Supplementary Material: Learning Heterogeneous Interaction Strengths by Trajectory Prediction with Graph Neural Network

Seungwoong Ha and Hawoong Jeong*
Department of Physics, Korea Advanced Institute of Science and Technology, Daejeon 34141, Korea
(Dated: November 16, 2022)

I. RAIN IMPLEMENTATION

test For RAIN, we globally adopt *Mish* [1] as a non-linear activation function which has a functional form as follows.

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

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. 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.

B. Pairwise attention

For pairwise attention (PA), we use transformer [2] 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.

C. 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 $2 \times$ (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].

* Also at Center for Complex Systems, Korea Advanced Institute of Science and Technology, Daejeon 34141, Korea; hjeong@kaist.edu

D. 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_{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.

E. 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,

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

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.

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.

II. 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 [3], 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.

For GATv2, we employed the exactly same encoder and decoder as our RAIN model, and switched the graph extraction model into GATv2 convolution layer with a single attention head.

III. DATASET DETAILS

A. 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 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.

B. Phase-coupled oscillators

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

$$\frac{d\phi_i}{dt} = w_i + \sum_{j \neq i} k_{ij} \sin(\phi_i - \phi_j) \tag{3}$$

with coupling constants k_{ij}, drawn uniformly from $[0, 2]$. We simulate the system by solving (3) 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.

C. Motion capture data

As mentioned in the main manuscript, we used the data of subject #35 from CMU motion databse [4]. The trial number spans from 1 to 16, and 28 to 34, following [5]. We used data processing codes from [5] 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.

IV. FURTHER ANALYSIS

The effect of PA mechanism is further illustrated by plotting distributions of correlations in figure 2. 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 ρ_{tot}, while averaging the right histogram would yield ρ_{sample}.

V. COMPARISON WITH NRI WITH MORE CATEGORIES

In Table 1, compare our model's prediction performances with NRI with higher number of categories, NRI(8), NRI(16), and NRI(32), along with NRI(2) and NRI(4), whici is already reported in the main manuscript. For the demonstration, the Kuramoto model with 5 oscillators is used. We omit the correlation comparison since it needs $8! = 40320$, $16! = 2.09 \times 10^{13}$, and $32! = 2.63 \times 10^{35}$ cases to test for NRI(8), NRI(16), and NRI(32), respectively, in order to find the best match with the ground-truth interaction strength.

VI. PERFORMANCE OF RAIN ON LARGE SYSTEM

To demonstrate the cabaility of inferring large system, we test RAIN's performance in a spring-ball system with $N = 30$. After the training, the total correlation is $\rho_{tot} = 0.7862$, and Figure 2 is the visualization of the trajectories and adjacecy matrix inference from RAIN+PA which shows good agreement with the ground-truth value.

---

[1] D. Misra, Mish: A self regularized non-monotonic activation function, arXiv preprint arXiv:1908.08681 (2019).

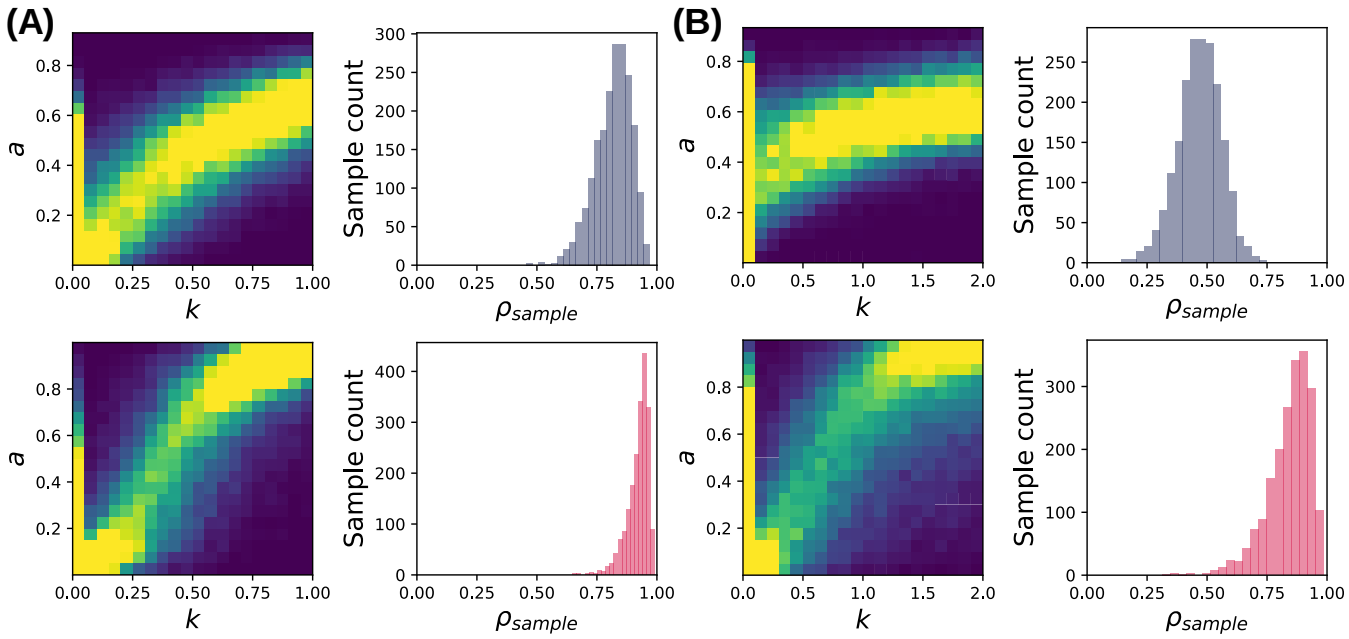

FIG. 1. Correlation analysis results from (A) spring-ball system and (B) kuramoto oscialltors with 10 agents. For each panel, the results from (top) RAIN without PA and (bottom) RAIN with PA is shown. The left 2D histograms for each panel shows the overall correlation between the ground-truth weight (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.

TABLE I. Mean squared error (MSE) in predicting future states for the Kuramoto model with 5 objects.

Model	Kuramoto		
Prediction steps	10	30	50
	5 objects		
NRI(2)	0.0268	0.1615	0.3275
NRI(4)	0.0308	0.1498	0.3050
NRI(8)	0.0335	0.1361	0.2664
NRI(16)	0.0306	0.1349	0.2807
NRI(32)	0.0262	0.1340	0.3050
RAIN + PA	**0.0041**	**0.0645**	**0.2059**
RAIN (true graph)	0.0037	0.0068	0.0122

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, Attention is all you need, in *Advances in neural information processing systems* (2017) pp. 5998–6008.

[3] T. Kipf, E. Fetaya, K.-C. Wang, M. Welling, and R. Zemel, Neural relational inference for interacting systems, arXiv preprint arXiv:1802.04687 (2018).

[4] Cmu. carnegie-mellon motion capture database. http://mocap.cs.cmu.edu (2003).

[5] C. Graber and A. Schwing, Dynamic neural relational inference for forecasting trajectories, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops* (2020) pp. 1018–1019.

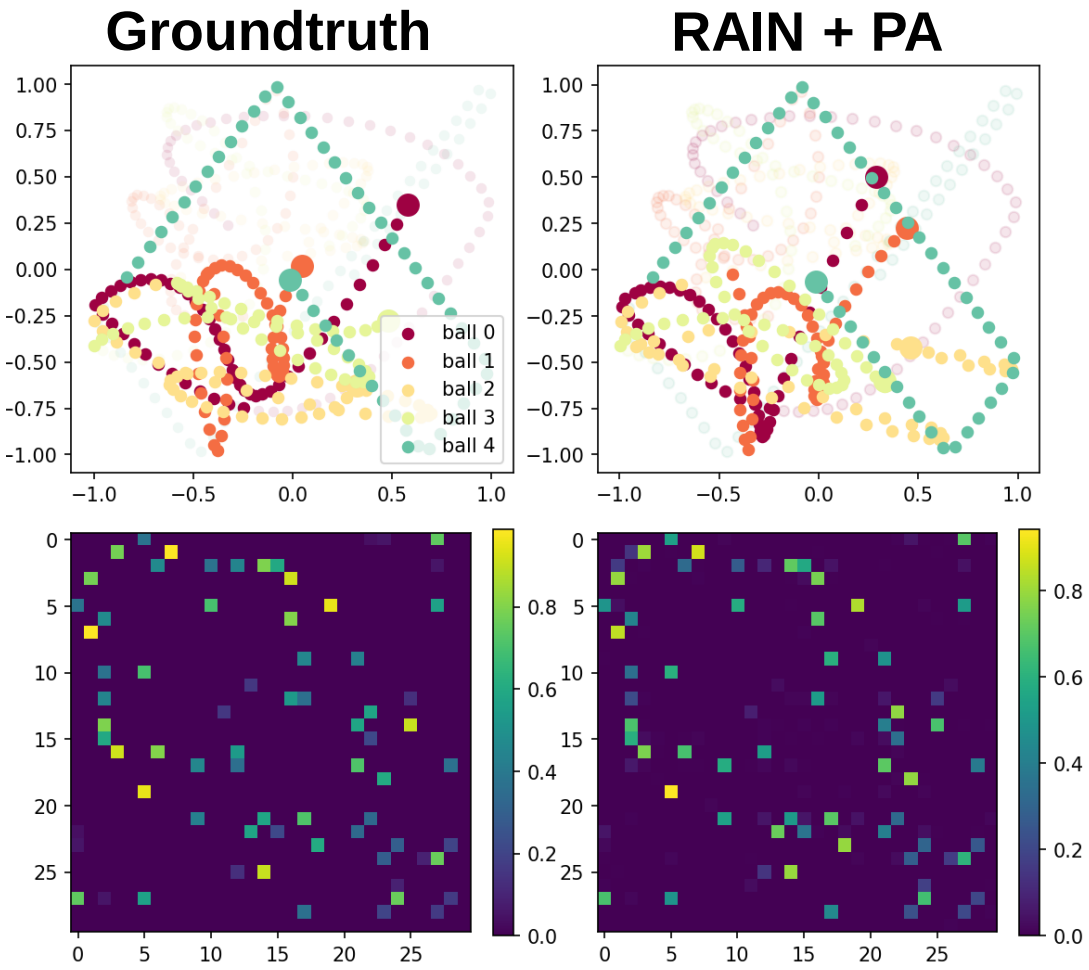

FIG. 2. Visualization of the trajectory predictions (upper) and retrieved connectivity matrices(lower) for the spring-ball system with $N = 30$. Here, 5 out of 30 balls are drawn for trajectory visualization, and semi-transparent paths indicate the first 50 steps of input trajectories while solid paths denote 50 steps of predicted future trajectories.