Active Learning based Structural Inference

Aoran Wang [1] **Jun Pang** [1 2]

Abstract

In this paper, we propose a novel framework *Active Learning based Structural Inference* (ALaSI), to infer the existence of directed connections from observed agents' states over a time period in a dynamical system. With the help of deep active learning, ALaSI is competent in learning the representation of connections with a relatively small pool of prior knowledge. Moreover, based on information theory, the proposed inter- and out-of-scope message learning pipelines are remarkably beneficial to structural inference for large dynamical systems. We evaluate ALaSI on various large datasets including simulated systems and real-world networks, to demonstrate that ALaSI is able to outperform previous methods in precisely inferring the existence of connections in large systems under either supervised learning or unsupervised learning.

1 Introduction

Dynamical systems are commonly observed in real-world, including physical systems (Kwapień & Drożdż, 2012; Ha & Jeong, 2021), biological systems (Tsubaki et al., 2019; Pratapa et al., 2020), and multi-agent systems (Brasó & Leal-Taixé, 2020; Li et al., 2022). A dynamical system can be described as a set of three core elements: (a) the state of the system in a time period, including the state of the individual agents, and can be viewed as time series; (b) the state-space of the system; and (c) the state-transition function (Irwin & Wang, 2017). Knowing these core elements, we can describe and predict how a dynamical system behaves. Yet the three elements are not independent of each other. The evolution of the state is affected by the state-transition function, which suggests that the future state may be predicted based on the current state and the entities which affect the agents (i.e. *connectivity*). Moreover, the state-transition function is often deterministic (Katok & Hasselblatt, 1995), which simplifies the derivation of the future state as a Markovian transition function.

However, in most cases, we hardly have access to the connectivity, or only have limited knowledge about the connectivity. Is it possible to infer the connectivity from the observed states of the agents over a time period? We formulate it as the problem of *structural inference*, and several machine learning frameworks have been proposed to address it (Kipf et al., 2018; Webb et al., 2019; Alet et al., 2019; Chen et al., 2021; Löwe et al., 2022; Wang & Pang, 2022). Although these frameworks can accurately infer the connectivity, as they perform representation learning on a fully connected graph, these methods can only work for small systems (up to dozens of agents), and cannot scale well to real-world large dynamical systems, for example, with hundreds of agents. Besides, as we show in the experiment and appendix sections in this work, the integration of prior knowledge about partial connectivity of the system is quite problematic among these methods.

In this work, we propose a novel structural inference framework, namely, **A**ctive **L**earning b**a**sed **S**tructural **I**nference (ALaSI), which is designed for the structural inference of large dynamical systems based on Deep Active Learning (DeepAL) (Ren et al., 2022), and is suitable for the integration of prior knowledge. In order to perform structural inference on large dynamical systems, unlike ordinal deep active learning methods that build feature pools on batches (Kirsch et al., 2019; Zhdanov, 2019; Ash et al., 2020; Gentile et al., 2022), the pools of ALaSI are built on agents, and the framework can consequently infer the existence of directed connections with a little prior knowledge of the connections. ALaSI leverages query strategy with dynamics for agent-wise selection to update the pool with the most informative partial system, which encourages ALaSI to infer the connections efficiently and accurately with partial prior knowledge of the connectivity (named 'scope'). Based on information theory, ALaSI learns both inter-scope (IS) and out-of-scope (OOS) messages from the current scope to distinguish the information which represents connections from agents within the scope and from agents out of the scope, which reserves redundancy when new agents come into scope. Moreover, with oracle such as

[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>, Jun Pang <jun.pang@uni.lu>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023.

Partial Information Decomposition (PID) (Williams & Beer, 2010), ALaSI can infer the connectivity even without prior knowledge and be trained in an unsupervised way. We show with extensive experiments that ALaSI can infer the directed connections of dynamical systems with up to 1.5K agents with either supervised learning or unsupervised learning. The main contribution of this paper is the following:

- We propose a novel structural inference algorithm, ALaSI, tailored to infer the connection of large dynamical systems based on DeepAL. It is the first attempt to structural inference with DeepAL to the best of our knowledge.
- We design a novel dynamic query strategy, which queries the most informative agents to be labeled based on the dynamic error, and enables ALaSI to learn efficiently on prior knowledge of the partial dynamical system.
- Based on information theory, we propose IS and OOS representation learning pipelines, which facilitate the learning of OOS connections from the current scope of the system, and reserve redundancy for new agents to be added to the current scope.
- We experimentally evaluate ALaSI with seven large dynamical systems, and show that ALaSI manages to precisely and efficiently infer the connections under both supervised and unsupervised settings.

2 Related Work

Structural inference. The aim of structural inference is to accurately reconstruct the connections between the agents in a dynamical system with observational agents' states. Among the wide variety of methods, Neural Relational Inference (NRI) (Kipf et al., 2018) was the first to address the problem of structural inference based on observational agents' states with the help of a Variational Auto-encoder (VAE) operating on a fixed fully connected graph structure. Several works have been proposed based on further improvement on NRI. Such as extending to multi-interaction systems (Webb et al., 2019), integrating efficient message-passing mechanisms (Chen et al., 2021), using modular meta-learning (Alet et al., 2019), and eliminating indirect connections with iterative process (Wang & Pang, 2022). From the aspect of Granger-causality, amortized causality discovery (ACD) (Löwe et al., 2022) attempted to infer a latent posterior graph from temporal conditional dependence, while Wu et al. (2020) proposed the Minimum Predictive Information Regularization (MPIR) model and used a learnable noise mask on nodes to reduce the computational cost. In addition to the work mentioned above, several frameworks inferred the connectivity with different problem settings. Some approaches fitted a dynamics model and then produced a causal graph estimate of the model by using recurrent models (Tank et al., 2021; Khanna & Tan, 2020), or inferred the connections by generating edges sequentially (Johnson, 2017; Li et al., 2018), or were specially designed to infer the connections of dynamic graphs (Ivanovic & Pavone, 2019; Graber & Schwing, 2020; Li et al., 2022). However, because of the fixed latent space in VAE or exponential computational efficiency, most of the methods mentioned above are incapable of structural inference on large dynamical systems and have difficulties in the efficient utilization of prior knowledge.

Deep Active learning. ALaSI follows the strategy of DeepAL (Gal et al., 2017; Pop & Fulop, 2018; Kirsch et al., 2019; Tran et al., 2019; Ren et al., 2022), attempting to combine the strong learning capability of deep learning in the context of high-dimensional data processing and the significant potential of Active Learning (AL) in effectively reducing labeling costs. To solve the problem of insufficient labeled sample data, (Tran et al., 2019) leveraged generative networks for data augmentation, and (Wang et al., 2016) expanded the labeled training set with pseudo-labels. Moreover, Hossain & Roy (2019) and Siméoni et al. (2020) used labeled and unlabeled datasets to combine supervised and semisupervised training with AL methods. Several works have been proposed on how to improve the batch sample query strategy (Shi & Yu, 2019; Kirsch et al., 2019; Zhdanov, 2019; Ash et al., 2020). As we will show, by leveraging the advantages of DeepAL, ALaSI is competent in efficiently and accurately inferring the existence of directed connections with a small labeled pool of prior knowledge.

Partial Information Decomposition. Partial Information Decomposition (PID) explicitly quantifies the information associated with two or more information sources that is not present in any subset of those sources (Williams & Beer, 2010; Lizier et al., 2013; Pakman et al., 2021). Therefore, PID is widely utilized to uncover the underlying connections between the agents in the dynamical systems in the field of physics (Barrett, 2015; Makkeh et al., 2018) and biology (Chan et al., 2017; Cang & Nie, 2020). Moreover, Lizier et al. (2013) extended the ordinary PID to cases with two or more sources and also considered past ego state as a source. Based on (Lizier et al., 2013), we derive a novel method for learning OOS messages from the current scope. Besides that, we also extend the original symmetric formulation of PID to unsymmetric cases by integrating temporal information, to enable ALaSI to infer the existence of directed connections even without any prior knowledge.

3 Preliminaries

3.1 Notations and General Problem Definition

We view a dynamical system $\mathcal{S}$ as $\mathcal{S} = \{\mathcal{V}, \mathcal{E}\}$, in which $\mathcal{V}$ represents the set of n agents in the system: $\mathcal{V} = \{v_i, 1 \leq i \leq n\}$, and $\mathcal{E}$ denotes the directed connections between the agents: $(v_i, v_j) \in \mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$. We focus on the cases

where we have recordings of the agents' states over a time period: $\mathcal{V} = \{V^t, 0 \leq t \leq T\}$, where T is the total number of time steps, and V^t is the set of features of all the n agents at time step t: $V^t = \{v_1^t, v_2^t, \ldots, v_n^t\}$. We name the recordings as trajectories. Based on the trajectories, we aim to infer the existence of directed connections between any agent-pair in the system. The connections are represented as $\mathcal{E} = \{\mathbf{e}_{ij} \in \{0, 1\}\}$, where $\mathbf{e}_{ij} = 1$ (or 0) denotes the existence of connection from agent i to j (or not). We sample a total number of K trajectories. With the notations above, the dynamics for agents within the system is:

$$v_i^{t+1} = v_i^t + \Delta \cdot \sum_{j \in \mathcal{U}_i} f\Big(\|v_i, v_j\|_\alpha\Big), \tag{1}$$

where Δ denotes a time interval, $\mathcal{U}_i$ is the set of agents connected with agent i, and $f(\cdot)$ is the state-transition function deriving to dynamics caused by the edge from agent j to i, and $\|\cdot, \cdot\|_\alpha$ denotes the α-distance. We state the problem of structural inference as searching for a combinatorial distribution to describe the existence of a directed connection between any agent pair in the dynamical system.

3.2 Problem Definition in the Context of DeepAL

Assume we have two sets of trajectories, the set of trajectories without knowing connectivity $\mathcal{D}_{\text{pool}} = \{\mathcal{V}_{\text{pool}}, \mathcal{E}_\emptyset\}$, and the set of trajectories for training $\mathcal{D}_{\text{train}} = \{\mathcal{V}_{\text{train}}, \mathcal{E}_{\text{train}}\}$, where $\mathcal{E}_\emptyset$ denotes the empty set of connectivity. We consider two scenarios: in the first scenario we have access to the ground truth of connectivity $\mathcal{E}$ in the system, and we perform a supervised-learning-based DeepAL with ALaSI:

$$\min_{\mathbf{s}^L : |\mathbf{s}^L| < \mathcal{K}} \mathbb{E}_{e \sim P_{\mathcal{E}_{\text{train}}}, v \sim P_{\mathcal{V}_{\text{train}}}} [\mathcal{L}(e, v; A_{\mathbf{s}^0 \cup \mathbf{s}^L})], \tag{2}$$

where $\mathbf{s}^0$ is the initial pool of m agents chosen from $\mathcal{D}_{\text{train}}$, as well as the connectivity between them, $\mathbf{s}^L$ is the extra pool with budget $\mathcal{K}$, A represents the algorithm of ALaSI, $\mathcal{L}$ denotes the learning objective and we denote P_x as the sampling space of variable x. The second scenario is where the ground-truth connectivity is inaccessible during training, and we show that ALaSI is competent to infer the connections in an unsupervised setting with an oracle: PID (Williams & Beer, 2010; Lizier et al., 2013). Thus, instead of having $\mathcal{E}_{\text{train}}$ available in $\mathcal{D}_{\text{train}}$, we leverage PID to calculate the connectivity between the agents in the pool at every round of sampling:

$$\min_{\mathbf{s}^k : |\mathbf{s}^k| < \mathcal{K}} \mathbb{E}_{e \sim P_{\mathcal{E}_{\text{PID}}}, v \sim P_{\mathcal{V}_{\text{train}}}} [\mathcal{L}(e, v; A_{\mathbf{s}^0 \cup \mathbf{s}^k})], \tag{3}$$

where $\mathbf{s}^k = \{\mathcal{V}_{\text{train}}, \mathcal{E}_{\text{PID}}\}$ denotes the pool, with $\mathcal{E}_{\text{PID}}$ denoting the connections generated by PID operating on the agents in the pool, and the number of agents in $\mathbf{s}^k$ has a budget $\mathcal{K}$. PID set up the initial set $\mathbf{s}^0$ as that of $\mathbf{s}^k$, but with a different size of agents m. We consider ALaSI with both supervised and unsupervised learning and conduct experiments on both settings, to demonstrate its performance.

3.3 Background on PID

PID of two sources X_1, X_2 amounts to expressing the mutual information (MI) of X_1, X_2 with a target Y as a sum of four non-negative terms (Pakman et al., 2021):

$$\begin{aligned} I(Y; (X_1, X_2)) = U(Y; X_1) + U(Y; X_2) \\ + R(Y; X_1, X_2) + Sy(Y; X_1, X_2), \end{aligned} \tag{4}$$

corresponding to unique (U_1, U_2), redundant (R) and synergistic (Sy) contributions, respectively. To calculate the PID terms, the redundant information is first calculated using the specific information I_{spec}, which quantifies the information provided by one variable about a specific state of another variable (Chan et al., 2017), such as from X_1 about state y of variable Y:

$$I_{spec}(y; X_1) = \sum_{x \in X_1} p(x|z) \Big(\log \frac{1}{p(z)} - \log \frac{1}{p(z|x)} \Big). \tag{5}$$

Then the redundant contribution is calculated by comparing the amount of information provided by each source within set $B = \{X_1, X_2\}$ about each state of the target Y:

$$R(Y; X_1, X_2) = \sum_{y \in Y} p(y) \min_B I_{spec}(y; B). \tag{6}$$

The unique information and the synergistic information can be calculated from the redundant information based on the consistency equations (Williams & Beer, 2010):

$$U(Y; X_1) = I(X_1; Y) - R(Y; X_1, X_2), \tag{7}$$

$$Sy(Y; X_1, X_2) = II(Y; X_1; X_2) + R(Y; X_1, X_2), \tag{8}$$

where the interaction information (McGill, 1954) of three variables $II(a; b; c)$ is calculated as $I(a; b|c) - I(a; b)$. In a system of n agents, given a pair of agents X_1 and Y, there are $n-2$ triplets involving the pair. The MI between X_1 and Y is unaffected by the choice of a third agent X_2, because MI is a pairwise measure. But $U(Y; X_1)$ varies depending on X_2, and the difference between $I(Y; X_1)$ and $U(Y; X_1)$ is equal to the redundancy between all three agents (Equation 7). So a popular method (Chan et al., 2017) is calculating the ratio score $r = U(Y; X_1)/I(Y; X_1)$ as capturing the proportion of MI that is accounted for by unique information between X_1 and Y, as opposed to redundant information between all three agents. If X_1 and Y are connected, their r is higher than any other pairs, and we can follow this method to infer the connectivity by calculating the ratio scores for all agent pairs.

4 Method

In this section, we present ALaSI, a scalable structural inference framework based on agent-wise DeepAL. We start by formulating such a learnable framework in Section 4.1.

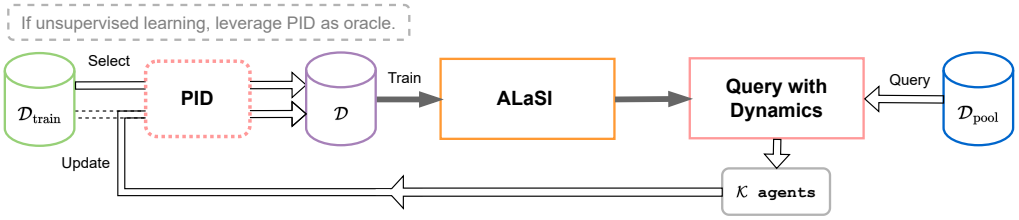

Figure 1. Overview of the pipeline of ALaSI.

After that, we describe the IS and OOS operations in Section 4.2, which are of great significance to make the framework scalable. Especially, we propose the hybrid loss and the query strategy with dynamics in Sections 4.3 and 4.4, respectively. Last but not least, we discuss the integration of PID into ALaSI in Section 4.5, which enables ALaSI to infer the connectivity with unsupervised learning.

4.1 Active Structural Inference with Dynamics

The basic idea behind ALaSI is to infer the existence of directed connection between two agents with the help of dynamics. According to Equation 1, we may describe it as: the correct inference of the connectivity enables the algorithm to predict the future states of the agent with smaller error. We formulate the statement as:

$$\arg\min_{\mathcal{U}_i \subseteq \mathcal{V}} \mathbb{E}_{\theta \sim p(\theta|\{\mathcal{V},\mathcal{E}\})} \mathcal{R}\big(v_i^{t+1}, P(\hat{v}_i^{t+1}|v_i^t, \mathcal{U}_i, \theta)\big), \quad (9)$$

where $\mathcal{U}_i$ represents the agents connected to agent i, $\mathcal{R}$ is the loss function to quantify the dynamics prediction error between actual dynamics v_i^{t+1} and predicted dynamics $\hat{v}_i^{t+1}$, and θ is the parameters of the model. The problem setting in Equation 9 is also widely adopted (Kipf et al., 2018; Webb et al., 2019; Löwe et al., 2022; Wang & Pang, 2022). For small dynamical systems, we can directly follow this formulation and leverage generative models such as a VAE to work on a fully-connected initial graph, in order to infer the connectivity of the whole system. However, for large dynamical systems, it is impractical and unattainable to infer the connectivity in the same way, which is also a common problem observed in the literature on structural inference.

In this work, we extend Equation 9 for large dynamical systems with the help of DeepAL. Unlike previous DeepAL algorithms, which train models on batch-wise selections (Gal et al., 2017; Kirsch et al., 2019; Pop & Fulop, 2018; Tran et al., 2019), we design ALaSI to train on agent-wise selections. The pool consists of features of different agents, and the directed connections between these agents. By training ALaSI on the pool, we try to encourage the framework to capture the statistics to describe the existence of connections between any agent-pair:

$$\arg\min_{\mathcal{U}_i \subseteq \mathcal{D}} \mathbb{E}_{\theta \sim p(\theta|\mathcal{D})} \mathcal{R}\big(v_i^{t+1}, Q(\hat{v}_i^{t+1}|v_i^t, \mathcal{U}_i, \theta)\big). \quad (10)$$

Different from Equation 9, we have a limited scope $\mathcal{D}$ on the available agents and their features, and we can only learn the representation of connections based on current scope $\mathcal{D}$. However, there possibly simultaneously exist connections between the OOS agents and the agents inside the scope, and discarding the influences of these OOS connections would lead to inaccurate inference results. As a consequence, we need to design the model Q so that it can distinguish the portion of information related to OOS connections and the portion of information coming from connections in the scope, in order to learn the representation of connection precisely and also reserve redundancy for new agents to be added into the pool. We describe the pipeline of ALaSI in Figure 1 and Algorithm 2 in the appendix.

4.2 Inter- / Out-of-Scope Operations

Previous works leveraged a fixed scope on the entire set of agents of the dynamical system, and thus struggled with the curse of scalability (Kipf et al., 2018; Webb et al., 2019; Löwe et al., 2022; Wang & Pang, 2022). To address this issue, we propose a set of inter-/out-of-scope operations in order to make ALaSI scalable. Suppose we have a partial view of n_p agents in the dynamical system ($n_p < n$), and we call the partial view a scope. For any agent i in the scope, it is possible that it has connections within the scope and also has connections from agents out of the scope simultaneously. We demonstrate an example in Figure 2.

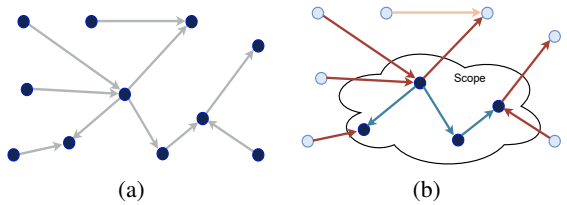

Figure 2. (a) A dynamical system with 11 agents. (b) A scope of 4 agents on the same system.

In Figure 2(a), there is a dynamical system consisting of 11 agents and directed connections. Then suppose we have a scope with 4 agents on the system and is shown as the cloud line in Figure 5(b). Based on the segmentation of the scope, the agents in the system are divided into two groups: IS agents (in original color), and OOS agents (in nattier blue). But the connections are segmented into three groups:(a) IS connections (in blue), (b) OOS connections (in red) and (c)

non-observable connections (in cream). Besides agents in the scope, IS agents may also be affected by OOS agents, thus we need to take the OOS connections into consideration and separate their influence.

We denote $\mathcal{V}^t_{\text{inter}}$ as the set of IS agents' states and Z^t_{oos} as the summary of OOS agents' states for an ego agent i at time-step t. $\mathcal{V}^t_{\text{inter}}$ and Z^t_{oos} share many characteristics: (1) Since both of them represent the features within the same system, the connections between either IS agents or OOS agents and agent i have the same dynamic function as shown in Equation 9; (2) From the perspective of information theory (Kraskov et al., 2004; Belghazi et al., 2018), we can easily reach the statement that: $I(v^t_i; \mathcal{V}^t_{\text{inter}}) \neq 0$ and $I(v^t_i; Z^t_{\text{oos}}) \neq 0$, where v^t_i represents the features of agent i at time step t, and $I(\cdot\ ;\cdot)$ denotes the MI between two entities. Therefore, we reformulate Equation 10 as:

$$\arg\min_{\mathcal{U}_i \subseteq \mathcal{D}} \mathbb{E}_{\theta\sim p(\theta|\mathcal{D})} \mathcal{R}\big(v^{t+1}_i, Q(\hat{v}^{t+1}_i | v^t_i, \mathcal{V}^t_{\text{inter}}, Z^t_{\text{oos}}, \theta)\big). \tag{11}$$

Yet the calculation of Z^t_{oos} is agnostic, it is necessary to have another set of derivations:

Proposition 4.1. *If we assume Z^t_{oos} only captures the information that affects v^t_i and is different from $\mathcal{V}^t_{inter}$, we can reach the following statements:*

$$\begin{aligned} &I(\mathcal{V}^t_{inter}; Z^t_{oos}) < I(v^{t+1}_i; Z^t_{oos}), \\ &\textit{and } I(\mathcal{V}^t_{inter}; Z^t_{oos}) < I(v^{t+1}_i; \mathcal{V}^t_{inter}). \end{aligned} \tag{12}$$

Proposition 4.1 infers that the MI between $\mathcal{V}^t_{\text{inter}}$ and Z^t_{oos} is the smallest among the MI between any pair from $\mathcal{V}^t_{\text{inter}}$, Z^t_{oos} and v^{t+1}_i. It also suggests that we can infer information about Z^t_{oos} from v^{t+1}_i. We prove the proposition in Section B.1 in the appendix. Based on the MI of time series between two sources and its own present state (Lizier et al., 2013), as well as the Markovian assumption, we have:

$$\begin{aligned} I(v^{t+1}_i; v^t_i, \mathcal{V}^t_{\text{inter}}, Z^t_{\text{oos}}) = I(v^{t+1}_i; v^t_i) + I(v^{t+1}_i; \mathcal{V}^t_{\text{inter}} |\ v^t_i) \\ + I(v^{t+1}_i; Z^t_{\text{oos}} |\ v^t_i, \mathcal{V}^t_{\text{inter}}). \end{aligned} \tag{13}$$

Since MI terms are non-negative by design, the last term on the right of Equation 13 suggests that given v^{t+1}_i, we can derive the information about Z^t_{oos} conditional on v^t_i and $\mathcal{V}^t_{\text{inter}}$. Therefore, we implement the inter-/out-of-scope message learning pipelines with neural networks and the pipeline of which is shown in the following equations:

$$Z_i = f_{\text{inter1}}([v^t_i, \mathcal{V}^t_{\text{inter}}]), \tag{14}$$

$$e_{\text{inter}} = f_{\text{inter2}}([Z_i \odot \mathcal{V}^t_{\text{inter}}, v^t_i]), \tag{15}$$

$$e_{\text{oos}} = f_{\text{oos2}}([f_{\text{oos1}}(v^t_i), e_{\text{inter}}]), \tag{16}$$

$$e_{\text{out}} = f_{\text{output}}(f_{\text{dynamics}}(e_{\text{inter}}, e_{\text{oos}}), v^t_i), \tag{17}$$

where e_{inter} and e_{oos} are learned inter-/out-of-scope representations ($\mathcal{V}^t_{\text{inter}}$ / Z^t_{oos}), respectively, $[\cdot,\cdot]$ is the concatenation operation, f_{inter1} is the neural network to learn the existence of connections between agent i and the agents inside the current scope, Z_i represents the connectivity inside the scope with regards to agent i, and $\odot$ is the operation to select agents based on connectivity. Suppose we have $\mathcal{K}$ agents in the scope, then $Z_i \in [0,1]^{\mathcal{K}}$. So for any agent i, j in the scope, we have $z_{ij} \in [0,1]$, representing the connectivity from agent i to agent j. In practice, we reparametrize z_{ij} with Gumbel-Softmax (Jang et al., 2017) to enable backpropagation (see Section C.5 in the appendix for implementation). Besides that, f_{inter2}, f_{oos1}, and f_{oos2} are the neural networks to learn representations of IS messages, OOS embeddings, and OOS messages, respectively. Finally, in Equation 17, we learn the representations for dynamics with f_{dynamics}, and output the future state of agent i (e_{out}) with f_{output}. In addition to the operations mentioned above, we leverage loss functions (in Section 4.3) to encourage ALaSI to extract OOS messages from v^t_i and $\mathcal{V}^t_{\text{inter}}$.

4.3 Train with a Hybrid Loss

The loss function has three roles: (a) encouraging the model to learn OOS representations; (b) calculating dynamics error; and (c) estimating the connectivity prediction error. As mentioned in Section 4.2, the OOS message Z^t_{oos} can be derived from v^{t+1}_i, v^t_i and $\mathcal{V}^t_{\text{inter}}$. Based on the triplet loss (Schultz & Joachims, 2003; Schroff et al., 2015) and Proposition 4.1, we derive the following loss function to learn the OOS message:

$$\mathcal{L}_{\text{oos}} = \frac{1}{(T-1)\cdot|\mathcal{D}|} \sum_{t}^{T-1} \sum_{i\in\mathcal{D}} \big[- I(Z^t_{\text{oos}}; v^{t+1}_i) \big], \tag{18}$$

where T represents the total count of time-steps, $\mathcal{D}$ represents the current scope, $|\mathcal{D}|$ denotes the number of agents in the scope. (We discuss the derivation in Section B.2.) We implement the calculation and maximization of mutual information with the help of DeepInfoMax (Hjelm et al., 2019). However, we have to introduce a regularization term to encourage the learned representations of Z^t_{oos} and $\mathcal{V}^t_{\text{inter}}$ to be independent of each other, and we leverage distance correlation (Székely et al., 2007). As already proved (Székely & Rizzo, 2009; 2012; 2014), the distance correlation between two variables is zero only when two variables are independent of each other. Therefore, we calculate and minimize the distance correlation between Z^t_{oos} and $\mathcal{V}^t_{\text{inter}}$:

$$\mathcal{L}_{\text{dc}} = \frac{1}{(T-1)\cdot|\mathcal{D}|} \sum_{t}^{T-1} \sum_{i\in\mathcal{D}} \frac{\text{dCov}^2(Z^t_{\text{oos}}, \mathcal{V}^t_{\text{inter}})}{\sqrt{\text{dVar}(Z^t_{\text{oos}})\text{dVar}(\mathcal{V}^t_{\text{inter}})}}, \tag{19}$$

where dCov and dVar are the squared sample distance covariance and the distance variance, respectively, and we describe the procedures for calculating these terms in Section C.4.2 in the appendix. Besides that, we also need the

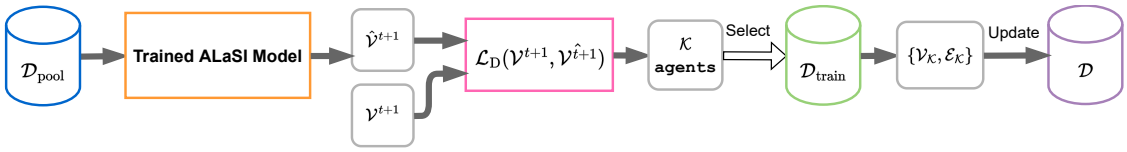

Figure 3. Query strategy with dynamics in ALaSI.

loss function for dynamics:

$$\mathcal{L}_{\text{D}} = -\frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\frac{||v_i^{t+1}-\hat{v}_i^{t+1}||^2}{2\sigma^2} + \text{const}, \tag{20}$$

where v^{t+1} and $\hat{v}^{t+1}$ are the ground-truth dynamics and predicted dynamics, respectively, and σ is the variance. Moreover, we have the loss function for connectivity:

$$\mathcal{L}_{\text{con}} = -\frac{1}{|D|^2}\sum_{i,j\in\mathcal{D}} z_{ij}\log\left(f(\hat{z}_{ij})\right), \tag{21}$$

where $f(\cdot)$ denotes the softmax function, z_i and $\hat{z}_i$ represent the ground-truth connectivity and predicted connectivity in the scope, respectively. With the proposed terms above, we can summarize the hybrid loss function $\mathcal{R}$ as:

$$\mathcal{R} = \alpha\cdot\mathcal{L}_{\text{D}} + \beta\cdot\mathcal{L}_{\text{con}} + \gamma\cdot\mathcal{L}_{\text{oos}} + \eta\cdot\mathcal{L}_{\text{dc}}, \tag{22}$$

where α, β, γ and η are the weights for the loss terms, trying to match the scales of the last three loss terms with the dynamic loss $\mathcal{L}_{\text{D}}$. We discuss the details of loss terms in Section C.4 in the appendix.

4.4 Query with Dynamics

Interestingly, AL is also called "query learning" in the statistics literature (Settles, 2009), indicating the importance of query strategies in the algorithms of AL. Query strategies are leveraged to decide which instances are most informative and aim to maximize different performance metrics (Settles, 2009; Konyushkova et al., 2017). Query strategies select queries from the pool and update the training set accordingly. In this work, we propose a novel pool-based strategy:

Algorithm 1 Query with Dynamics $\mathcal{Q}$.

Input: $\mathcal{D}_{\text{train}}$ = a pool of labeled trajectories $\{\mathcal{V}_{\text{train}}, \mathcal{E}\}$,
Input: $\mathcal{D}_{\text{pool}}$ = a pool of test trajectories $\{\mathcal{V}_{\text{pool}}\}$,
Input: $\mathcal{D}$ = a pool of agents we have for training,
Parameters: Query Size: $\mathcal{K}$,
Model Weights: θ, **Dynamic Loss:** $\mathcal{L}_{\text{D}}$,
Output: Query of $\mathcal{K}$ agents,
Calculate dynamics loss $\mathcal{L}_{\text{D}}$ on all of the agents in $\mathcal{D}_{\text{pool}}$ with only one other agent in scope,
Select $\mathcal{K}$ agents with largest dynamics prediction error,
Return: $\mathcal{K}$ agents and update $\mathcal{D} = \mathcal{D}\cup\mathcal{V}_i, i\in\{\mathcal{K}\}$ with features and connectivity from $\mathcal{D}_{\text{train}}$.

Query with Dynamics, which selects queries of $\mathcal{K}$ agents with the largest dynamics prediction error $\mathcal{L}_{\text{D}}$ from the pool $\mathcal{D}_{\text{pool}}$, and then we update training set $\mathcal{D}$ with the features and connectivity of $\mathcal{K}$ agents from $\mathcal{D}_{\text{train}}$. If we have no access to the connectivity as in unsupervised learning, we run PID to align directed connections to the agents in pool $\mathcal{D}$ with additional $\mathcal{K}$ agents (as shown in Algorithm 2). We describe the query strategy in Algorithm 1 and Figure 3. It is notable that although we have labels on the existence of connections, we do not query agents purely on them. On one hand, the characteristic of dynamical systems (Equation 9) provides strong support that the wrong alignment of connections leads to large dynamics error $\mathcal{L}_{\text{D}}$. On the other hand, we try to reserve redundancy for unsupervised learning cases, where ALaSI has no access to ground-truth connections. In this case, we ought to use alternative algorithms as an oracle, such as PID, to estimate the existence of connections and build $\mathcal{D}_{\text{train}}$. However, it may be risky that the oracle has a strong bias on the set for training $\mathcal{D}$, and thus errors in this set are unavoidable. As a result, we query agents from the entire pool $\mathcal{D}_{\text{pool}}$ according to their dynamics error $\mathcal{L}_{\text{D}}$, thus wrong connections would be recognized by our query strategy.

4.5 Structural Inference with PID

As mentioned above, it is possible that we have no access to the ground-truth connectivity of the dynamical system. ALaSI manages to infer the connections with the help of an oracle: PID (Williams & Beer, 2010; Lizier et al., 2013). The PID framework decomposes the information that a source of variables provides about a destination variable (Lizier et al., 2013). In our cases to infer the existence of directed connections between a pair of agents i and j, we extend the formulation in (Pratapa et al., 2020) with temporal ordering to infer the direction of connections. We first decompose the features of agent i in the scope: V_i, into two sets: $X_i^{0:T-1} = \{v_i^t, 0\le t\le T-1\}$ and $X_i^{1:T} = \{v_i^t, 1\le t\le T\}$. Then we calculate the r_{ij} between two agents i and j over all other agents in the scope:

$$r_{ij} = U(X_j^{1:T}; X_i^{0:T-1})/I(X_j^{1:T}; X_i^{0:T-1}), \tag{23}$$

where r_{ij} is the ratio score for connection from agent i to j, and is then ranked with the results obtained from all of the agent pairs in the scope. So the temporal information derives directed connections in this formulation. We summarize the details of PID in ALaSI in Algorithm 3 in the appendix. With the help of PID, ALaSI can infer the existence of di-

rected connections even without any prior knowledge about the connectivity, which broadens the application scenarios of ALaSI. It is possible to use other methods as an oracle for ALaSI, such as pure mutual-information-based methods, SCODE (Matsumoto et al., 2017) or even classic VAE-based structural inference methods (Kipf et al., 2018; Webb et al., 2019; Alet et al., 2019; Löwe et al., 2022), which shows a high ability of adaption of ALaSI.

5 Experiments

We test ALaSI on seven different large dynamical systems, including simulated networks and real-world gene regulatory networks (GRNs). Implementation details can be found in Section C in the appendix. Besides that, we include additional experiments on the integration of prior knowledge with unsupervised learning and ablation study in Section D.

Datasets. We first test our framework on physical simulations of spring systems, which is also mentioned in (Kipf et al., 2018). Different from that in (Kipf et al., 2018), we sample the trajectories of balls in the system with fixed connectivity, but with different initial conditions. We sample the trajectories by varying the number of balls: $\{50, 100, 200, 500\}$, and we name the corresponding datasets as: "Springs50", "Springs100", "Springs200", and "Springs500". Moreover, we collect three real-world GRNs from literature, namely single cell dataset of embryonic stem cells (ESC) (Biase et al., 2014), a cutoff of Escherichia coli microarray data (E. coli) (Jozefczuk et al., 2010), and a cutoff of Staphylococcus aureus microarray data (S. aureus) (Marbach et al., 2012). And the three GRNs have 96, 1505 and 1084 agents, respectively.

Baselines and metrics. We compare ALaSI with the state-of-the-art baseline methods:

- NRI (Kipf et al., 2018): a variational-auto-encoder model for relational inference.
- fNRI (Webb et al., 2019): an NRI-based model with a multiplex graph, allowing each layer to encode for each connection-type.
- 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 across samples with different underlying causal graphs.
- MPIR (Wu et al., 2020): a model based on minimum predictive information regularization.
- PID (Williams & Beer, 2010): computes the ratio between unique mutual information between any agent-pair in the system and aligns connections according to the ranking.

Despite NRI, fNRI, MPM and ACD being originally designed to operate with unsupervised learning, we follow the instruction in their paper and only train the encoders to show their results of supervised learning. We describe the implementation details of the baseline methods in Section C.6. The evaluation results are demonstrated with the Area Under the Receiver Operating Characteristic (AUROC), showing the model's ability to discriminate between cases (positive examples) and non-cases (negative examples), and in this paper, it is used to make clear the method's ability to distinguish actual connections and non-connections.

5.1 Experimental Results of Supervised Learning

We first train ALaSI and baseline methods with supervised learning. It is worth mentioning that despite our efforts, we did not find an approach to train MPIR and PID in a supervised way without violating their inference mechanisms. For the rest of the baseline methods, we follow the instruction in their paper and only train the encoders on the partial knowledge of connections. The experimental results of ALaSI and baseline methods are shown in Figure 4. We report the results as the average AUROC values of ten runs and as a function of the proportion of labeled connections. The number of labeled connections is calculated as the square of the number of agents in the scope, where we mark both connections and non-connections as labeled. We subtract the number of labeled connections with the square of the total number of agents in the system to obtain the proportion of labeled connections. Each sub-figure corresponds to the experimental results on a specific dataset.

As shown in Figure 4, the results of baseline methods are positively affected by the proportion of labeled connections during training, and only MPM is marginally better than the other baseline methods on most of the datasets. The rest of the baselines perform almost equally. The results of ALaSI are also positively correlated with the proportion of labeled connections, but the results are much better than any of the baselines. Although ALaSI is only marginally better than any of the baselines on the datasets of "Springs50" and "Springs100" when the proportion of labeled connections is relatively small (smaller than 0.1), ALaSI outperforms baselines greatly when the proportion of labeled connections is greater than 0.2 on these datasets. ALaSI also infers connectivity with remarkably higher accuracy than baseline methods on the rest of the datasets.

Moreover, we also observe that ALaSI learns the connectivity of large dynamical systems more efficiently than baselines. For example, as shown in the experimental results on all of the datasets except "Springs200", with only 60% of the prior knowledge, ALaSI reaches higher inference accuracy than any baseline methods operating with 80% of the prior knowledge. And this phenomenon is more remarkable in "Springs100", "Springs500" and "E. coli", where ALaSI outperforms baselines with only 50% of the prior

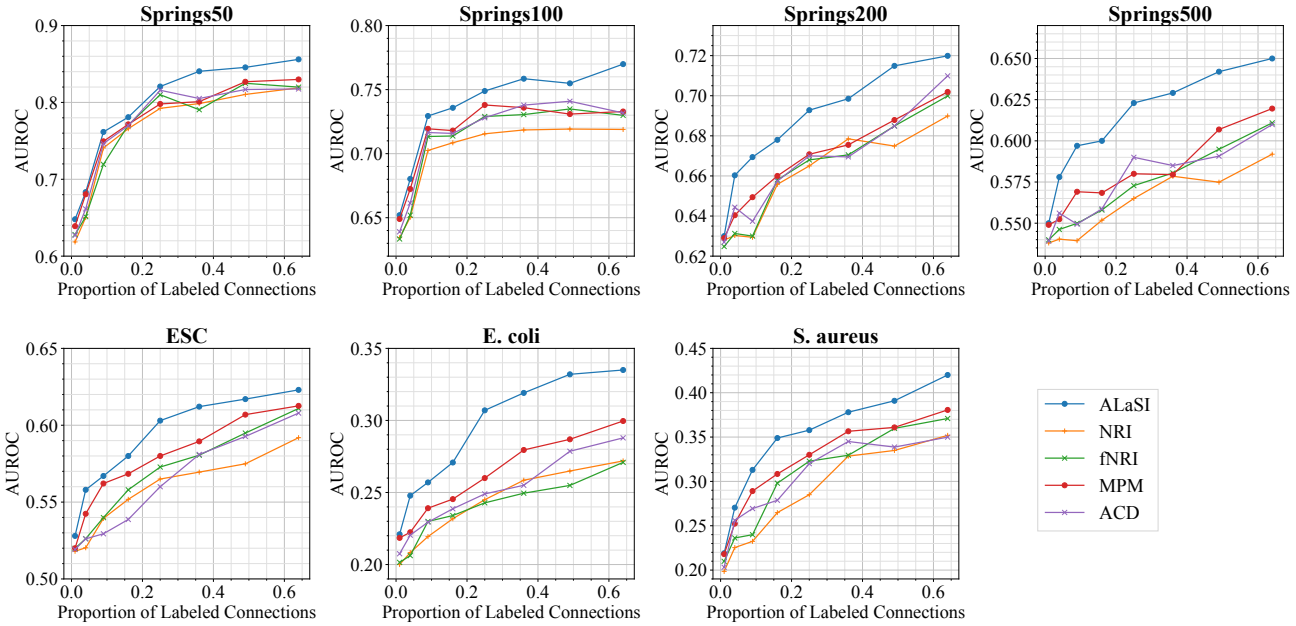

Figure 4. Averaged AUROC results of ALaSI and baseline methods as a function of the proportion of labeled connections. Baseline methods are modified to be trained in a supervised way.

Table 1. Averaged AUROC results (in %) of baseline methods and ALaSI with unsupervised learning.

Method	Springs50	Springs100	Springs200	Springs500	ESC	E. coli	S. aureus
NRI	61.7± 0.04	55.2± 0.05	53.7± 0.05	51.1± 0.06	39.2± 0.06	15.6± 0.06	35.1± 0.07
fNRI	62.1± 0.03	56.7± 0.04	54.0± 0.04	51.6± 0.03	39.8± 0.07	15.4± 0.06	35.1± 0.06
MPM	63.1± 0.05	59.0± 0.05	54.4± 0.04	51.8± 0.04	40.2± 0.07	17.0± 0.05	37.5± 0.05
ACD	62.0± 0.04	58.9± 0.03	53.9± 0.05	51.5± 0.03	38.7± 0.04	16.2± 0.07	36.9± 0.08
MPIR	49.7± 0.02	44.4± 0.03	42.0± 0.03	41.1± 0.03	30.6± 0.05	15.1± 0.04	33.1± 0.04
PID	67.8± 0.01	63.0± 0.02	59.2± 0.03	54.7± 0.03	45.1± 0.04	19.5± 0.03	37.8± 0.03
ALaSI	**73.5**± 0.03	**69.8**± 0.04	**66.1**± 0.05	**63.2**± 0.06	**57.3**± 0.07	**23.4**± 0.05	**39.2**± 0.05

Table 2. Averaged training time (in hours) of baseline methods and ALaSI with unsupervised learning.

Method	Springs50	Springs100	Springs200	Springs500	ESC	E. coli	S. aureus
NRI	29.2± 0.02	40.6± 0.03	57.1± 0.05	85.1± 0.03	39.4± 0.04	118.6± 0.06	101.7± 0.06
fNRI	31.0± 0.03	49.0± 0.04	58.0± 0.03	86.8± 0.05	42.0± 0.06	121.4± 0.07	105.3± 0.05
MPM	35.9± 0.02	51.6± 0.03	57.4± 0.03	85.6± 0.04	44.1± 0.05	124.0± 0.06	105.9± 0.06
ACD	49.0± 0.04	82.4± 0.02	63.9± 0.03	90.0± 0.04	80.4± 0.04	130.9± 0.06	113.5± 0.05
MPIR	**12.6**± 0.01	**20.7**± 0.02	**42.0**± 0.03	**51.5**± 0.03	**19.5**± 0.04	**65.1**± 0.03	**47.6**± 0.02
PID	51.6± 0.01	100.2± 0.01	151.0± 0.02	183.4± 0.02	89.3± 0.02	267.1± 0.02	230.8± 0.01
ALaSI	25.5± 0.04	33.8± 0.03	46.1± 0.04	60.3± 0.04	37.2± 0.05	87.0± 0.04	72.9± 0.05

knowledge. Thanks to DeepAL and query with dynamics, ALaSI can update the labeling pool with the most informative addition of agents. Besides that, the IS and OOS operations encourage the model to learn connections within the scope and meanwhile also reserve redundancy for possible OOS connections. Consequently, ALaSI is able to learn the connectivity of large dynamical systems with less prior knowledge under supervised learning.

5.2 Experimental Results of Unsupervised Learning

We report the final average AUROC values and standard deviations of ALaSI and baseline methods under unsupervised learning from ten runs in Table 1, the average training time and standard deviations in Table 2, as well as the number of required GPUs in Table 3. We can observe from Table 1 that all of the methods unsurprisingly perform worse than themselves in supervised learning, which is also stated in (Kipf et al., 2018; Chen et al., 2021). ALaSI performs better than

Table 3. Number of utilized GPU cards of baseline methods and ALaSI with unsupervised learning.

Method	Springs50	Springs100	Springs200	Springs500	ESC	E. coli	S. aureus
NRI	1	2	4	6	1	8	6
fNRI	1	2	4	6	1	8	6
MPM	1	2	4	6	1	8	6
ACD	1	2	4	6	1	8	6
MPIR	1	1	1	1	1	1	1
PID	1	1	1	1	1	1	1
ALaSI	1	1	1	1	1	1	1

any of the baseline methods on all of the datasets with large margins (up to 17.1%), which certainly verifies the inference accuracy of ALaSI on the unsupervised structural inference of large dynamical systems. Moreover, the average training time of ALaSI and baseline methods is shown in Table 2. It is worth mentioning that as shown in Table 3, most of the baseline methods are trained on multiple GPU cards when the dataset has more than 100 agents, while ALaSI is trained on a single GPU card. Experimental settings with details may refer to Section C.1. The averaged training time of ALaSI is only longer than MPIR across all of the datasets, while much more accurate than MPIR.Although the AUROC values of PID are the highest among baseline methods, its operation time is much longer than the rest, and it is nevertheless less accurate than ALaSI. Compared with the rest of the baselines, thanks to the query strategy with dynamics and the OOS operation, ALaSI manages to infer the connections for large dynamical systems with higher efficiency even with unsupervised learning. These results demonstrate the computational efficiency and effectiveness of ALaSI for structural inference on large dynamical systems. More experimental results on noisy data and ablation studies can be found in Section D in the appendix.

6 Conclusion

This paper has introduced ALaSI, a scalable structural inference framework based on DeepAL. The query with dynamics encourages the framework to select the most informative agents to be labeled based on dynamics error, and thus leads to faster convergence. The OOS operation enables the framework to distinguish IS messages and OOS messages based on the current view of the partial system, which on the other hand promotes the scalability of ALaSI. The experimental results on the seven large datasets have validated the scalability and inference accuracy of ALaSI. The experiments under supervised settings suggest the possibility of leveraging ALaSI to infer the connectivity of large dynamical systems with less prior knowledge. Moreover, the experiments under unsupervised settings demonstrate the broad application scenarios of ALaSI to infer the connectivity even without prior knowledge. Future research includes structural inference based on causality and structural inference for systems with changing agents and connections.

Acknowledgment

Author Jun Pang acknowledges financial support from the Institute for Advanced Studies of the University of Luxembourg through an Audacity Grant (AUDACITY-2021).

References

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 (NeurIPS)*, volume 32, pp. 11804–11815, 2019.

Ash, J. T., Zhang, C., Krishnamurthy, A., Langford, J., and Agarwal, A. Deep batch active learning by diverse, uncertain gradient lower bounds. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

Barrett, A. B. Exploration of synergistic and redundant information sharing in static and dynamical gaussian systems. *Physical Review E*, 91(5):052802, 2015.

Belghazi, M. I., Baratin, A., Rajeshwar, S., Ozair, S., Bengio, Y., Courville, A., and Hjelm, D. Mutual information neural estimation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pp. 531–540. PMLR, 2018.

Biase, F. H., Cao, X., and Zhong, S. Cell fate inclination within 2-cell and 4-cell mouse embryos revealed by single-cell RNA sequencing. *Genome Research*, 24(11): 1787–1796, 2014.

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.

Cang, Z. and Nie, Q. Inferring spatial and signaling relationships between cells from single cell transcriptomic data. *Nature Communications*, 11(1):1–13, 2020.

Chan, T. E., Stumpf, M. P., and Babtie, A. C. Gene regulatory network inference from single-cell data using multivariate information measures. *Cell Systems*, 5(3): 251–267.e3, 2017.

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.

Cowen-Rivers, A., Lyu, W., Tutunov, R., Wang, Z., Grosnit, A., Griffiths, R.-R., Maravel, A., Hao, J., Wang, J., Peters, J., and Bou Ammar, H. Hebo: Pushing the limits of sample-efficient hyperparameter optimisation. *Journal of Artificial Intelligence Research*, 74, 07 2022.

Eriksson, D., Pearce, M., Gardner, J., Turner, R. D., and Poloczek, M. Scalable global optimization via local Bayesian optimization. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019.

Gal, Y., Islam, R., and Ghahramani, Z. Deep bayesian active learning with image data. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1183–1192. PMLR, 2017.

Gentile, C., Wang, Z., and Zhang, T. Achieving minimax rates in pool-based batch active learning. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, pp. 7339–7367. PMLR, 2022.

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.

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

Hjelm, R. D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., and Bengio, Y. Learning deep representations by mutual information estimation and maximization. In *Proceddings of the 7th International Conference on Learning Representations (ICLR)*, 2019.

Hossain, H. M. S. and Roy, N. Active deep learning for activity recognition with context aware annotator selection. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pp. 1862–1870. ACM, 2019.

Irwin, M. and Wang, Z. *Dynamic Systems Modeling*, pp. 1–12. John Wiley & Sons, Ltd, 2017.

Ivanovic, B. and Pavone, M. The Trajectron: Probabilistic multi-agent trajectory modeling with dynamic spatiotemporal graphs. In *Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2375–2384, 2019.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Johnson, D. D. Learning graphical state transitions. In *Proceedings of the 5th International Conference on Learning Representations (ICLR)*, 2017.

Jozefczuk, S., Klie, S., Catchpole, G., Szymanski, J., Cuadros-Inostroza, A., Steinhauser, D., Selbig, J., and Willmitzer, L. Metabolomic and transcriptomic stress response of Escherichia coli. *Molecular Systems Biology*, 6(1):364, 2010.

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

Khanna, S. and Tan, V. Y. F. Economy statistical recurrent units for inferring nonlinear granger causality. In *Proceedings of the 8th International Conference on Learning Representations (ICLR)*, 2020.

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.

Kirsch, A., van Amersfoort, J., and Gal, Y. BatchBALD: Efficient and diverse batch acquisition for deep bayesian active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 7024–7035, 2019.

Konyushkova, K., Sznitman, R., and Fua, P. Learning active learning from data. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, pp. 4225–4235, 2017.

Kraskov, A., Stögbauer, H., and Grassberger, P. Estimating mutual information. *Physical Review E*, 69:066138, 2004.

Kwapień, J. and Drożdż, S. Physical approach to complex systems. *Physics Reports*, 515(3-4):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. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):10556–10569, 2022.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., and Battaglia, P. W. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018.

Lizier, J. T., Flecker, B., and Williams, P. L. Towards a synergy-based approach to measuring information modification. In *Proceedings of the 2013 IEEE Symposium on Artificial Life (ALife)*, pp. 43–51. IEEE, 2013.

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.

Makkeh, A., Theis, D. O., and Vicente, R. Broja-2pid: A robust estimator for bivariate partial information decomposition. *Entropy*, 20(4):271, 2018.

Marbach, D., Costello, J. C., Küffner, R., Vega, N. M., Prill, R. J., Camacho, D. M., Allison, K. R., Kellis, M., Collins, J. J., and Stolovitzky, G. Wisdom of crowds for robust gene network inference. *Nature Methods*, 9(8):796–804, 2012.

Matsumoto, H., Kiryu, H., Furusawa, C., Ko, M. S., Ko, S. B., Gouda, N., Hayashi, T., and Nikaido, I. SCODE: an efficient regulatory network inference algorithm from single-cell RNA-Seq during differentiation. *Bioinformatics*, 33(15):2314–2321, 2017.

McGill, W. Multivariate information transmission. *Transactions of the IRE Professional Group on Information Theory*, 4(4):93–111, 1954.

Pakman, A., Nejatbakhsh, A., Gilboa, D., Makkeh, A., Mazzucato, L., Wibral, M., and Schneidman, E. Estimating the unique information of continuous variables. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 34, 2021.

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 33, pp. 8024–8035, 2019.

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.

Pop, R. and Fulop, P. Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles. *arXiv preprint arXiv:1811.03897*, 2018.

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.

Ren, P., Xiao, Y., Chang, X., Huang, P., Li, Z., Gupta, B. B., Chen, X., and Wang, X. A survey of deep active learning. *ACM Computing Surveys*, 54(9):1–40, 2022.

Schroff, F., Kalenichenko, D., and Philbin, J. FaceNet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

Schultz, M. and Joachims, T. Learning a distance metric from relative comparisons. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pp. 41–48. MIT Press, 2003.

Settles, B. Active learning literature survey. Technical report, University of Wisconsin-Madison, 2009.

Shi, W. and Yu, Q. Integrating bayesian and discriminative sparse kernel machines for multi-class active learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, pp. 2282–2291, 2019.

Siméoni, O., Budnik, M., Avrithis, Y., and Gravier, G. Rethinking deep active learning: Using unlabeled data at model training. In *Proceedings of the 25th International Conference on Pattern Recognition (ICPR)*, pp. 1220–1227. IEEE, 2020.

Székely, G. J. and Rizzo, M. L. Brownian distance covariance. *The Annals of Applied Statistics*, 3(4):1236–1265, 2009.

Székely, G. J. and Rizzo, M. L. On the uniqueness of distance covariance. *Statistics & Probability Letters*, 82 (12):2278–2282, 2012.

Székely, G. J. and Rizzo, M. L. Partial distance correlation with methods for dissimilarities. *The Annals of Statistics*, 42(6):2382–2412, 2014.

Székely, G. J., Rizzo, M. L., and Bakirov, N. K. Measuring and testing dependence by correlation of distances. *The Annals of Statistics*, 35(6):2769–2794, 2007.

Tank, A., Covert, I., Foti, N., Shojaie, A., and Fox, E. B. Neural granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4267–4279, 2021.

Tran, T., Do, T., Reid, I. D., and Carneiro, G. Bayesian generative active deep learning. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pp. 6295–6304. PMLR, 2019.

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.

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

Wang, K., Zhang, D., Li, Y., Zhang, R., and Lin, L. Cost-effective active learning for deep image classification. *IEEE Transactions on Circuits and Systems for Video Technology*, 27(12):2591–2600, 2016.

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

Williams, P. L. and Beer, R. D. Nonnegative decomposition of multivariate information. *arXiv preprint arXiv:1004.2515*, 2010.

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

Zhdanov, F. Diverse mini-batch active learning. *arXiv preprint arXiv:1901.05954*, 2019.

Zhen, X., Meng, Z., Chakraborty, R., and Singh, V. On the versatile uses of partial distance correlation in deep learning. *arXiv preprint arXiv:2207.09684*, 2022.

A What are IS and OOS Connections?

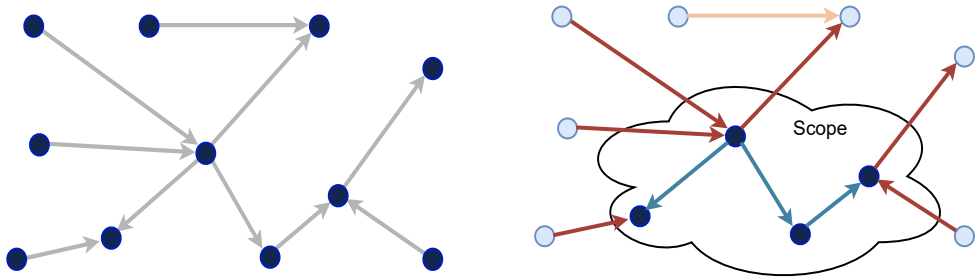

Figure 5. (a) A dynamical system with 11 agents. (b) A scope of 4 agents on the same system.

In Figure 5(a), there is a dynamical system consisting of 11 agents and directed connections. Then suppose we have a scope with 4 agents on the system and is shown as the cloud line in Figure 5(b). Based on the segmentation of the scope, the agents in the system are divided into two groups: IS agents (in original color), and OOS agents (in nattier blue). But the connections are segmented into three groups:(a) IS connections (in blue), (b) OOS connections (in red) and (c) non-observable connections (in cream). During the training of ALaSI, the query strategy with dynamics (Section 4.4) selects the most informative agents and builds a scope upon these agents. If ALaSI operates without the OOS message learning pipeline, at every update of the scope, ALaSI can only learn the representation of the connections within the scope. Yet IS agents may also be affected by OOS agents, thus we need to take the OOS connections into consideration and separate their influence. Therefore, the OOS message learning pipeline expands the learning field of ALaSI to OOS connections, even though one agent of every OOS connection is not observable in the current scope, which significantly promotes the learning efficiency of ALaSI.

B Proofs

B.1 Proof of Proposition 4.1

We prove Proposition 4.1 in this section. Since we assume the independence between $\mathcal{V}^t_{\text{inter}}$ and Z^t_{oos}, based on the definition of mutual information between two independent variables, we can easily get to the first statement:

$$I(\mathcal{V}^t_{\text{inter}}; Z^t_{\text{oos}}) \approx 0. \tag{24}$$

Moreover, from the proposed PI-diagram of information in a target decomposed from three source variables (Lizier et al., 2013), we have the following statement:

$$I(v_i^{t+1}; v_i^t, \mathcal{V}^t_{\text{inter}}, Z^t_{\text{oos}}) > 0. \tag{25}$$

We refer to Figure 3 in (Lizier et al., 2013) and search for the terms related to X^t_{inter} and Z^t_{oos}:

$$I(v_i^{t+1}; \mathcal{V}^t_{\text{inter}}) = \{\mathcal{V}^t_{\text{inter}}\} + \{\mathcal{V}^t_{\text{inter}}\}\{v_i^t, Z^t_{\text{oos}}\} + \{v_i^t\}\{\mathcal{V}^t_{\text{inter}}\}\{Z^t_{\text{oos}}\} + \{\mathcal{V}^t_{\text{inter}}\}\{Z^t_{\text{oos}}\} + \{v_i^t\}\{\mathcal{V}^t_{\text{inter}}\} \gg 0, \tag{26}$$

$$I(v_i^{t+1}; Z^t_{\text{oos}}) = \{Z^t_{\text{oos}}\} + \{Z^t_{\text{oos}}\}\{v_i^t, \mathcal{V}^t_{\text{inter}}\} + \{v_i^t\}\{\mathcal{V}^t_{\text{inter}}\}\{Z^t_{\text{oos}}\} + \{\mathcal{V}^t_{\text{inter}}\}\{Z^t_{\text{oos}}\} + \{v_i^t\}\{Z^t_{\text{oos}}\} \gg 0, \tag{27}$$

where $\{\cdot\}\{\cdot\}$ denotes the redundant information in the two sources, $\{\cdot\}\{\cdot\}\{\cdot\}$ denotes the redundant information in the three sources, $\{\cdot\}$ represents the unique information in the single source, and $\{\cdot\ ,\ \cdot\}$ is the synergistic information from the sources. We summarize the results from Equation 24 to 27, and can derive to:

$$I(\mathcal{V}^t_{\text{inter}}; Z^t_{\text{oos}}) < I(v_i^{t+1}; Z^t_{\text{oos}}), \text{ and } I(\mathcal{V}^t_{\text{inter}}; Z^t_{\text{oos}}) < I(v_i^{t+1}; \mathcal{V}^t_{\text{inter}}), \tag{28}$$

which is Proposition 4.1.

B.2 Derivation of OOS Loss Function

We describe the derivation procedure for Equation 18 in this section. As mentioned in Section 4.2, we can derive the OOS message Z^t_{oos} from v_i^{t+1}, v_i^t and $\mathcal{V}^t_{\text{inter}}$. Based on the triplet loss (Schroff et al., 2015; Schultz & Joachims, 2003) and

Proposition 4.1, we derive the following loss function to learn OOS message:

$$\mathcal{L}_{\text{oos}} = \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[I(\mathcal{V}_{\text{inter}}^{t};Z_{\text{oos}}^{t}) - I(v_i^{t+1};Z_{\text{oos}}^{t}) + \alpha_1 + I(\mathcal{V}_{\text{inter}}^{t};Z_{\text{oos}}^{t}) - I(v_i^{t+1};\mathcal{V}_{\text{inter}}^{t}) + \alpha_2\big], \tag{29}$$

where T represents the total count of time-steps, $\mathcal{D}$ represents the current scope, $|\mathcal{D}|$ denotes the number of agents in the scope, and α_1 and α_2 are margins to regulate the distance between two pairs of mutual information, respectively, in order to encourage larger values of $I(v_i^{t+1};Z_{\text{oos}}^{t})$ and $I(v_i^{t+1};\mathcal{V}_{\text{inter}}^{t})$ compared to $I(\mathcal{V}_{\text{inter}}^{t};Z_{\text{oos}}^{t})$. It is notable that Z_{oos}^{t} and $\mathcal{V}_{\text{inter}}^{t}$ are calculated according to every agent in the scope, respectively. We omit the subscript of Z_{oos}^{t} and $\mathcal{V}_{\text{inter}}^{t}$ for agent i in Equation 29 for concise. Then we can derive:

$$\begin{aligned}
\mathcal{L}_{\text{oos}} =& \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[I(\mathcal{V}_{\text{inter}}^{t};Z_{\text{oos}}^{t}) - I(v_i^{t+1};Z_{\text{oos}}^{t}) + \alpha_1 + I(\mathcal{V}_{\text{inter}}^{t};Z_{\text{oos}}^{t}) - I(v_i^{t+1};\mathcal{V}_{\text{inter}}^{t}) + \alpha_2\big] \\
=& \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[H(Z_{\text{oos}}^{t}) - H(Z_{\text{oos}}^{t}|\mathcal{V}_{\text{inter}}^{t}) - \big(H(Z_{\text{oos}}^{t}) - H(Z_{\text{oos}}^{t}|v_i^{t+1})\big) + \alpha_1 + H(\mathcal{V}_{\text{inter}}^{t}) - H(\mathcal{V}_{\text{inter}}^{t}|Z_{\text{oos}}^{t}) \\
& - \big(H(\mathcal{V}_{\text{inter}}^{t}) - H(\mathcal{V}_{\text{inter}}^{t}|v_i^{t+1})\big) + \alpha_2\big] \\
=& \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[H(Z_{\text{oos}}^{t}|v_i^{t+1}) - H(Z_{\text{oos}}^{t}|\mathcal{V}_{\text{inter}}^{t}) + \alpha_1 + H(\mathcal{V}_{\text{inter}}^{t}|v_i^{t+1}) - H(\mathcal{V}_{\text{inter}}^{t}|Z_{\text{oos}}^{t}) + \alpha_2\big].
\end{aligned}$$

We assume Z_{oos}^{t} and $\mathcal{V}_{\text{inter}}^{t}$ are independent of each other, and we can reformulate the equation as:

$$\begin{aligned}
\mathcal{L}_{\text{oos}} =& \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[H(Z_{\text{oos}}^{t}|v_i^{t+1}) - H(Z_{\text{oos}}^{t}) + \alpha_1 + H(\mathcal{V}_{\text{inter}}^{t}|v_i^{t+1}) - H(\mathcal{V}_{\text{inter}}^{t}) + \alpha_2\big] \\
=& \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[-I(Z_{\text{oos}}^{t};v_i^{t+1}) - I(\mathcal{V}_{\text{inter}}^{t};v_i^{t+1}) + \alpha_1 + \alpha_2\big].
\end{aligned}$$

Since the mutual information between two fixed variables is certain, we omit the second term in the above derivation. Besides that, since the target is minimization, the constant term has no effect on the formulation. As a result, we can obtain:

$$\mathcal{L}_{\text{oos}} = \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}}\big[-I(Z_{\text{oos}}^{t};v_i^{t+1})\big],$$

which is the formulation in Equation 18. As a result, we only need to minimize $-I(Z_{\text{oos}};v_i^{t+1})$, and we can implement it with DeepInfoMax (Hjelm et al., 2019) algorithm. DeepInfoMax maximizes the mutual information between input data and learned high-level representations with the help of global and local information.

C Implementation

C.1 General Settings

We implement ALaSI in PyTorch (Paszke et al., 2019) with the help of Scikit-Learn (Pedregosa et al., 2011) to calculate various metrics. We run experiments of ALaSI on a single NVIDIA Tesla V100 SXM2 graphic card, which has 32 GB graphic memory and 5120 NVIDIA CUDA Cores. We attach our pseudocode and implementation as the supplementary document to this paper. During training, we set batch size as 64 for datasets which have less than 100 agents, for those equal or more than 100 agents, we set batch size as 16. We train our ALaSI model with 500 epochs for each updated label pool on every dataset.

As for baseline methods, since the training under supervised settings only requires the encoder of the model, which demands moderate space, we managed to run the methods on a single NVIDIA Tesla V100 SXM2 graphic card, and the batch sizes are the same as ALaSI. However, when it came to unsupervised learning, the computational requirement of variational auto-encoder-based methods increased significantly. As a result, in order to run these methods on scalable datasets with more than 100 agents, we use "DistributedDataParallel" of PyTorch to enable the parallel training of these models. And we ran these methods on four NVIDIA Tesla V100 SXM2 graphic cards, with a batch size of 128. For the experiments

on datasets with less than 100 agents, we just ran the baselines on a single NVIDIA Tesla V100 SXM2 graphic card with a batch size of 64. For MPIR, since the model is super small and the computational requirement is the smallest among all of the baselines, we ran it on a single NVIDIA Tesla V100 SXM2 graphic card with a batch size of 64. For all of the experiments, we train ALaSI with a learning rate of 0.0005.

C.2 Hyper-parameters

We have the following hyper-parameters: initial sample size m, query size $\mathcal{K}$, number of epochs E, number of selection rounds N, variance σ of $\mathcal{L}_{\text{dc}}$, weights α, β, γ, ξ in hybrid loss, and proportion of rank in PID η. We utilized grid search for the rough values of these hyper-parameters, and show them in Table 4. We reported the choice of parameters based on the values that can match all of the loss terms into the same scale. And even based on these easy searches, ALaSI managed to outperform other baseline methods. We think that it is feasible to tune these parameters with the help of Bayesian Optimization packages, such as HEBO (Cowen-Rivers et al., 2022) and TuRBO (Eriksson et al., 2019).

Table 4. Hyper parameter choices for every dataset.

DATASET	m	$\mathcal{K}$	E	N	σ	α	β	γ	ξ	η
Springs50	5	0.10	500	12	0.0008	0.05	0.8	20	2	0.3
Springs100	5	0.05	500	15	0.0008	0.02	0.8	30	2	0.3
Springs200	10	0.04	500	20	0.0008	0.02	0.5	20	3	0.2
Springs500	20	0.02	600	30	0.0008	0.02	0.6	40	3	0.2
ESC	5	0.05	500	20	0.0008	0.02	0.5	50	2	0.2
E. coli	20	0.02	600	50	0.0008	0.01	0.4	40	3	0.3
S. aureus	20	0.02	600	50	0.0008	0.01	0.4	20	3	0.3

C.3 Details of Pipelines

In this section, we first demonstrate the general pipeline of ALaSI in Algorithm 2. Then we show the description of PID algorithm in ALaSI in Algorithm 3, which is followed by the implementation of ALaSI in Algorithm 4.

C.4 Details of Loss Function

In this section, we discuss and state the details of loss terms and the implementation details of the proposed loss terms in hybrid loss (Equation 22).

C.4.1. OOS Loss

In this section, we describe the implementation of OOS loss function (Equation 18). As shown in Section B.2, the loss function is simplified as the maximization of mutual information between Z^t_{oos} and v_i^{t+1} for all $0 \leq t \leq T-1$, and for all agent i in the current scope. As mentioned in Section 4.3, we leverage DeepInfoMax (Hjelm et al., 2019) to maximize $I(Z^t_{\text{oos}}, v_i^{t+1})$. We follow the implementation of DeepInfoMax at: `https://github.com/DuaneNielsen/DeepInfomaxPytorch`, which is a pytorch version of official implementation at `https://github.com/rdevon/DIM`. Interestingly, DeepInfoMax requires output variables, input variables and also the negative samples of input variables. As a result, besides Z^t_{oos} and v_i^{t+1}, we also feed $\mathcal{V}^t_{\text{inter}}$ to DeepInfoMax, as the negative samples.

C.4.2. Distance Correlation

In this section, we firstly describe the procedures to calculate distance correlation $\mathcal{L}_{\text{dc}}$ in Equation 19, then we describe the implementation of distance correlation in our work.

Procedures. We firstly pair the K samples of Z^t_{oos} and $\mathcal{V}^t_{\text{inter}}$ as pairs: $(z_p, x_p)_{p\in K}$. Then we calculate the distance matrices $\mathbf{A}, \mathbf{B} \in \mathbb{R}^{K\times K}$ as:

$$\mathbf{A}_{pq} = ||z_p - z_q||_F, \text{ and } \mathbf{B}_{pq} = ||x_p - x_q||_F,\ p, q = 1, ..., K.$$

Algorithm 2 Pipeline of ALaSI.

Input: $\mathcal{D}_{\text{train}}$ = a pool of labeled trajectories $\{\mathcal{V}_{\text{train}}, \mathcal{E}\}$,
Input: $\mathcal{D}_{\text{pool}}$ = a pool of test trajectories $\{\mathcal{V}_{\text{pool}}\}$,
Parameters: initial sample size m, query size $\mathcal{K}$, number of epochs E, number of selection rounds N,
Model Weights: θ, **Hybrid Loss:** $\mathcal{R}$, **Query with Dynamics:** $\mathcal{Q}$,
Output: Trained Active Structural Inference Model $\mathcal{M}$,
if Supervised learning **then**
  Set of data points $\mathcal{D} \leftarrow$ Select m agents with features $\mathcal{V}_m$ and connectivity $\mathcal{E}_m$ from $\mathcal{D}_{\text{train}}$,
else
  Select m agents with features $\mathcal{V}_m$ from $\mathcal{D}_{\text{train}}$,
  Run PID on $\mathcal{V}_m$ and obtain connections between m nodes: $\mathcal{E}_{PID0}$,
  Set of data points $\mathcal{D} \leftarrow \{\mathcal{V}_m, \mathcal{E}_{PID0}\}$,
end if
Train model $\mathcal{M}$ E epochs with loss $\mathcal{R}$ on $\mathcal{D}$ and obtain parameters θ_0,
Query $\mathcal{K}$ agents with the strategy of query with dynamics $\mathcal{Q}(\theta_0, \{\mathcal{V}_{\text{pool}}\}, \mathcal{E})$,
if Supervised learning **then**
  Update $\mathcal{D}$ with $\mathcal{K}$ agents with features $\mathcal{V}_\mathcal{K}$ and connectivity $\mathcal{E}_\mathcal{K}$ from $\mathcal{D}_{\text{train}}$,
else
  Select m agents with features $\mathcal{V}_m$ from $\mathcal{D}_{\text{train}}$,
  Run PID on features $\mathcal{V}_\mathcal{K}$ and obtain connections between $\mathcal{K}$ nodes: $\mathcal{E}_{PIDK}$,
  Update $\mathcal{D} \leftarrow \{\mathcal{V}_\mathcal{K}, \mathcal{E}_{PIDK}\}$,
end if
while Round $i < N$ **do**
  Train model $\mathcal{M}$ E epochs with loss $\mathcal{R}$ on $\mathcal{D}$ and obtain parameters θ_i,
  Query agent features with $\mathcal{Q}(\theta_i, \{\mathcal{V}_{\text{pool}}\}, \mathcal{E})$ and choose $\mathcal{K}$ agents,
  if Supervised learning **then**
    Update $\mathcal{D}$ with $\mathcal{K}$ agents with features $\mathcal{V}_\mathcal{K}$ and connectivity $\mathcal{V}_\mathcal{K}$ from $\mathcal{D}_{\text{train}}$,
  else
    Select m agents with features $\mathcal{V}_m$ from $\mathcal{D}_{\text{train}}$,
    Run PID on features $\mathcal{V}_\mathcal{K}$ and obtain connections between $\mathcal{K}$ nodes: $\mathcal{E}_{PIDK}$,
    Update $\mathcal{D} \leftarrow \{\mathcal{V}_\mathcal{K}, \mathcal{E}_{PIDK}\}$,
  end if
end while
Return: trained model $\mathcal{M}$ and parameters θ.

Algorithm 3 PID Algorithm in ALaSI.

Input: $\{\mathcal{V}_{\text{pool}}\}$ = a pool of trajectories of p agents,
Parameters: Rank or proportion of rank: ξ, Total number of time steps of features: T,
Output: $\mathcal{D}_{\text{train}}$ = a pool of labeled trajectories $\{\mathcal{V}_{\text{pool}}, \mathcal{E}\}$,
for agent i in total p agents **do**
  for agent j in $p-1$ agents **do**
    for agent r in $p-2$ agents **do**
      Compute the unique component I_{Uni} between $X_i^{1:T-1}$ and $X_j^{2:T}$ given $X_r^{2:T}$,
      Compute the mutual information I between $X_i^{1:T-1}$ and $X_j^{2:T}$ given $X_r^{2:T}$,
      Compute the ratio q_r between the I_{Uni} and I,
    end for
    Calculate the sum of q_r over all agents r as q_{ij},
  end for
end for
Rank all q_{ij}, and select ξ (or $\xi \cdot p$) agent-pairs with highest q_{ij},
Mark the connections from i to j in these pairs as exist, the rest as non-exist,
Return: the connectivity between p agents.

After that, we double center the distance matrices to get $\tilde{\mathbf{A}}_{pq}$, $\tilde{\mathbf{B}}_{pq}$:

$$\tilde{\mathbf{A}}_{pq} = \mathbf{A}_{pq} - \bar{\mathbf{A}}_{p\cdot} - \bar{\mathbf{A}}_{\cdot q} + \bar{\mathbf{A}}_{\cdot\cdot},$$

where $\bar{\mathbf{A}}_{i\cdot}$ denoted the mean of row i, $\bar{\mathbf{A}}_{\cdot j}$ denotes the mean of column j, $\bar{\mathbf{A}}_{\cdot\cdot}$ denotes the overall mean of $\mathbf{A}$. So this centers both the rows and columns of A, B. All rows and columns of $\tilde{\mathbf{A}}$ and $\tilde{\mathbf{B}}$ sum to 0. In short notation:

$$\tilde{\mathbf{A}}_{qm} = (I - M)\mathbf{A}(I - M), \text{and } \tilde{\mathbf{B}}_{qm} = (I - M)\mathbf{A}(I - M),$$

where $M = \frac{1}{K}\mathbf{1}\mathbf{1}^T$. The distance covariance of Z^t_{oos} and $\mathcal{V}^t_{\text{inter}}$ is defined as the square root of:

$$\text{dcov}^2(Z^t_{\text{oos}}, \mathcal{V}^t_{\text{inter}}) = \frac{1}{K^2}\sum_{p,q=1}^{K} \tilde{\mathbf{A}}_{pq}\tilde{\mathbf{B}}_{pq}.$$

And the distance variance is defined as: $\text{dvar}^2(x) = \text{dcov}^2(x, x)$. Thus we can calculate the distance correlation with:

$$\mathcal{L}_{\text{dc}} = \frac{1}{(T-1)\cdot|\mathcal{D}|}\sum_{t}^{T-1}\sum_{i\in\mathcal{D}} \frac{\text{dCov}^2(Z^t_{\text{oos}}, \mathcal{V}^t_{\text{inter}})}{\sqrt{\text{dVar}(Z^t_{\text{oos}})\text{dVar}(\mathcal{V}^t_{\text{inter}})}},$$

which is the same formulation as Equation 19.

Implementation. As for the implementation of distance correlation, we originally follow the official implementation of distance correlation implementation of (Zhen et al., 2022) at `https://github.com/zhenxingjian/Partial_Distance_Correlation`. We then extend the implementation to suit batch-wise calculation and GPU acceleration.

C.4.3. Discussion on the Loss Terms

In this section, we would like to discuss the importance of different terms in the hybrid loss mentioned in Section 4.3. The hybrid loss consists of four terms, $\mathcal{L}_{oos}$: to learn OOS messages, $\mathcal{L}_{dc}$: to ensure the independence assumption of learning OOS messages, $\mathcal{L}_D$: loss function for dynamics, and $\mathcal{L}_{con}$: loss function for connectivity. Among the four terms, $\mathcal{L}_{oos}$ and $\mathcal{L}_{dc}$ should appear in pairs to learn OOS messages (as stated and proved in Section B.2). $\mathcal{L}_D$ and $\mathcal{L}_{con}$ are the very important terms to make ALaSI work (terms for AL training), which cannot be discarded. Therefore, we conducted ablation studies to check the importance of terms for OOS message learning, and presented the results in Section D.3. In the ablation studies, we state "ALaSI-no OOS" as the one without $\mathcal{L}_{oos}$ and $\mathcal{L}_{dc}$ by setting γ and η as zero. And Figure 9 clearly shows the importance of $\mathcal{L}_{oos}$ and $\mathcal{L}_{dc}$. Without these two terms, the algorithm can only learn about the representations of connections within the scope and cannot extrapolate onto OOS connections, which results in an almost linear dependence between AUROC and the proportion of labeled connections.

C.5 Implementation of Pipelines

We first briefly describe the pipeline of learning of ALaSI in Algorithm 4. We then describe the components of several networks mentioned in Algorithm 4. The design of f_{inter1}, f_{inter2}, f_{oos1}, f_{oos2} and f_{dynamics} follows modular-design practice and are based on a multi-layer-perceptron shown in Algorithm 5. We name the functional pipeline shown in Algorithm 5 as MLP, and we can represent the networks in Algorithm 4 as: $f_{\text{inter1}} = MLP(MLP(\cdot))$, $f_{\text{inter2}} = MLP(\cdot)$, $f_{\text{oos1}} = MLP(\cdot)$, $f_{\text{oos2}} = MLP(\cdot)$ and $f_{\text{dynamics}} = MLP(\cdot)$. We briefly report the dimension of the layers of each networks in Table 5, where $f^{'}_{\text{inter1}}$ represents the second $MLP(\cdot)$ of f_{inter1}, $f^{'}_{\text{inter1}}$ represents the first $MLP(\cdot)$ of f_{inter1}, x_{dim} is the number of dimensions of an agent at a time step, and $|T|$ represents the total time steps of the trajectory.

C.6 Implementation of Baselines

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

MPM. We use the official implementation code by the author from `https://github.com/hilbert9221/NRI-MPM`

Algorithm 4 Pipeline of learning in ALaSI.

Input: $\mathcal{V}$ = set of agent features of current scope,
Input: n = number of agents in the current scope,
Input: Z_{gt} = ground truth connectivity in the current scope,
Connection Learning Network: f_{inter1}, IS Message Network: f_{inter2}, OOS Embedding Network: f_{oos1}, OOS Message Network: f_{oos2}, Dynamics Learning Network: f_{dynamics}, Output Function: f_{output}, DeepInfoMax: f_{DIM}
Split agent features according to time steps: $\mathcal{V}_\tau = \mathcal{V}^{0:T-1}$ for training, $\mathcal{V}_\psi = \mathcal{V}^{1:T}$ for loss calculation, where T represents the total time steps,
Learn representation of connections: $Z = f_{\text{inter1}}(\mathcal{V}_\tau, n)$,
Summarize connectivity inside the scope: $\hat{Z} = GumbelSoftmax(Z)$,
Learn inter scope messages: $e_{\text{inter}} = f_{\text{inter2}}(\mathcal{V}_\tau, \hat{Z})$,
Learn OOS messages: $e_{\text{oos}} = f_{\text{oos2}}(f_{\text{oos1}}(\mathcal{V}_\tau, e_{\text{inter}}))$,
Learn dynamics: $e_{\text{out}} = f_{\text{output}}(f_{\text{dynamics}}(e_{\text{inter}}, e_{\text{oos}}), \mathcal{V}_\tau)$,
Calculate OOS loss with DeepInfoMax: $\mathcal{L}_{\text{OOS}} = f_{\text{DIM}}(e_{\text{inter}}, e_{\text{oos}}, \mathcal{V}_\tau)$,
Calculate distance correlations: $\mathcal{L}_{\text{dc}}$ from e_{oos} and $\mathcal{V}_\tau$ for each agent in the scope,
Calculate dynamics prediction loss: $\mathcal{L}_{\text{D}} \leftarrow \{e_{\text{out}}, \mathcal{V}_\psi\}$,
Calculate connectivity loss: $\mathcal{L}_{\text{con}} \leftarrow \{\hat{Z}, Z_{\text{gt}}\}$,
Summarize as the hybrid loss: $\mathcal{R} \leftarrow \{\mathcal{L}_{\text{D}}, \mathcal{L}_{\text{con}}, \mathcal{L}_{\text{OOS}}, \mathcal{L}_{\text{dc}}\}$,
Update parameters with back-propagation,
Return: trained model.

Algorithm 5 The Multi-layer-perceptron.

Input: features $input$
x = elu(Linear1($input$))
x = dropout(x)
x = elu(Linear2(x))
out = batch_norm(x)
Return: out

with customized data loader for our chosen datasets. We add our metric-evaluation for AUROC in "evaluate()" function of class "XNRIDECIns" in the original code.

Table 5. Dimension of the layers and dropout rates.

Parameters	f''_{inter1}	f'_{inter1}	f_{inter2}	f_{oos1}	f_{oos2}	f_{dynamics}
Linear1	$2 \cdot x_{dim} \cdot \lvert T \rvert$	256	$2 \cdot x_{dim}$	$(x_{dim} + 256) * (\lvert T \rvert - 1)$	$2 \cdot x_{dim}$	256
Dropout	0.0	0.0	0.0	0.5	0.0	0.0
Linear2	256	2	256	$x_{dim} * (\lvert T \rvert - 1)$	256	256

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 customized data loader for our datasets. We implement the metric-calculation pipeline in the "forward_pass_and_eval()" function.

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

PID. Based on the Julia implementation of PID in `https://github.com/Tchanders/InformationMeasures.jl`, we implement PID in Python. Then we implement the mutual information calculation of PID with KDTree (see `https://github.com/paulbrodersen/entropy_estimators`), in order to enable PID to operate on continuous high-dimensional data. Different from other methods, we run PID on all of the dataset we have in experiments. For instance, when running experiments on "Springs50", PID infer the connections of the entire dynamical system based on a union set of the trajectories for training, validation and testing.

C.7 Further Details about Datasets

Spring Datasets To generate these springs datasets ("Springs50", "Springs100", "Springs200", and "Springs500"), we follow the description of the data in (Kipf et al., 2018) but with fixed connections. To be specific, at the beginning of the data generation for each springs dataset, we randomly generate a ground truth graph and then simulate 12000 trajectories on the same ground truth graph, but with different initial conditions. The rest settings are the same as that mentioned in (Kipf et al., 2018). We collect the trajectories and randomly group them into three sets for training, validation and testing with the ratio of 8: 2: 2, respectively.

GRN Datasets Different from springs datasets, GRN datasets (ESC, E. coli, and S. aureus) are sampled from publicly available data sources. We download the datasets from the links mentioned in the corresponding literature, sample the trajectories with the same amount of time steps as of springs datasets, and randomly group the trajectories of gene expressions into three sets for training, validation and testing with the ratio of 8: 2: 2, respectively.

D Further Experimental Results

In this section, we demonstrate additional experimental results as the supplement to Section 5.

D.1 Integration of Prior Knowledge with Unsupervised Learning

We conduct the integration of prior knowledge with unsupervised learning with ALaSI. At the beginning of every experiment, we randomly assign a portion of agents with true connectivity, and keep the remaining settings the same as those in Section 5.2. During a query, if the agents with true connectivity are selected and the connections of these agents assigned by PID are contrary to the true label, we set the connectivity the same as the label and maintain the connections of the rest agents. We summarize the results and plot them in Figure 6, where we plot the AUROC results of fully supervised

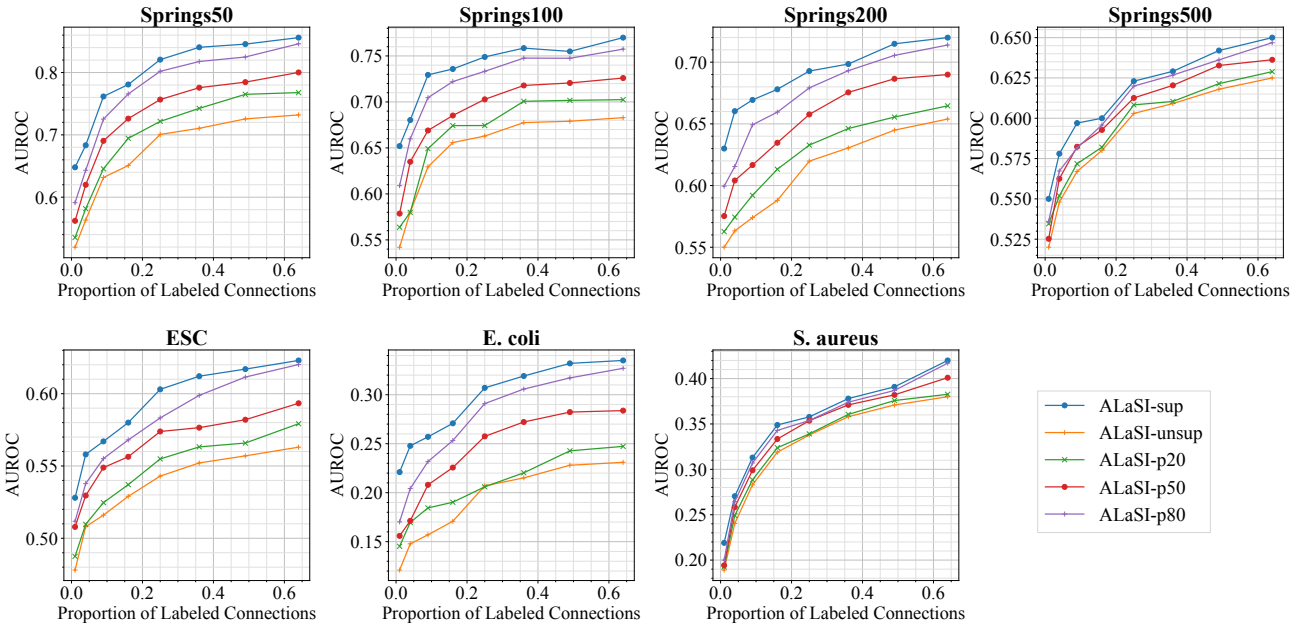

Figure 6. Averaged AUROC results of ALaSI-sup, ALaSI-unsup, ALaSI-p20, ALaSI-p50, and ALaSI-p80 as a function of the proportion of labeled connections on "Springs50", "Springs100", "Springs200", "Springs500", ESC, E. coli and S. aureus datasets.

ALaSI (ALaSI-sup), fully unsupervised ALaSI (ALaSI-unsup), and unsupervised ALaSI with 20%, 50% and 80% of prior knowledge on agents (ALaSI-p20, ALaSI-p50 and ALaSI-p80). As we can observe from the plots, ALaSI is capable of being integrated with prior knowledge, and the AUROC value is positively correlated with the proportion of integrated prior knowledge. Interestingly, ALaSI-p80 moves generally closer to the fully supervised ALaSI, which on the other hand verifies the data efficiency of ALaSI. ALaSI has the capability of accurately inferring accurate connectivity of dynamical systems with less prior knowledge. In comparison, we also tested the integration of prior knowledge with baseline methods that uses

VAE under unsupervised settings, but surprisingly we observed performance drops in terms of AUROC. We think the reason might be the integration of prior knowledge happened in the latent space, violating the generation process of these methods. We leave the study of these performance drops to future work.

D.2 Robustness Tests of ALaSI

Although ALaSI is tested on several real-world datasets and the results are reported in Sections 5.1 and 5.2, it is interesting to carry out more experiments to further test the robustness of ALaSI. We generate a series of "Springs50" datasets with different levels of Gaussian noise. The Gaussian noise is added to the features of the agents and the levels Δ amplify the noise as follows:

$$\tilde{v}_i^t = v_i^t + \zeta \cdot 0.02 \cdot \Delta, \text{ where } \zeta \sim \mathcal{N}(0, 1), \tag{30}$$

where v_i^t represents raw feature vector of agent i at time t. And we plot the experimental results of ALaSI on these datasets in Figure 7. As shown in Figure 7, noises in the agents' features have an effect on the performance of ALaSI. The effect

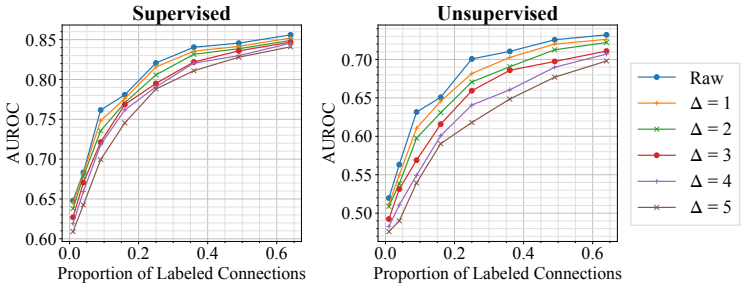

Figure 7. Averaged AUROC results of ALaSI as a function of the proportion of labeled connections on "Springs50" dataset with different levels of noise under supervised or unsupervised setting.

is minor when ALaSI is trained in supervised setting. But under unsupervised setting, especially when the proportion of labeled connections in the pool is smaller than 0.4, ALaSI faces a bigger challenge to infer the connections compared with it when under supervised setting. When the proportion of labeled connections increases, the effect of noises becomes smaller and smaller. So in summary, although noises have a negative impact on the performance, ALaSI still can infer the connections with moderate to high accuracy. Besides that, we also test the baseline methods on the dataset of "Springs50" with different levels of Gaussian noise, and plot the results in Figure 8. Each subplot in Figure 8 reports the performance of ALaSI and baseline methods on the "Springs50" dataset with a certain noise level, respectively. As we can learn from the figure, although the baseline methods are trained under supervised settings, compared to ALaSI, they are more sensitive to noises. The margins between the AUROC results of ALaSI and the best baseline methods become larger when the noise level increases. We think the reason may come from the baseline methods utilizing a full-sized computational graph, so during training, all of the connections within the system are learned simultaneously. Therefore, a high level of noise leads to an enormous uncertainty in the loss functions of these methods (their loss functions are summations of errors of all the connections in the system). Different from baseline methods, ALaSI learns the connections agent-wise, which eases the uncertainty in the loss function. Besides that, the query with dynamics can correctly select the most informative agent to be added to the scope, regardless of the noise level. We think a combination of these two functioning mechanisms helps ALaSI to reduce the uncertainty created by noisy data.

D.3 Ablation Study

We conduct ablation studies on the effectiveness of query with dynamics, as well as OOS operation. We modify ALaSI into (a) ALaSI-ran: where we replace the query with dynamics strategy with a random sampling strategy on agents; and (b) ALaSI-no OOS: where we remove the pipeline for OOS representation learning and the corresponding terms in the loss function. We report the results of unsupervised learning, which we believe is closer to real-world scenarios, and report the averaged AUROC results of these variants as a function of the proportion of labeled connections by PID.

As shown in Figure 9, ALaSI with query strategy with dynamics and OOS operation outperforms its variants, ALaSI-random and ALaSI-no OOS. Despite the inference accuracy of all these methods increasing when a large portion of agents are labeled, we observe that ALaSI converges much faster than the other two methods. Besides that, OOS operation is of

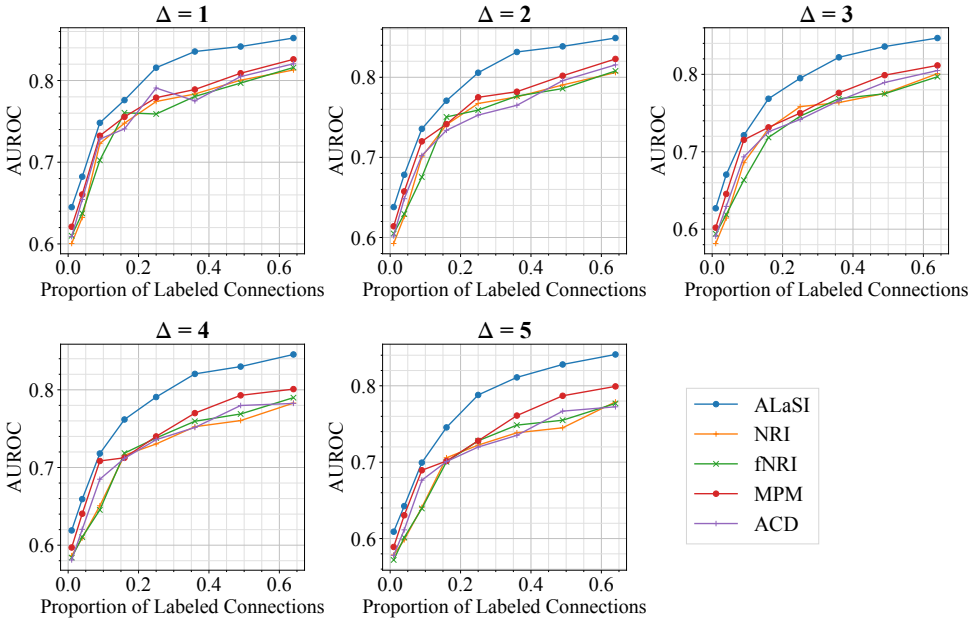

Figure 8. Averaged AUROC results of ALaSI and baseline methods as a function of the proportion of labeled connections on "Springs50" dataset with different levels of noise under supervised setting.

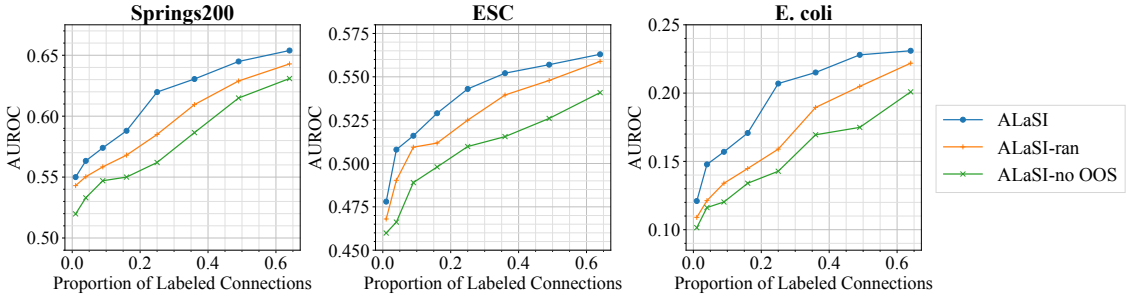

Figure 9. Averaged AUROC results of ALaSI, ALaSI-ran and ALaSI-no OOS as a function of the proportion of labeled connections under unsupervised learning.

great importance to the design of a scalable structural inference method. It is commonly observed among the subplots that ALaSI-no OOS can only learn about the representations of connections within the scope and cannot extrapolate onto OOS connections, which results in an almost linear dependence between AUROC and the proportion of labeled connections. Therefore, the query strategy with dynamics and the OOS operation of ALaSI effectively encourage faster convergence under unsupervised settings.

E Limitation of ALaSI

Besides the datasets mentioned in this work, we also test ALaSI on the physic simulation datasets mentioned in NRI (Kipf et al., 2018). Most of the physic simulation datasets have no more than 10 agents in the system, which are much smaller than the ones used in this work. Based on the experiments on these datasets, ALaSI cannot outperform baseline methods when the size of the dynamical system is small. Since ALaSI works on agent-wise selection to build the pool for training, when the total count of agents is small, ALaSI cannot benefit from the mechanism of active learning. Moreover, if there exist multiple types of connections in the dynamical system, we doubt whether ALaSI can be qualified as the structural inference method for this kind of system. We think it is possible to extend the application scenario of ALaSI to these systems with a built-in multiplex graph, and we leave this for future work.

F Broader Impact

ALaSI allows researchers in the field of network science, biology and physics to study the underlying interacting structure of large dynamical systems, which is the first algorithm targeting the structural inference of large systems. We have shown that ALaSI has outstanding performance facing large dynamical systems even with additive Gaussian noise, which proves its broad application scenarios. While the emergence of structural inference technology for large systems may be helpful for many, it can be potentially misused either. For example, it can be likely to be used to reveal private anonymous connections which could erode privacy and anonymity.

G Ethics Statement

ALaSI is a framework for structural inference of dynamical systems. No matter how effective it is at this task, there may still be failure modes ALaSI will not catch. So far in this work, we haven't seen any issue with ethics.

H Reproducibility

We will make the implementation public on GitHub. We will include the code of ALaSI, and the procedures for accessing the dataset we used in this work. Please refer to it as the implementation of ALaSI.