Effective and Efficient Structural Inference with Reservoir Computing

Aoran Wang [1] Tsz Pan Tong [1 2] Jun Pang [1 2]

Abstract

In this paper, we present an effective and efficient structural inference approach by integrating a Reservoir Computing (RC) network into a Variational Auto-encoder-based (VAE-based) structural inference framework. With the help of Bi-level Optimization, the backbone VAE-based method follows the Information Bottleneck principle and infers a general adjacency matrix in its latent space; the RC net substitutes the partial role of the decoder and encourages the whole approach to perform further steps of gradient descent based on limited available data. The experimental results on various datasets including biological networks, simulated fMRI data, and physical simulations show the effectiveness and efficiency of our proposed method for structural inference, either with much fewer trajectories or with much shorter trajectories compared with previous works.

1. Introduction

As widely observed in the real world, many dynamical systems can be modeled as agents interacting with each other and can be viewed as a graph whose nodes represent the agents, edges represent the interactions between the agents, and adjacency matrix represents the underlying structure. The examples include the underlying interacting graphs of physical systems (Kwapień & Drożdż, 2012; Ha & Jeong, 2021), multi-agent systems (Brasó & Leal-Taixé, 2020; Li et al., 2021) and biological systems (Tsubaki et al., 2019; Pratapa et al., 2020). As there emerges a call of understanding dynamical systems, it is fundamental to reveal the underlying structure. The acknowledgment of underlying structure evidently benefits the understanding of the intrinsic mechanisms of dynamical systems, and can further facilitate the prediction and control of these systems.

However, in many cases, only the observable features of agents are available within a given time period, leaving the underlying structure partially or fully concealed beneath the intricate dynamics. Therefore, it is imperative to develop an approach that can unveil the hidden structure of dynamical systems using the observable features of the agents. We call the collection of observed features of all agents in a dynamical system in a time period *a trajectory*. Thus, the trajectory contains rich information about the evolution of the features of agents as a result of their previous features and interactions with other agents (Katok & Hasselblatt, 1995). Many data-driven approaches have been proposed, aiming at inferring the underlying structure of the dynamical systems based on the obtained trajectories (Kipf et al., 2018; Webb et al., 2019; Alet et al., 2019; Graber & Schwing, 2020; Chen et al., 2021; Löwe et al., 2022; Wang & Pang, 2022). These approaches are VAE-based, infer the adjacency matrix of the dynamical systems following the *Information Bottleneck* (IB) principle, and require thousands of trajectories for training. But in many real-world scenarios, the acquisition of trajectories is either expensive or time-consuming, which consequently becomes a remarkable challenge for these VAE-based methods.

In this work, we propose an approach to integrate an RC network into a VAE-based structural inference framework. RC is well-known for its low parameter and low data demand while high accuracy in dynamics prediction (Gallicchio & Micheli, 2011; Gauthier et al., 2021), but since its internal weights are fixed after initialization, the IB principle fails in this case. As a result, the direct substitution of the decoder in the VAE-based structural inference framework with an RC is impracticable. Therefore, we integrate the RC network as a branch diverging from the decoder of the backbone framework and take advantage of Bi-level Optimization (BO) to bridge the nested optimization of both branches. Thanks to the easy-to-train RC network and the proper integration with BO, the new framework requires much less training data compared with previous methods (Kipf et al., 2018; Webb et al., 2019; Chen et al., 2021; Löwe et al., 2022; Wang & Pang, 2022) while maintaining high accuracy. The experimental results show the integration of an RC network promotes the performance of the structural inference method and outperforms previous methods on datasets with

[1]Faculty of Science, Technology and Medicine, University of Luxembourg, Luxembourg [2]Institute for Advanced Studies, University of Luxembourg, Luxembourg. Correspondence to: Aoran Wang <aoran.wang@uni.lu>, Tsz Pan Tong <tszpan.tong@uni.lu>, Jun Pang <jun.pang@uni.lu>.

Proceedings of the 40th International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

much fewer trajectories or with much shorter trajectories. Extensive experiments in the appendix also show the robustness and the time efficiency of the proposed approach. Specifically, our central technical contributions include:

- We propose a novel structural inference approach with an integrated RC network, which is formed as two branches and optimized with the help of BO. According to the best of our knowledge, it is the very first attempt of utilizing an RC network for structural inference.
- The proposed methodology can infer the underlying structure of the dynamical system accurately with much fewer or much shorter trajectories.
- We experimentally show that the integration of RC network will not be a burden to the training process and the framework is robust to the additive Gaussian noise.

2. Related Work

VAE by Kingma & Welling (2013) gained huge success by learning the distribution of latent variables, and VAE has been extensively applied in computer vision (Salimans et al., 2017) and generative models (Goodfellow et al., 2020; Van Den Oord et al., 2017). However, it was not until Kipf et al. (2018) applied VAE on structure inference in their Neural Relational Inference (NRI) framework. NRI uses a message-passing design in its encoder and interprets the latent variable as the network structure. Webb et al. (2019) further proposed factorized Neural Relational Inference (fNRI) to generalize NRI to a multi-interaction system. Chen et al. (2021) modified the message-passing algorithm by incorporating spatio-temporal information and structural prior. Alemi et al. (2017) suggested the principle of Variational Information Bottleneck (VIB) to generalize VAE with a solid theoretical ground based on information theory, and Wang & Pang (2022) have proposed the iterative Structural Inference of Directed Graphs (iSIDG) based on VIB and NRI. iSIDG encouraged a tighter bound of VIB and achieved a 5-10% boost compared to the aforementioned models. In another line of work on Granger Causality (GC), Wu et al. (2020) proposed the Minimum Predictive Information Regularization (MPIR) model and used a learnable noise mask on each node to lower the computational cost. Löwe et al. (2022) introduced Amortized Causality Discovery (ACD) under the VAE framework to learn causality strength between nodes. However, these models either require a vast amount of data or have relatively poor performance, making them inapplicable in some fields where high fidelity of structure inference is needed but data collection is expensive.

Echo State Network (ESN) and Liquid State Machine (LSM) were individually proposed by Jaeger (2001) and Maass et al. (2002), respectively, later unified under the name of Reservoir Computing (RC) (Schrauwen et al., 2007). Both ESN and LSM have a recurrent structure and only the output layer is trained, but LSM uses biologically inspired sparsely connected leaky integrate-and-fire neurons, while ESN uses ordinary neurons. A mass collection of RC-based frameworks for time-series prediction have been proposed (Wyffels & Schrauwen, 2010; Pathak et al., 2018; Dong et al., 2019; Bianchi et al., 2020). These frameworks are characterized by a relatively small amount of data required. Interestingly, among them, there are several attempts trying to combine an RC with a VAE (Suh et al., 2016; Cabrera et al., 2017; Canaday et al., 2021). Interested readers may read more about the recent advancements in the field of RC in literature (Nakajima & Fischer, 2021; Gauthier et al., 2021). It is worth mentioning that none of the work explicitly infers the structure of the underlying interacting graph. The RC net in this paper follows a modernized version of the hierarchical ESN, the DeepESN (Gallicchio et al., 2017).We integrate the RC net into the structural inference method as a branch for future feature prediction, hoping it can effectively promote the data efficiency of the whole method.

Bi-level Optimization (BO) is a typical mathematical problem in which the feasible region of the upper-level optimization problem is constrained by the solution set of the lower-level optimization problem (Stackelberg et al., 1952; Dempe, 2020b), and some crucial issues in machine learning, such as hyperparameter optimization (Franceschi et al., 2018) and reinforcement learning (Hong et al., 2020), can be reformulated as BO problems. Conventional methods in this field include hypergradient descent (Pedregosa, 2016; Grazzi et al., 2020), stationary-seeking method (Mehra & Hamm, 2021), and value function approach (Outrata, 1990; Liu et al., 2021). In this work, we utilize a BO algorithm to bridge the training of both branches as nested upper-level and lower-level optimization problems while satisfying the principle of VIB.

3. Preliminaries

3.1. Notations and Problem Formulation

We view a dynamical system as a directed graph, by representing the agents of the system as the nodes of the graph, and the directed interactions between the agents as the directed edges in the graph. The directed graph is denoted as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is the set of features of n nodes, and $\mathcal{E}$ is the set of edges. Based on $\mathcal{E}$, we derive an asymmetric adjacency matrix $\mathbf{A} \in \mathbb{R}^{n\times n}$, where $\mathbf{a}_{ij} \in \{0, 1\}$ indicates the presence ($\mathbf{a}_{ij} = 1$) or the absence ($\mathbf{a}_{ij} = 0$) of the edge from node i to j. The edges are assumed to be static and do not evolve with time. The nodes' features over a time period are represented as trajectories: $\mathcal{V} = \{V^{(0)}, V^{(1)}, \dots, V^{(T)}\}$, with $T + 1$ time steps, and $V^{(t)}$ is the set of features of all n nodes at time step t: $V^{(t)} = \{v_0^{(t)}, v_1^{(t)}, \dots, v_n^{(t)}\}$. Given the node features over a time period, the structural inference problem in this paper is to reconstruct the asymmetric adjacency matrix $\mathbf{A}$ of the whole graph in an unsupervised way.

We state the problem of *structural inference* as searching for a combinatorial distribution to describe the existence of the edges between any of the node pairs in the graph. Despite many works, which are based on the principle of IB, have been proposed to solve the problem of structural inference (Kipf et al., 2018; Webb et al., 2019; Chen et al., 2021; Löwe et al., 2022; Wang & Pang, 2022), they demand a huge amount of data, either sampling thousands of trajectories or require relatively long trajectories. In order to address this problem, we integrate an RC network into the structural inference framework and utilize a BO algorithm to deal with fewer and shorter available trajectories.

3.2. Structural Inference with Information Bottleneck

The principle of IB provides an explanation and understanding of learning with deep neural networks (Tishby et al., 1999; Tishby & Zaslavsky, 2015; Shwartz-Ziv & Tishby, 2017). In general, for the input data X and its target label Y, the IB aims to learn the minimal sufficient representation $Z = \arg\min_Z I(Z; X) - \mathfrak{u} \cdot I(Z; Y)$, where $I(\cdot, \cdot)$ represents the mutual information between two variables, and $\mathfrak{u}$ is the Lagrangian multiplier to balance sufficiency and minimality.

Moreover, Alemi et al. (2017) presents a variational approximation to the IB theory (VIB), and proves variational auto-encoder (VAE) objective is a special case of the variational approximation. Previous structural inference methods utilize a VAE as the backbone (Kipf et al., 2018; Webb et al., 2019; Chen et al., 2021; Löwe et al., 2022; Wang & Pang, 2022), and obtain the adjacency matrix $\mathbf{A}$ from the latent space $\mathbf{Z}$ of VAE:

$$\mathbf{Z} = \arg\min_{\mathbf{Z}} I(\mathbf{Z}; V^t) - \mathfrak{u} \cdot I(\mathbf{Z}; V^{t+1}). \tag{1}$$

Therefore, the VAE learns the minimal sufficient statistics from the present node features, which are sufficient to derive the future node features. In the case of inferring the structure of dynamical systems, the Markovian assumption works and ensures the formulation: $V^{t+1} \leftarrow \{V^t; \mathbf{A}\}$ (Wang & Pang, 2022), so that in these VAE-based methods, we can sample the adjacency matrix of the graph from the minimal sufficient statistics obtained in the latent space. But these works require an enormous amount of data, such as at least 12000 trajectories, and at least 49 time steps in each trajectory (Kipf et al., 2018; Webb et al., 2019; Wang & Pang, 2022), which definitely becomes a considerable challenge for real-world applications, especially when obtaining of such trajectories is either expensive or time-consuming.

3.3. Reservoir Computing

In this section, we provide a brief discussion on the background knowledge of Reservoir Computing (RC). The exact implementation of RC may refer to Sections 4.1 and 4.2. RC is a machine learning paradigm that is especially suitable for learning dynamical systems, even when the systems exhibit chaotic or complex spatio-temporal behaviors (Dong et al., 2020; Gauthier et al., 2021). An RC is based on a recurrent artificial neural network with a pool of interconnected neurons - the reservoir, and an input layer feeding sequential input data to the network, and an output layer weighting the network's states (Gauthier et al., 2021). We denote $\mathbf{u}^{(t)} \in \mathbb{R}^d$ as the input sequential data, $\mathbf{h}^{(t)} \in \mathbb{R}^N$ represents the state of the reservoir, and N represents the number of neurons in the reservoir. In general, the dynamics can be formulated as:

$$\mathbf{h}^{(t+1)} = \frac{1}{\sqrt{N}} f\Big(\mathbf{W}_r \mathbf{h}^{(t)} + \mathbf{W}_n \mathbf{u}^{(t)}\Big), \tag{2}$$

where $\mathbf{W}_r \in \mathbb{R}^{N \times N}$ and $\mathbf{W}_n \in \mathbb{R}^{N \times d}$ are the reservoir and input weight matrices, respectively. These weights are fixed after initialization, by sampling the weights according to an i.i.d. Gaussian distribution and regularizing the spectral radius of the weight matrix (Gallicchio & Micheli, 2011). $f(\cdot)$ is an element-wise non-linearity. In most cases, it is a hyperbolic tangent (Dong et al., 2020). We may also stack multiple RCs, but in this section, we will keep the minimal formulation in Equation 2 for simplicity. The RC is utilized to predict a given output $\mathbf{o}^{(t+1)} \in \mathbb{R}^c$, and we obtain it after passing the final layer:

$$\hat{\mathbf{o}}^{(t+1)} = \mathbf{W}_o \mathbf{h}^{(t+1)}, \tag{3}$$

where $\mathbf{W}_o$ denotes the weight of the final output layer. RCs are characterized by fixed internal weights and only the last linear layer is trained. Therefore, training RCs requires a relatively small amount of data in comparison with ordinary deep learning frameworks. The expressivity and the power of RC rather lie in the high-dimensional non-linear dynamics of the reservoir. As a result, by integrating an RC into a VAE-based structural inference framework, we try to fulfill our objective: reducing the amount of data needed for structural inference. However, since it has not been solidly and theoretically justified whether the training process of RC follows IB, we cannot simply replace either the encoder or decoder of the VAE with an RC. Here, the RC is designed as nesting with the general pipeline of structural inference, which demands a BO algorithm.

3.4. Bi-level Optimization

Bi-level Optimization (BO) is a hierarchical mathematical programming task where the feasible region of one optimization task is restricted by the solution set mapping of another optimization task (Liu et al., 2022b). Thus it contains two levels of optimization tasks, where one is nested within the other. The inner (or nested) and outer optimization tasks are often respectively referred to as the Lower-Level (LL) and Upper-Level (UL) subproblems (Dempe, 2020a). A standard BO problem can be formally expressed as:

$$\min_{\psi,\theta} F(\psi,\theta) \text{ s.t. } \theta \in \arg\min_{\theta'} g(\psi,\theta'), \tag{4}$$

where the goal is to minimize the upper objective F whose variables include the solution of another minimization problem w.r.t. the lower objective g. The θ and ψ are the lower and upper variables, respectively. A major class of BO methods is based on direct gradient descent on the upper variable ψ while viewing the optimal lower variable $\theta^*(\psi) = \arg\min_\theta g(\psi,\theta)$ as a (uniquely defined) function of ψ. Yet the calculation of the derivative $\nabla_\psi \theta^*(\psi)$ requires expensive manipulation of the Hessian matrix of g. In this work, we follow the idea of BOME (Liu et al., 2022a), which is a simple first-order BO algorithm that depends only on first-order gradient information and requires no implicit differentiation. We show that with the help of BOME, RC can be integrated into the frameworks for structural inference, and thus the strength of RC and VAE-based structural inference methods can be combined and unified.

4. Model Design

4.1. Reservoir Computing for Structural Inference

RC has proven excellent performance in time series prediction and has small demand on the amount of training data (Cabrera et al., 2017; Gauthier et al., 2021; Srinivasan et al., 2022). Because the weights in the reservoir are randomly generated and do not change during training, it is difficult to follow IB, and searching for a layer in the reservoir that can be a minimal sufficient representation for the adjacency matrix of the system is impractical if we want to train the RC in an end-to-end fashion. Inspired by a series of Siamese networks (Guo et al., 2017; Zhang & Peng, 2019; Javed et al., 2022), which have network branches sharing partial weights, we align the RC as a branch from the decoder of VAE (see the upper part in Figure 1). Recall the mechanisms of the decoder in VAE-based structural inference methods, the decoder receives the inferred adjacency matrix and the present node features as input, trying to predict the future node features:

$$\tilde{\mathbf{d}}_{ij}^{(t)} = z_{ij} \cdot f_e(v_i^{(t)}, v_j^{(t)}), \tag{5}$$

$$\mathbf{u}_j^{(t)} = v_j^{(t)} + f_v \sum_{i \neq j} \tilde{\mathbf{d}}_{ij}^{(t)}, \tag{6}$$

where z_{ij} represents the inferred edge from node i to j in the latent space $\mathbf{Z}$, f_e and f_v denote the embedding networks, and are implemented as multi-layer perceptrons. In addition, there are three linear layers attached after these two steps for output. We feed the learned representation of every node $\mathbf{u}_j^{(t)}$ to RC, and perform the following operation:

$$\hat{\mathbf{o}}_j^{(t+1)} = \mathcal{J}_{RC}(\mathbf{u}_j^{(t)}), \tag{7}$$

where $\mathcal{J}_{RC}(\cdot)$ represents the RC, and $\hat{\mathbf{o}}_j^{(t+1)}$ is the predicted future features of node j. The RC in this work consists of three identical RC cells and a linear output layer:

$$\mathcal{J}_{RC}(\mathbf{u}_j^{(t)}) = f_{out}(\mathcal{J}_{Cell,1}, \mathcal{J}_{Cell,2}, \mathcal{J}_{Cell,3}) \tag{8}$$

where $\mathcal{J}_{Cell,n}$ represents the output from RC cell n, and f_{out} denotes the final linear output layer. These RC cells are identical in architecture but are initialized independently. RC cells receive cascade inputs, but their outputs are collected parallelly. Details about RC cells are in Section 4.2. The integration of RC into the structural inference framework hence has the following advantages:

1. The structural invariance of graph data (Battaglia et al., 2018) is taken care by the message-passing-like operations in the decoder. In this case, the number of possible neighbors is the most concerning invariance, and RC cannot deal with it directly.

2. Both node features and inferred structures are contained in the learned representations $\mathbf{u}_j^{(t)}$, and RC can consequently acquire both types of information simultaneously.

3. Compared with the rest layers in the decoder after the bifurcation and the stacked linear layers after the decoder, RC has much fewer parameters. As a consequence, it is easier to train and leads to a more accurate prediction.

4. The IB principle of VAE-based structural inference method $\mathcal{J}_{SI}$ is not violated, since the divergence starts from the second layer in the decoder, the latent space in VAE still infers the adjacency matrix of the graph.

5. RC has proven robustness against noise (Jalalvand et al., 2018; Vlachas et al., 2020; Chen et al., 2020). The integration of RC into the structural inference framework may improve the robustness of the whole approach.

Although the RC branch is assumed not to violate the principle of VIB in structural inference methods, it is still necessary to incorporate the optimization of both branches with the help of BO, and it is discussed in Section 4.3.

4.2. RC Cells

The idea of RC cells comes from Vlachas et al. (2020), which consists of an Elman-RNN with proper initialization. Gallicchio et al. (2017) propose a reservoir design with leaky integrator units, and we follow this design. Given

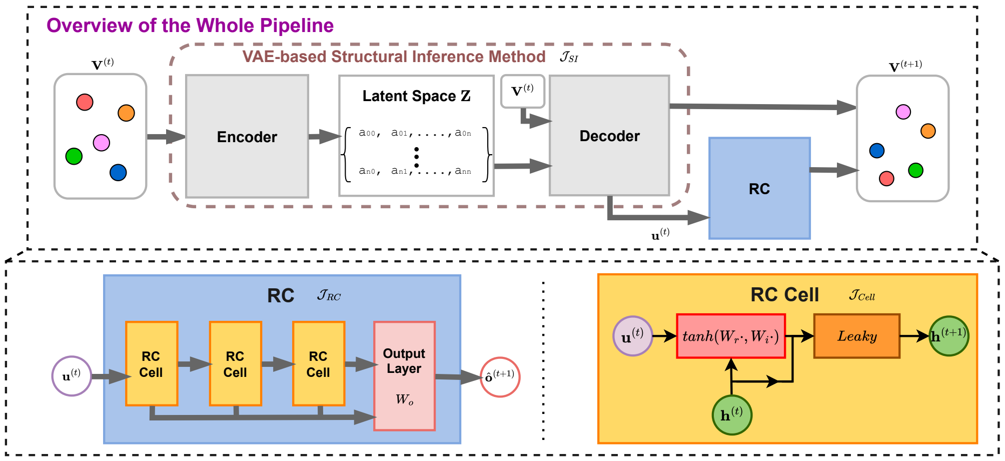
Overview of the Whole Pipeline
VAE-based Structural Inference Method $\mathcal{J}_{SI}$
$\mathbf{v}^{(t)}$
Encoder
Latent Space Z
$a_{00}, a_{01}, \ldots, a_{0n}$
$a_{n0}, a_{n1}, \ldots, a_{nn}$
Decoder
$\mathbf{v}^{(t+1)}$
RC
$\mathbf{u}^{(t)}$
RC $\mathcal{J}_{RC}$
RC Cell
Output Layer
W_o
$\hat{\mathbf{o}}^{(t+1)}$
RC Cell $\mathcal{J}_{Cell}$
$tanh(W_r\cdot, W_i\cdot)$
Leaky
$\mathbf{h}^{(t+1)}$
$\mathbf{h}^{(t)}$

Figure 1. **(Above)** The overview of the whole pipeline. **(Below)** Details about the RC and RC cell in this work.

input feature of node j at time t: $\mathbf{u}_j^{(t)}$, we have:

$$\mathbf{h'}_j^{(t)} = f(\mathbf{W}_r \cdot \mathbf{h}_j^{(t-1)} + \mathbf{W}_n \cdot \mathbf{u}_j^{(t)}), \tag{9}$$

$$\mathbf{h}_j^{(t)} = (1-\alpha)\mathbf{h}_j^{(t-1)} + \alpha\mathbf{h'}_j^{(t)}, \tag{10}$$

where $\mathbf{h}^{(t)}$ denotes the state of the RC cell at time t, $f(\cdot)$ is a hyperbolic tangent function, $\mathbf{W}_r$ and $\mathbf{W}_n$ represent reservoir state weight matrix and input weight matrix, respectively. Moreover, $\alpha \in [0, 1]$ is the leaking rate. Suppose we have time series $\{\mathbf{u}_j^{(0)}, \mathbf{u}_j^{(1)}, ..., \mathbf{u}_j^{(T-1)}\}$, an RC cell will run on all of these time steps and we obtain the output:

$$\mathcal{J}_{Cell} = [\mathbf{h}_j^{(0)}, \mathbf{h}_j^{(1)}, ..., \mathbf{h}_j^{(T-1)}]_t, \tag{11}$$

where $[\cdot,\cdot]_t$ denotes concatenation along the dimension of time. We may also extract the state of the RC cell at time t for node j: $\mathbf{h}_j^{(t)}$ for further calculation. Then three RC cells are stacked to build a DeepESN (Gallicchio et al., 2017), and the following RC cell just takes in the computed states of previous RC cell $\mathcal{J}_{Cell}$ as input. Then we compute the output at time t from the global state of all RC cells:

$$\hat{\mathbf{o}}_j^{(t+1)} = f_{out}([\mathbf{h}_{j,1}^{(t)}, \mathbf{h}_{j,2}^{(t)}, \mathbf{h}_{j,3}^{(t)}]), \tag{12}$$

where $[\cdot,\cdot]$ denotes concatenation, and $\mathbf{h}_{j,n}^{(t)}$ denotes the state of RC cell n at time t for node j, while f_{out} represents a linear output layer with weight $\mathbf{W}_o$. The utilization of a DeepESN instead of a "standard shallow RC", which only consists of one RC cell, has the following advantages:

1. Compared with a standard shallow RC, the structured state space organization with multiple time-scale dynamics DeepESN is intrinsic to the nature of compositionality of recurrent neural models (Gallicchio et al., 2017).

2. As proved by Gallicchio et al. (2017), DeepESN architecture can be seen as a simplification of the standard shallow RC, leading to a reduction in the absolute number of recurrent weights. So the standard shallow RC can be seen as a special case of DeepESN.

However, similar to standard shallow RC, which should be initialized to fulfill the echo state property (ESP) (Gallicchio & Micheli, 2011; Yildiz et al., 2012), DeepESN also requires special initialization of the weights in the reservoir of RC cells. We follow the necessary and sufficient conditions proved by Gallicchio & Micheli (2017), and initialize the weights in RC cells by first randomly drawing weights $\mathbf{W}_r$ and $\mathbf{W}_n$ from a uniform distribution in $[-1, 1]$, and then rescale them so that their spectral radius is smaller than one or the Lipschitz constant of the state transition function in DeepESN is smaller than one. For details about the initialization, please refer to Section A.1 of this paper.

4.3. Training with Bi-level Optimization

Since we have two branches, they have different loss functions. For the branch of the VAE-based structural inference method, the loss function $\mathcal{L}_{SI}$ can be summarized as:

$$\mathcal{L}_{SI} = \mathcal{L}_p + \beta \cdot \mathcal{L}_{KL} + \gamma \cdot \mathcal{R}, \tag{13}$$

where $\mathcal{L}_p$ and $\mathcal{L}_{KL}$ are the loss terms for prediction and Kullback–Leibler (KL) divergence, $\mathcal{R}$ represents the regularization terms, and $\{\beta, \gamma\}$ are weight terms (Kipf et al., 2018; Webb et al., 2019; Chen et al., 2021; Wang & Pang, 2022). For the branch of RC, the loss function $\mathcal{L}_{RC}$ is calculated with Normalized Root Mean Square Error (NRMSE):

$$\mathcal{L}_{RC} = \frac{1}{N}\sum_j \sqrt{\frac{\sum_{t=0}^{T-1}(\hat{\mathbf{o}}_j^{(t+1)} - v_j^{(t+1)})^2}{T-1}} \cdot \frac{1}{\bar{\mathbf{o}}_j}, \tag{14}$$

where N represents the total number of nodes in the graph, and $\bar{\mathbf{o}}_j = \sum_{t=0}^{T-1} \hat{\mathbf{o}}_j^{(t+1)}/(T-1)$. The loss terms $\mathcal{L}_{SI}$ and $\mathcal{L}_{RC}$ are divergent, and so are the branches. Although both branches output the prediction on future node features, the

Algorithm 1 Pipeline of BO

```
**Goal**: $\min_{\psi,\theta} \mathcal{J}_{UL}(\psi,\theta)$ s.t. $\theta \in \arg\min_{\theta'} \mathcal{J}_{LL}(\psi,\theta')$.
**Input**: Initialization $(\psi_0,\theta_0)$; inner step $T'$; outer and inner step size $\xi$, $\tau$ (set $\tau = \xi$ by default).
**for** iteration $k$ **do**
    1. Get $\theta_k^{(T')}$ by $T'$ steps of gradient descent on $\mathcal{J}_{LL}(\psi_k,\cdot)$ starting from $\theta_k$.
    2. Set $\hat{q}(\psi,\theta) = \mathcal{J}_{LL}(\psi,\theta) - \mathcal{J}_{LL}(\psi,\theta_k^{(T')})$.
    3. Update $(\psi,\theta)$ : $(\psi_{k+1},\theta_{k+1}) \leftarrow (\psi_k,\theta_k) - \xi(\nabla\mathcal{J}_{UL}(\psi_k,\theta_k) + \lambda_k\nabla\hat{q}(\psi_k,\theta_k))$
    where $\lambda_k = \max\left(\frac{\phi_k - \langle\nabla\mathcal{J}_{UL}(\psi_k,\theta_k),\ \nabla\hat{q}(\psi_k,\theta_k)\rangle}{||\hat{q}(\psi_k,\theta_k)||^2}, 0\right)$,
    and $\phi_k = \eta||\hat{q}(\psi_k,\theta_k)||^2$ (default),
    or $\phi_k = \eta\hat{q}(\psi_k,\theta_k)$ with $\eta > 0$.
**end for**
```

layer setups are divergent and do not have any shared weight. Therefore, training in our model is not comparable with the training of a Siamese network, which has many shared weights and similar layer setups. Here comes the challenge: how to train both branches so that either of them won't be negatively affected by the other?

We utilize BO, which is a class of optimization problems with nested tasks (Liu et al., 2022b). Recall the general pipeline described in Section 4.1 and shown in the upper part in Figure 1, the optimization of the whole model consists of two tasks: (a) the LL task: $\mathcal{J}_{LL}(\psi,\theta') = \mathcal{L}_{SI}$, which corresponds to the training of VAE-based structural inference branch; and (b) the UL task: $\mathcal{J}_{UL}(\psi,\theta) = \mathcal{L}_{RC}$, which corresponds to the training of RC branch, and $\{\psi,\theta\}$ are the parameters to be optimized. As mentioned in Section 4.1, the RC branch diverges from the decoder of VAE, so in practice, we formalize the UL task as $\mathcal{J}_{UL}(\psi,\theta) = \mathcal{L}_{RC} + \beta\cdot\mathcal{L}_{KL} + \gamma\cdot\mathcal{R}$, which also takes care of the partial VAE. So the BO utilized in this work is:

$$\min_{\psi,\theta} \mathcal{J}_{UL}(\psi,\theta) \ \text{ s.t. } \theta \in \arg\min_{\theta'} \mathcal{J}_{LL}(\psi,\theta'). \qquad (15)$$

We follow BOME (Liu et al., 2022a) as the BO algorithm in this work, which uses a modified dynamic barrier gradient descent on the value function of BO, and only requires first-order gradients. We describe the pipeline of BO in this work in Algorithm 1, where $\langle\cdot,\cdot\rangle$ represents element-wise multiplication. Interestingly, Liu et al. (2022a) introduce the notion of attraction points (see Section A.3 for details) during gradient descent. The attraction of (ψ,θ) is where the gradient descent algorithm can not make improvement and is implemented with an approximation in step 1 of Algorithm 1. BOME can optimize the UL objective function from this attraction of LL optimization without being trapped to the discontinuous attraction points on the boundary of the attraction basin. In our case, after setting the value of T' properly, the attraction is where the optimization of $\mathcal{J}_{LL}$ stagnates, and the inferred adjacency matrix by the framework at this step is no less accurate than the one inferred by vanilla VAE-based method. Then the training process turns to optimize the upper-level objective function $\mathcal{J}_{UL}$. With the help of BOME, the optimization of RC branch can also benefit the other branch, and further promote the performance of structural inference.

Besides that, as mentioned in previous works, the two basic elements supporting the VAE-based structural inference methods are the Markovian assumption and the principle of IB (Kipf et al., 2018; Wang & Pang, 2022). We denote $\mathbf{H}$ as the layer in the decoder whose outputs are fed to RC. When $\mathcal{J}_{LL}$ stagnates, following the principles of IB, and according to Lagrangian $I(\mathbf{H};V^{(t)}) - \text{u}\cdot I(\mathbf{H};V^{(t+1)})$, the obtained representation from layer $\mathbf{H}$ is the minimal sufficient statistics and simplest mapping of $V^{(t)}$ to $V^{(t+1)}$). Based on the Markovian assumption $V^{t+1} \leftarrow \{V^t;\mathbf{A}\}$ (Wang & Pang, 2022), $\mathbf{H}$ outputs the minimal sufficient representation for both present node features V^t and inferred adjacency matrix $\mathbf{A}$. As a result, the training of $\mathcal{J}_{UL}$ starts from the input of minimal sufficient representation, and will further update this representation and encourage the whole framework to infer the adjacency matrix more precisely. Please refer to the appendix for more details about BO in this work. We summarize the whole pipeline in Figure 1, and in Algorithm 2 in the appendix. To conclude, the utilization of BOME in this work has the following advantages:

1. BOME is a first-order BO method with fast optimization. Since the framework consists of two branches, BOME can efficiently optimize the nested branches and will not be a burden to the computational resources.
2. With a proper setting of T', BOME ensures the optimization of upper-level objective starts from the stagnation of lower-level objective. In the problem setting of this work, RC will be optimized based on the minimal sufficient representation of both present node states and inferred adjacency matrix from VAE.
3. From a broader perspective, the utilization of a BO algorithm can help with data hyper-cleaning (Liu et al., 2022a), which increases the robustness of the structural inference framework against noise.

5. Experiments

We test our method on twelve different datasets, with different numbers of available trajectories and various lengths. More experimental results can be found in Section C.

5.1. General Settings

Backbone VAE-based Method. We choose iSIDG (Wang & Pang, 2022) with GIN (Xu et al., 2019) encoder and MLP decoder as the backbone VAE-based method. iSIDG is the state-of-the-art structural inference method for both directed

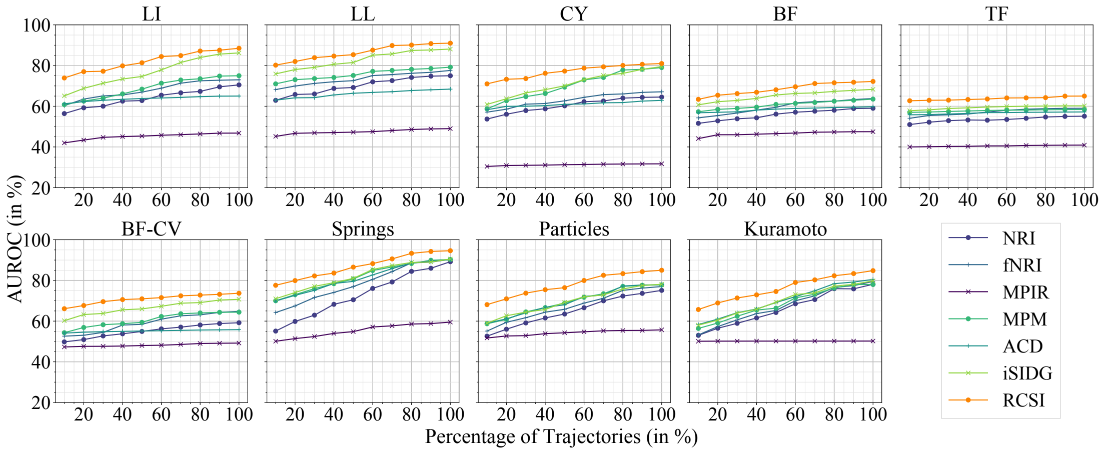

Figure 2. Plots of AUROC values (in %) of different methods as a function of the percentage of the available trajectories. The results are the average of ten runs. 100% trajectories equal 12000 trajectories. The percentages of the trajectories in the figure are 10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90% and 100%, respectively. The subplots share the same x-axis and y-axis.

and undirected graphs. It updates the assumed adjacency matrix at the input side of the encoder with direction information from inferred adjacency matrix, and it has been experimentally evaluated on more than 15 datasets. We simply replace the $\mathcal{M}_{SI}$ in Algorithm 2 (in Appendix) as iSIDG, and replace the loss function $\mathcal{L}_{SI}$ as the one from iSIDG. For the hyperparameters such as the weights combining different terms in the loss function, and the weights for stop condition, we keep the same values as those of iSIDG. We name the new approach RCSI.

Datasets. Similar to iSIDG, we test our model on the six directed synthetic biological networks (Pratapa et al., 2020), namely Linear (LI), Linear Long (LL), Cycle (CY), Bifurcating (BF), Trifurcating (TF), and Bifurcating Converging (BF-CV) networks, which are essential components leading to a variety of different trajectories that are commonly observed in differentiating and developing cells (Saelens et al., 2019). We use BoolODE (Pratapa et al., 2020) to simulate the process of developing cells with these synthetic networks and record the raw trajectories with 49 time steps. But the number of trajectories used in the following experiments varies, which is different from iSIDG (Wang & Pang, 2022), and we perform subsampling of time steps on the raw trajectories. We randomly divide the trajectories into training set, validation set and test set with a ratio of $8:2:2$. The features at every node are one-dimensional mRNA expression levels.

We also test our model on NetSim datasets (Smith et al., 2011) of simulated fMRI data. The NetSim datasets consist of simulated blood-oxygen-level-dependent imaging data across different regions within the human brain, which are asymmetric relation networks. Since the datasets are small, we use the same amount of trajectories as (Löwe et al., 2022; Wang & Pang, 2022). We sample 49 snapshots on each trajectory with equal intervals, but afterward, we subsample trajectories with fewer time steps for further experiments. The node features at every time step are one-dimensional.

Besides, we also select three physical simulations mentioned in (Kipf et al., 2018), namely springs, charged particles and phase-coupled oscillators (Kuramoto model). We keep the symmetric interactions in the setting of these networks, and follow the sampling process used by iSIDG, except with a different number of trajectories for each experiment. We also subsample the trajectories with different counts of time steps, and randomly divide them into training set, validation set and test set with a ratio of $8:2:2$. The features at every node are four-dimensional.

Baselines and metrics. We compare RCSI with the state-of-the-art methods for structural inference:

- NRI (Kipf et al., 2018): a VAE-based model for unsupervised relational inference.
- fNRI (Webb et al., 2019): an NRI-based model with additional latent space for every factorized interaction type.
- MPIR (Wu et al., 2020) a model based on minimum predictive information regularization.
- MPM (Chen et al., 2021): an NRI-based method with a relation interaction mechanism and a spatio-temporal message passing mechanism.
- ACD (Löwe et al., 2022): a variational model that leverages shared dynamics to infer causal relations.
- iSIDG (Wang & Pang, 2022): a VAE-based model which iteratively updates the adjacency matrix to be fed to encoder with direction information.

We describe the implementation details in Section B. We compare our evaluation results using the following metrics: the area under the receiver operating characteristic (AUROC) of the inferred adjacency matrix to the ground truth.

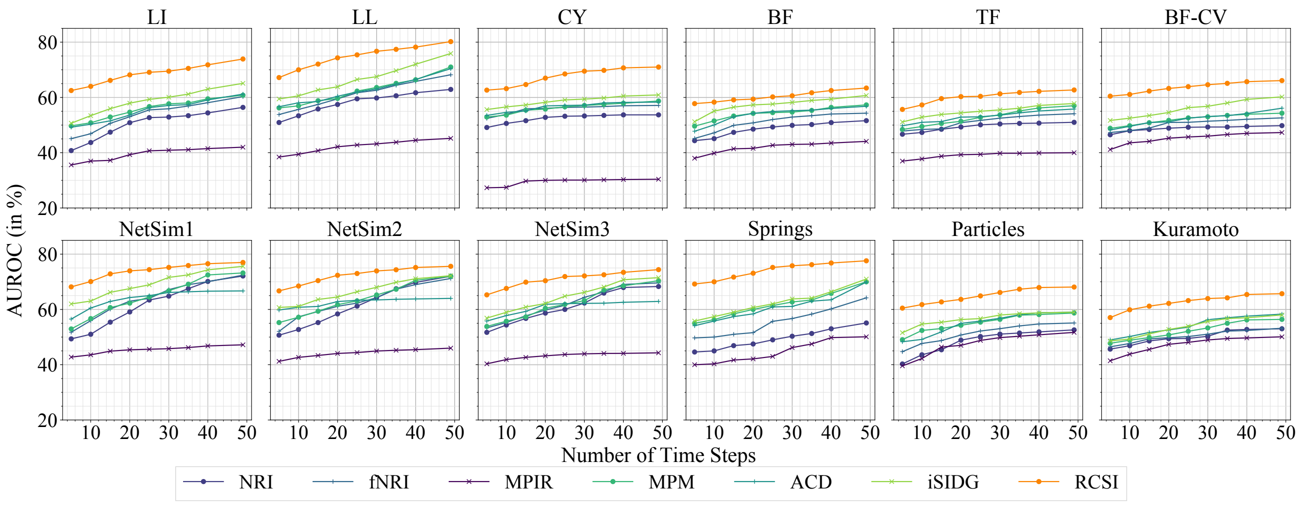

Figure 3. Plots of AUROC values (in %) as a function of the number of time steps. The results are the averaged of ten runs. The numbers of time steps in the figure are 5, 10, 15, 20, 25, 30, 35, 40, and 49, respectively. The subplots share the same x-axis and y-axis.

5.2. Experiments on Fewer Trajectories

In this section, we study the performance of all of the methods when the input trajectories are fewer. The term "fewer" represents that the amount of available trajectories is less than those in previous works (Kipf et al., 2018; Webb et al., 2019; Wang & Pang, 2022). The experimental results of the proposed model and baseline methods on synthetic networks and physical simulations as a function of the percentage of trajectories are summarized in Figure 2. The trajectories have 49 time steps. Since the numbers of trajectories in NetSim datasets are relatively small, we test RCSI and baseline methods on NetSim datasets with 100% trajectories, and the results are shown in Table 3 in the appendix. As shown in Figure 2, with only 60% trajectories, RCSI can outperform almost all of the baseline methods running with 100% trajectories on all of the nine datasets, which experimentally verifies the effectiveness and data-efficiency of RCSI. Specifically, on the dataset of "TF", RCSI outperforms all of the baseline methods with only 10% trajectories. Although iSIDG performs conspicuously on the datasets of directed graphs, no matter how many trajectories are available, RCSI still manages to outperform iSIDG on these datasets. Despite the decrease in performance of all of the methods with fewer trajectories, RCSI succeeds in holding its leading position. Besides, when the available trajectories become fewer, the advantage of RCSI over the baseline methods also becomes greater, which clearly reveals the merit of RCSI in the application scenario of fewer available data.

5.3. Experiments on Shorter Trajectories

In this section, we study the performance of all of the methods when the trajectories are shorter. The term "shorter" represents that the amount of sampled time steps in each trajectory is less than those in Section 5.2. We first sample 1,200 trajectories for synthetic networks and physical simulations, then we subsample the trajectories with different numbers of time steps. For NetSim datasets, we perform subsampling on time steps from raw trajectories directly. The results are shown in Figure 3. As expected, all of the methods perform worse when the trajectories become shorter, while RCSI maintains a noticeable advantage over all of the baseline methods. It is worth mentioning that just with trajectories of 15 time steps, RCSI outperforms all of the baseline methods with 49 time steps of data on almost all of the datasets. In some datasets such as CY, BF-CV, and Springs, the results of RCSI with only 5 time steps are even more accurate than the results of other baseline methods with 49 time steps. These results experimentally evaluated the effectiveness and data efficiency of RSCI when provided with shorter trajectories. However, similar to the baseline methods, RCSI also suffers from shorter trajectories, which suggests that longer time series are embedded with rich information that cannot be easily modeled by RC. Despite that, RCSI still manages to outperform all of the baseline methods. But since RC net in this work follows the Echo State Property, and the steps to reach echo state vary from system to system, it would be interesting to study the least sufficient time steps to infer an essentially accurate adjacency matrix for a specific type of dynamical system.

5.4. Ablation Study

We conduct an ablation study to investigate the impact of different designs of the RC network on the performance of RCSI. The following RC network designs are considered:

- Single RC cell: A single RC cell with varying reservoir sizes of 20, 40, 60, or 80 neurons.
- Two RC cells: Two RC cells, each with reservoir sizes of 10, 20, 30, or 40 neurons.

Table 1. Ablation study on the design of the RC network.

RC Setup	Springs Dataset		
	50% traj. w. 49 T.S.	Full traj. w. 49 T.S.	Full traj. w. 25 T.S.
1 RC @ 20	78.6± 0.10	86.1± 0.10	66.4± 0.10
1 RC @ 40	81.8± 0.10	91.0± 0.08	70.0± 0.10
1 RC @ 60	84.2± 0.05	93.0± 0.05	71.5± 0.05
1 RC @ 80	85.6± 0.04	94.1± 0.04	74.6± 0.04
2 RCs @ 10	81.3± 0.05	91.0± 0.05	69.8± 0.05
2 RCs @ 20	83.2± 0.04	92.2± 0.04	71.2± 0.04
2 RCs @ 30	85.2± 0.05	94.1± 0.05	74.8± 0.04
2 RCs @ 40	86.2± 0.04	94.5± 0.04	75.0± 0.05
3 RCs @ 10	83.1± 0.04	92.8± 0.04	73.8± 0.05
3 RCs @ 20	86.5± 0.04	94.6± 0.04	75.2± 0.06
3 RCs @ 30	87.0± 0.03	94.8± 0.04	75.2± 0.05
3 RCs @ 40	87.1± 0.02	94.9± 0.03	75.3± 0.03

- Three RC cells: Three RC cells, each with reservoir sizes of 10, 20, 30, or 40 neurons.

To evaluate the performance of each RCSI variation, we examined three different sets of trajectories from springs dataset: 50% of trajectories with 49 timesteps (50% traj. w. 49 T.S.), the full set of trajectories with 49 timesteps (Full traj. w. 49 T.S.), and the full set of trajectories with 25 timesteps (Full traj. w. 25 T.S.). The results, including the averaged AUROC values and standard deviations over ten runs, are presented in Table 1. In the table, the notation "1 RC @ 20" refers to an RC network comprising a single RC cell with a reservoir size of 20 neurons, while "2 RCs @ 20" denotes an RC network consisting of two RC cells, each with a reservoir size of 20 neurons.

Based on the findings presented in the table, it is evident that incorporating multiple RC cells enhances the structural inference accuracy of RCSI when the total count of neurons in the RC remains constant. For instance, the configuration "3 RCs @ 20" consistently outperforms both "1 RC @ 60" and "2 RCs @ 30" across all three experiments, despite having an equivalent total number of parameters. However, our observations also reveal that increasing the reservoir size of each individual cell within the stacked RC setup has a marginal impact on RCSI's inference accuracy beyond a certain threshold. In the provided table, we identify the critical number of neurons per reservoir as 20. When the reservoir size exceeds this value (e.g., 30 or 40 neurons, as indicated in the table), the performance improvement becomes negligible. Moreover, such setups exhibit a larger number of training parameters in the overall RC network, ranging from 1.5 to 2 times more than that of "3 RCs @ 20", resulting in reduced computational efficiency. Considering the trade-off between inference accuracy and efficiency, we have chosen the configuration of "3 RCs @ 20" as our preferred RC network setup for RCSI. This choice strikes a balance between achieving high accuracy while maintaining computational efficiency. More experimental results, such as the detailed performance results of the methods, results on noisy data and training time, can be found in Appendix C.

6. Limitations

RCSI, like other structural inference methods, possesses certain limitations that are worth noting. Firstly, RCSI is applicable only to dynamical systems with a static underlying interacting structure. When the underlying structure varies over time, the simple setup of RC cells struggles to capture the changing structure effectively. Furthermore, the backbone VAE-based method has not been tested on dynamic graphs. However, we believe it is feasible to extend RCSI to dynamic graphs by incorporating additional hidden variables to capture the temporal changes in the structure. Secondly, the scalability of RCSI remains uncertain. As the backbone VAE-based method operates on a full-graph setting, the memory requirements for training scale at least quadratically with the total number of nodes ($O(n^2)$), where 'n' denotes the node count. To address this issue, it is necessary to either develop a memory-efficient backbone method or integrate the RC into the VAE framework. Nevertheless, these limitations present intriguing avenues for future research aimed at enhancing RCSI. By overcoming these challenges, we can broaden the application scenarios of RCSI, thereby benefiting researchers across various fields.

7. Conclusion and Future Work

This paper has introduced a novel structural inference approach with an integrated RC net. The integration is enabled by a Bi-level Optimization algorithm - BOME, which manages to optimize the backbone of the structural inference method and the RC net in a nested and efficient way. The experimental results verify the advantage of the RC-integrated structural inference method on datasets with fewer trajectories and datasets with shorter trajectories.

Currently, we only generally study structural inference with integrated RC for dynamical systems. Future research includes research on specific cases when the dynamical system is either Lyapunov stable, bifurcated, or damping, as these factors may affect the exact choice of the variables of Echo State Property. Furthermore, we will also study how to improve the robustness of the RC-integrated structural inference method against noises in node features.

Acknowledgment

Authors Tsz Pan Tong and Jun Pang acknowledge financial support from the Institute for Advanced Studies of the University of Luxembourg through an Audacity Grant (AUDACITY-2021).

References

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Alet, F., Weng, E., Lozano-Pérez, T., and Kaelbling, L. P. Neural relational inference with fast modular meta-learning. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, pp. 11804–11815, 2019.

Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Bianchi, F. M., Scardapane, S., Løkse, S., and Jenssen, R. Reservoir computing approaches for representation and classification of multivariate time series. *IEEE transactions on neural networks and learning systems*, 32(5): 2169–2179, 2020.

Brasó, G. and Leal-Taixé, L. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6247–6257, 2020.

Cabrera, D., Sancho, F., and Tobar, F. Combining reservoir computing and variational inference for efficient one-class learning on dynamical systems. In *2017 International Conference on Sensing, Diagnostics, Prognostics, and Control (SDPC)*, pp. 57–62. IEEE, 2017.

Canaday, D., Pomerance, A., and Girvan, M. A meta-learning approach to reservoir computing: Time series prediction with limited data. *arXiv preprint arXiv:2110.03722*, 2021.

Chen, P., Liu, R., Aihara, K., and Chen, L. Autoreservoir computing for multistep ahead prediction based on the spatiotemporal information transformation. *Nature Communications*, 11(1):1–15, 2020.

Chen, S., Wang, J., and Li, G. Neural relational inference with efficient message passing mechanisms. In *Proceedings of the 35th AAAI Conference on Artificial Intelligence (AAAI)*, pp. 7055–7063, 2021.

Dempe, S. *Bilevel Optimization: Theory, Algorithms, Applications and a Bibliography*, pp. 581–672. Springer International Publishing, Cham, 2020a.

Dempe, S. Bilevel optimization: theory, algorithms, applications and a bibliography. In *Bilevel optimization*, pp. 581–672. Springer, 2020b.

Dong, J., Rafayelyan, M., Krzakala, F., and Gigan, S. Optical reservoir computing using multiple light scattering for chaotic systems prediction. *IEEE Journal of Selected Topics in Quantum Electronics*, 26(1):1–12, 2019.

Dong, J., Ohana, R., Rafayelyan, M., and Krzakala, F. Reservoir computing meets recurrent kernels and structured transforms. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020.

Franceschi, L., Frasconi, P., Salzo, S., Grazzi, R., and Pontil, M. Bilevel programming for hyperparameter optimization and meta-learning. In *International Conference on Machine Learning (ICML)*, pp. 1568–1577. PMLR, 2018.

Gallicchio, C. and Micheli, A. Architectural and markovian factors of echo state networks. *Neural Networks*, 24(5): 440–456, 2011. ISSN 0893-6080.

Gallicchio, C. and Micheli, A. Echo state property of deep reservoir computing networks. *Cognitive Computation*, 9 (3):337–350, 2017.

Gallicchio, C., Micheli, A., and Pedrelli, L. Deep reservoir computing: A critical experimental analysis. *Neurocomputing*, 268:87–99, 2017.

Gauthier, D. J., Bollt, E., Griffith, A., and Barbosa, W. A. S. Next generation reservoir computing. *Nature Communications*, 12:5564, September 2021.

Gong, C., Liu, X., and Liu, Q. Automatic and harmless regularization with constrained and lexicographic optimization: A dynamic barrier approach. *Advances in Neural Information Processing Systems (NeurIPS)*, 34, 2021.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Graber, C. and Schwing, A. G. Dynamic neural relational inference for forecasting trajectories. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pp. 4383–4392, 2020.

Grazzi, R., Franceschi, L., Pontil, M., and Salzo, S. On the iteration complexity of hypergradient computation. In *International Conference on Machine Learning (ICML)*, pp. 3748–3758. PMLR, 2020.

Guo, Q., Feng, W., Zhou, C., Huang, R., Wan, L., and Wang, S. Learning dynamic siamese network for visual object tracking. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

Ha, S. and Jeong, H. Unraveling hidden interactions in complex systems with deep learning. *Scientific Reports*, 11(1):1–13, 2021.

Hong, M., Wai, H.-T., Wang, Z., and Yang, Z. A two-timescale framework for bilevel optimization: Complexity analysis and application to actor-critic. *arXiv preprint arXiv:2007.05170*, 2020.

Jaeger, H. 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(34): 13, 2001.

Jalalvand, A., De Neve, W., Van de Walle, R., and Martens, J.-P. Towards using reservoir computing networks for noise-robust image recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1666–1672, 2016.

Jalalvand, A., Demuynck, K., De Neve, W., and Martens, J.-P. On the application of reservoir computing networks for noisy image recognition. *Neurocomputing*, 277:237–248, 2018.

Javed, S., Danelljan, M., Khan, F. S., Khan, M. H., Felsberg, M., and Matas, J. Visual object tracking with discriminative filters and siamese networks: a survey and outlook. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Katok, A. and Hasselblatt, B. *Introduction to the Modern Theory of Dynamical Systems*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1995.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 2688–2697. PMLR, 2018.

Kwapień, J. and Drożdż, S. Physical approach to complex systems. *Physics Reports*, 515(3):115–226, 2012.

Li, J., Ma, H., Zhang, Z., Li, J., and Tomizuka, M. Spatio-temporal graph dual-attention network for multi-agent prediction and tracking. *arXiv preprint arXiv:2102.09117*, 2021.

Liu, B., Ye, M., Wright, S., Stone, P., and qiang liu. BOME! bilevel optimization made easy: A simple first-order approach. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022a.

Liu, R., Liu, X., Zeng, S., Zhang, J., and Zhang, Y. Value-function-based sequential minimization for bi-level optimization. *arXiv preprint arXiv:2110.04974*, 2021.

Liu, R., Gao, J., Zhang, J., Meng, D., and Lin, Z. Investigating bi-level optimization for learning and vision from a unified perspective: A survey and beyond. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (12):10045–10067, 2022b.

Löwe, S., Madras, D., Shilling, R. Z., and Welling, M. Amortized causal discovery: Learning to infer causal graphs from time-series data. In *Proceedings of the 1st Conference on Causal Learning and Reasoning (CLeaR)*, pp. 509–525. PMLR, 2022.

Maass, W., Natschläger, T., and Markram, H. Real-time computing without stable states: A new framework for neural computation based on perturbations. *Neural computation*, 14(11):2531–2560, 2002.

Mehra, A. and Hamm, J. Penalty method for inversion-free deep bilevel optimization. In *Asian Conference on Machine Learning (ACML)*, pp. 347–362. PMLR, 2021.

Nakajima, K. and Fischer, I. *Reservoir Computing*. Springer, 2021.

Outrata, J. V. On the numerical solution of a class of stackelberg problems. *Zeitschrift für Operations Research*, 34 (4):255–277, 1990.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems(NeurIPS)*, volume 32, pp. 8024–8035, 2019.

Pathak, J., Hunt, B., Girvan, M., Lu, Z., and Ott, E. Model-free prediction of large spatiotemporally chaotic systems from data: A reservoir computing approach. *Physical review letters*, 120(2):024102, 2018.

Pedregosa, F. Hyperparameter optimization with approximate gradient. In *International conference on machine learning (ICML)*, pp. 737–746. PMLR, 2016.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Pratapa, A., Jalihal, A. P., Law, J. N., Bharadwaj, A., and Murali, T. Benchmarking algorithms for gene regulatory network inference from single-cell transcriptomic data. *Nature Methods*, 17(2):147–154, 2020.

Saelens, W., Cannoodt, R., Todorov, H., and Saeys, Y. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, 2019.

Salimans, T., Karpathy, A., Chen, X., and Kingma, D. P. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017.

Schrauwen, B., Verstraeten, D., and Van Campenhout, J. An overview of reservoir computing: theory, applications and implementations. In *Proceedings of the 15th european symposium on artificial neural networks.*, pp. 471–482, 2007.

Shao, C. and Feng, Y. Overcoming catastrophic forgetting beyond continual learning: Balanced training for neural machine translation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL)*, pp. 2023–2036, Dublin, Ireland, May 2022. Association for Computational Linguistics.

Shwartz-Ziv, R. and Tishby, N. Opening the black box of deep neural networks via information. *arXiv preprint arXiv:1703.00810*, 2017.

Smith, S. M., Miller, K. L., Salimi-Khorshidi, G., Webster, M., Beckmann, C. F., Nichols, T. E., Ramsey, J. D., and Woolrich, M. W. Network modelling methods for FMRI. *Neuroimage*, 54(2):875–891, 2011.

Srinivasan, K., Coble, N., Hamlin, J., Antonsen, T., Ott, E., and Girvan, M. Parallel machine learning for forecasting the dynamics of complex networks. *Physical Review Letters*, 128(16):164101, 2022.

Stackelberg, H. v. et al. Theory of the market economy. 1952.

Suh, S., Chae, D. H., Kang, H.-G., and Choi, S. Echo-state conditional variational autoencoder for anomaly detection. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pp. 1015–1022. IEEE, 2016.

Tishby, N. and Zaslavsky, N. Deep learning and the information bottleneck principle. In *Proceedings of 2015 IEEE Information Theory Workshop (ITW)*, pp. 1–5. IEEE, 2015.

Tishby, N., Pereira, F., and Biale, W. The information bottleneck method. In *Proceedings of the 37th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 368–377. IEEE, 1999.

Tsubaki, M., Tomii, K., and Sese, J. Compound–protein interaction prediction with end-to-end learning of neural networks for graphs and sequences. *Bioinformatics*, 35 (2):309–318, 2019.

Van Den Oord, A., Vinyals, O., et al. Neural discrete representation learning. *Advances in Neural Information Processing Systems (NIPS)*, 30, 2017.

Vlachas, P. R., Pathak, J., Hunt, B. R., Sapsis, T. P., Girvan, M., Ott, E., and Koumoutsakos, P. Backpropagation algorithms and Reservoir Computing in Recurrent Neural Networks for the forecasting of complex spatiotemporal dynamics. *Neural Networks*, 126:191–217, 2020.

Wang, A. and Pang, J. Iterative structural inference of directed graphs. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, 2022.

Webb, E., Day, B., Andres-Terre, H., and Lió, P. Factorised neural relational inference for multi-interaction systems. *arXiv preprints arXiv:1905.08721*, 2019.

Wu, T., Breuel, T., Skuhersky, M., and Kautz, J. Discovering nonlinear relations with minimum predictive information regularization. *arXiv preprint arXiv:2001.01885*, 2020.

Wyffels, F. and Schrauwen, B. A comparative study of reservoir computing strategies for monthly time series prediction. *Neurocomputing*, 73(10-12):1958–1964, 2010.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *Proceedings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Yildiz, I. B., Jaeger, H., and Kiebel, S. J. Re-visiting the echo state property. *Neural Networks*, 35:1–9, 2012.

Zhang, Z. and Peng, H. Deeper and wider siamese networks for real-time visual tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

A. More Details about RCSI

The whole pipeline of RCSI may refer to Algorithm 2.

Algorithm 2 Pipeline of RCSI

```
**Goal**: Infer adjacency matrix $\mathbf{A}$ from observational node features $V = \{V^{(0)}, V^{(1)}, ..., V^{(T)}\}$.
**Input**: Node features $V = \{V^{(0)}, V^{(1)}, ..., V^{(T)}\}$; inner step $T'$; outer and inner step size $\xi$, $\tau$ (set $\tau = \xi$ by default);
weights in loss $\beta$, $\gamma$; inner rounds $T' = 10$; weight for BOME $\eta = 0.5$.
**Models**: VAE-based structural inference model $\mathcal{M}_{SI}$; RC net $\mathcal{M}_{RC}$.
**Loss**: Loss for VAE-based structural inference model $\mathcal{J}_{LL}$; Loss for RC net $\mathcal{J}_{UL}$.
**Model Parameters**: VAE-based structural inference model $\theta$; RC net $\psi$.
Split $V$ into $V^T = \{V^{(0)}, V^{(1)}, ..., V^{(T-1)}\}$ and $V^{T+1} = \{V^{(1)}, V^{(2)}, ..., V^{(T)}\}$
Initialize $\psi$ according to Sections 4.2 and A.1
**while** Epoch < MaxEpoch **do**
  **while** Inner Iteration $k < T'$ **do**
    Now the model parameters are $\{\psi_k, \theta_k\}$
    $\mathbf{A} \leftarrow$ Encoder($V^T$)
    $\hat{V}^{T+1} \leftarrow$ Decoder($V^T, \mathbf{A}$)
    Backpropagate $\mathcal{J}_{LL}$ to obtain updated parameters $\theta_k^{(T')}$
  **end while**
  Calculate $\hat{q}(\psi_k, \theta_k) = \mathcal{J}_{LL}(\psi, \theta) - \mathcal{J}_{LL}(\psi, \theta_k^{(T')})$
  Update $(\psi, \theta)$ : $(\psi_{k+1}, \theta_{k+1}) \leftarrow (\psi_k, \theta_k) - \xi(\nabla \mathcal{J}_{UL}(\psi_k, \theta_k) + \lambda_k \nabla \hat{q}(\psi_k, \theta_k))$
  where $\lambda_k = \max\left(\frac{\phi_k - \langle \nabla \mathcal{J}_{UL}(\psi_k, \theta_k),\ \nabla \hat{q}(\psi_k, \theta_k)\rangle}{||\hat{q}(\psi_k, \theta_k)||^2}, 0\right)$, and $\phi_k = \eta ||\hat{q}(\psi_k, \theta_k)||^2$
**end while**
**Return: A**
```

A.1. Initialization of RC

As mentioned in Section 4.1 and in various literature (Jalalvand et al., 2016; Dong et al., 2020; Vlachas et al., 2020; Gauthier et al., 2021), the initialization of any RC net is of great importance. Before the description on the initialization in this work, we would like to recall some basic properties and functions from literature that play an important role in the initialization of RC net. First are the two theorems from (Gallicchio et al., 2017):

Theorem A.1. *(Necessary Condition for the Echo State Property of DeepESN) Consider the case given input* $\mathbf{x}$*, the dynamics of a DeepESN with hidden state* $\mathbf{h}$ *are ruled by:*

$$\begin{aligned}\mathbf{h}^{(t+1)} =& F(\mathbf{h}^{(t)}, \mathbf{x}^{(t)}) \\ =& \Big(F_1(\mathbf{h}_1^{(t)}, \mathbf{x}^{(t)}), ..., F_N(\mathbf{h}_1^{(t)}, \mathbf{h}_N^{(t)}\mathbf{x}^{(t)})\Big),\end{aligned} \tag{16}$$

where $F = (F_1, ..., F_N)$ *is a composition of layer-wise applied state transition function,* $\mathbf{h}^{(t)} = (\mathbf{h}_1^{(t)}, ..., \mathbf{h}_N^{(t)})$ *is the global state of the whole RC net and is a composition of reservoir states of all the levels of hierarchy. And the DeepESN is implemented in terms of leaky integrator reservoir units, and assumes that the null sequence is an admissible input for the system. Then a necessary condition for the ESP to hold is provided by the following equation:*

$$\max_{i=1,...,N} \rho_i = \max_{i=1,...,N} \rho((1-\alpha_i)\mathbf{I} + \alpha_i \mathbf{W}_{r,i}) < 1, \tag{17}$$

where $\rho(\cdot)$ *is the spectral radius operator (i.e. the maximum absolute eigenvalue of its matrix argument),* $\mathbf{I}$ *is the identity matrix, and* $\alpha_i \in [0, 1]$ *is the leaking rate of level i.*

Theorem A.2. *(Sufficient Condition for the Echo State Property of DeepESN) Consider a DeepESN whose dynamics are ruled by Equation 16, implemented in terms of leaky integrator reservoir units, and with tanh non-linearity as the activation function. If the DeepESN is featured by globally contractive dynamics then it satisfies the ESP. Accordingly, a sufficient condition for the ESP to hold is given by the following equation:*

$$\max_{i=1,...,N} C_i < 1, \tag{18}$$

where C_i denotes the Lipschitz constant of the state transition function F_i of the ith reservoir level, and it is computed as:

$$C_i = \begin{cases} (1-\alpha_1) + \alpha_1 ||\mathbf{W}_{r,1}|| & \text{if } i = 1 \\ (1-\alpha_i) + \alpha_i (C_{i-1}||\mathbf{W}_{n,i}|| + ||\mathbf{W}_{r,1}||) & \text{if } i > 1, \end{cases} \tag{19}$$

where $||\cdot||$ is the matrix norm induced by the L_2-norm defined on the corresponding state spaces.

A simple approach to initialize the reservoir weights in DeepESN is then to randomly draw the elements in $\mathbf{W}_{n,i}$ and in $\mathbf{W}_{r,i}$, e.g., from a uniform distribution in $[-1, 1]$, and then rescale them in order to meet one of the conditions expressed by Theorems A.1 and A.2. The initialization of the weights in the RC cells in this work follows this simple approach with a check on the spectral radius, which is faster to compute.

A.2. Bi-level Optimization

The BO in this work follows BOME (Liu et al., 2022a), and this section provides more details about the implemented BOME in this work. We follow the same notions used in Section 3.4. We consider a value function, which yields natural first-order algorithms for non-convex lower-level objective and requires no computation of Hessian matrices (Liu et al., 2022a). And thus Equation 15 is equivalent to the following constrained optimization

$$\min_{\psi,\theta} \mathcal{J}_{UL}(\psi,\theta) \ \text{ s.t. } q(\psi,\theta) := \mathcal{J}_{LL}(\psi,\theta) - \mathcal{J}^*_{LL}(\psi) \leq 0, \tag{20}$$

where $\mathcal{J}^*_{LL}(\psi) = \min_\theta \mathcal{J}_{LL}(\psi,\theta) = \mathcal{J}_{LL}(\psi,\theta^*(\psi))$ is known as the value function, and $\theta^*(\psi) = \arg\min_\theta \mathcal{J}_{LL}(\psi,\theta)$ is the optimum. Compared with hypergradient approach, this formulation does not require the calculation of the implicit derivative $\nabla_\psi \theta^*(\psi)$ (Liu et al., 2022a). Although $\mathcal{J}_{LL}(\psi)$ depends on $\theta^*(\psi)$, its derivative $\nabla_\psi \mathcal{J}^*_{LL}(\psi)$ does not depend on $\nabla_\psi \theta^*(\psi)$ by Danskin's theorem:

$$\nabla_\psi \mathcal{J}^*_{LL}(\psi) = \nabla_1 \mathcal{J}_{LL}(\psi,\theta^*(\psi)) + \nabla_\psi \theta^*(\psi) \nabla_2 \mathcal{J}_{LL}(\psi,\theta^*(\psi)) = \nabla_1 \mathcal{J}_{LL}(\psi,\theta^*(\psi)), \tag{21}$$

while the second term vanishes because the definition of the optimum $\theta^*(\psi)$: $\nabla_2 \mathcal{J}_{LL}(\psi,\theta^*(\psi)) = 0$. Therefore, provided that we can evaluate $\theta^*(\psi)$ at each iteration, solving Equation 20 yields an algorithm for BO that requires no Hessian computation. BOME uses the dynamic barrier gradient descent (Gong et al., 2021), which is an elementary first-order algorithm for solving constrained optimization, but it applies only to a special case of the bi-level problem. Therefore, the authors of BOME extend it to handle the general case. The original idea of dynamic barrier gradient descent is to iteratively update the parameter (ψ,θ) to reduce $\mathcal{J}_{UL}$ while controlling the decrease of the constraint q, ensuring that q decreases whenever $q > 0$. Specifically, denote ξ as the step size, the update at each step is:

$$(\psi_{k+1},\theta_{k+1}) \leftarrow (\psi_k,\theta_k) - \xi\delta_k, \tag{22}$$

$$\text{where } \delta_k = \arg\min_\delta ||\nabla \mathcal{J}_{UL}(\psi_k,\theta_k)||^2 \ \text{ s.t. } \langle \nabla q(\psi_k,\theta_k), \delta \rangle \geq \phi_k. \tag{23}$$

Here $\nabla \mathcal{J}_{UL} = \nabla_{(\psi_k,\theta_k)} \mathcal{J}_{UL}(\psi_k,\theta_k)$, $\nabla q = \nabla_{(\psi_k,\theta_k)} q(\psi_k,\theta_k)$, $\langle\cdot,\cdot\rangle$ is the inner-production, and $\phi_k \geq 0$ is a non-negative control barrier.

Two choices of ϕ_k that satisfy the condition above are $\phi_k = \eta q(\psi_k,\theta_k)$ and $\phi_k = \eta||\nabla q(\psi_k,\theta_k)||^2$ with $\eta > 0$. In this work, we follow the second choice and use $\phi_k = \eta||\nabla q(\psi_k,\theta_k)||^2$ as default. So that the optimization of Equation 23 yields a closed form solution:

$$\delta_k = \nabla \mathcal{J}_{UL}(\psi_k,\theta_k) + \lambda_k \nabla q(\psi_k,\theta_k), \ \text{with } \lambda_k = \max\Big(\frac{\phi_k - \langle \nabla \mathcal{J}_{UL}(\psi_k,\theta_k), \ \nabla q(\psi_k,\theta_k)\rangle}{||q(\psi_k,\theta_k)||^2}, 0\Big), \tag{24}$$

and $\lambda_k = 0$ in case of $||\nabla q(\psi_k,\theta_k)|| = 0$.

Yet the method above requires the calculation of $q(\psi_k,\theta_k)$ and $\nabla q(\psi_k,\theta_k)$ which require evaluation of $\theta^*(\psi_k)$. We also follow the approximation in BOME which approximates $\theta^*(\psi_k)$ by $\theta_k^{(T')}$, where $\theta_k^{(T')}$ is obtained by running T' steps of gradient steps of $\mathcal{J}_{LL}(\psi_k,\cdot)$ w.r.t. θ starting from θ_k. That is, we set $\theta^{(0)} = \theta_k$ and let

$$\theta_k^{(t+1)} = \theta_k^{(t)} - e\nabla_\theta \mathcal{J}_{LL}(\psi_k, \theta_k^{(t}), \ \ t = 0, ..., T' - 1, \tag{25}$$

for some step size parameter $e > 0$. We therefore obtain an estimate of $q(\psi,\theta)$ at iteration k by replacing $\theta^*(\psi_k)$ with

$\theta_k^{(T')}$: $\hat{q}(\psi,\theta) = \mathcal{J}_{LL}(\psi,\theta) - \mathcal{J}_{LL}(\psi,\theta^*(\psi_k))$. We substitute $\hat{q}(\psi,\theta)$ into Equation 23 to obtain the update direction δ_k, and the full algorithm is summarized in Algorithm 1.

A.3. Attraction Points

The notion of attraction points is introduced and proved in the work of BOME (Liu et al., 2022a):

Definition A.3. (Attraction Points) Given any (ψ,θ), we say that $\theta^{\diamondsuit}(\psi,\theta)$ is the attraction point of (ψ,θ) with step size $e > 0$ if the sequence $\{\theta^{(t)}\}_{t=0}^{\infty}$ generated by gradient descent $\theta^{(t)} = \theta^{(t-1)} - e\nabla_{\theta}\mathcal{J}_{LL}(\psi,\theta^{(t-1)})$ starting from $\theta^{(0)} = \theta$ converges to $\theta^{\diamondsuit}(\psi,\theta)$.

Intuitively, the attraction of (ψ,θ) is where the gradient descent algorithm cannot make any improvement. The set of (ψ,θ) that has the same attraction point forms an attraction basin. But there is a key challenge that $\theta^{\diamondsuit}(\psi,\theta)$ and hence $q^{\diamondsuit}(\psi,\theta) = \mathcal{J}_{LL}(\psi,\theta) - \mathcal{J}_{LL}(\psi,\theta^{\diamondsuit}(\psi,\theta))$ can be discontinuous w.r.t. θ when it is on the boundary of different attraction basins. However, these boundary points are not stable stationary points and it is possible to use arguments based on the stable manifold theorem to show that an algorithm with random initialization will almost surely not visit them (Liu et al., 2022a). This statement ensures the stability of BOME with the approximation terms.

B. Further Implementation Details

B.1. Basic Settings

We implement RCSI in Pytorch (Paszke et al., 2019) with the help of Scikit-Learn (Pedregosa et al., 2011) to calculate metrics. The experiments are run on one NVIDIA Tesla V100 SXM2 32G graphic card, with two Xeon Gold 6132 @ 2.6GHz CPUs. We set the maximum epoch as 1000, and set batch size as 128 for datasets that have no more than 10 nodes, and 64 for datasets having more than 10 nodes. We use Adam optimizer (Kingma & Ba, 2015) for the training of both branches with the learning rate of $5e-4$, and we reduce the learning rate to 50% if there is no loss drop in the past 100 epochs.

B.2. Further details of datasets

Synthetic networks. The six directed Boolean networks (LI, LL, CY, BF, TF, BF-CV) are the most often observed fragments in many gene regulatory networks, each has 7, 18, 6, 7, 8 and 10 nodes, respectively. Thus by carrying out experiments on these networks, we can get acknowledge on the performance of the chosen methods on the structural inference of real-world biological networks. We collect the six ground-truth directed Boolean networks from (Pratapa et al., 2020) and simulate the single-cell evolving trajectories with BoolODE (Pratapa et al., 2020) `https://github.com/Murali-group/BoolODE` with default settings mentioned in that paper for every network. We first sample a total number of 12000 raw trajectories. We then sample different numbers of trajectories from raw trajectories and randomly group them into three datasets: for training, for validation, and for testing, with a ratio of $8:2:2$. After that we sample different numbers of snapshots according to the requirements of experiments in Sections 5.2 and 5.3 with equal time intervals in every trajectory and save them as ".npy" files for data loading.

NetSim datasets. The NetSim datasets simulate blood-oxygen-level-dependent imaging data across different regions within the human brain and is described in (Smith et al., 2011) and `https://www.fmrib.ox.ac.uk/datasets/netsim/`. We target inferring the existence of directed connections between different brain areas. Among the total 28 datasets in NetSim, we choose the first three datasets (NetSim1, NetSim2 and NetSim3) which have 5, 10, and 15 nodes, respectively. We sample different numbers of snapshots according to the requirements of experiments in Sections 5.2 and 5.3 on each trajectory with equal intervals and randomly group them into three sets for training, validation and testing with the ratio of 8: 2: 2, respectively.

Physical simulations. To generate these physical simulations (springs, charged particles and phase-coupled oscillators), we follow the description of the data in (Kipf et al., 2018) but with fixed interactions. To be specific, at the beginning of the data generation for each physical simulation, we randomly generate a ground truth graph and then simulate 12000 raw trajectories on the same ground truth graph, but with different initial conditions. We then sample different numbers of trajectories from raw trajectories and randomly group them into three datasets: for training, for validation, and for testing, with a ratio of $8:2:2$. After that we sample different numbers of snapshots according to the requirements of experiments in Sections 5.2 and 5.3 with equal time intervals in every trajectory. The rest settings for the simulations are the same as those mentioned in (Kipf et al., 2018). It is worth mentioning that the connections in the physical simulations are indirected, which are different from those in the synthetic networks and NetSim datasets. We collect the trajectories and randomly group them into three sets for training, validation and testing with the ratio of 8: 2: 2, respectively.

Datasets with Noise. We generate a dataset with different levels of noise based on springs dataset of physical simulations. We first sample trajectories the way same as those for physical simulations. After that, we add Gaussian noise with different levels Δ on the obtained node features $v_i^{(t)}$:

$$\hat{v}_i^{(t)} = v_i^{(t)} + \kappa \cdot 0.02 \cdot \Delta, \text{ where } \kappa \sim \mathcal{N}(0, 1). \tag{26}$$

B.3. Implementation details of RCSI

For details about the implementation please refer to the link attached in the supplementary material. The choice of hyperparameters in this work follows the choices in BOME and iSIDG. For example, we set $T' = 10$ and $\eta = 0.5$ for BOME in this work, and set the weights in the loss functions identical as iSIDG (Wang & Pang, 2022). We briefly describe the implementation of the main blocks in RCSI in the following paragraphs.

Backbone VAE-based Structural Inference Method. The backbone VAE-based structural inference method utilized in this work is iSIDG (Wang & Pang, 2022). We follow the setup and implementation described in the work of iSIDG. We also would like to thank the authors of iSIDG for the code. The hyperparameters and the implementation of loss functions of the VAE-based structural inference method in RCSI are identical to those in iSIDG.

RC Net. The RC net in this work consists of three identical RC cells and a final linear output layer. The setup of RC cells in this work follows the " ReservoirCell" class from `https://github.com/Pervasive-AI-Lab/ContinualLearning-EchoStateNetworks`. The initialization of the RC cells are identical to the default used in the class, which regularizes the spectral radius of weight matrices, and is implemented according to Theorem A.1. The dimensionality of the RC cells is set to match the output from the diverging layer in the decoder of iSIDG. For example, "input_size" is set as 256, while the rest are set the same as default in the class. The final linear output layer has the size to match the concatenated output from three RC cells, and outputs the node features to match the settings of the datasets (4 for physical simulations, and 1 for the rest). It is worth mentioning that in the training process of RC nets for feature prediction, warm-up sessions are necessary, to ensure that the reservoir state does not depend on the RC initial conditions (Gauthier et al., 2021). However, as the task of RCSI is to infer the underlying structure of a dynamical system, the initial dependence plays minor role here. As a result, RCSI does not necessarily need a warm-up period.

BOME. We would like to thank the authors of BOME (Liu et al., 2022a) for the implementation details of the algorithm. The implementation of BOME in this work follows the general implementation pipeline of BOME, and simply replaces the learning objectives $f(\cdot)$ and $g(\cdot)$ with the iSIDG objective $\mathcal{J}_{UL}$ and the RC net objective $\mathcal{J}_{LL}$.

B.4. Implementation details of baselines

NRI. We use the official implementation code by the author from `https://github.com/ethanfetaya/NRI` with a customized data loader for our chosen datasets. We add our metric evaluation in the "test" function, after the calculation of accuracy in the original code.

fNRI. We use the official implementation code by the author from `https://github.com/ekwebb/fNRI` with a customized data loader for our chosen datasets. We add our metric evaluation in the "test" function, after the calculation of accuracy and the selection of the correct order for the representations in latent spaces in the original code.

MPIR. We follow the official implementation from `https://github.com/tailintalent/causal` as the model for MPIR. We run the model with a customized data loader for the chosen datasets. After the obtain of the results, we run another script to calculate the metrics.

MPM. We use the official implementation code by the author from `https://github.com/hilbert9221/NRI-MPM` with a customized data loader for our chosen datasets. We add our metric evaluation for AUROC in the "evaluate()" function of class "XNRIDECIns" in the original code.

ACD. We follow the official implementation code by the author as the framework for ACD (`https://github.com/loeweX/AmortizedCausalDiscovery`). We run the code with a customized data loader for the datasets in this work. We implement the metric-calculation pipeline in the "forward_pass_and_eval()" function.

iSIDG. We ask the authors of iSIDG for the code and follow the instructions on the settings of hyper-parameters in their work. We disable the metric evaluations for AUPRC and Jaccard index in the original implementation of iSIDG for faster computation.

C. Further Experimental Results

Because of the page limitation, more experimental results are reported in this section.

C.1. Exact results on datasets of only 10% trajectories

The experimental results of the proposed model and the baseline methods on synthetic networks, three NetSim datasets and physical simulations are summarized in Tables 2 and 3, which consists of mean and standard deviation of AUROC values from 10 experiments on each. The numbers of trajectories in this section are various. For synthetic networks and physical simulations, the numbers of trajectories are 1200 in total, respectively, but for NetSim datasets, we keep the number of trajectories as identical to the default, as the trajectories are already few.

As shown in Tables 2 and 3, RCSI outperforms all baseline methods on all 12 datasets. It is worth mentioning that the amount of available data in the experiments on synthetic networks and physical simulations is only 10% to that investigated in previous works (Kipf et al., 2018; Webb et al., 2019; Wang & Pang, 2022). So in comparison with the results reported in previous works (Kipf et al., 2018; Wang & Pang, 2022), baseline methods perform much worse. In sharp contrast, RCSI manages to infer the adjacency matrix of the graph with the highest AUROC value among all of the investigated methods, with a margin of up to 14.3%, which supports that the integration of RC into the framework of structural inference can help the method to overcome the curse of the amount of data.

As the feature dimension of the nodes varies from one to four in the datasets, the AUROC values of RCSI suggest that the feature dimension has minor influence on the performance. Recall that iSIDG (Wang & Pang, 2022) is slightly inferior to other methods when node features are rich, the integration of RC succeeds in helping iSIDG to overcome this challenge.

Table 2. AUROC values (%) on synthetic networks and physical simulations with a total of 1200 trajectories and 49 time steps.

Methods	Datasets								
	LI	LL	CY	BF	TF	BF-CV	Springs	Particles	Kuramoto
NRI	56.4± 0.12	62.9± 0.12	53.7± 0.10	51.6± 0.16	51.0± 0.10	49.8± 0.14	55.1± 0.06	52.6± 0.05	53.0± 0.04
fNRI	60.3± 0.05	68.2± 0.05	57.1± 0.06	54.3± 0.05	54.1± 0.06	52.6± 0.08	64.2± 0.04	55.1± 0.04	53.2± 0.04
MPIR	42.0± 0.06	45.2± 0.04	30.4± 0.09	44.1± 0.06	40.0± 0.06	47.3± 0.05	50.1± 0.04	51.7± 0.04	50.1± 0.03
MPM	60.8± 0.03	71.0± 0.04	58.7± 0.05	57.3± 0.05	57.0± 0.06	54.3± 0.05	70.0± 0.05	58.7± 0.05	56.4± 0.04
ACD	61.2± 0.03	63.0± 0.04	58.3± 0.07	56.8± 0.07	55.8± 0.05	54.1± 0.07	69.9± 0.04	59.0± 0.05	58.3± 0.04
iSIDG	65.1± 0.04	75.9± 0.05	60.9± 0.05	60.7± 0.06	57.8± 0.05	60.2± 0.05	71.0± 0.04	59.0± 0.04	58.0± 0.04
RCSI	**73.9**± **0.06**	**80.2**± **0.04**	**71.0**± **0.05**	**63.4**± **0.04**	**62.7**± **0.05**	**66.1**± **0.06**	**77.6**± **0.04**	**68.1**± **0.05**	**65.7**± **0.04**

C.2. Results on datasets of 25 time steps

In this section, we would like to study the performance of all of the methods when the trajectories consist of 25 time steps, which are shorter than those in Section 5.2. The number of trajectories in this experiment is 1200 in total for physical simulations and synthetic networks, and identical for NetSim datasets. From the sampled trajectories that are used in Section 5.2, we subsample the first 25 time steps in each trajectory to create the datasets used in this study, and we show AUROC results in Tables 4 and 5. The results in these tables consist of averaged AUROC values and standard deviations from 10 runs. As expected, all of the methods perform worse than experiments on datasets of 49 time steps, while RCSI still unsurprisingly outperforms all of the baselines. Since the available data is cut to almost half, data-driven structural inference methods are facing bigger challenges. The experimental results in this section support the performance of RCSI for the task of structural inference on fewer and shorter trajectories.

Table 3. AUROC values (%) on NetSim datasets. The trajectories in these datasets consist of 49 time steps.

Methods	Datasets		
	NetSim1	NetSim2	NetSim3
NRI	72.1 ± 0.03	72.0 ± 0.04	68.3 ± 0.03
fNRI	72.4 ± 0.05	71.2 ± 0.05	69.6 ± 0.04
MPIR	47.2 ± 0.03	46.0 ± 0.04	44.3 ± 0.02
MPM	73.2 ± 0.03	72.0 ± 0.03	70.4 ± 0.03
ACD	66.7 ± 0.04	64.0 ± 0.03	62.9 ± 0.03
iSIDG	75.6 ± 0.05	72.2 ± 0.05	71.5 ± 0.04
RCSI	**77.0** ± **0.05**	**75.6** ± **0.05**	**74.4** ± **0.04**

Table 4. AUROC values (%) on synthetic networks and physical simulations with 25 time steps.

Methods	Datasets								
	LI	LL	CY	BF	TF	BF-CV	Springs	Particles	Kuramoto
NRI	52.7± 0.10	59.5± 0.10	53.2± 0.10	49.3± 0.08	50.1± 0.05	49.2± 0.09	49.0± 0.05	50.2± 0.04	49.5± 0.04
fNRI	55.5± 0.05	61.7± 0.04	56.4± 0.05	52.0± 0.05	51.7± 0.05	51.0± 0.07	55.7± 0.05	52.2± 0.04	50.2± 0.04
MPIR	40.7± 0.06	42.8± 0.04	30.1± 0.09	42.7± 0.06	39.4± 0.06	45.7± 0.05	43.0± 0.06	48.8± 0.05	48.1± 0.03
MPM	56.7± 0.04	62.3± 0.04	56.6± 0.05	54.5± 0.05	52.8± 0.05	52.6± 0.04	61.4± 0.05	55.4± 0.05	52.1± 0.04
ACD	56.3± 0.03	62.0± 0.04	57.1± 0.04	54.9± 0.05	53.0± 0.05	52.5± 0.05	60.9± 0.05	55.8± 0.05	53.6± 0.04
iSIDG	59.3± 0.04	66.5± 0.04	59.1± 0.04	57.6± 0.05	55.0± 0.05	56.3± 0.05	62.0± 0.05	56.1± 0.04	53.9± 0.04
RCSI	**69.1**± **0.05**	**75.4**± **0.05**	**68.5**± **0.04**	**60.2**± **0.04**	**60.4**± **0.04**	**63.9**± **0.05**	**75.2**± **0.06**	**64.9**± **0.04**	**63.2**± **0.05**

Table 5. AUROC values (%) on NetSim datasets. The trajectories in these datasets consist of 25 time steps.

Methods	Datasets		
	NetSim1	NetSim2	NetSim3
NRI	63.4 ± 0.05	61.2 ± 0.05	60.0 ± 0.04
fNRI	64.0 ± 0.04	62.3 ± 0.04	61.5 ± 0.04
MPIR	45.6 ± 0.03	44.4 ± 0.04	43.7 ± 0.03
MPM	64.5 ± 0.04	63.1 ± 0.04	61.8 ± 0.03
ACD	65.0 ± 0.04	63.3 ± 0.04	62.1 ± 0.03
iSIDG	68.9 ± 0.05	66.4 ± 0.05	64.8 ± 0.03
RCSI	**74.4** ± **0.04**	**73.0** ± **0.05**	**71.9** ± **0.05**

C.3. Results on Noisy Datasets

Since RC is claimed to be robust against noise (Gauthier et al., 2021), and BO can also deal with noisy data (Liu et al., 2022a), we would like to study the robustness of the proposed RC structural inference method. We generate a series of datasets with noises based on the springs dataset, with different levels of Gaussian noise. The Gaussian noise is added to the node features and the level Δ amplifies the noise as: $\tilde{v}_i^{(t)} = v_i^{(t)} + \kappa \cdot 0.02 \cdot \Delta$, where $\kappa \sim \mathcal{N}(0, 1)$. We sample the trajectories with the same amount and same time steps as in Section C.2, and we report the results in Table 6. As shown in the table, RCSI still outperforms all baseline methods on the experiments of noisy data. However, averaged AUROC values of all methods experience a decrease, and the decrease is approximately linearly correlated with the levels of the Gaussian noise Δ. Besides that, the margins between the AUROC results of RCSI and other methods decrease as the level of noise increases. As a result, we can reach the conclusion that the robustness of RC integrated structural inference method is higher than vanilla structural inference methods, but the degree of robustness decreases as the level of noise increases. We think this phenomenon may come from the naive design of the RC net in this work. The RC net is designed as the combination of several RC cells to reduce the number of parameters and the number of neurons in the whole RC net, in order to fit the case of fewer data, and also reduce the difficulty of initialization. But compared to the RC used in (Vlachas et al., 2020) and (Srinivasan et al., 2022), the RC setup in this work is much easier and much simpler. Despite the simple setup, the integration of RC can still improve the performance of structural inference methods and infer the adjacency matrices more accurately. We leave the work on the study of increasing the robustness with an optimal design of the RC net for future work.

Table 6. AUROC results (%) on springs dataset with noise.

Methods	Noise Levels				
	$\Delta = 1$	$\Delta = 2$	$\Delta = 3$	$\Delta = 4$	$\Delta = 5$
NRI	49.0	50.2	49.5	50.2	49.5
fNRI	55.7	52.2	50.2	52.2	50.2
MPIR	43.0	48.8	48.1	48.8	48.1
MPM	61.4	55.4	52.1	55.4	52.1
ACD	60.9	55.8	53.6	55.8	53.6
iSIDG	62.0	56.1	53.9	56.1	53.9
RCSI	**75.2**	**64.9**	**63.2**	**64.9**	**63.2**

C.4. Results on Time Efficiency

Table 7. Training time (in hour) of RCSI and baseline methods on datasets with 100% trajectories and with 49 time steps.

Methods	Datasets											
	LI	LL	CY	BF	TF	BF-CV	Springs	Particles	Kuramoto	NetSim1	NetSim2	NetSim3
iSIDG	48.1	50.6	40.9	44.6	40.3	44.0	42.2	36.0	39.2	20.7	36.9	50.8
RCSI	48.4	51.0	41.5	44.8	41.0	44.7	42.9	36.5	39.8	21.2	37.3	51.3

We summarize the training time of RCSI and iSIDG on all of the datasets with 100% trajectories and with 49 time steps in Table 7. The results are the averaged values of 10 runs. Testing the running time on full trajectories and with full time steps is the most extreme case which has the most amount of data and therefore requires the longest training time among all of the experiments. There are some differences between the reported training time of iSIDG in Table 7 and those in its original work (Wang & Pang, 2022), but the differences are minor. As shown in the table, RCSI is only slightly slower than its vanilla backbone method iSIDG, and the margins are negligible. And the margins are not affected by whether the graph is directed or not. The choice of RC net with very few parameters and the BOME with efficient first-order gradient descent ensures that the addition of RC branch won't be a burden to the training of the whole framework.

C.5. Results on Fewer and Shorter Trajectories

Table 8. Performance comparison between RCSI and iSIDG. RCSI works on LI dataset but with different percentages of trajectories and different lengths of trajectories, while iSIDG is tested on LI dataset with 49 time steps and 100% trajectories. Mark "+" represents RCSI on current dataset is better than iSIDG on full dataset, while "-" is on the contrary.

Number of Time Steps	Percentage of the Available Trajectories									
	10	20	30	40	50	60	70	80	90	100
5	-	-	-	-	-	-	-	-	-	-
10	-	-	-	-	-	-	-	-	-	+
15	-	-	-	-	-	-	-	-	+	+
20	-	-	-	-	-	-	-	-	+	+
25	-	-	-	-	-	-	-	+	+	+
30	-	-	-	-	-	-	-	+	+	+
35	-	-	-	-	-	-	+	+	+	+
40	-	-	-	-	-	-	+	+	+	+
49	-	-	-	-	-	-	+	+	+	+

Since RCSI performs the best among all of the investigated methods with fewer trajectories and shorter trajectories, we would like to figure out how much data RCSI requires to match the performance of vanilla iSIDG on 100% trajectories and 49 time steps. We summarize the performance comparison in terms of average AUROC values of ten runs of both methods on the LI dataset, and show the results in Table 8. The average AUROC result of iSIDG on LI dataset with full trajectories is 86.2% (Wang & Pang, 2022). As shown by the "+" marks in the table, RCSI outperforms iSIDG with fewer trajectories and shorter trajectories. For example, RCSI can infer the structure more accurately than iSIDG with 70% trajectories and 30 time steps, as well as with 90% trajectories and 15 time steps. It is also shown in the table that the number of trajectories and the lengths of trajectories have almost the same affect on the performance of RCSI, as the frontier boundary is almost linear. The results shown in Table 8 strongly support the outstanding performance of RCSI on fewer and shorter trajectories.

C.6. Discussion on Further Ablation Study

In addition to the ablation study in Section 5.4, we discuss other ablation studies in this section. As mentioned in Section B.3, the choice of hyperparameters in this work simply follows the work of BOME (Liu et al., 2022a) and iSIDG (Wang & Pang, 2022), and RCSI manages to outperform the baseline methods with such hyperparameter settings. Besides that, there is no additional hyperparameter, which further suggests that RCSI is an easy-to-use structural inference method. As for the ablation studies on the important components of RCSI, we aim to either run RCSI without BOME, or simply just run a vanilla RC net for structural inference. The BOME is of great importance for combining the training processes of the backbone VAE-based method and the RC net. We ran several training processes of RCSI without BOME, and the results are terrible. At every epoch, the training of the backbone VAE-based method worked properly, but then the training of RC net pulled the progress backward, which was quite similar to the catastrophic forgetting in continual learning (Shao & Feng, 2022). This phenomenon supported the necessity of a BO to combine the training of both branches. We also tested a variation of RCSI which only consisted of a vanilla RC net. Although we tried our best to find a certain layer or reservoir that can be utilized as the generating space for the adjacency matrix, but we failed. Because the training of RC does not strictly follow the principle of IB, thus it was ambitious to find a certain layer that has the minimal statistic and can be viewed as the adjacency matrix. But we think that it would be interesting if we can dig deeper into the mechanism of RC net or hierarchical ESN, and find a stable way to use it as a generative model. We leave this for future work.

D. Broader Impact

Similar to NRI, fNRI, iSIDG, and other structural inference methods, RCSI allows for numerous researchers in the field of physics, chemistry, sociology, and biology to study the underlying interacting structure of the dynamical systems. We have shown that RCSI is effective and efficient in the case of fewer and shorter available trajectories, and is insensitive to the dimensionality of the node features, which proves its broad application. While the emergence of structural inference technology may be extremely helpful for many, potentially it can be misused. For example, it can be likely to be used to detect the intimate relationship between users on a social network, which could erode privacy.