# Learning Heterogeneous Interaction Strengths by Trajectory Prediction with Graph Neural Network

**Seungwoong Ha**
KAIST
Daejeon 34141, Republic of Korea
skyround2002@gmail.com

**Hawoong Jeong**
KAIST
Daejeon 34141, Republic of Korea
hjeong@kaist.ac.kr

## Abstract

Dynamical systems with interacting agents are universal in nature, commonly modeled by a graph of relationships between their constituents. Recently, various works have been presented to tackle the problem of inferring those relationships from the system trajectories via deep neural networks, but most of the studies assume binary or discrete types of interactions for simplicity. In the real world, the interaction kernels often involve continuous interaction strengths, which cannot be accurately approximated by discrete relations. In this work, we propose the relational attentive inference network (RAIN) to infer continuously weighted interaction graphs without any ground-truth interaction strengths. Our model employs a novel pairwise attention (PA) mechanism to refine the trajectory representations and a graph transformer to extract heterogeneous interaction weights for each pair of agents. We show that our RAIN model with the PA mechanism accurately infers continuous interaction strengths for simulated physical systems in an unsupervised manner. Further, RAIN with PA successfully predicts trajectories from motion capture data with an interpretable interaction graph, demonstrating the virtue of modeling unknown dynamics with continuous weights.

## 1 Introduction

Dynamical systems with interactions provide a fundamental model for a myriad of academic fields, yet finding out the form and strength of interactions remains an open problem due to its inherent degeneracy and complexity. Although it is crucial to identify the interaction graph of a complex system for understanding its dynamics, disentangling individual interactions from trajectory data without any ground-truth labels is a notoriously hard inverse problem.

There has been a long history and substantial amount of work on both inferring the network topology and the nonlinear interactions between interacting constituents from data [3, 5, 4, 22, 10, 8], along with the development of various measures to capture the relation between constituents (e.g., Pearson correlations, mutual information, transfer entropy, Granger causality, and variants thereof [21]). Many of these inferences focus on specific systems with the necessity for a model prior, such as domain knowledge of the agent characteristics, proper basis construction, and detailed assumptions on the system dynamics.

Recently, by phenomenal advances in machine learning, adopting a neural network as a key component of the interaction inference has gained attention from researchers [24, 14, 26]. The key strength of these approaches comes from the fact that a neural network enables relatively free-form modeling of the system. One influential work in this direction, neural relational inference (NRI) [14], explicitly infers edges by predicting the future trajectories of the given system. But previous studies for extracting interaction graphs with neural networks [14, 25, 9] mainly focused on inferring edges with discrete edge types, which means that they are incapable of distinguishing the edges of different

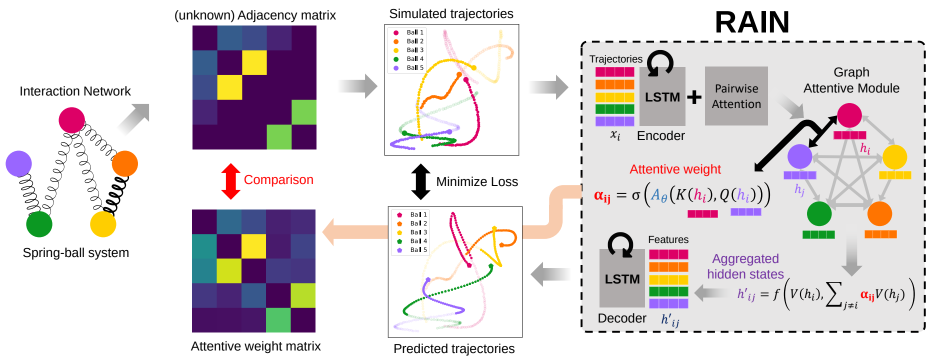

Figure 1: Overview of system formulation and RAIN architecture. RAIN encodes each agent's trajectory with an LSTM encoder and applies pairwise attention (PA) to the hidden states for constructing a pair of embeddings for each agent pair. Then the graph attentive module extracts the interaction strength from a pair of embeddings in the form of an attention weight. The decoder module finally predicts the future trajectories of each agent with an LSTM decoder, but here, each prediction can only employ the weighted information from other agents. This restriction on information induces the attention weights in the learning process to properly reflect the strengths of the connections.

interaction strengths with the same type. Considering the common occurrence of such heterogeneous interaction strengths throughout diverse systems, the assumption of discrete edge types severely limits the expressibility of the model.

A weighted interaction graph can be expressed in the form of a connectivity matrix, which is a conventional adjacency matrix with continuous-valued entries that indicate the interaction strengths. One recent study [19] tackles this problem of inferring the connectivity matrix by first training the neural network and then additionally applying a 'graph translator' to extract continuous graph properties, which needs a ground-truth label to train. Also, various attempts for continuous strength inference from machine learning researchers [11, 12, 2] often lack verifications through the dataset with ground-truth interaction strength. Several studies from physicists [27, 17] employed perturbation analysis and response dynamics to infer continuous interaction strengths, and reported a good agreement with the ground-truth strength. However, these correlation-based methods usually require more than $10^5$ to $10^7$ data points for the inference, which is a feasible size for an entire dataset but not for a single instance, and thus the direct application to experimental data is difficult.

In the current work, we propose a neural network called Relational Attentive Inference Network (RAIN) to address the problem of inferring weighted interaction graphs from multivariate trajectory data in an unsupervised manner. RAIN infers the interaction strength between two agents from previous trajectories by learning the attentive weight while simultaneously learning the unknown dynamics of the system and thus is able to precisely predict the future trajectories. Our model employs the attention mechanism twice: once for the construction of pairwise trajectory embedding and once for the actual graph weight extraction. Differing from previous approaches such as the graph attention network (GAT) [24], RAIN aims to infer the absolute interaction strength that governs the system dynamics. By comparing the inferred interaction strengths of simulated physical systems with ground-truth values that are not provided at the training stage, we verify that RAIN is capable of inferring both system dynamics and weighted interaction graphs solely from multivariate data. We further show that RAIN outperforms discrete baselines on real-world motion capture data, representing a system in which we cannot be certain whether a continuous form of interaction strengths even exists. In this way, we demonstrate that the rich flexibility and expressibility of the continuous modeling of interaction strengths are crucial for the accurate prediction of the future dynamics of an unknown empirical system.

## 2 Model description

Our RAIN model as shown in Fig. 1 comprises three parts trained jointly: an encoder that compresses time series data, a graph extraction module that infers the interaction weight between every pair of agents, and a decoder that predicts the future trajectories of each agent. Note that RAIN does not

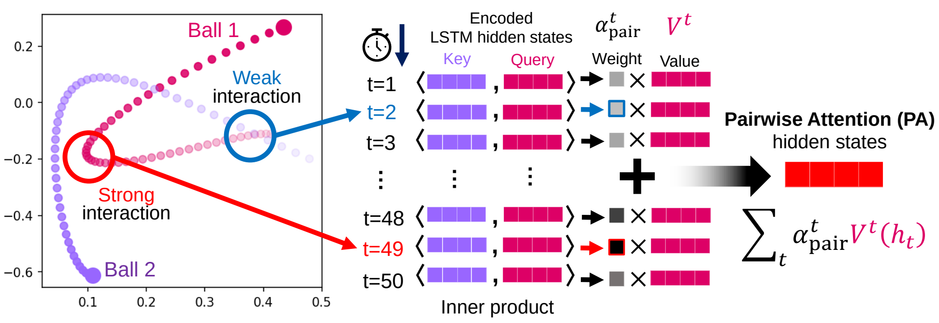

Figure 2: Overview of the PA mechanism, which facilitates effective interaction inference by emphasizing the critical part of the time series. Here, two balls weakly interacted at first but showed strong interaction at the end. The circled points at timestep 2 and 49 are emphasized as exemplary timestep where the interaction was weak and strong, respectively. PA achieves refined trajectory representations by comparing encoded LSTM hidden states at the same time. Due to the asymmetric nature of the transformer architecture, the weighted hidden state for A to B and B to A can be drastically different, which lets PA properly handle the directional asymmetry of interaction.

require a ground-truth interaction graph for training; instead, it produces an interaction graph as a byproduct of future trajectory prediction. In the following, we formalize our model and describe each component in detail.

## 2.1 Encoder and pairwise attention

The long short-term memory (LSTM) encoder is composed of a single layer of gated recurrent unit (GRU) [6] with hidden state dimension $d_{\text{lstm}} = 128$. The encoder receives $T_{\text{enc}}$ steps of trajectories $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^{T_{\text{enc}}}$, each consisting of $R$ state variables of $N$ agents, and produces corresponding $T_{\text{enc}}$ hidden states, $\mathbf{h}^1, \mathbf{h}^2, \ldots, \mathbf{h}^{T_{\text{enc}}}$. The last hidden state $\mathbf{h}^{T_{\text{enc}}}$ could preserve the information from the entire trajectory in theory. But we found that a naive final hidden state is insufficient to fully capture the interaction strength in large-scale systems. In this study, we propose a *pairwise attention* (PA) mechanism to effectively infer the interaction strength between two trajectories. The intuition behind PA is straightforward: a single hidden state cannot be a suitable choice for extracting the interaction strengths from *every* possible pair. Thus, we calculate the attention between same-time hidden states to assign weights to their contribution, as depicted in Fig. 2. We use a slightly modified transformer [23] with $m = 4$ heads and a head dimension of $d_{\text{h}} = d_{\text{lstm}/m=32}$ to emphasize strong interaction. See [23] for a detailed description of the transformer architecture. Formally, the LSTM hidden states are processed into Key$(K)$, Query$(Q)$, and Value$(V)$ matrices as follows,

$$\mathbf{h}^t_i = f_{\text{LSTM}}(\mathbf{h}^{t-1}_i, \mathbf{x}^t_i) \tag{1}$$

$$X^t_{\text{pair},i} = f_{\text{pair},X}(\mathbf{h}^t_i) \quad \text{where } X \in \{K, Q, V\} \tag{2}$$

$$X^t_{\text{pair},i} = X^{t,1}_{\text{pair},i} \oplus X^{t,2}_{\text{pair}i} \oplus \ldots \oplus X^{t,m}_{\text{pair},i}, \tag{3}$$

where the symbol $\oplus$ indicates the concatenation of the matrices at the last dimension, $f_{\text{LSTM}}$ is the GRU layer, $\mathbf{x}^t_i$ are the state variables of the $i$th agent at time $t$, and $f_{\text{pair},X}$ is a stack of multilayer perceptron (MLP) layers for $K_{\text{pair}}$, $Q_{\text{pair}}$, and $V_{\text{pair}}$. The superscripts $t$ and $m$ on $X$ indicate the time and head number, respectively. The attention-weighted hidden state $\tilde{\mathbf{h}}$ is expressed as follows,

$$\alpha^{t,m}_{\text{pair},ij} = \text{softmax}(K^{t,m}_{\text{pair},i}(Q^{t,m}_{\text{pair},j})^T / \sqrt{d_{\text{h}}}) \tag{4}$$

$$\tilde{\mathbf{h}}^m_i = \sum_{t=1}^{T_{\text{enc}}} \alpha^{t,m}_{\text{pair},ij} V^{t,m}_i \tag{5}$$

$$\tilde{\mathbf{h}}_i = \tilde{\mathbf{h}}^1_i \oplus \tilde{\mathbf{h}}^2_i \oplus \ldots \oplus \tilde{\mathbf{h}}^m_i, \tag{6}$$

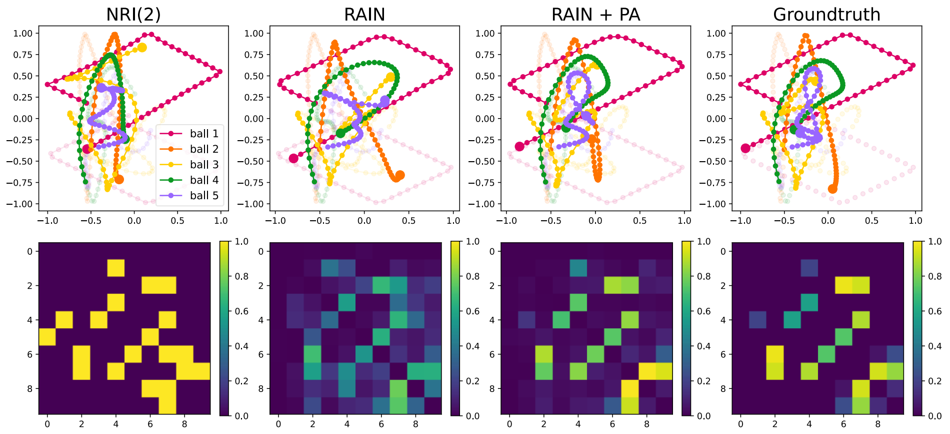

Figure 3: Visualization of the trajectory predictions (upper) and retrieved connectivity matrices (lower) for the spring-ball system. The results from left to right are from NRI(2), RAIN without PA, RAIN with PA, and ground-truth. Here, 5 out of 10 balls are drawn for trajectory visualization, and semi-transparent paths indicate the first 50 steps of input trajectories while solid paths denote 50 steps of predicted future trajectories.

where $\alpha^{t,m}_{\text{pair},ij}$ is an attentive weight between agents $i$ and $j$, and $\tilde{\mathbf{h}}^m_i$ is a weighted hidden state for the $m$th attention head. RAIN passes $\tilde{\mathbf{h}}$ to the graph extraction module.

By employing the PA mechanism, we can refine the LSTM hidden states by focusing on the specific time of interactions, which is unique to each agent pair. We highlight that comparing the same-time hidden states only, rather than considering the full attention as in conventional settings, significantly reduces the time complexity of the inference while achieving the goal of the extraction of temporal information. Also, since the transformer can handle asymmetric relationships by differentiating key and query, PA can capture directional connections with ease.

### 2.2 Graph extraction module

The graph extraction module of RAIN is similar to the GAT [24]. The difference between ours and the original GAT is that RAIN has no prior knowledge of the underlying graph and thus the $\alpha_{\text{graph}}$ attention value should infer the presence of the edge itself as well as its strength, while GAT aims to find the relative importance between fixed graph edges. Also, the conventional inner-product attention of GAT is replaced with a transformer architecture in RAIN to achieve a more flexible representation. We use attention-weighted hidden states from the PA mechanism to apply the transformer as follows,

$$X_{\text{graph},i} = f_{\text{graph},X}(\tilde{\mathbf{h}}_i) \quad \text{where } X \in \{K, Q, V\} \tag{7}$$
$$\alpha_{\text{graph},ij} = \sigma(K_{\text{graph},i}(Q_{\text{graph},j})^T/\sqrt{d_h}), \tag{8}$$

where $f_{\text{graph},X}$ is a stack of MLP layers for $K_{\text{graph}}$, $Q_{\text{graph}}$, and $V_{\text{graph}}$. Note that instead of softmax which normalizes the attention weight, sigmoid activation $\sigma$ is used for graph attention to obtain the *absolute* interaction strength. This is because our main goal is to infer the ground-truth interaction strength, not the relative strength for a single instance. Extracted graph attention $\alpha_{\text{graph},ij}$ becomes a weight for the decoder module.

### 2.3 Decoder

The decoder shares the same GRU layer with the encoder module and the employ new value function, $V_{\text{dec},i}$, for the message aggregation. For agent $i$, RAIN concatenates attention-weighted value vectors from other agents along with its own value vector as follows,

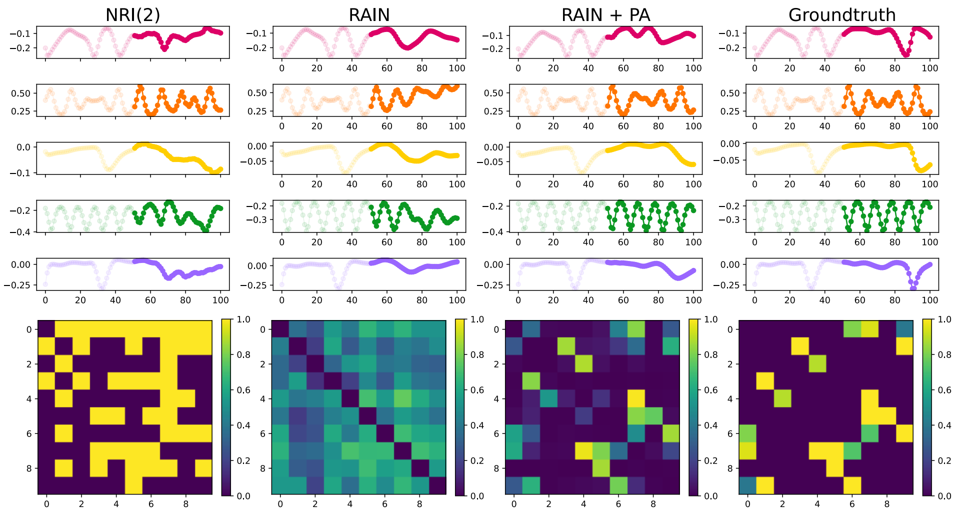

Figure 4: Visualization of the trajectory predictions (upper) and retrieved connectivity matrices (lower) for coupled Kuramoto oscillators. The results from left to right are from NRI(2), RAIN without PA, RAIN with PA, and ground-truth. Here, 5 out of 10 oscillators are drawn for trajectory ($\frac{d\phi_i}{dt}$) visualization, and semi-transparent paths indicate the first 50 steps of input trajectories while solid paths denote 50 steps of predicted future trajectories.

$$\mathbf{h}_i^{T_{\text{enc}}+t} = f_{\text{LSTM}}(\mathbf{h}_i^{T_{\text{enc}}+t-1}, \mathbf{x}_i^{T_{\text{enc}}+t}) \tag{9}$$

$$V_{\text{dec},i}^{T_{\text{enc}}+t} = f_{\text{pair},V}(\mathbf{h}_i^{T_{\text{enc}}+t}) \tag{10}$$

$$\tilde{\mathbf{h}}_{\text{dec},i}^{T_{\text{enc}}+t} = \sum_{j\neq i} \alpha_{\text{graph}} V_{\text{dec},j}^{T_{\text{enc}}+t} \oplus V_{\text{dec},i}^{T_{\text{enc}}+t}. \tag{11}$$

RAIN finally produces a prediction of the state variables of the next step using a stack of MLP layers $f_{\text{dec}}$, $f_\mu$, and $f_\sigma$,

$$\mathbf{h}_{\text{dec}}^{T_{\text{enc}}+t} = f_{\text{dec}}(\tilde{\mathbf{h}}_{\text{graph},i}^{T_{\text{enc}}+t}) \tag{12}$$

$$\mu_i^{T_{\text{enc}}+t} = f_\mu(\mathbf{h}_{\text{dec}}^{T_{\text{enc}}+t}) \tag{13}$$

$$\sigma_i^{T_{\text{enc}}+t} = f_\sigma(\mathbf{h}_{\text{dec}}^{T_{\text{enc}}+t}). \tag{14}$$

The outputs of $f_\mu$ and $f_\sigma$ are $r$-dimensional vectors, each representing the means and variances of the difference of state variables. RAIN samples the next state from a Gaussian distribution with $f_\mu$ and $f_\sigma$ and adds the values to the previous state. The decoder iterates this for $T_{dec}$ steps to predict the future states. For training, we employ negative log-likelihood (NLL) loss for a Gaussian distribution,

$$\mathcal{L}_{\text{NLL}} = \sum_{t=T_{\text{enc+1}}}^{T_{\text{dec}}} \sum_{q}^{R} \sum_{i}^{N} -\frac{1}{2}\log(2\sigma_i^{t,q}) + \frac{(\Delta y_i^{t,q} - \mu_i^{t,q})^2}{2(\sigma_i^q)^2}, \tag{15}$$

where $\Delta y_i^{t,q}$, $\mu_i^{t,q}$, and $\sigma_i^{t,q}$ are the true future state difference, predicted mean, and predicted variance of the state variable $q$ of agent $i$ at time $t$, respectively.

## 3 Experiments

We demonstrate the capability of RAIN by performing inference tasks with various model systems ranging from simulated physical systems to real motion capture data from a walking human. All of

Table 1: Total Pearson correlation ($\rho_{\text{tot}}$) and the sample Pearson correlation ($\rho_{\text{sample}}$) between retrieved interaction weights and true interaction weights for simulations with 5 and 10 interacting objects.

| Model | Spring | | Kuramoto | |
|---|---|---|---|---|
| Corr. | $\rho_{\text{tot}}$ | $\rho_{\text{sample}}$ | $\rho_{\text{tot}}$ | $\rho_{\text{sample}}$ |
| | 5 objects | | | |
| Corr. (Path) | 0.2917 | 0.2944 | 0.0243 | -0.1206 |
| Corr. (LSTM) | 0.0979 | 0.0952 | -0.1369 | -0.0751 |
| NRI(2) | 0.8482 | 0.8660 | 0.0027 | -0.0048 |
| NRI(4) | 0.9250 | 0.9214 | 0.0230 | 0.0192 |
| RAIN | 0.8411 | 0.8770 | 0.5213 | 0.4987 |
| RAIN + PA | **0.9400** | **0.9568** | **0.8731** | **0.8925** |
| | 10 objects | | | |
| Corr. (Path) | 0.2654 | 0.2732 | 0.1047 | 0.0682 |
| Corr. (LSTM) | 0.1196 | 0.1096 | 0.0024 | -0.0012 |
| NRI(2) | 0.8602 | 0.8593 | 0.0393 | 0.0413 |
| NRI(4) | 0.8921 | 0.8966 | 0.0438 | 0.0454 |
| RAIN | 0.7787 | 0.8164 | 0.4560 | 0.4683 |
| RAIN + PA | **0.9117** | **0.9221** | **0.8265** | **0.8411** |

the models and baselines are implemented with `PyTorch` and optimized with Adam [13]. Data and models are available at github.com/nokpil/RAIN. See the Supplementary Materials for details on the data generation, model architectures, and training details.

## 3.1 Physical simulations

We simulated the trajectories of two physical systems: a spring-ball system and phase-coupled Kuramoto oscillators [16]. Unlike previous studies [14, 25] in which every interaction strength $w$ in the system is discrete and constant (commonly $w = 1$), we consider a more general setting where interaction strengths are drawn from a continuum and are thus heterogeneous. Assuming that two agents $i$ and $j$ are interacting, the interaction strength for each system would be a spring constant $k_{ij}$ for a spring-ball system and a coupling weight $w_{ij}$ for a Kuramoto model. Although our model can handle asymmetric interaction strength—thanks to the asymmetric nature of the transformer—we take the symmetric form of connectivity matrix $k_{ij} = k_{ji}$ for the simulated systems. For the spring-ball and Kuramoto systems, we first select the edges between $n$ nodes with probability $p$ (excluding self-connections) to construct an interaction graph, and then assign a randomly sampled interaction strength to each edge from a uniform distribution $U[0, 1]$ while expressing non-assigned edges with zero interaction strength. We generate 10k training samples and 2k validation samples for all simulated tasks. The state variables consisting of trajectories are $x, y, v_x, v_y$ (positions and velocities) for the spring-ball system. For the Kuramoto oscillators, a concatenated vector of $\frac{d\phi}{dt}$, $\sin\phi$, and intrinsic frequency $\omega$ are used to construct the trajectories, where $\phi$ is the phase of an oscillator.

For evaluation, we measure how accurately the model predicts future trajectories by mean squared error (MSE) and how well the model retrieves the original connectivity matrix by two forms of Pearson correlation. We first gather every retrieved interaction strength $a$ and corresponding true interaction strength $k$ from all 2k validation samples and calculate the correlation in total ($\rho_{\text{tot}}$). Secondly, we independently calculate the correlation for each validation sample and take the average of the 2k samples ($\rho_{\text{sample}}$). One may interpret $\rho_{\text{tot}}$ as the overall performance of the model, while $\rho_{\text{sample}}$ indicates the expected correlation for a single instance at a test time. We excluded diagonal trivial zeroes (due to no self-loop interaction) while calculating correlation, so the reported value is strictly lower compared to the case where every value in the connectivity matrix is used. Correlations from the baseline models with discrete edge types are obtained by assigning continuous weights to each edge type according to every possible permutation and choosing the best one. More precisely, we assign $n$ weights of $0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, 1$ to $n$ edge types with $n!$ different assignments and report the highest correlation.

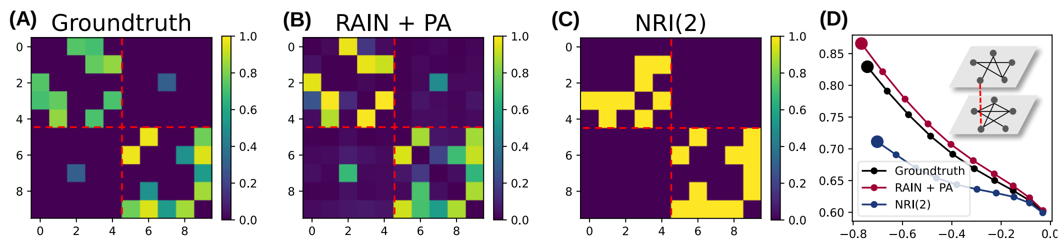

Figure 5: (A) Ground-truth connectivity matrix of the given multilayer network with a single inter-layer link. Inferred interaction weights from (B) RAIN with PA and (C) NRI(2) are shown. (D) Trajectory predictions (10 steps) for ball 3. The inset depicts a diagram of the given multilayer networks, where the inter-layer link is highlighted with the red dotted line.

### 3.1.1 Results

Table 1 shows the correlations between the true interaction graph and the predicted interaction strengths from various models. Following [14], we measure the correlation between trajectory feature vectors as Corr. (Path) and the correlation between trained LSTM feature vectors as Corr. (LSTM). NRI(2) and NRI(4) indicate the NRI from [14] with 2 and 4 edges types, respectively. We check the effect of PA by testing the RAIN model both with and without the PA mechanism. In Table 1, we can clearly see that RAIN with PA significantly outperforms every other model at inferring interaction strengths. Note that the high correlations from the NRI models for the spring-ball system can only be achieved if and only if we know the ground-truth interaction strength and choose the best permutation. Considering the huge performance difference between RAIN without and with PA, our model and PA mechanism are both crucial for accurate interaction strength retrieval. We show MSE results for predicting 50 future states in Table 2, Here, again following [14], SingleLSTM runs a single LSTM for each object separately, while JointLSTM concatenates all state vectors (thus may handle a fixed number of agents only) and trains a single LSTM to jointly predict all future states. See the Supplementary Materials for further analysis, including full histograms of the correlation distributions.

Figures 3 and 4 show the results of future prediction and interaction strength inference for 10 interacting agents, where 5 trajectories are drawn for visualization. In Fig. 3, we can observe that all models, including NRI(2), succeed in capturing the existence of interaction between agents. But it is apparent that the NRI(2) model fails to capture the interactions with small weights, while RAIN without PA yields in numerous false positives with spurious weights. Such weakness of each model is reflected in relatively large prediction errors compared to RAIN with PA, especially for ball 4 (green) and ball 5 (purple). The power of the PA mechanism becomes more evident with the Kuramoto oscillators (Fig. 4), where only RAIN with PA succeeds in retrieving meaningful interaction weights. To sum up, we can conclude that the PA mechanism clearly helps the refinement of hidden states and thus yields better results with a greater correlation with the ground-truth interaction weights.

### 3.1.2 Impact of weak interaction

By inferring interaction with a continuous strength as RAIN does, we can capture an entire spectrum of interactions with a single model. Particularly, we find that our model is able to detect weak interactions that are often ignored by the discrete NRI models due to their limited capacity. In Figure 5, we emphasize the significance of the inference of weak interaction by constructing the connectivity matrix in a form of a multilayer network [15] with 2 layers. Here, we prepare two densely connected layers of springs where their spring constants are sampled from $[0.5, 1]$, uniformly. Between the two layers, we set a single inter-layer link (connecting ball 3 and ball 8) with a spring constant of $0.3$, which is much smaller than the intra-layer interaction weights. Since the synchronization of a multilayer network largely depends on the strength and structure of inter-layer coupling [18, 7], capturing this weak but significant connection between the two layers is critical for trajectory prediction. As Fig. 5 shows, RAIN with PA accurately captures this weak interaction, while NRI(2) misses it and thus its predicted trajectory considerably deviates from the true one.

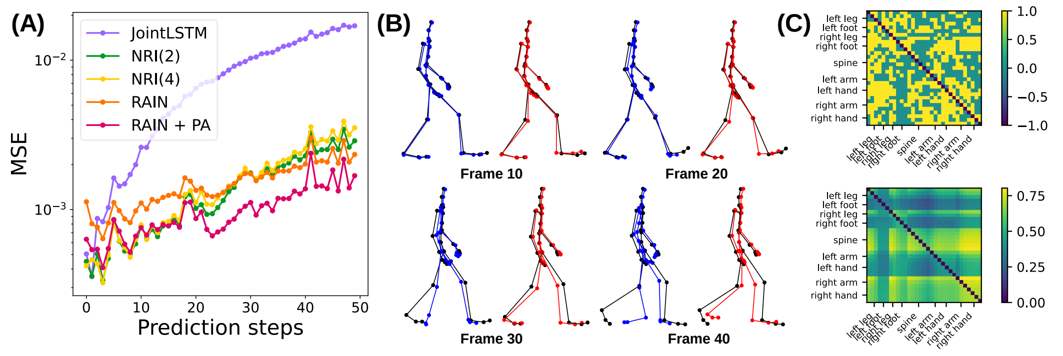

Figure 6: (A) Validation MSE comparison for motion capture data. The prediction results of $50$ time steps from JointLSTM, NRI(2), NRI(4), RAIN without PA, and RAIN with PA are shown. (B) Sample predictions from NRI(2) (blue) and RAIN with PA (red) for a validation trajectory of motion capture data with the ground-truth states (black). (C) Inferred interaction weights from NRI(2) (top) and RAIN with PA (bottom) are shown. Here, the diagonal entries from NRI(2) are marked with $-1$ to distinguish them from the two edge types, $0$ and $1$.

### 3.2 Motion capture data

Next, we test our model in inferring interactions between the joints of a walking human data from the CMU motion database [1]. Different from the physical simulations, this real-world system does not possess any well-known dynamics with a continuous interaction strength. Following [14] and [9], we use the data from subject #35 with 31 joints. We split the training and validation data sets into a $4:1$ ratio. We use the same protocol as with the physical systems: provide $50$ time steps of trajectories and let the model predict $50$ unseen time steps afterward. For the motion data, we train the model to predict the raw value of the future state instead of the difference from the current state.

Figure 6A shows the errors from each model's prediction, where RAIN with PA achieved the lowest MSE. An example visualization of the prediction is shown in Fig. 6B. Without dynamically re-evaluating the interaction graph at every time step as reported in [14] and [9], our model precisely predicts the future states from static but continuous interaction weights. Also, as shown in Fig. 6C, our model produces a smooth and block-wise connectivity matrix which is much easier to interpret compared to that from the discrete NRI(2). This is because there is no restriction or natural meaning for each edge type in the discrete model, and thus the manual sparsity prior is needed to handle the 'no interaction' edges if an abundance is expected. On the other hand, the attention value of RAIN is expected to convey the strength of interaction directly. For instance, it is clear from RAIN's connectivity matrix that the dynamics of the feet are less correlated with the rest of the joints and also that the spine moves along with the right leg, arm, and hand, both of which agree with the qualitative movements of a human while walking.

## 4 Conclusion

In this work, we introduced RAIN, a model for inferring continuous weighted interaction graphs from trajectory data in an unsupervised manner. With the PA mechanism that computes the attention between same-time LSTM hidden states between agents, we can sharply refine the representation for the interaction weight extraction. Our model successfully inferred the absolute interaction weights from simulated physical systems and further showed great prediction performance with an empirical system, demonstrating the advantage of continuous weight modeling in relational learning. Notably, RAIN needs only $50$ time steps of data to infer the interaction weights, and thus requires less data by several orders of magnitude than correlation-based theoretical methods [27, 17]. Also, we found that the refinement of the LSTM hidden state with PA is critical for meaningful performance. Since the PA mechanism is generally applicable to a neural model where its trajectory is represented by a recurrent neural network, one may expect an increase in performance by employing PA in other relational models without increasing the inference time significantly. In the real world, interactions between agents in complex systems possess a broad range of characteristics. For instance, they can

Table 2: Mean squared error (MSE) in predicting future states for simulations with 5 and 10 interacting objects. Underlined entries show better results than those from RAIN with PA.

| Model | Spring | | | Kuramoto | | |
|---|---|---|---|---|---|---|
| Prediction steps | 10 | 30 | 50 | 10 | 30 | 50 |
| | | | 5 objects | | | |
| Static | 0.6665 | 1.3614 | 1.2340 | 1.0671 | 0.9901 | 1.0108 |
| SingleLSTM | 0.1969 | 0.5643 | 0.6048 | 0.5006 | 0.4957 | 0.5247 |
| JointLSTM | 0.0704 | 0.3913 | 0.6362 | 0.0391 | 0.1083 | 0.1749 |
| NRI(2) | 0.0171 | 0.2232 | 0.5263 | 0.0268 | 0.1615 | 0.3275 |
| NRI(4) | 0.0122 | 0.1395 | 0.3573 | 0.0308 | 0.1498 | 0.3050 |
| RAIN | 0.0259 | 0.1322 | 0.3113 | 0.0388 | 0.1705 | 0.3675 |
| RAIN + PA | **0.0084** | **0.0714** | **0.2193** | **0.0041** | **0.0645** | **0.2059** |
| RAIN (true graph) | 0.0062 | 0.0181 | 0.0732 | 0.0037 | 0.0068 | 0.0122 |
| | | | 10 objects | | | |
| Static | 0.6070 | 1.1253 | 1.0779 | 1.0380 | 0.9935 | 0.9799 |
| SingleLSTM | 0.2028 | 0.4991 | 0.5189 | 0.5511 | 0.5138 | 0.5173 |
| JointLSTM | 0.1317 | 0.4823 | 0.5840 | 0.0953 | 0.2490 | 0.3832 |
| NRI(2) | 0.0078 | 0.1158 | 0.3169 | 0.0392 | 0.2341 | 0.4451 |
| NRI(4) | 0.0061 | 0.0866 | 0.2440 | 0.0372 | 0.2411 | 0.4385 |
| RAIN | 0.1665 | 0.3109 | 0.4996 | 0.0307 | 0.1902 | 0.3942 |
| RAIN + PA | **0.0069** | **0.0892** | **0.2351** | **0.0115** | **0.1586** | **0.4016** |
| RAIN (true graph) | 0.0059 | 0.0163 | 0.0504 | 0.0009 | 0.0063 | 0.0301 |

be either positive (excitatory or encouraging) or negative (inhibitory or suppressing), time-delayed with heterogeneous time scales, and noisy both inherently and externally. Complex systems in nature generally contain every aspect of these characteristics, such as neural signals in a human brain. Reinforcing the current RAIN architecture to handle data of such complex nature with a single model would be a promising future direction to explore.

## 5 Data availability

Simulation code and data files are available at https://github.com/nokpil/RAIN.

## 6 Acknowledgement

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea NRF-2022R1A2B5B02001752 (HJ) and National Research Foundation of Korea Grant funded by the Korean Government NRF-2018S1A3A2075175 (SH).

## References

[1] Carnegie-mellon university. carnegie-mellon motion capture database. http://mocap.cs.cmu.edu, 2003.

[2] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks? *arXiv preprint arXiv:2105.14491*, 2021.

[3] Jose Casadiego, Mor Nitzan, Sarah Hallerberg, and Marc Timme. Model-free inference of direct network interactions from nonlinear collective dynamics. *Nature communications*, 8(1):1–10, 2017.

[4] Yen-Yu Chang, Fan-Yun Sun, Yueh-Hua Wu, and Shou-De Lin. A memory-network based solution for multivariate time-series forecasting. *arXiv preprint arXiv:1809.02105*, 2018.

[5] Emily SC Ching and HC Tam. Reconstructing links in directed networks from noisy dynamics. *Physical Review E*, 95(1):010301, 2017.

[6] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[7] Fabio Della Rossa, Louis Pecora, Karen Blaha, Afroza Shirin, Isaac Klickstein, and Francesco Sorrentino. Symmetries and cluster synchronization in multilayer networks. *Nature communications*, 11(1):1–17, 2020.

[8] Keisuke Fujii, Naoya Takeishi, Kazushi Tsutsui, Emyo Fujioka, Nozomi Nishiumi, Ryoya Tanaka, Mika Fukushiro, Kaoru Ide, Hiroyoshi Kohno, Ken Yoda, et al. Learning interaction rules from multi-animal trajectories via augmented behavioral models. *Advances in Neural Information Processing Systems*, 34:11108–11122, 2021.

[9] Colin Graber and Alexander Schwing. Dynamic neural relational inference for forecasting trajectories. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 1018–1019, 2020.

[10] Seungwoong Ha and Hawoong Jeong. Unraveling hidden interactions in complex systems with deep learning. *Scientific reports*, 11(1):1–13, 2021.

[11] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *Advances in Neural Information Processing Systems*, 30, 2017.

[12] Zijie Huang, Yizhou Sun, and Wei Wang. Coupled graph ode for learning interacting system dynamics. In *KDD*, pages 705–715, 2021.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. Neural relational inference for interacting systems. *arXiv preprint arXiv:1802.04687*, 2018.

[15] Mikko Kivelä, Alex Arenas, Marc Barthelemy, James P Gleeson, Yamir Moreno, and Mason A Porter. Multilayer networks. *Journal of complex networks*, 2(3):203–271, 2014.

[16] Yoshiki Kuramoto. Self-entrainment of a population of coupled non-linear oscillators. In *International symposium on mathematical problems in theoretical physics*, pages 420–422. Springer, 1975.

[17] Pik-Yin Lai. Reconstructing network topology and coupling strengths in directed networks of discrete-time dynamics. *Physical Review E*, 95(2):022311, 2017.

[18] I Leyva, R Sevilla-Escoboza, I Sendiña-Nadal, R Gutiérrez, JM Buldú, and S Boccaletti. Inter-layer synchronization in non-identical multi-layer networks. *Scientific Reports*, 7(1):1–9, 2017.

[19] Zhe Li, Andreas S Tolias, and Xaq Pitkow. Learning dynamics and structure of complex systems using graph neural networks. *arXiv preprint arXiv:2202.10996*, 2022.

[20] Diganta Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[21] Thomas Schreiber. Measuring information transfer. *Physical review letters*, 85(2):461, 2000.

[22] Rundong Shi, Weinuo Jiang, and Shihong Wang. Detecting network structures from measurable data produced by dynamics with hidden variables. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 30(1):013138, 2020.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[24] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[25] Ezra Webb, Ben Day, Helena Andres-Terre, and Pietro Lió. Factorised neural relational inference for multi-interaction systems. *arXiv preprint arXiv:1905.08721*, 2019.

[26] Zhang Zhang, Yi Zhao, Jing Liu, Shuo Wang, Ruyi Tao, Ruyue Xin, and Jiang Zhang. A general deep learning framework for network reconstruction and dynamics learning. *Applied Network Science*, 4(1):1–17, 2019.

[27] Zhaoyang Zhang, Zhigang Zheng, Haijing Niu, Yuanyuan Mi, Si Wu, and Gang Hu. Solving the inverse problem of noise-driven dynamic networks. *Physical Review E*, 91(1):012814, 2015.

## Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] See Abstract and Introduction.
   (b) Did you describe the limitations of your work? [Yes] See Conclusion.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See Data availability.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See Supplemental material.
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See Supplemental material.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See the manuscript where we cite [1] for its dataset.
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See Data availability.

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

## A Appendix

### A.1 RAIN implementation

For RAIN, we globally adopt *Mish* [20] as a non-linear activation function which has a functional form as follows.

$$\text{Mish}(x) = x \tanh(\text{softplus}(x)) = x \tanh(\ln(1 + e^x)) \tag{16}$$

Hereafter, we denote the batch dimension as $B$, the number of input time step as $T$, the number of agent as $N$, the number of state variables as $S$, the agent number as $N$. Typically, $B = 128$, $T = 50$, $S = 3$(Kuramoto), 4(Spring), 6(motion), and $N = 5, 10, 31$(motion).

#### A.1.1 Encoder

Intially, the model recieves the input data with the size of $B \times T \times N \times S$ at every batch. We first encoding each state with MLP with a 2-layer MLP with hidden dimension of 64 and output dimension of 128. The encoded state is then feeded to a sigle GRU layer with hidden dimension of 256, which produces the LSTM hidden states with the size of $B \times N \times 256$ for both future predictions and the grpah extraction. For the initial hidden state, we also train a single-layer MLP with output dimension of 256, which recieves the raw state variables (at time 0) and produces the initial hidden states for our GRU layer.

#### A.1.2 Pairwise attention

For pairwise attention (PA), we use transformer [23] with 4 heads with 32 dimensions each, total of 128 dimension. Hence, the key (L), query (Q), and value (V) is calculated via a linear layer with input dimension of 256 and output dimension of 128 (to match with the transformer dimension, $32 \times 4 = 128$). After we perform PA by calculating the same-time attention and weighted average, the refined hidden states goes to the graph extraction module. Since PA produces a single hidden states for every pair of agents, we now have the output with the size of $B \times N \times N \times 128$, at this stage.

#### A.1.3 Graph extraction module

For graph extraction, we employ a 3-layer MLP with hidden dimension of 32, 16 and output dimension of 1. This module makes the refined hidden states into the size of $B \times N \times N \times 1$, which indicates the $N \times N$ weighted interaction graphs that our module has inferred. Note that the RAIN model without PA uses two concatenated individual LSTM hidden states (from agent $i$ and $j$) for the graph extraction, and thus the input dimension of the graph extraction module becomes $2 \times (\text{LSTM dimension}) = 256$ (and the same as RAIN with PA afterwards). Finally, we add diagonal mask filled with $-10000$ to the result and apply a sigmoid function. This effectively reduces every diagonal entries into 0 ($\sigma(-10000) \approx 0$) and normalize the scale of the interaction strength into $[0, 1]$.

### A.1.4 Decoder

In decoding stage, we first feed the current trajectory data to the LSTM encoder and gets the hidden states for current time step. We then calcualte the another value function with a linear layer, $V_{\text{dec}}$, with output dimension of 128. For a prediction of single agent, we concatenate two vectors; its own value and weighted average of all other value, according to the weights from the graph extraction module. Hence, the input dimension of the decoder module should be $2 \times 128 = 256$ (since two value functions are concatenated).

The decoder of RAIN is consists of three MLPs: one primary 2-layer MLP with hidden and output dimension of 256 for hidden states encoding, and two secondary 2-layer MLPs with hidden dimension of 256 and output dimension of $S$. After we passes the concatenated values to the primary MLP, we further feeds the results to two secondary MLPs for yielding means and variances of the desired output.

### A.1.5 Training details

For training, we iterate the decoder 50 times while sampling the trajectory data from generated means and variances, and compared it with state difference to calculate NLL loss. We perform gradient descent at each iteration step, let gradient flows through backward time and yield 50 losses in total. We average all the NLL losses and perform backward propagation.

## A.2 Baseline implementation

In the current work, we trained various baselines to compare the performance with our model. More precisely, we trained SingleLSTM, JointLSTM, NRI(2), NRI(4), and both RAIN with and without PA.

The SingleLSTM model consists of a single 2-lyaer LSTM with hidden dimension of 256. Similar to RAIN, we train the model by feeding 50 previous states and aim to predict 50 future states, but individually. Hence, there are no information communicated between agetns. Furthermore, we used the hidden states of the second layer of LSTM in SingleLSTM for calcualting Corr (LSTM), by measuring the Pearson correlation between two agent's hidden states and again maesure those correlations to the ground-truth interaction strengths.

For the JointLSTM model, we first concatenate all states from the agents, which yields the vector with dimension of $N \times S$. A single LSTM is trained to recieve this concatenated vector, and produce a single (again, concatenated) vector to predict the future states of all agents at once. This is the most naive form of neural network that (in theory) enables the information communication between agents. Other trainig protocols are same as RAIN.

For NRI(2) and NRI(4), we faithfully followed the original training protocol of [14], including 10 steps of burning-in process. We set the hidden state dimension as 256 and also used learning rate scheduling as mentioned in the original paper. We did not employ dynamical graph re-evaluation, since we wanted to compare the performance between static graphs.

## A.3 Dataset details

### A.3.1 Spring-ball system

Spring-ball systems are consist of $N$ balls and connecting springs between them. The interaction between balls are Hookean force, $\mathbf{F}_{ij} = -k_{ij}\mathbf{x_{ij}}$ where $k_{ij}$ is a spring constant and $x_{ij}$ is a relative position vector between two balls. In this study, we aim to infer $k_{ij}$ without knowing any prior information about the dynamics.

All of the $N$ balls are initially start with random 2D positions and velocities that is drawn from the normal distribution $\mathcal{N}(0, 1)$. Here, velocities are further normalized to have a vector norm of 1. The balls are kinetically moving in a square box with sides of length 5, centered at the origin. The collision between every object is perfectly elastic (In practice, the size of the ball is ignored and the collision between balls never happens). We simulate each trajectory 1000 times steps with time interval $dt = 0.005$, and it is further subsampled every 10 steps to construct total 100 steps of

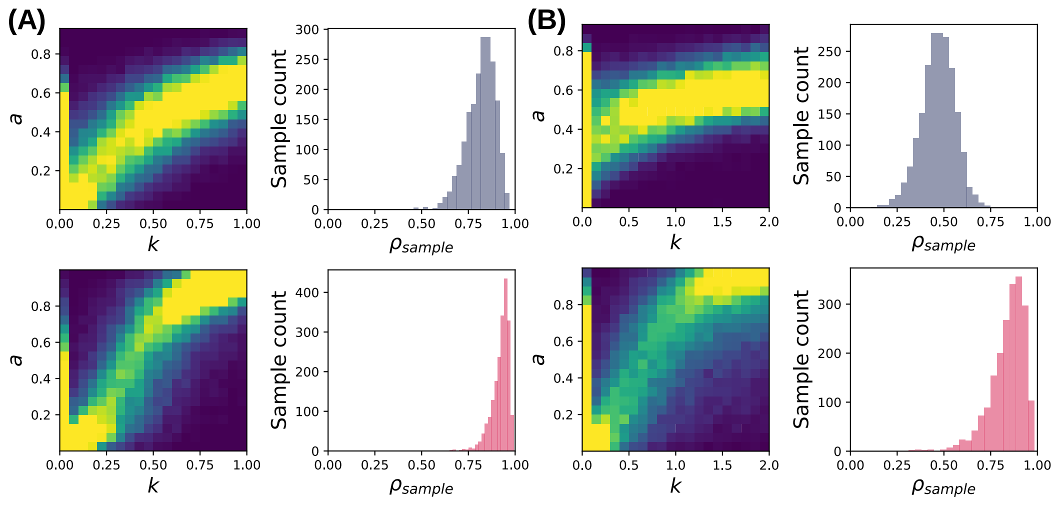

Figure 7: Correlation analysis results from (A) spring-ball system and (B) kuramoto oscialltors with 10 agents. For each panel, the results from (top) RAIN without PA and (bottom) RAIN with PA is shown. The left 2D histograms for each panel shows the overall correlation between the ground-truth weight ($k$) and the inferred weight ($a$) for the entire validation dataset, while the right histograms for each panel shows the same correlation but calculated from each validation samples independently. The line at $k = 0$ is due to the relative abundance of zero interaction weight, i.e., agent pairs with no interaction.

training data; 50 steps for input and 50 steps for validation. In the simulation, the spring constant is drawn uniformly from $[0, 1]$ multiplied by a constant factor 0.05 for stability.

### A.3.2 Phase-coupled oscillators

For Kuramoto oscillators, each ball its intrinsic frequency $w_i$ and interacting with each other by following differential equation:

$$\frac{d\phi_i}{dt} = w_i + \sum_{j \neq i} k_{ij} \sin(\phi_i - \phi_j) \tag{17}$$

with coupling constants $k_{ij}$, drawn uniformly from $[0, 2]$. We simulate the system by solving (17) with fourth-order Runge-Kutta integrator with a step size $dt = 0.01$. Intrinsic frequency $w_i \in \mathbb{N}$ and initial phase $\phi_i^{t=0}$ is drawn uniformly from $[1, 10]$ and $[0, 2\pi)$, respectively.

### A.3.3 Motion capture data

As mentioned in the main manuscript, we used the data of subject #35 from CMU motion databse [1]. The trial number spans from 1 to 16, and 28 to 34, following [9]. We used data processing codes from [9] to preprocess the amc files into the trainig and validation dataset. By split the data by a length of 100 (50 input steps and 50 future steps), we got 63 sequences of train data and 23 sequences of validation data.

## A.4 Further analysis

The effect of PA mechanism is further illustrated by plotting distributions of correlations in Figure 7. Here, left 2D histograms on each panel shows the scatter plot of true weight $k$ and inferred weight $a$ among entire validation dataset and the right histogram shows the distribution of each validation sample's correlation. Thus, calculating correlation on left 2D histogram would yield $\rho_{\text{tot}}$, while averaging the right histogram would yield $\rho_{\text{sample}}$.