α_{ij} as training parameters and the temperature τ as a hyper-parameter, which can be written as:

$$a_{ij} = \frac{\exp((\log(\alpha_{ij}) + \xi_{ij})/\tau)}{\exp((\log(\alpha_{ij}) + \xi_{ij})/\tau) + \exp((\log(1-\alpha_{ij}) + \xi'_{ij})/\tau)} \quad , \tag{1}$$

Algorithm 1: Gumbel Graph Network, GGN

```
$P \leftarrow Length\ of\ Prediction\ Steps$;
$S_n \leftarrow Network\ Generator\ Train\ Steps$;
$S_d \leftarrow Dynamics\ Learner\ Train\ Steps$;
$lr \leftarrow Learning\ Rate$;
Input: $X = \{X^0, X^1, ..., X^P\}$
# Initialization:
Initialize:
Network Generator parameters $\alpha$;
Dynamics Learner parameters $\theta$;
# Training:
for each epoch do
    $A \leftarrow Gumbel\ Generator(\alpha)$;
    # Train Dynamics Learner;
    for $m = 1, ..., S_n$ do
        $X^0_{predict} \leftarrow X^0$;
        for $t = 1, ..., P$ do
            $X^t_{predict} \leftarrow Dynamics\ Learner(A, X^{t-1}_{predict}, \theta)$;
        end
        $loss \leftarrow Compute\ Loss(\{X^1, ..., X^P\}, \{X^1_{predict}, ..., X^P_{predict}\})$;
        $\delta\theta \leftarrow BackPropagation$;
        $\theta \leftarrow \theta - lr * \delta\theta$;
    end
    # Train Network Generator;
    for $n = 1, ..., S_d$ do
        $A \leftarrow Gumbel\ Generator(\alpha)$;
        $X^0_{predict} \leftarrow X^0$;
        for $t = 1, ..., P$ do
            $X^t_{predict} \leftarrow Dynamics\ Learner(A, X^{t-1}_{predict}, \theta)$;
        end
        $loss \leftarrow Compute\ Loss(\{X^1, ..., X^P\}, \{X^1_{predict}, ..., X^P_{predict}\})$;
        $\delta\alpha \leftarrow BackPropagation$;
        $\alpha \leftarrow \alpha - lr * \delta\alpha$;
    end
end
Output: $A$, $X_{predict} = \{X^1_{predict}, ..., X^P_{predict}\}$
```

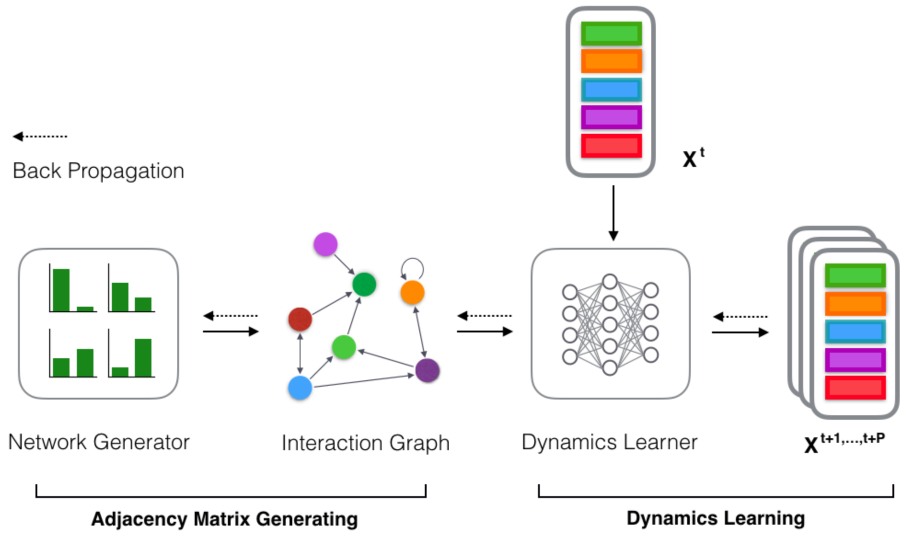

Figure 1: **Basic structure of GGN.** Our framework contains two main parts. First, the Adjacency Matrix is generated by the Network Generator via Gumbel softmax sampling; then the adjacency matrix and X^t (node state at time t) are fed to Dynamics Learner to predict the node state in future P time step. The back-propagation process runs back through all the computations.

where ξ_{ij}s and ξ'_{ij}s are i.i.d. random numbers following the gumbel distribution[28]. This calculation uses a continuous function with random noise to simulate a discontinuous sampling process. And the temperature parameter τ adjusts the sharpness of the output. When $\tau \to 0$, a_{ij} will take 1 with probability α_{ij} and 0 with probability $1 - \alpha_{ij}$.

α_{ij}s are all trainable parameters, which can be adjusted according to the back propagation algorithm. Thanks to the features of Gumbel-softmax trick, the gradient information can be back propagated through the whole computation graph although the process of sampling random numbers is non-differentiable.

2.3 Dynamics Learner

Learning with graph-structured data is a hot topic in deep learning research areas. Recently, Graph networks (GNs) [4] have been widely investigated and have achieved compelling performance in node classification, link prediction, et al. In general, a GN uses the graph structure A and node features X as its input to learn the representation of each node. Specifically, the graph information used here is the adjacency matrix constructed by the generator. The whole dynamics learner can be presented as a function:

$$X^t_{predict} = f(X^{t-1}, A) \tag{2}$$

where X_t is the state vector of all N nodes at time step t, and A is the adjacency matrix constructed by the network generator. If we want to make multiple predictions, we have:

$$X^{t,\cdots,T}_{predict} = f(f(f\cdots(X^{t-1}, A))), \tag{3}$$

where $X_{t,\cdots,T}$ represents the states of the system from time t to T. f is the transitional function, whose structure can be illustrated by Figure 2, and f will be repeatedly applied for $T - t$ times.

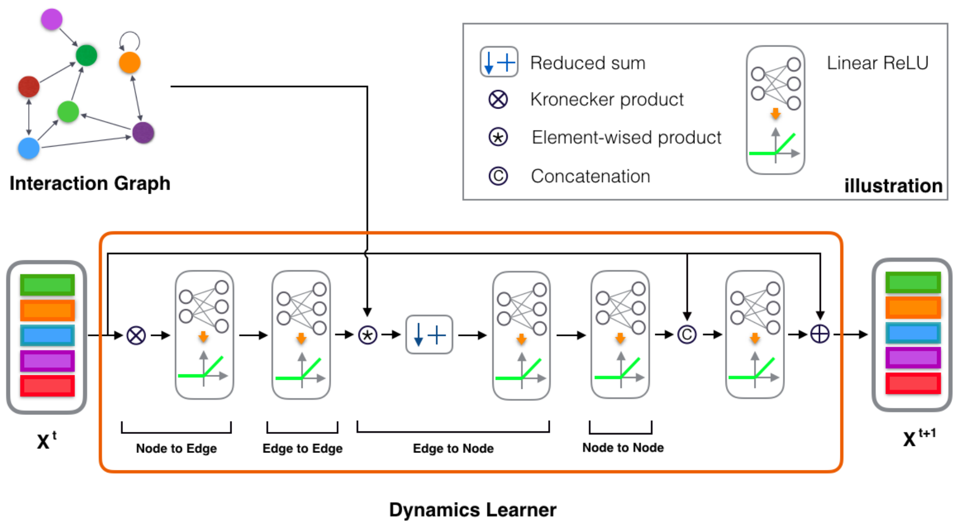

Figure 2: **The Structure of the Dynamics Learner.** Dynamics Learner takes the graph structure (here we use Adjacency Matrix) and node states X as its input to predict node states at next time step(s). Inside the Dynamics Learner, there would be four main parts operating in succession to accomplish the whole process: Node to Edge, Edge to Edge, Edge to Node and Node to Node.

In Figure 2, the symbol $\otimes$ represents the Kronecker product while the time operator is replaced with the operator to make two elements a pair. Therefore, if X is a vector, then $X \otimes X^T$ is an $N \times N$ matrix, and the element at the ith row and jth column is $< x_i, x_j >$, where x_i is the ith element in X. The symbol $*$ represents the element-wised product, and the symbol C represents the concatenation operator.

The dynamics learner comprises of four main steps:

Node-to-Edge: Firstly, nodes' states at time $t-1$ are fed into the model to generate the hidden states of all possible node pairs. This process can be presented as a function:

$$H_e^{t-1} = f_{v\to e}(\left[X^{t-1}, (X^{t-1})^T\right]) \tag{4}$$

where $[.,.]$ denotes the concatenation operator. X^{t-1}, $\left(X^{t-1}\right)^T$ denotes the set of states of nodes at time $t-1$ and its transposition, respectively. Thus $\left[X^{t-1}, \left(X^{t-1}\right)^T\right]$ contains the input nodes' and output nodes' states of the all possible node pairs at time $t-1$. $f_{v\to e}$ represents the node-to-edge($v \to e$) mapping. While there are a number of mapping functions to choose from, including Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), here we use Multi-Layer Perceptron (MLP) which consists of one or more layers of the combination of a linear layer and a rectified linear unit (ReLU).H_e^{t-1} denotes the hidden states on node pairs, which captures the influence of nodes on possible edges.

Edge-to-Edge: We use MLP to decode the edges' hidden states, and aim to capture more information in possible edges.

$$H_e^{t-1} = f_e(H_e^{t-1}) \tag{5}$$

Edge-to-Node: Edge-to-node denotes the aggregation process from hidden states of all incoming possible edges.

$$H_v^t = f_{e\to v}(A * H_e^{t-1}) \tag{6}$$

$A * H_e^{t-1}$ sets elements 0 if there is no connection between two nodes (the corresponding elements in A are 0). $f_{e\to v}$ represents the reduced-sum operation and generates H_v^t. H_v^t denotes the hidden state at time t of all nodes which aggregating the edges' hidden states at time $t-1$.

Node-to-Node: We adopt MLP similar to that in edge-to-edge step to get more information in nodes.

$$H_v^t = f_v(H_v^t) \tag{7}$$

Finally, we introduce skip-connection in ResNet [15] to improve the gradient flow through the network, which enhances the performance of the Dynamics Learner. X^t denotes the nodes' states at time t. f_{output} is another MLP.

$$X_{predict}^t = f_{output}(\left[X^{t-1}, H_v^t\right]) + X^{t-1} \tag{8}$$

After all the above processes, we complete one step prediction of all nodes' states. To make multi-step predictions, we feed in the output states and reiterate until we get the prediction sequence $X_{predict} = (X_{predict}^1, ..., X_{predict}^T)$. Then we back propagate the Mean Squared Error (MSE) or Negative Log Likelihood (NLL) Loss between model prediction and the ground truth.

3 Experiments

3.1 Overview

We experimented with three types of model: Boolean Network[21], Kuramoto [24], and Coupled Map Lattice [19, 20], which exhibit binary, continuous, and discrete trajectories, respectively. Here we attempt to train our model to learn the dynamics and reconstruct the interactions between particles, or the adjacency matrices, under all three circumstances.

In this article, we test the performance of GGN with three main experiments: one concerns different net size and different level of chaos (subsection 3.1); one features different type of network topology (subsection 3.2), and one studies the relationship between data size and accuracy (subsection 3.3). Our full code implementations are as shown on Github [https://github.com/bnusss/GGN].

3.2 Model Simulations

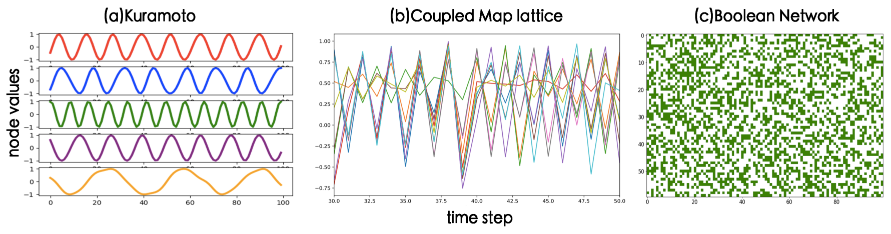

Figure 3: **Time evolution of node values in three types of network simulations.** Kuramoto (a), Coupled Map Lattice (b) and Boolean Network (c).

3.2.1 Boolean Network

In Boolean Network system, every variable has a possible value of 0 or 1 and a Boolean function is assigned to the node. The function takes the states of its neighbors as inputs and returns a binary value determining the state of the current node.

In simulation. We set the structure of the network as a directed graph with the degree of each node as K, and different K determines whether the network will evolve chaotically or non-chaotically. All nodes follow the same randomly generated table of dynamical rules. The training data we generated contains 5k pairs of state transition sequences. Meanwhile, we simulated 1k validation set and 1k test set.

3.2.2 Kuramoto Model

The Kuramoto model [24] is a nonlinear system of phase-coupled oscillators. Specifically, we study the system

$$\frac{d\phi_i}{dt} = \omega_i + k \sum_{j \neq i} A_{ij} \sin(\phi_i - \phi_j) \tag{9}$$

Where ω_i are intrinsic frequencies sampled from a given distribution $g(\omega)$, and here we use a uniform distribution on $[1, 10)$; k is the coupling strength; $A_{ij} \in \{0, 1\}$ are the elements of $N \times N$ adjacency matrix, and for undirected random networks we study, $A_{ij} = A_{ji}$. The Kuramoto network have two types of dynamics, synchronized and non-synchronized. According to studies by Restrepo et. al [35], the transition from coherence to incoherence can be captured by a critical coupling strength $k_c = k_0/\lambda$, where $k_0 = 2/\pi g(5.5)$ in our case, and λ is the largest eigenvalue of the adjacent matrix. The network synchronizes if $k > k_c$, and otherwise fails to synchronize. We simulate and study both coherent and incoherent cases.

For simulation, we solve the 1D differential equation with 4th-order Runge-Kutta method with a step size of 0.01. Our training sets include 5k samplings, validation set 1k, and test set 1k, each sampling covers $d\phi_i/dt$ and $\sin(\phi_i)$ in 10 time-steps.

3.2.3 Coupled Map Lattice

Coupled map lattices represent a dynamical model with discrete time and continuous state variables[19, 20]. The model is originally defined on a chain with a periodic boundary condition but can be easily extended to any type of topology:

$$x_{t+1}(i) = (1-s)f(x_t(i)) + \frac{s}{\deg(i)} \sum_{j \in \text{neighbor}(i)} f(x_t(j)), \tag{10}$$

where s is the coupling constant and $\deg(i)$ is the degree of node i. We choose the following logistic map function:

$$f(x) = \lambda x(1-x). \tag{11}$$

We simulated $N \in \{10, 30\}$ coupled map lattices with initial states $x_0(i)$ sampling uniformly from $[0, 1]$ for random regular graphs. Notice that when setting coupling constant $s = 0$, the system reduces to N independent logistic map. The training sets also include 5k samplings, 1k validation set, and 1k test set, each sampling covers x_i in 10 time-steps.

4 Results

In each experiment listed below, we set the hyper-parameters S_n and S_d of the Boolean Network model to 20 and 10, respectively, while in Coupled Map Lattice model and Kuramoto model they are 5 and 30. In Coupled Map Lattice model and Kuramoto model, the prediction steps P is 9, which means that the current state is used to predict the node state of the next 9 time steps, while in the Boolean Network, it is set to 1. In all the experiments, we've set the hidden size in all the MLP networks of the dynamics learner module of the GGN model to 256. All the presented results are the mean value over five times of repeated experiments. The horizontal lines: "-" in the table indicates that the amount of data exceeds the model processing limitation, the model becomes so unstable that outputs may present as "nan" during training.

We compare our model with following baseline algorithms:

- **LSTM**(Long Short-Term Memory Network) is a well-known recurrent neural network and has been shown to be very suitable for sequence prediction problems. To do network reconstruction with LSTM, some previous work [22] used thresholded correlation matrix to represent the adjacency matrix. But according to our experiments, this method would only yield all-zero or all-one matrices, therefore cannot serve as a satisfactory way of deriving adjacency matrices. Hence, we use LSTM only for node state prediction.

- **NRI**(Neural Relational Inference Model) is able to reconstruct the underlying network structure and predict the node state in future time steps simultaneously by observing the node state sequence. We compare our model against it in both tasks. Here we use settings similar to that in Kipf's original paper[22]: all our experiments use MLP decoders, and with the Kuramoto model, we use CNN encoder and other models the MLP encoder.

We use the following indicators to evaluate the results of the experiments:

- **TPR**(true positive rate) measures the proportion of positive instances that are correctly identified. We consider 1 in the adjacency matrix as a positive element, whereas 0 as a negative one.
- **FPR**(false positive rate) computes the proportion of negative instances that are incorrectly identified in the adjacency matrix generated.
- **MSE**(mean square error) measures the average of the squares of the errors, that is the average squared difference between the estimated values and data. The MSE we showed below is the average mean square error of next P time steps.
- **ACC(net)** is the proportion of correctly identified elements of the Adjacency Matrix.
- **ACC(dyn)**, In our experiment on the Boolean Network, we use indices ACC(dyn) to measure the proportion of nodes whose states are predicted accurately in the next time step.

4.1 Experiments with Different Dynamics

Node Num	State	LSTM	NRI				GGN			
		ACC(dyn)	ACC(net)(%)	TPR	FPR	ACC(dyn)	ACC(net)(%)	TPR	FPR	ACC(dyn)
10	non-chaotic	**0.841**	56.8	0.422	0.395	0.820	**99.1**	1	0.008	0.694
10	chaotic	**0.789**	48.1	0.458	0.465	0.528	**99.4**	0.983	0	0.693
30	non-chaotic	0.912	40.9	0.590	0.591	0.798	**92.6**	0.476	0.036	**0.937**
30	chaotic	**0.765**	46.0	0.549	0.547	0.721	**90.0**	0.601	0.034	0.699
100	non-chaotic	**0.933**	-	-	-	-	**84.0**	0.505	0.153	0.9267
100	chaotic	**0.796**	-	-	-	-	**95.7**	0.25	0.013	0.7483

Table 1: Results with Boolean Network.

In our experiments, we set the network topology of a Boolean Network as a directed graph, with the indegree of each node being k, and k determines whether the system is chaotic or not. For all systems, we set $k = 2$ for non-chaotic cases; and for systems with $10, 30, 100$ nodes, k is set to $7, 5$ and 4, respectively, to obtain chaotic dynamics. As shown in Table 1, the GGN model recovers the ground-truth interaction graph with an accuracy significantly higher than competing method, and the recovery rate in non-chaotic regimes is better than those in chaotic regime.

Node Num	State	LSTM	NRI				GGN			
		MSE	ACC(%)	TPR	FPR	MSE	ACC(%)	TPR	FPR	MSE
10	non-chaotic	1.92e-2	53.1	0.446	0.588	1.69e-4	**100**	1	0	**5.63e-6**
10	chaotic	2.54e-2	54.7	0.459	0.605	4.04e-4	**99.3**	1	0.013	**3.24e-5**
30	non-chaotic	4.11e-2	-	-	-	-	**100**	1	0	**3.29e-6**
30	chaotic	5.03e-2	-	-	-	-	**99.9**	1	0.0017	**3.41e-6**

Table 2: Results with CML model.

Here we presented our results obtained on coupled map lattice model in Table 2. In our experiments, the network topology is random 4-regular graph, and we set coupling constant $s = 0.2$ fixed. Because $r \approx 3.56995$ is the onset of chaos in the logistic map, we chose $r = 3.5$ and $r = 3.6$ to represent non-chaotic and chaotic dynamics respectively. For a random 4-regular graph with 10 nodes, our GGN model has obtained approximately 100% accuracy in the task of network reconstruction. For a system with 30 nodes, it is still able to achieve a high accuracy and the performance obtained on non-chaotic dynamics is better than that on chaotic dynamics.

Node Num	State	LSTM	NRI				GGN			
		MSE	ACC(%)	TPR	FPR	MSE	ACC(%)	TPR	FPR	MSE
10	coherent	2.67e-2	51.8	0.543	0.505	9.63e-2	**99.8**	0.994	0.004	**1.12e-3**
10	incoherent	2.85e-2	51.7	0.543	0.508	1.11e-1	**99.8**	0.994	0.004	**8.36e-4**
30	coherent	3.12e-2	-	-	-	-	**89.8**	0.920	0.169	**3.96e-4**
30	incoherent	3.35e-2	-	-	-	-	**81.0**	0.700	0.124	**1.90e-4**

Table 3: Results with Kuramoto model.

In the experiment concerning Kuramoto Model, we used Erdos-Renyi random graph with a connection possibility of 0.5. As the onset of synchronization is at $k = k_c$ (in our cases, $k_c = 1.20$ for 10 nodes system and $k_c = 0.41$ for 30 nodes system), we chose $k = 1.1k_c$ and $k = 0.9k_c$ to represent coherent and incoherent dynamics respectively. Here we used data of only two dimensions (speed and amplitude), as opposed to four dimensions in NRI's original setting (speed, phase, amplitude and intrinsic frequency), so the performance of NRI model here is lower than that presented in Kipf's original paper [22]. Similar to BN and CML, our GGN model attains better accuracy in coherent cases, which are more regular than incoherent ones.

To sum up, the above experiments clearly demonstrates the compelling competence and generality of our GGN model. As shown in the tables, GGN achieves high accuracy on all three network simulations, and its performance remains good when the number of nodes increases and the system transform from non-chaotic to chaotic. Although we note that our model achieves relatively lower accuracy in chaotic cases, and it is not perfectly stable under the current data size (in CML and Kuramoto experiments, there is a 1/5 chance that performance would decrease), the overall results are satisfactory.

4.2 Reconstruction Accuracy with Network Structure

In this section, we used 100-node Boolean Network with the Voter dynamics [25] (for a node with degree k, and m neighbours in state 1, in the next time step it has a probability of m/k to be in state 1, and a probability of $(k-m)/k$ to be in state 0) to study how network structure affects the network reconstruction performance of our GGN model. Specifically, we studied WS networks and BA networks, and examined how the reconstruction accuracy would change under different network parameters. We also experimented with two different data sizes: 500 and 5000 pairs of state transition sequences, to see how network structure would affect the needed amount of data.

In the first experiment, we studied WS networks of different re-connection possibility p. We note that the reconstruction accuracy declines slowly between $p =$

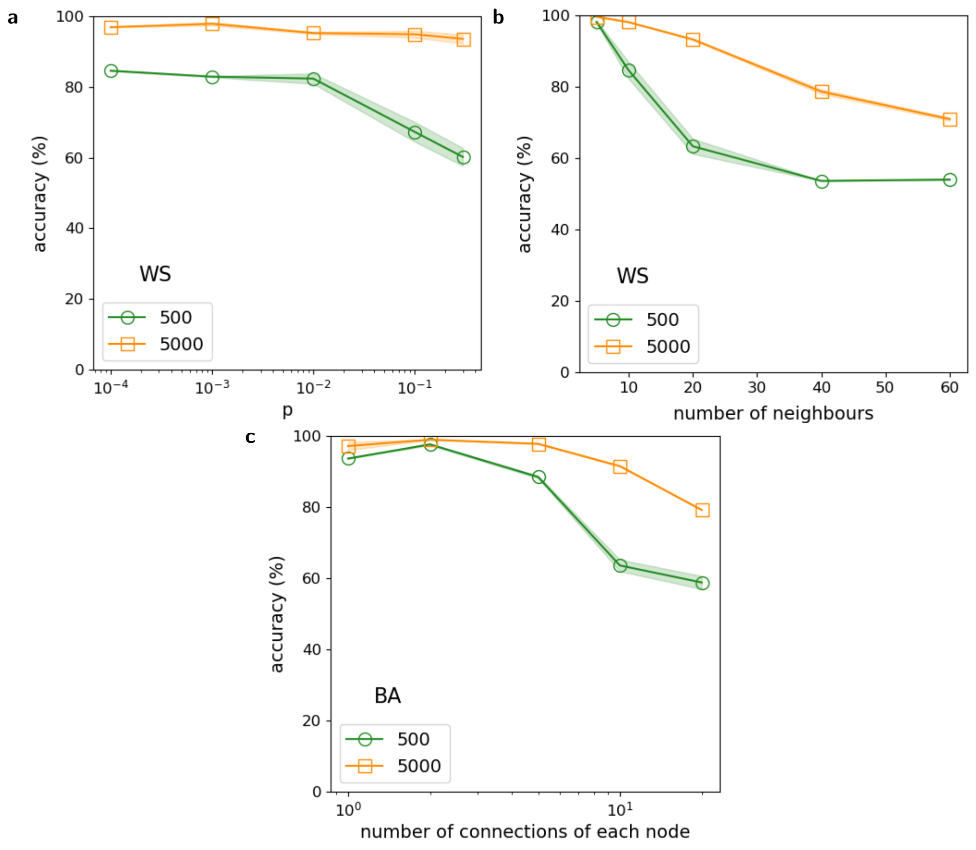

Figure 4: **Accuracy of reconstruction with different network structures.** Accuracy of reconstruction with **a**: WS networks under different re-connection possibility p (while neighbours=20); **b**: WS networks under different number of neighbours (while $p = 0.3$); **c**: BA networks under different number of connections of each node. We experimented with two different data sizes: 500 and 5000 pairs of state transition sequences, represented in each plot by green and orange line, respectively.

$[10^{-4}, 10^{-2}]$, but drops sharply when p is larger than 10^{-2}. As the average distance of the network drops quickly before 10^{-2}, but our reconstruction accuracy remains roughly the same, it seems that the reconstruction accuracy is insensible to it. On the other hand, the Clustering Coefficient of the network drops quickly when p is larger than 10^{-2}, while declining slowly when p is smaller [44], which correspond with our curves of accuracy. Therefore, we may conclude that the reconstruction accuracy is directly affected by Clustering Coefficient of the network. However, as the data size increases, the performance is significantly augmented under all different values of p, and the slope is greatly reduced. So increasing the data size can effectively solve the problem brought by increasing re-connection possibility.

In the second experiment, we studied WS networks of different number of neighbours. Here the situation is much simpler: as number of neighbours increases, the complexity of network also goes up, and it in turn makes learning more difficult. So the need for data is increasing along with the number of neighbours.

In the third experiment, we studied BA networks of different number of connections of each node. The result is similar to the second experiment, but here, increasing the data size receives a smaller response than in the WS networks. That is probably because in BA networks, a few nodes can greatly affect the dynamics of the whole network, which makes the complexity even higher, therefore the need for data would be greater for BA networks.

4.3 Reconstruction Accuracy with Data Size

In this section we study the dependency between the amount of data and the accuracy of reconstruction. We performed our experiments on CML model with chaotic dynamics. As illustrated in Figure 5, the accuracy of reconstruction significantly improves when feeding more data to the model. We also noticed that an insufficient amount of data can lead to a high deviation, which means that our method can either produce results with high accuracy or fail.

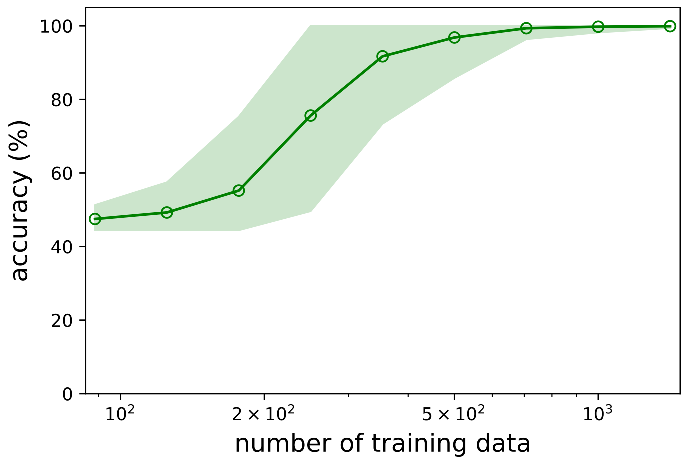

Figure 5: **Accuracy of reconstruction versus the size of training data.** The amount of training data is ranging from about 10^2 to 10^3 with each sampling evolving for $T = 100$ time steps. The results are obtained on CML model with 10 nodes and the network topology is random 4-regular graphs. We set $s = 0.2$ and $r = 3.6$ to generate chaotic dynamics.

5 Conclusion

In this work we introduced GGN, a model-free, purely data-driven method that can simultaneously reconstruct network topology and perform dynamic prediction from time series data of node state. Without any prior knowledge of the network structure and with only the time series of node states, it is able to complete both tasks with high accuracy.

In a series of experiments, we demonstrated that GGN is able to be applied to a variety of dynamical systems, including continuous, discrete, and even binary ones. And we found that in most cases, GGN can reconstruct the network better from non-chaotic data. In order to further explore GGN's properties and to better know its upper limit, we conducted experiments under different network topologies and different data volumes. The results show that the network reconstruction ability of our model is strongly correlated with the complexity of dynamics and the Clustering Coefficient of the network. It is also demonstrated that increasing the data size can significantly improve GGN's net reconstruction performance, while a small data size can result in large deviation and unstable performance.

However, we are well aware that the current work has some limitations. It now focuses only on static graph, and Markovian dynamics. Besides, as the limitation of our computation power, the maximum network size we can process is limited up to 100 nodes. In future works, we will further enhance the capacity of our model so as to process larger networks.

6 Declarations

6.1 Availability of data and material

The datasets generated and/or analysed during the current study are available in https://github.com/bnusss/GGN.

6.2 Competing interests

The authors declare that they have no competing interests.

6.3 Funding

The research is supported by the National Natural Science Foundation of China (NSFC) under the grant numbers 61673070.

6.4 Authors' contributions

JZ conceived and designed the study. ZZ, YZ, SW, JL and RX performed the experiments. ZZ, YZ, SW, JL and RX wrote the paper. JZ reviewed and edited the manuscript. All authors read and approved the manuscript.

6.5 Acknowledgements

We thank professor Wenxu Wang and Qinghua Chen from School of Systems Science, Beijing Normal University for discussion and their generous help.

Author details

[1]School of Systems Science, Beijing Normal University, 100875 Beijing, People's Republic of China. [2]ColorfulClouds Pacific Technology Co., Ltd., No.04, Building C, 768 Creative Industrial Park, Compound 5A, Xueyuan Road, Haidian District, 100083 Beijing, People's Republic of China. [3]School of Physics, Nanjing University, 210093 Nanjing, People's Republic of China.

References

1. Réka Albert and Albert-László Barabási. Statistical mechanics of complex networks. *Reviews of modern physics*, 74(1):47, 2002.
2. Baruch Barzel and Albert-László Barabási. Network link prediction by global silencing of indirect correlations. *Nature biotechnology*, 31(8):720, 2013.
3. Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Advances in neural information processing systems*, pages 4502–4510, 2016.
4. Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.
5. Stefano Boccaletti, Vito Latora, Yamir Moreno, Martin Chavez, and D-U Hwang. Complex networks: Structure and dynamics. *Physics reports*, 424(4-5):175–308, 2006.
6. Niklas Boers, Bedartha Goswami, Aljoscha Rheinwalt, Bodo Bookhagen, Brian Hoskins, and Jürgen Kurths. Complex networks reveal global pattern of extreme-rainfall teleconnections. *Nature*, page 1, 2019.
7. Aleksandar Bojchevski, Oleksandr Shchur, Daniel Zügner, and Stephan Günnemann. Netgan: Generating graphs via random walks. *arXiv preprint arXiv:1803.00816*, 2018.
8. Andrea Brovelli, Mingzhou Ding, Anders Ledberg, Yonghong Chen, Richard Nakamura, and Steven L Bressler. Beta oscillations in a large-scale sensorimotor cortical network: directional influences revealed by granger causality. *Proceedings of the National Academy of Sciences*, 101(26):9849–9854, 2004.

9. Jose Casadiego, Mor Nitzan, Sarah Hallerberg, and Marc Timme. Model-free inference of direct network interactions from nonlinear collective dynamics. *Nature communications*, 8(1):2192, 2017.
10. Nicola De Cao and Thomas Kipf. Molgan: An implicit generative model for small molecular graphs. *arXiv preprint arXiv:1805.11973*, 2018.
11. Victor M Eguiluz, Dante R Chialvo, Guillermo A Cecchi, Marwan Baliki, and A Vania Apkarian. Scale-free brain functional networks. *Physical review letters*, 94(1):018102, 2005.
12. Soheil Feizi, Daniel Marbach, Muriel Médard, and Manolis Kellis. Network deconvolution as a general method to distinguish direct dependencies in networks. *Nature biotechnology*, 31(8):726, 2013.
13. Matthias Feurer, Aaron Klein, Katharina Eggensperger, Jost Springenberg, Manuel Blum, and Frank Hutter. Efficient and robust automated machine learning. In *Advances in neural information processing systems*, pages 2962–2970, 2015.
14. Timothy S Gardner, Diego Di Bernardo, David Lorenz, and James J Collins. Inferring genetic networks and identifying compound mode of action via expression profiling. *Science*, 301(5629):102–105, 2003.
15. Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
16. Geoffrey Hinton, Li Deng, Dong Yu, George Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Brian Kingsbury, et al. Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29, 2012.
17. Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5308–5317, 2016.
18. Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.
19. Kunihiko Kaneko. Pattern dynamics in spatiotemporal chaos: Pattern selection, diffusion of defect and pattern competition intermettency. *Physica D: Nonlinear Phenomena*, 34(1-2):1–41, 1989.
20. Kunihiko Kaneko. Overview of coupled map lattices. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 2(3):279–282, 1992.
21. Stuart A Kauffman. Metabolic stability and epigenesis in randomly constructed genetic nets. *Journal of theoretical biology*, 22(3):437–467, 1969.
22. Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.
23. Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.
24. Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics*, pages 420–422. Springer, 1975.
25. Jingwen Li, Wen-Xu Wang, Ying-Cheng Lai, and Celso Grebogi. Reconstructing complex networks with binary-state dynamics. *arXiv preprint arXiv:1511.06852*, 2015.
26. Yaguang Li, Rose Yu, Cyrus Shahabi, and Yan Liu. Diffusion convolutional recurrent neural network: Data-driven traffic forecasting. *arXiv preprint arXiv:1707.01926*, 2017.
27. Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.
28. Saralees Nadarajah and Samuel Kotz. The beta gumbel distribution. *Mathematical Problems in engineering*, 2004(4):323–332, 2004.
29. Mark EJ Newman. The structure and function of complex networks. *SIAM review*, 45(2):167–256, 2003.
30. Nathaniel Virgo Nicholas Guttenberg, Hidetoshi Aoki Olaf Witkowski, and Ryota Kanai. Permutation-equivariant neural networks applied to dynamics prediction. 2016.
31. Théophane Weber Razvan Pascanu Peter Battaglia Daniel Zoran Nicholas Watters, Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. 2017.
32. Hae-Jeong Park and Karl Friston. Structural and functional brain networks: from connections to cognition. *Science*, 342(6158):1238411, 2013.
33. Yao Quanming, Wang Mengshuo, Jair Escalante Hugo, Guyon Isabelle, Hu Yi-Qi, Li Yu-Feng, Tu Wei-Wei, Yang Qiang, and Yu Yang. Taking human out of learning applications: A survey on automated machine learning. *arXiv preprint arXiv:1810.13306*, 2018.
34. Christopher J Quinn, Todd P Coleman, Negar Kiyavash, and Nicholas G Hatsopoulos. Estimating the directed information to infer causal relationships in ensemble neural spike train recordings. *Journal of computational neuroscience*, 30(1):17–44, 2011.
35. Juan G Restrepo, Edward Ott, and Brian R Hunt. Onset of synchronization in large networks of coupled oscillators. *Physical Review E*, 71(3):036151, 2005.
36. Zhesi Shen, Wen-Xu Wang, Ying Fan, Zengru Di, and Ying-Cheng Lai. Reconstructing propagation networks with natural diversity and identifying hidden sources. *Nature communications*, 5:4323, 2014.
37. Steven H Strogatz. Exploring complex networks. *nature*, 410(6825):268, 2001.
38. Joshua M Stuart, Eran Segal, Daphne Koller, and Stuart K Kim. A gene-coexpression network for global discovery of conserved genetic modules. *science*, 302(5643):249–255, 2003.
39. Marc Timme. Revealing network connectivity from response dynamics. *Physical review letters*, 98(22):224101, 2007.
40. Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.
41. Wen-Xu Wang, Ying-Cheng Lai, Celso Grebogi, and Jieping Ye. Network reconstruction based on evolutionary-game data via compressive sensing. *Physical Review X*, 1(2):021021, 2011.
42. Wen-Xu Wang, Rui Yang, Ying-Cheng Lai, Vassilios Kovanis, and Celso Grebogi. Predicting catastrophes in nonlinear dynamical systems by compressive sensing. *Physical review letters*, 106(15):154101, 2011.

43. Wen-Xu Wang, Rui Yang, Ying-Cheng Lai, Vassilios Kovanis, and Mary Ann F Harrison. Time-series–based prediction of complex oscillator networks via compressive sensing. *EPL (Europhysics Letters)*, 94(4):48006, 2011.
44. Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393(6684):440, 1998.
45. Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.
46. Jiaxuan You, Rex Ying, Xiang Ren, William L Hamilton, and Jure Leskovec. Graphrnn: a deep generative model for graphs. *arXiv preprint arXiv:1802.08773*, 2018.
47. Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. *arXiv preprint arXiv:1709.04875*, 2017.
48. Jiani Zhang, Xingjian Shi, Junyuan Xie, Hao Ma, Irwin King, and Dit-Yan Yeung. Gaan: Gated attention networks for learning on large and spatiotemporal graphs. *arXiv preprint arXiv:1803.07294*, 2018.
49. Fengwen Chen Guodong Long Chengqi Zhang Philip S. Yu Zonghan Wu, Shirui Pan. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2018.