HOPE: High-order Graph ODE For Modeling Interacting Dynamics

Xiao Luo [1] **Jingyang Yuan** [2] **Zijie Huang** [1] **Huiyu Jiang** [3] **Yifang Qin** [2] **Wei Ju** [2] **Ming Zhang** [2] **Yizhou Sun** [1]

Abstract

Leading graph ordinary differential equation (ODE) models have offered generalized strategies to model interacting multi-agent dynamical systems in a data-driven approach. They typically consist of a temporal graph encoder to get the initial states and a neural ODE-based generative model to model the evolution of dynamical systems. However, existing methods have severe deficiencies in capacity and efficiency due to the failure to model high-order correlations in long-term temporal trends. To tackle this, in this paper, we propose a novel model named High-Order graPh ODE (HOPE) for learning from dynamic interaction data, which can be naturally represented as a graph. It first adopts a twin graph encoder to initialize the latent state representations of nodes and edges, which consists of two branches to capture spatio-temporal correlations in complementary manners. More importantly, our HOPE utilizes a second-order graph ODE function which models the dynamics for both nodes and edges in the latent space respectively, which enables efficient learning of long-term dependencies from complex dynamical systems. Experiment results on a variety of datasets demonstrate both the effectiveness and efficiency of our proposed method.

1. Introduction

Dynamic interacting systems are ubiquitous in the real world, such as social networks (Hafiene et al., 2020; Huang et al., 2019; Liao et al., 2021) and moving planets (Cranmer et al., 2020). In these systems, a set of agents are connected and exhibit complicated behaviors over time, forming an evolving graph structure. For example, the spread of COVID-19 can be regarded as a dynamical system where each node represents a state with an evolving daily death rate and these nodes are connected according to their geographical distances. Consequently, modeling and understanding the intricate dynamics of complex relational systems has become a major area of research with various downstream tasks (Hsieh et al., 2021; Wang & Yu, 2021; Hackl et al., 2021; Groß et al., 2019).

In literature, a range of deep neural network approaches have been proposed for modeling dynamic interaction systems (Battaglia et al., 2016; Kipf et al., 2018; Veličković et al., 2018). Typically, they use graph neural networks (GNNs) to learn node representations at each timestamp and predict their future trends. However, these discrete models cannot deal with irregular-sampled observations and require the observation of each node to be accessible at every timestamp (Huang et al., 2020; 2021). In contrast, neural ordinary differential equation (ODE) methods are effective to model system dynamics with missing data (Chen et al., 2018). Recent efforts (Poli et al., 2019; Huang et al., 2020; 2021; Gupta et al., 2022) have extended this technique into modeling interacting dynamical systems. Generally speaking, these methods usually combine GNNs with neural ODE models to capture spatio-temporal relationships in dynamical systems. In particular, they first utilize GNNs to initialize state representations and then develop a neural ODE model for both nodes and edges, which drives the evolution of the dynamical system. In the end, a decoder is adopted to output the prediction and the whole generative model is optimized through variational inference.

However, existing graph ODE approaches fail to model complex high-order correlations from two perspectives, which results in inferior performance in reality. On the one hand, in the graph encoder, these methods (Huang et al., 2020; 2021) usually construct a temporal graph and then utilize a spatial-based GNN to learn node representations. However, spatial-based GNNs could miss the high-order non-neighborhood semantics embedded in graph spectrum, which may fail to provide proper initialization of latent state representations. On the other hand, in the generative ODE model, they typically involve the first-order derivatives. However, second-order laws are indispensable in abundant dynamical systems of physical science (Aydin & Coban, 2022; Hao

[1]Department of Computer Science, University of California, Los Angeles, USA [2]National Key Laboratory for Multimedia Information Processing, School of Computer Science, Peking University [3]Department of Statistics and Applied Probability, University of California, Santa Barbara, USA. Correspondence to: Xiao Luo <xiaoluo@cs.ucla.edu>.

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

et al., 2022), including Newton's equations of motion as well as oscillators. Therefore, current approaches are typically incapable of capturing these high-order dependencies in latent representations, implying the shortage of model capacity. Worse yet, it has been validated that first-order neural ODEs are less efficient owing to the high number of function evaluations (NFEs) required for optimization and inference (Norcliffe et al., 2020).

In this paper, we propose a novel method named High-Order graPh ODE (HOPE) for learning from dynamic interaction data. Generally, HOPE learns high-order spatio-temporal correlations in both the graph encoder and graph ODE generative model. In particular, to initialize the latent state representations for objects, we first construct a temporal graph and then extract holistic spatio-temporal relationships using two parallel GNN branches from complementary views. On the one hand, we employ the self-attention mechanism to adaptively aggregate neighborhood information to learn object representations from a spatial perspective. On the other hand, second-order spectral convolution is adopted to learn high-order correlations between non-neighborhood nodes in a less-parametric manner, hence exploiting semantics embedded in graph spectrum. In addition, we provide a generative model involving a second-order graph ODE function to learn long-term temporal dependency, which models dynamic nodes and edges in interacting systems effectively. Our second-order graph ODE function is derived from the momentum updating rule, indicating a faster convergence rate empirically (Perantonis & Karras, 1995). Eventually, a decoder is used to output the prediction and the whole generative model is trained by maximizing the evidence lower bound of the likelihood loss. We undertake extensive experiments on three benchmark datasets to verify the efficacy of our proposed methods and the results show that HOPE achieves state-of-the-art performance over competing baselines in terms of both accuracy and efficiency.

The contributions of this work can be summarized as below:

- **Higher-order Architecture.** Our approach incorporates second-order graph convolution to capture non-neighborhood semantic information, as well as second-order graph ODEs to model higher-order temporal dependencies. These specific architectural choices have been tailored to address the challenges posed by the problem, ultimately resulting in superior performance on benchmark datasets.

- **Theoretical Analysis.** We offer a theoretical analysis that connects the message passing algorithm with the second-order graph ODE (Lemma 3.1), illustrates the implementation of the second-order graph ODE (Lemma 3.2), and establishes the local uniqueness of solutions to our ODE (Lemma 3.3).

- **Extensive Experiments.** We have conducted comprehensive experiments across three benchmark datasets in various settings. Our in-depth analyses demonstrate the effectiveness of the individual components proposed within HOPE.

- **Application Prospect.** Our comprehensive continuous model has practical applicability to interacting systems, offering valuable insights into physical simulations and biological processes governed by second-order laws.

2. Preliminaries

2.1. Problem Definition

We assume that a dynamical system contains N interacting objects. The input is comprised of the trajectories of the objects and their underlying weighted interaction graph which evolves continuously. The snapshots of the interaction graph is denoted as $\mathcal{G} = \{G^1, \cdots, G^T\}$ in which $G^t = \{\mathcal{V}, \mathcal{E}^t\}$ denotes the interaction graph at the timestamp t and the set $\mathcal{E}^t$ denotes the set of weighted edges. $w_{ij}^t \in \mathsf{R}$ is the weight of the edge from node i to node j at the timestamp t. The object trajectory sequence is denoted as $\mathcal{X} = \{\boldsymbol{X}^1, \cdots, \boldsymbol{X}^T\}$ and $\boldsymbol{X}^t$ is the feature matrix at the timestamp t. The adjacency matrix sequence is denoted as $\mathcal{A} = \{\boldsymbol{A}^1, \cdots, \boldsymbol{A}^T\}$, where $\boldsymbol{A}^t(i,j) = w_{ij}^t$. Given the observed information in dynamical systems, we aim to model the latent dynamics based on the object embeddings and use them to predict the trajectories in the future, i.e., $\boldsymbol{X}^t(t > T)$.

2.2. Neural ODEs for Modeling Dynamical Systems

Neural ODEs have been proposed for modeling multi-agent dynamical systems (Huang et al., 2020; 2021). Taking single-agent dynamic systems as a start, the evolution of state representations are modeled by a given first-order ODE, i.e., $\frac{d\boldsymbol{z}^t}{dt} = f(\boldsymbol{z}^t)$, which drives the dynamical systems to evolve continuously. At this time, the whole trajectory of the object is decided by the initial state representations $\boldsymbol{z}^0$ as follows:

$$\boldsymbol{z}^T = \boldsymbol{z}^0 + \int_{t=0}^{T} f\left(\boldsymbol{z}^t\right) dt. \tag{1}$$

Further, $\boldsymbol{z}^t$ can be obtained via numerical methods such as Runge-Kutta (Schober et al., 2019) and Adams (Odibat & Baleanu, 2020), producing the trajectory at every timestamp with a decoder. When it comes to multi-agent dynamical systems, Equation 1 is extended into a system of ODE equations as follows (Huang et al., 2021):

$$\boldsymbol{z}_i^T = \boldsymbol{z}_i^0 + \int_{t=0}^{T} f_i\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right) dt, \tag{2}$$

where $\boldsymbol{z}_i^t$ denotes the latent state representation for object i at the timestamp t into $\mathbf{x}_i^t$, and f_i captures the complex

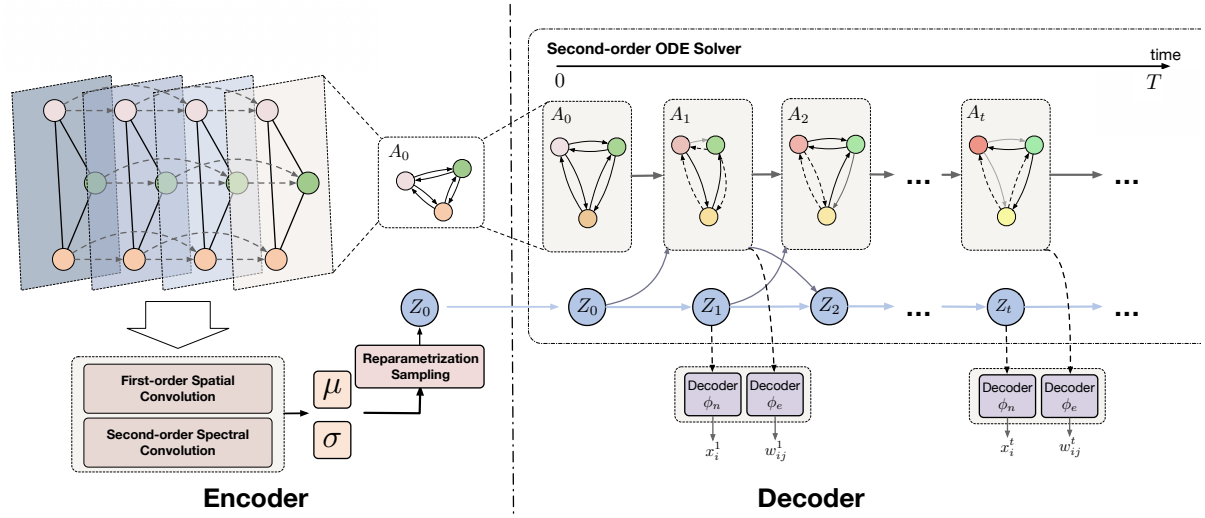

Figure 1. Overview of our proposed HOPE. To begin, the twin encoder utilizes two branches of graph convolution to extract spatio-temporal relationships for the initialization of latent state representations. Then, a generative model utilizes second-order ODEs to simulate the evolution of both nodes and edges. In the end, HOPE feeds state representations into the decoders to output the predicted nodes and edges. We maximize the evidence lower bound (ELBO) of the likelihood during optimization.

interaction among various objects.

3. Methodology

In this paper, we propose a novel method named HOPE for modeling interacting system dynamics, which learns high-order spatio-temporal correlations to enhance the model capacity and the optimization efficiency. Our HOPE is mainly comprised of three components: (1) a twin graph encoder to initialize state representations for objects, which adopts two GNN branches to explore object interactions from complementary views; (2) a continuous-time generative model which utilizes the second-order graph ODEs to model long-term dynamics in interacting systems with enhanced efficiency; (3) a decoder which outputs the prediction values for nodes and edges based on latent state representations. More details can be found in Figure 1.

3.1. Twin Graph Encoder

Our twin graph encoder seeks to learn complex spatio-temporal correlations to initialize the latent state representations of objects and edges. We begin with building a temporal graph to characterize both temporal and spatial patterns, which are then encoded into object representations by two branches of complementary graph convolution operations. On the one hand, we combine spatial convolution with the self-attention mechanism to adaptively learn neighborhood information in the temporal graph. On the other hand, a second-order spectral graph convolution is employed to explore non-neighborhood semantic information embedded in graph spectrum in a less-parametric fashion. Finally, we aggregate these temporal object representations into sequential representations to initialize the latent state of each object and edge.

Temporal Graph Construction. To concurrently describe both temporal and spatial correlations, we introduce a temporal graph where each node represents the observation of an object at a given timestamp. Our graph contains two types of edges, i.e., spatial and temporal edges. Spatial edges are based on weighted edges between two objects at the same timestamps while temporal edges are between every two consecutive observations for each object.

Specifically, in the constructed temporal graph $\mathcal{G}^H$, there are nodes i_t indicating the observations of i at time step t. The adjacent matrix $\tilde{\boldsymbol{A}}$ contains both spatial and temporal edges as follows:

$$\tilde{\boldsymbol{A}}(i_t, j_{t'}) = \begin{cases} w_{ij}^t & t = t^0 \\ 1 & i = j, t^0 = t+1 \\ 0 & \text{otherwise} \end{cases} . \tag{3}$$

First-order Spatial Convolution. In recent years, graph neural networks (GNNs) have achieved significant success in modeling graph-structured data (Kipf & Welling, 2017; Xu et al., 2019). Generally, spatial-based GNNs utilize the message passing paradigm to provide discriminative node embeddings. To be specific, at each layer, they embed node semantic features into deep representations via extracting information from the first-order neighborhood of each node To adaptively infer interaction between each node and its neighbors, we leverage the attention mechanism during the convolution process.

In detail, we first calculate the interaction scores between each central node and its neighbors, which are used to aggregate the embeddings of its neighbors at the previous layer. Formally, given the embedding $\boldsymbol{h}_i^{t,(k)}$ of i_t at the k-th layer, the interaction scores between i_t and its neighbor $j_{t'}$ in the temporal graph is derived from both the adjacent matrix and their features:

$$s^{(k)}(v_i^t, v_j^{t'}) = \tilde{\boldsymbol{A}}(i_t, j_{t'}) cos(\boldsymbol{W}_{query}\boldsymbol{h}_i^{t,(k)}, \boldsymbol{W}_{key}\boldsymbol{h}_j^{t',(k)}), \tag{4}$$

where $cos(\cdot,\cdot)$ calculates the cosine similarity between two vectors. $\boldsymbol{W}_{query}$ and $\boldsymbol{W}_{key}$ are two matrices for similarity calculation. Then the updated node representation at the $k+1$-th layer is as follows:

$$\boldsymbol{h}_i^{t,(k+1)} = \boldsymbol{h}_i^{t,(k)} + \sigma\left(\sum_{v_j^{t'} 2\boldsymbol{U}_{v_i^t}} s^{(k)}(v_i^t, v_j^{t'})\boldsymbol{W}_{value}\boldsymbol{h}_j^{t',(k)}\right) \tag{5}$$

where $\sigma(\cdot)$ denotes a non-linear activation function and $\boldsymbol{W}_{value}$ is a learnable transformation matrix. $\boldsymbol{U}_{v_i^t}$ collects the first-order neighbors of v_i^t. After stacking K layers, we can get each node representation, i.e., $\boldsymbol{h}_i^t = \boldsymbol{h}_i^{t,(K)}$ in an adaptive fashion.

Second-order Spectral Convolution. However, spatial graph convolution could be ineffective to explore non-neighborhood correlations. Toward this end, we further introduce spectral graph convolution as a supplement to explore semantic information buried in spectral domains. In particular, we leverage a less-parametric way to implicitly exploit high-order non-neighboring correlations embedded in graph spectrum from the adjacent matrix.

We first recall that the Chebyshev polynomial is defined in a recursive manner with $T_0(x) = 1$, $T_1(x) = x$ and $T_m(x) = 2xT_{m\ \ 1}(x) - T_{m\ \ 2}(x)$. All the node features are stacked into a matrix $\boldsymbol{E}^{(0)} = \boldsymbol{X}$. Given the normalized graph Laplacian $\boldsymbol{L} = \boldsymbol{I} - \tilde{\boldsymbol{D}}^{\ \frac{1}{2}}\boldsymbol{A}\tilde{\boldsymbol{D}}^{\ \frac{1}{2}}$ with the degree matrix $\tilde{\boldsymbol{D}}$ and the identity matrix $\boldsymbol{I}$, the node representation matrix at k-th layer $\boldsymbol{E}^{(k)}$ using the second-order Chebyshev graph convolution (Defferrard et al., 2016) is formulated as:

$$\boldsymbol{E}^{(k)} = \sum_{m=0}^{2} T_m(\tilde{\boldsymbol{L}})\boldsymbol{E}^{(k\ \ 1)}\boldsymbol{W}_m^{(k)} \tag{6}$$

where $\tilde{\boldsymbol{L}} = 2\boldsymbol{L}/\lambda_{\max} - \boldsymbol{I}$ and $\lambda_{\max}$ denotes the maximum eigenvalue of $\boldsymbol{L}$, and $\boldsymbol{W}_m^{(k)} \in \mathsf{R}^{d\ \ d}$ is a learnable weight matrix at the k-th layer. We also stack K layers, producing each representation vector $\boldsymbol{e}_i^t$ from $\boldsymbol{E}^{(K)} \in \mathsf{R}^{NT\ \ d}$ with semantics from a spectral perspective.

Sequence Representation Learning. Our twin branches probe spatio-temporal relationships from complementary views. To generate sequence representations for initialization, we first combine both representations and then leverage the attention mechanism to aggregate these temporal representations into a sequence representation for each object. On this basis, we obtain the posterior distributions of initial latent state representations.

In particular, we first combine both representations at the last layer with the temporal embeddings, and then utilize a multi-layer perception (MLP) to obtain the final representation for each v_i^t. In formulation,

$$\boldsymbol{q}_i^t = \delta([\boldsymbol{e}_i^t, \boldsymbol{h}_i^t]) + TE(t) \tag{7}$$

$$\mathrm{TE}(t)[2i] = \sin\left(\frac{\Delta t}{10000^{2i/d}}\right), \tag{8}$$

$$\mathrm{TE}(t)[2i+1] = \cos\left(\frac{\Delta t}{10000^{2i/d}}\right), \tag{9}$$

where $[2i]$ and $[2i+1]$ represent the element indexes for the even and odd positions in temporal embeddings, respectively and $\delta(\cdot)$ is an learnable MLP projector.

Afterward, we summarize these node representations at every timestamp into a summarized representation $\boldsymbol{u}_i$ by generating weights for each timestamp and then taking the weighted average. In particular, for each object, our HOPE first takes the average of all observations to generate weights for different timestamps. In formulation,

$$\alpha_i^t = tanh((\frac{1}{T}\sum_{t=1}^{T}\boldsymbol{q}_i^t)\boldsymbol{W}_{sum}) \cdot \boldsymbol{q}_i^t, \tag{10}$$

where $\boldsymbol{W}_{sum}$ is a learnable matrix for linear transformation. Then, we take the weighted average of all observations with a non-linear activation function $\sigma(\cdot)$ embedded to provide non-linearity as follows:

$$\boldsymbol{u}_i^t = \frac{1}{T}\sum_{t=1}^{T}\sigma(\alpha_i^t\boldsymbol{q}_i^t). \tag{11}$$

In the end, we initialize state representations for the sequential generative model by sampling from the approximate posterior distribution, i.e., $q(\boldsymbol{Z}^0|\mathcal{X},\mathcal{A})$ based on sequence representations. Moreover, the posterior distribution should approach a prior distribution $p(\boldsymbol{Z}^0)$ during optimization as an regularization. To accomplish this, we measure the mean and variance of the posterior distribution to minimize their difference and employ the "reparametrization" trick (Kingma & Welling, 2013) when initializing state vectors $\boldsymbol{z}_i^0$ and $\boldsymbol{\Pi}_{ij}^0$ for object i and the edge between nodes i and j, respectively from the posterior distribution. In formulation, we have:

$$q(\boldsymbol{z}_i^0|\mathcal{X},\mathcal{A}) = \mathcal{N}(\psi^m(\boldsymbol{u}_i), \psi^v(\boldsymbol{u}_i)), \tag{12}$$

$$\boldsymbol{z}_i^0 \sim p(\boldsymbol{z}_i^0) \approx q(\boldsymbol{z}_i^0|\mathcal{X},\mathcal{A}), \tag{13}$$

$$\mathbf{\Pi}_{ij}^0 = \psi^e(\boldsymbol{z}_i^0||\boldsymbol{z}_j^0||\boldsymbol{z}_i^0 \odot \boldsymbol{z}_j^0), \tag{14}$$

where $\mathcal{N}$ denotes a normal distribution, $\psi^m(\boldsymbol{u}_i)$ and $\psi^v(\boldsymbol{u}_i)$ are two MLPs to calculate its mean and variance, $||$ denotes the concatenation of vectors, $\odot$ denotes the Hadamard product of the two vectors, and ψ^e is an MLP to initialize the state representations for edges.

3.2. Neural Coupled Graph ODE Model

With the latent states for basic constituents (i.e., nodes and edges) in the graph, we introduce a neural graph ODE model to study the dynamics of interacting systems in a generative manner. Existing methods often adopt first-order ODE functions, which are incapable of effectively capturing long-term dependencies. Worse yet, first-order neural ODEs have been shown lack of efficiency due to the requirement for a high number of function evaluations (NFEs) in both optimization and inference (Norcliffe et al., 2020). To address these issues, we aim to incorporate second-order neural ODEs into continuous GNNs. Second-order ODEs can model the majority of underlying high-order laws in complicated physical systems which first-order ODEs fail to model. Additionally, second-order ODEs have been investigated in various numeric optimization scenarios with accelerated convergence rates (Zhang et al., 2019; Polyak, 1964; Xia et al., 2021).

To be specific, we introduce a second-order graph ODE function on latent state representations, which is written as:

$$\frac{d^2\boldsymbol{Z}^t}{dt^2} + \gamma\frac{d\boldsymbol{Z}^t}{dt} = \sigma((\hat{\boldsymbol{D}}^t)^{-1}\hat{\boldsymbol{A}}^t\boldsymbol{Z}^t\boldsymbol{W}_p) - \boldsymbol{Z}^t, \tag{15}$$

where $\boldsymbol{Z}^t = [\boldsymbol{z}_1^t, \cdots, \boldsymbol{z}_N^t] \in \mathsf{R}^{N \times d}$ represents the latent state representation matrix at timestamp t, $\boldsymbol{W}_p \in \mathsf{R}^{d \times d}$ is matrix for linear transformation, γ is a pre-defined coefficient to balance the first-order and second-order ODEs, $\hat{\boldsymbol{A}}^t$ denotes the updated adjacent which will be introduced later, and $\hat{\boldsymbol{D}}^t$ is a degree matrix used to normalize $\hat{\boldsymbol{A}}^t$. The right hand side term follows the paradigm of message passing to discover spatial relations in the system. Therefore, using a numerical solver to solve Equation 15 is equivalent to aggregating neighborhood information and then iteratively updating state representations as time passes.

Further, to acquire the updated adjacent matrix $\hat{\boldsymbol{A}}^t$ embedded with the dynamic edge information, we introduce a second ODE to simulate the development of edges, which is determined by both node state representations and the current edge states. In detail, the ODE function for edges is written as follows:

$$\frac{d^2\mathbf{\Pi}_{ij}^t}{d^2t} + \gamma\frac{d\mathbf{\Pi}_{ij}^t}{dt} = \rho_n\left(\left[\boldsymbol{z}_i^t\|\boldsymbol{z}_j^t\|\boldsymbol{z}_i^t \odot \boldsymbol{z}_j^t\right]\right) + \rho_e\left(\mathbf{\Pi}_{ij}^t\right), \tag{16}$$

where ρ_n and ρ_e are two MLPs to transform the input to embedding space R^d. Finally, a projector ρ_s transfers each edge state $\mathbf{\Pi}_{ij} \in \mathsf{R}^d$ to a scalar in the updated adjacent matrix as follows:

$$\hat{\boldsymbol{A}}_{ij}^t = \rho_s(\mathbf{\Pi}_{ij}). \tag{17}$$

An Optimization Perspective. Taking Equation 15 as an example, we illustrate our second-order ODE function using the momentum optimization algorithm. To begin with, we extend the momentum optimization algorithm (Polyak, 1964) to graphs as follows:

$$\boldsymbol{Z}^{t+1} = (1-\lambda)\boldsymbol{Z}^t + \lambda\sigma((\hat{\boldsymbol{D}}^t)^{-1}\hat{\boldsymbol{A}}^t\boldsymbol{Z}^t\boldsymbol{W}_p) + \beta\left(\boldsymbol{Z}^t - \boldsymbol{Z}^{t-1}\right), \tag{18}$$

where $\beta > 0$ is a parameter to control the updating rule. Equation 18 is comprised of three terms: the first one denotes the state representation matrix at the last timestamp; the second one provides the neighborhood information following the message passing paradigm; and the last one is the momentum term recording the updating trend, which is commonly used in accelerating the convergence in optimization algorithms (Polyak, 1964; Sutskever et al., 2013).

Lemma 3.1. *Given the momentum updating algorithm for GNNs, we have Equation 15 when $\lambda \to 0$.*

To facilitate the off-the-shelf ODE solvers during implementation, we transform the second-order ODE into a first-order ODE using the following lemma.

Lemma 3.2. *Our ODE formalization can be transformed into an augmented first-order ODE.*

Discussion. We regard second-order ODEs as a specific type of augmented first-order ODEs (Dupont et al., 2019), which expands the feature space to increase the expressive capability. Differently, our augmented features come from the gradients, i.e., $\dot{\boldsymbol{Z}}^t$, which has a definite form with better interpretability for modeling long-term dependency in dynamical interacting systems.

Based on the augmented first-order ODE expression, we can also show the local uniqueness of solutions to our ODE system. This analysis of uniqueness can help us design the architectures of neural networks (Kong et al., 2020; Chen et al., 2018).

Lemma 3.3. *Given the initial state $(t_0, \mathbf{Z}^{t_0})$, we claim that there exists $\varepsilon > 0$, s.t. Equation 15 has a unique solution in the interval $[t_0 - \varepsilon, t_0 + \varepsilon]$ when the activation function $\sigma(\cdot)$ is ReLU and $\boldsymbol{A}^t$ is continuous.*

The proofs of three lemmas can be found in Appendix A, B and C, respectively.

3.3. Decoder and Optimization

In this part, we introduce two decoders to reconstruct the inputs (i.e., nodes and edges) and then formalize the training optimization procedure for the whole framework.

Table 1. Results of compared methods on COVID-19 with the prediction length one week, two weeks and three weeks. **Bold** numbers indicate the best performance whereas underline numbers indicate the second best performance.

Methods	1-week-ahead			2-week-ahead			3-week-ahead			Average		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
LSTM	231.7	370.4	0.0816	482.3	771.8	0.1608	547.5	865.9	0.1809	420.5	669.4	0.1411
GRU	234.9	373.8	0.0786	491.8	780.3	0.1544	555.3	874.0	0.1739	427.3	676.0	0.1356
NODE	103.1	184.7	0.0349	183.7	293.4	0.0516	255.7	400.1	0.0677	180.8	292.7	0.0514
HBNODE	243.8	383.6	0.0712	220.1	380.8	0.0527	283.3	464.2	0.0767	249.1	409.5	0.0669
DGCRN	102.4	161.5	0.0279	191.7	301.0	0.0479	281.4	428.1	0.0739	191.8	296.9	0.0499
MPNODE	152.7	237.5	0.0357	272.0	549.4	0.0527	248.7	385.8	0.0696	224.5	390.9	0.0527
CG-ODE	91.59	146.9	0.0255	182.5	298.9	0.0431	251.1	385.6	0.0632	175.1	277.1	0.0439
HOPE	**85.64**	**146.0**	**0.0228**	**180.9**	**275.2**	**0.0397**	**243.1**	**373.3**	**0.0612**	**169.9**	**264.8**	**0.0412**

Table 2. Results of compared methods on Social Network with the prediction length 10, 20 and 40. **Bold** numbers indicate the best performance whereas underline numbers indicate the second best performance.

Methods	10			20			40			Average		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
LSTM	0.0933	0.1367	0.1269	0.1880	0.2719	0.2748	0.3677	0.5268	0.6276	0.2163	0.3118	0.3431
GRU	0.0939	0.1374	0.1280	0.1844	0.2724	0.2772	0.3916	0.5478	0.6566	0.2233	0.3192	0.3539
NODE	0.1276	0.1630	0.1161	0.2430	0.3066	0.2521	0.5383	0.6742	0.6025	0.3030	0.3813	0.3236
HBNODE	0.1230	0.1596	0.1149	0.2701	0.3358	0.2635	0.4877	0.6247	0.5934	0.2936	0.3734	0.3239
DGCRN	0.0913	0.1284	0.1192	0.2337	0.3046	0.2558	0.4764	0.5723	0.5614	0.2671	0.3351	0.3121
MPNODE	0.0887	0.1198	0.1151	0.1792	0.2554	0.2549	0.3678	0.5239	0.5214	0.2119	0.2997	0.2971
CG-ODE	0.0852	0.1236	0.1205	0.2073	0.2436	0.2432	0.3199	**0.4871**	0.4909	0.2041	0.2848	0.2849
HOPE	**0.0796**	**0.1167**	**0.1050**	**0.1543**	**0.2203**	**0.2174**	**0.3019**	0.4873	**0.4867**	**0.1786**	**0.2748**	**0.2697**

To be specific, our HOPE predicts object trajectories based on their latent states. To this end, we utilize two decoders to output the likelihood for each observation and interaction, i.e., $p(\boldsymbol{x}_i^t|\boldsymbol{z}_i^t)$ and $p(w_{ij}^t|\boldsymbol{\Pi}_{ij}^t)$, respectively. To simplify the computation, we merely output the mean of the distribution as follows:

$$\boldsymbol{\mu}_i^t = \phi_n(\boldsymbol{z}_i^t); \mu_{ij}^t = \phi_e(\boldsymbol{\Pi}_{ij}^t)), \tag{19}$$

where ϕ_n and ϕ_e are two different functions parameterized by two MLPs.

In our implementation, we partition each training example into two segments based on the time and utilize the first part to predict the second part. Then, the framework is optimized with the variational inference paradigm (Kingma & Welling, 2013), i.e., maximizing the evidence lower bound (ELBO) of the likelihood. In particular, we not only maximize the likelihood for nodes and edges, but also minimize the Kullback-Leibler (KL) divergence between the prior and posterior distributions. The objective is formalized as:

$$\begin{aligned}\ell_{ELBO} = & \mathsf{E}_{Z^0 \ \prod_{i=1}^N q\left(z_i^0 | X, \mathcal{A}\right)}\left[\log p(X, \mathcal{A})\right] \\ & - \mathrm{KL}\left[\prod_{i=1}^N q(\boldsymbol{z}_i^0|\mathcal{X}, \mathcal{A}) \| p\left(\boldsymbol{Z}^0\right)\right],\end{aligned} \tag{20}$$

where $KL(\cdot||\cdot)$ denotes the Kullback-Leibler (KL) divergence. After considering the likelihood independently for each node and each edge, Equation 20 can be rewritten as:

$$\begin{aligned}\ell_{ELBO} = & -\sum_i \sum_t \frac{\left\|\mathbf{x}_i^t \ \mu_i^t\right\|^2}{2\sigma^2} - \sum_i \sum_j \sum_t \frac{\left\|w_{ij}^t \ \mu_{ij}^t\right\|^2}{2\sigma^2} \\ & - \mathrm{KL}\left[\prod_{i=1}^N q(\boldsymbol{z}_i^0|\mathcal{X}, \mathcal{A}) \| p\left(\boldsymbol{Z}^0\right)\right],\end{aligned} \tag{21}$$

where σ^2 denotes the variance of the prior distribution. The overall algorithm is provided in Appendix.

4. Experiments

4.1. Experimental Settings

Datasets and Data Split. To evaluate our model, we utilize three datasets of interacting dynamical systems, i.e., **COVID-19** (Dong et al., 2020), **Social Network** (Gu et al., 2017) and **Spring Ocsillator** (Kipf et al., 2018). **COVID-19** contains daily tendency records from the Johns Hopkins University (JHU) Center for Systems Science and Engineering [1]. Following recent works (Huang et al., 2021), five dynamic features along with one static feature are selected as object attributes. **Social Network** models opinions migrating from individuals to individuals in a social network. **Spring Ocsillator** models a group of balls linked with springs. The location and velocity vectors are adopted as node representations. To facilitate our model optimization, each training

[1] https://github.com/CSSEGISandData/COVID-19

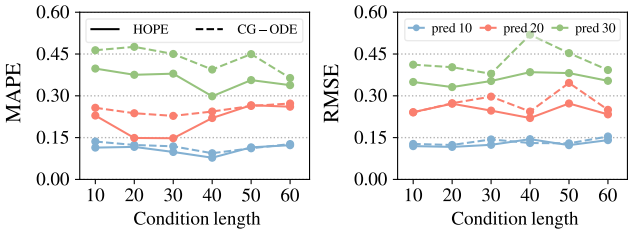

Figure 2. Performance comparison in different settings on Social Network. Our HOPE can achieves better performance in most cases.

Table 3. Results of compared methods on Spring Ocsillator with the prediction length 36, 48 and 60.

Methods	36		48		60	
	MAE	RMSE	MAE	RMSE	MAE	RMSE
LSTM	0.2661	0.3401	0.4120	0.5321	0.6257	0.7775
GRU	0.3110	0.3910	0.3940	0.4997	0.6369	0.7950
NODE	0.2757	0.3491	0.4569	0.5702	0.6259	0.7770
HBNODE	0.2969	0.3792	0.4512	0.5778	0.6486	0.8086
DGCRN	0.2798	0.3623	0.4013	0.5296	0.6159	0.7683
MPNODE	0.3473	0.4340	0.4161	0.5199	0.6133	0.7648
CG-ODE	0.2723	0.3515	0.4178	0.5382	0.6180	0.7729
HOPE	**0.2649**	**0.3387**	**0.3847**	**0.4841**	**0.5883**	**0.7359**

graph sequence is separated into a conditional segment and a prediction segment where the conditional segment is served as the input for the model encoder, while the prediction segment is used as supervision. Condition length and prediction length denote the size of two segments, respectively. We also make sure that there is no sequence overlap on training and testing sets as indicated in (Huang et al., 2021).

Baselines. We compare our models with a range of state-of-the-art methods, including three neural network-based methods (**LSTM** (Hochreiter & Schmidhuber, 1997), **GRU** (Cho et al., 2014) and **DGCRN** (Li et al., 2022)) and four neural ODE-based methods (**NODE** (Chen et al., 2018), **HBNODE** (Xia et al., 2021), **MPNODE** (Gupta et al., 2022) and **CG-ODE** (Huang et al., 2021)). Their details can be found in Appendix F.

Evaluation Metrics. To evaluate these competing methods, we leverage three standard evaluation metrics, i.e., Mean Absolute Error (MAE), Root Mean Squared Error (RMSE), and Mean Absolute Percentage Error (MAPE). Since the ground truth values in Spring Oscillator are shaking around zero, the metric MAPE is deprecated here. In addition, we vary the prediction length and conditional length with the same trend, which can evaluate competing methods both in the long run and short run.

Implementation Details. We adopt Pytorch (Paszke et al., 2017) and torchdiffeq (Kidger et al., 2021) for implementing all the baselines. In our method, we use one-layer graph convolution in the graph encoder. The coefficient γ is set to 3 as default. The parameter sensitivity will be further explored in Section 4.5. The embedding dimension of hidden embeddings is set to 64. During training, an Adam optimizer is used with the learning rate set to $5e-3$ and weight decay set to $1e-5$. The dropout rate is set to 0.2. The batch size is set to 8 and we train the model for 100 epochs. For fair comparisons, the parameters in the compared baselines are first set according to the corresponding papers and then tuned for the best performance.

4.2. Performance Comparison

Table 1, Table 2 and Table 3 summarize the performance of competing methods with varying prediction lengths on COVID-19, Social Network and Spring Ocsillator, respectively. From the compared results, we have the following observations: First, although the temporal GNN method (DGCRN) can obtain competitive performance on Spring Oscillator, it still performs much worse than most of the neural ODE methods (i.e., NODE, CG-ODE and HOPE). Therefore, modeling the continuous evolution of nodes and edges is indispensable in dynamical systems. Second, Graph ODE methods (i.e., CG-ODE and HOPE) overall perform better than neural ODE methods (i.e., NODE and HBNODE), implying that learning spatial relationships is crucial for modeling dynamical interacting systems. Third, our method outperforms all these baselines in most cases on three datasets. In particular, compared with the best baseline CG-ODE, our HOPE reduces the prediction error of 2.97%, 4.44% and 6.15% in terms of average MAE, RMSE and MAPE on COVID-19, respectively. We attribute the significant decreasement of prediction errors to two reasons. (i) Introduction of the second-order spectral convolution expands the receptive field of the encoder, improving the initialization of the latent state representations. (ii) Introduction of the second-order graph ODE allows the generative model to explore long-term dependency, which enhances model expressivity.

Effect of Different Condition and Prediction Lengths. Finally, we analyze the prediction performance with regard to various condition lengths and prediction lengths. In particular, we vary the condition length and prediction length in $\{10, 20, 30, 40, 50, 60\}$ and $\{10, 20, 30\}$, respectively. The compared results of CG-ODE and HOPE on Social Network is shown in Figure 2. It can be seen that our HOPE achieves better performance in most cases in terms of MAPE, which validates the superiority of our HOPE. In addition, we can find that when conducting the long-run prediction, the gap between CG-ODE and HOPE tends to be larger in most cases. Perhaps the reason is that the long-run prediction is more dependent on high-order dependency which can be captured in our HOPE.

Table 4. Ablation study on Social Network. **Bold** numbers indicate the best performance whereas underline numbers indicate the second best performance.

Methods	10			20			40			Avgence		
	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE	MAE	RMSE	MAPE
HOPE w.o. FC	0.09910	0.1222	0.1172	0.1856	0.2669	0.2624	0.3934	0.5360	0.5421	0.2260	0.3084	0.3072
HOPE w.o. FO	0.08477	0.1334	0.1237	0.2018	0.2761	0.2835	0.3244	0.5163	0.4935	0.2037	0.3086	0.3002
HOPE w.o. SC	0.08322	0.1169	0.1122	**0.1520**	0.2384	0.2682	0.3167	**0.4700**	0.5782	0.1840	0.2751	0.3195
HOPE w.o. SO	0.08067	0.1212	0.1099	0.1793	0.2497	0.2223	0.3029	0.4847	0.6470	0.1876	0.2852	0.3264
HOPE w.o. E	0.09612	0.1358	0.1249	0.1879	0.2648	0.2616	0.3612	0.6249	0.5238	0.2151	0.3418	0.3034
Full Model	**0.07958**	**0.1167**	**0.1050**	0.1543	**0.2203**	**0.2174**	**0.3019**	0.4873	**0.4867**	**0.1786**	**0.2748**	**0.2697**

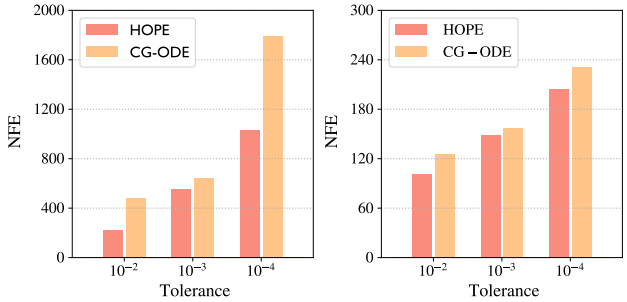

Figure 3. NFEs comparison between CG-ODE and HOPE on COVID-19 (left) and Social Network (right). Our model outperforms CG-ODE in terms of efficiency.

4.3. Efficiency Analysis

In this part, we evaluate the efficiency of two graph ODE models (i.e., CG-ODE and HOPE) by comparing the number of function evaluation (NFE) on forward or backward process. A higher NFE means a lower convergence rate in ODE solving process, which brings more computational resources consumption and more time cost. In particular, we vary the error tolerance using the same ODE solver to show the efficiency of these methods under different circumstances. The results on COVID-19 and Social Network are shown in Figure 3 and we can observe similar results on Spring Oscillator. From the results, it can be observed that the NFEs of both HOPE and CG-ODE both rise when the tolerance decreases. Since a lower error tolerance usually indicates a high accuracy, we can conclude a trade-off between the prediction accuracy and the computation cost. In addition, we can observe that HOPE requires fewer NFEs when the tolerance is the same, which indicates that HOPE is more efficient with a higher convergence rate using the second-order ODEs. This conclusion is also in accordance with the acceleration of the momentum updating algorithm.

4.4. Ablation Study

Furthermore, we conduct extensive ablation experiments to demonstrate the performance of the key components in HOPE. To be specific, we introduce the following model variants: (1): HOPE w.o. FC. It removes the first-order spatial convolution and only utilizes the second-order spectral convolution in the encoder. (2) HOPE w.o. SC. It removes the second-order spectral convolution and only utilizes the first-order spatial convolution in the encoder. (3) HOPE w.o. FO. It removes the term of the first-order derivative by setting γ in Equation 15. (4) HOPE w.o. SO. It removes the term of the second-order derivative in Equation 15. (5) HOPE w.o. E. It does not model the evolution of edge evolution using neural ODEs.

Table 4 shows the compared results on Social Network. We can have several observations as follows: (1) HOPE is superior to HOPE w.o. FC and HOPE w.o. SC mostly, which indicates that both first-order spatial and second-order spectral graph convolution is indispensable for learning complex spatio-temporal correlations in the temporal graph. (2) Without the second-order ODEs, HOPE w.o. SO achieves worse performance compared with our full model, which indicates the importance of capturing long-term dependency in complex dynamical systems. (3) Our full model outperforms HOPE w.o. FO consistently, which implies that modeling the short-term dependency in Equation 15 is also beneficial to the performance. (4) The performance achieved by HOPE w.o. E is much poorer than the full model, which demonstrates the importance of modeling the evolution of edges in dynamical interacting systems.

4.5. Sensitivity Analysis

In this part, we study how the hyperparameters (i.e., number of the convolution layer K and coefficient γ in Equation 15 influence the prediction performance of our HOPE in Figure 4 with varying prediction lengths. To begin with, we vary the layer number K with all other parameters fixed and reveal the performance in the left column. From the results, it can be observed that the performance is optimal with a one-layer twin graph encoder. Perhaps the reason is that the encoder could suffer from the over-smoothing issue. Furthermore, we analyze the influence of the coefficient γ in the right column. It can be found that when γ is zero, the performance tends to decrease, which validates the necessity of modeling the first-order dependency once more. Moreover, our HOPE

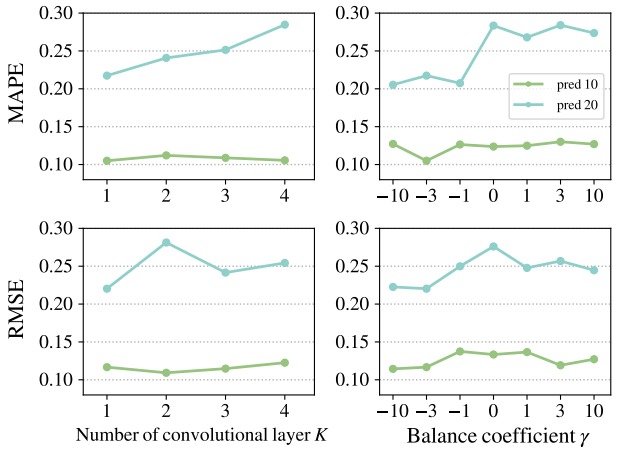

Figure 4. Performance sensitivity of K and γ on Social Network with different prediction lengths.

can obtain considerable performance when γ is around -3. Therefore, we set K and γ to 1 and -3, respectively.

5. Related Work

Neural Ordinary Differential Equation. Motivated by the fact that ResNets can be formalized as discretized Ordinary Differential Equations (ODEs) (Chen et al., 2018), a number of neural ODE approaches have been developed (Chen et al., 2018; Dupont et al., 2019), which characterize a e continuous-depth model by parameterizing the derivative of the hidden state. In recent years, numerous attempts have been made to increase the expressive capability of neural ODEs. For instance, Augmented Neural ODEs (Dupont et al., 2019) enlarge the embedding space, which enables neural ODEs to learn features with a different topology from the input space. In addition, a variety of techniques have integrated neural ODEs into GNNs (Xhonneux et al., 2020; Zhang et al., 2022; Qin et al., 2023). However, these models focus on designing continuous models to release oversmoothing on static graphs. Even worse, they do not study the transferability of the inductive bias to changing graph structure. In contrast, our work targets at dynamical interacting systems with complex spatio-temporal relationships and improves the performance on dynamic graph structures with coupled ODE systems.

Interacting Dynamical Systems Analysis. In recent years, learning from interacting dynamical systems has attracted considerable interest for a variety of applications, including traffic flow forecasting (Wang et al., 2020; Lan et al., 2022; Zhao et al., 2023) and stock price prediction (Deng et al., 2019; Sawhney et al., 2021; Yoo et al., 2021). To collect complex interacting information, a variety of GNN-based algorithms have been developed, which integrate GNNs with various sequential methodologies (Pan et al., 2021; Yuan et al., 2021; Chen et al., 2022; Dai et al., 2020; Sun et al., 2019; Guo et al., 2019). These approaches typically generate graphs based on distance and learn temporal and spatial dependence using various architectures in dynamical systems. In spite of their performance, the bulk of these works depend on the premise that the observation of each item is available at each timestamp. Nonetheless, this assumption cannot be compatible with the ground truth in practice (Huang et al., 2020; 2021). To address this issue, we investigate the architectures of neural ODEs for modeling interacting dynamics and offer a model HOPE to capture high-order dependency in dynamical interacting systems.

6. Conclusion

This paper studies the problem of modeling dynamic interaction systems and introduce a novel graph ODE method named HOPE to tackle it. The core of our HOPE is to learn high-order relationships from dynamic interaction data. HOPE adopts a graph encoder, which consists of two branches to capture complex spatio-temporal relationships in complementary manners. More importantly, it utilizes second-order graph ODE functions which model the dynamics for both nodes and edges based on their embeddings respectively, which enables efficient learning of long-term dependencies from complex dynamical systems. Experiment results on a variety of datasets demonstrate the effectiveness of our proposed method in terms of both accuracy and efficiency.

Limitations. While the incorporation of high-order dynamics in our HOPE model offers advantages in terms of improved performance and faster convergence, it also comes with the trade-off of increased computational space, particularly when implemented through augmented first-order ODEs. As a result, our proposed HOPE model may encounter scalability challenges when applied to large-scale dynamic systems. In some cases, even more complex higher-order dynamics may be present, which cannot be adequately captured by our second-order model. This limitation indicates that further enhancements may be necessary to address such complexities in specific applications. To overcome these limitations, our future research will concentrate on refining the implementation using parallel optimization techniques, which could help mitigate the scalability challenges. Furthermore, expanding the model to accommodate more generalized higher-order dynamics will enhance its versatility for a wider spectrum of dynamical systems.

Acknowledgement

This work was partially supported by NSF 2211557, NSF 1937599, NSF 2119643, NSF 2303037, NASA, SRC, Okawa Foundation Grant, Amazon Research Awards, Cisco research grant, Picsart Gifts, and Snapchat Gifts.

References

Aydin, M. N. and Coban, R. Pid sliding surface-based adaptive dynamic second-order fault-tolerant sliding mode control design and experimental application to an electromechanical system. *International Journal of Control*, 95(7):1767–1776, 2022.

Balcilar, M., Guillaume, R., Héroux, P., Gaüzère, B., Adam, S., and Honeine, P. Analyzing the expressive power of graph neural networks in a spectral perspective. In *ICLR*, 2021.

Battaglia, P., Pascanu, R., Lai, M., Jimenez Rezende, D., et al. Interaction networks for learning about objects, relations and physics. In *NeurIPS*, 2016.

Benson, A. and Kleinberg, J. Link prediction in networks with core-fringe data. In *WWW*, pp. 94–104, 2019.

Berndt, D. J. and Clifford, J. Using dynamic time warping to find patterns in time series. pp. 359–370, 1994.

Bruna, J., Zaremba, W., Szlam, A., and LeCun, Y. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

Chen, J., Chen, Z., Zheng, L., and Chen, X. A spatio-temporal data-driven automatic control method for smart home services. In *WWW*, pp. 948–955, 2022.

Chen, R. T., Rubanova, Y., Bettencourt, J., and Duvenaud, D. K. Neural ordinary differential equations. In *NeurIPS*, 2018.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Cranmer, M., Sanchez Gonzalez, A., Battaglia, P., Xu, R., Cranmer, K., Spergel, D., and Ho, S. Discovering symbolic models from deep learning with inductive biases. In *NeurIPS*, pp. 17429–17442, 2020.

Dai, R., Xu, S., Gu, Q., Ji, C., and Liu, K. Hybrid spatio-temporal graph convolutional network: Improving traffic prediction with navigation data. In *KDD*, pp. 3074–3082, 2020.

Defferrard, M., Bresson, X., and Vandergheynst, P. Convolutional neural networks on graphs with fast localized spectral filtering. In *NeurIPS*, 2016.

Deng, S., Zhang, N., Zhang, W., Chen, J., Pan, J. Z., and Chen, H. Knowledge-driven stock trend prediction and explanation via temporal convolutional network. In *WWW*, pp. 678–685, 2019.

Dong, E., Du, H., and Gardner, L. An interactive web-based dashboard to track covid-19 in real time. *The Lancet Infectious Diseases*, 20(5):533–534, 2020.

Dupont, E., Doucet, A., and Teh, Y. W. Augmented neural odes. In *NeurIPS*, 2019.

Fan, W., Ma, Y., Li, Q., He, Y., Zhao, E., Tang, J., and Yin, D. Graph neural networks for social recommendation. In *WWW*, pp. 417–426, 2019.

Groß, A., Kracher, B., Kraus, J. M., Kühlwein, S. D., Pfister, A. S., Wiese, S., Luckert, K., Pötz, O., Joos, T., Van Daele, D., et al. Representing dynamic biological networks with multi-scale probabilistic models. *Communications Biology*, 2(1):1–12, 2019.

Gu, Y., Sun, Y., and Gao, J. The co-evolution model for social network evolving and opinion migration. In *KDD*, pp. 175–184, 2017.

Guo, S., Lin, Y., Feng, N., Song, C., and Wan, H. Attention based spatial-temporal graph convolutional networks for traffic flow forecasting. In *AAAI*, pp. 922–929, 2019.

Gupta, J. K., Vemprala, S., and Kapoor, A. Learning modular simulations for homogeneous systems. In *NeurIPS*, 2022.

Hackl, J., Scholtes, I., Petrović, L. V., Perri, V., Verginer, L., and Gote, C. Analysis and visualisation of time series data on networks with pathpy. In *Companion Proceedings of the Web Conference*, pp. 530–532, 2021.

Hafiene, N., Karoui, W., and Romdhane, L. B. Influential nodes detection in dynamic social networks: A survey. *Expert Systems with Applications*, 159:113642, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *NeurIPS*, 2017.

Hao, Z., Hao, J., Peng, Z., Wang, S., Yu, P. S., Wang, X., and Wang, J. Dy-hien: Dynamic evolution based deep hierarchical intention network for membership prediction. In *WSDM*, pp. 363–371, 2022.

He, X., Deng, K., Wang, X., Li, Y., Zhang, Y., and Wang, M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pp. 639–648, 2020.

Henaff, M., Bruna, J., and LeCun, Y. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997.

Hsieh, T.-Y., Wang, S., Sun, Y., and Honavar, V. Explainable multivariate time series classification: a deep neural network which learns to attend to important variables as well as time intervals. In *WSDM*, pp. 607–615, 2021.

Huang, C., Wu, X., Zhang, X., Zhang, C., Zhao, J., Yin, D., and Chawla, N. V. Online purchase prediction via multi-scale modeling of behavior dynamics. In *KDD*, pp. 2613–2622, 2019.

Huang, Z., Sun, Y., and Wang, W. Learning continuous system dynamics from irregularly-sampled partial observations. In *NeurIPS*, pp. 16177–16187, 2020.

Huang, Z., Sun, Y., and Wang, W. Coupled Graph ODE for Learning Interacting System Dynamics. In *KDD*, 2021.

Ju, W., Fang, Z., Gu, Y., Liu, Z., Long, Q., Qiao, Z., Qin, Y., Shen, J., Sun, F., Xiao, Z., et al. A comprehensive survey on deep graph representation learning. *arXiv preprint arXiv:2304.05055*, 2023a.

Ju, W., Gu, Y., Luo, X., Wang, Y., Yuan, H., Zhong, H., and Zhang, M. Unsupervised graph-level representation learning with hierarchical contrasts. *Neural Networks*, 158:359–368, 2023b.

Kidger, P., Chen, R. T. Q., and Lyons, T. J. "hey, that's not an ode": Faster ode adjoints via seminorms. *ICML*, 2021.

Kim, D. and Oh, A. How to find your friendly neighborhood: Graph attention design with self-supervision. *arXiv preprint arXiv:2204.04879*, 2022.

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

Kipf, T., Fetaya, E., Wang, K.-C., Welling, M., and Zemel, R. Neural relational inference for interacting systems. In *ICML*, pp. 2688–2697, 2018.

Kipf, T. N. and Welling, M. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

Kong, L., Sun, J., and Zhang, C. Sde-net: Equipping deep neural networks with uncertainty estimates. *arXiv preprint arXiv:2008.10546*, 2020.

Lan, S., Ma, Y., Huang, W., Wang, W., Yang, H., and Li, P. Dstagnn: Dynamic spatial-temporal aware graph neural network for traffic flow forecasting. In *ICML*, pp. 11906–11917, 2022.

Li, F., Feng, J., Yan, H., Jin, G., Yang, F., Sun, F., Jin, D., and Li, Y. Dynamic graph convolutional recurrent network for traffic prediction: Benchmark and solution. *ACM Transactions on Knowledge Discovery from Data*, 2022.

Liao, S., Liang, S., Meng, Z., and Zhang, Q. Learning dynamic embeddings for temporal knowledge graphs. In *WSDM*, pp. 535–543, 2021.

Norcliffe, A., Bodnar, C., Day, B., Simidjievski, N., and Liò, P. On second order behaviour in augmented neural odes. In *NeurIPS*, 2020.

Odibat, Z. and Baleanu, D. Numerical simulation of initial value problems with generalized caputo-type fractional derivatives. *Applied Numerical Mathematics*, 156:94–105, 2020.

Pan, Z., Ke, S., Yang, X., Liang, Y., Yu, Y., Zhang, J., and Zheng, Y. Autostg: Neural architecture search for predictions of spatio-temporal graph. In *WWW*, pp. 1846–1855, 2021.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Perantonis, S. J. and Karras, D. A. An efficient constrained learning algorithm with momentum acceleration. *Neural Networks*, 8(2):237–249, 1995.

Poli, M., Massaroli, S., Park, J., Yamashita, A., Asama, H., and Park, J. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *Ussr Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Qin, Y., Ju, W., Wu, H., Luo, X., and Zhang, M. Learning graph ode for continuous-time sequential recommendation. *arXiv preprint arXiv:2304.07042*, 2023.

Qu, L., Zhu, H., Duan, Q., and Shi, Y. Continuous-time link prediction via temporal dependent graph neural network. In *WWW*, pp. 3026–3032, 2020.

Sawhney, R., Agarwal, S., Wadhwa, A., Derr, T., and Shah, R. R. Stock selection via spatiotemporal hypergraph attention network: A learning to rank approach. In *AAAI*, pp. 497–504, 2021.

Schober, M., Särkkä, S., and Hennig, P. A probabilistic model for the numerical solution of initial value problems. *Statistics and Computing*, 29(1):99–122, 2019.

Sun, C., Karlsson, P., Wu, J., Tenenbaum, J. B., and Murphy, K. Stochastic prediction of multi-agent interactions from partial observations. In *ICLR*, 2019.

Sutskever, I., Martens, J., Dahl, G., and Hinton, G. On the importance of initialization and momentum in deep learning. In *ICML*, pp. 1139–1147, 2013.

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. Graph attention networks. In *ICLR*. 2018.

Wang, R. and Yu, R. Physics-guided deep learning for dynamical systems: A survey. *arXiv preprint arXiv:2107.01272*, 2021.

Wang, X., He, X., Wang, M., Feng, F., and Chua, T.-S. Neural graph collaborative filtering. In *SIGIR*, pp. 165–174, 2019.

Wang, X., Ma, Y., Wang, Y., Jin, W., Wang, X., Tang, J., Jia, C., and Yu, J. Traffic flow prediction via spatial temporal graph neural network. In *WWW*, pp. 1082–1092, 2020.

Wang, Y., Wang, W., Liang, Y., Cai, Y., and Hooi, B. Curgraph: Curriculum learning for graph classification. In *WWW*, pp. 1238–1248, 2021.

Wu, M., Pan, S., Zhou, C., Chang, X., and Zhu, X. Unsupervised domain adaptive graph convolutional networks. In *WWW*, pp. 1457–1467, 2020.

Xhonneux, L.-P., Qu, M., and Tang, J. Continuous graph neural networks. In *ICML*, pp. 10432–10441, 2020.

Xia, H., Suliafu, V., Ji, H., Nguyen, T., Bertozzi, A., Osher, S., and Wang, B. Heavy ball neural ordinary differential equations. In *NeurIPS*, pp. 18646–18659, 2021.

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. How powerful are graph neural networks? In *ICLR*, 2019.

Yoo, J., Soun, Y., Park, Y.-c., and Kang, U. Accurate multivariate stock movement prediction via data-axis transformer with multi-level contexts. In *KDD*, pp. 2037–2045, 2021.

Yuan, Z., Liu, H., Liu, J., Liu, Y., Yang, Y., Hu, R., and Xiong, H. Incremental spatio-temporal graph learning for online query-poi matching. In *WWW*, pp. 1586–1597, 2021.

Zhang, J., Uribe, C. A., Mokhtari, A., and Jadbabaie, A. Achieving acceleration in distributed optimization via direct discretization of the heavy-ball ode. In *Proceedings of the American Control Conference*, pp. 3408–3413, 2019.

Zhang, Y., Gao, S., Pei, J., and Huang, H. Improving social network embedding via new second-order continuous graph neural networks. In *KDD*, pp. 2515–2523, 2022.

Zhao, T., Zhang, X., and Wang, S. Graphsmote: Imbalanced node classification on graphs with graph neural networks. In *WSDM*, pp. 833–841, 2021.

Zhao, W., Wang, W., and Tian, Y. Graformer: Graph-oriented transformer for 3d pose estimation. In *CVPR*, pp. 20438–20447, 2022.

Zhao, Y., Luo, X., Ju, W., Chen, C., Hua, X.-S., and Zhang, M. Dynamic hypergraph structure learning for traffic flow forecasting. In *ICDE*, 2023.

A. Proof of Lemma 3.1.

Proof. Assuming $\boldsymbol{R}^{t+1} := \left(\boldsymbol{Z}^{t+1} - \boldsymbol{Z}^{t}\right)/\sqrt{\lambda}$ and $\beta := 1 - \gamma\sqrt{\lambda}$ where γ is a given coefficient, the momentum updating algorithm can be rewritten as follows:

$$\boldsymbol{R}^{t+1} = (1 - \gamma\sqrt{\lambda})\boldsymbol{R}^{t} + \sqrt{\lambda}(\sigma((\hat{\boldsymbol{D}}^{t})^{\ 1}\hat{\boldsymbol{A}}^{t}\boldsymbol{Z}^{t}\boldsymbol{W}_{p}) - \boldsymbol{Z}^{t}). \tag{22}$$

When $\lambda \to 0$, we have

$$\frac{d\boldsymbol{Z}^{t}}{dt} = \boldsymbol{R}^{t}, \tag{23}$$

$$\frac{d\boldsymbol{R}^{t}}{dt} = -\gamma\boldsymbol{R}^{t} + (\sigma((\hat{\boldsymbol{D}}^{t})^{\ 1}\hat{\boldsymbol{A}}^{t}\boldsymbol{Z}^{t}\boldsymbol{W}_{p}) - \boldsymbol{Z}^{t}), \tag{24}$$

Finally, we combine Equation 23 and Equation 24 by deleting $\boldsymbol{R}^{t}$ as below:

$$\frac{d^{2}\boldsymbol{Z}^{t}}{dt^{2}} + \gamma\frac{d\boldsymbol{Z}^{t}}{dt} = \sigma((\hat{\boldsymbol{D}}^{t})^{\ 1}\hat{\boldsymbol{A}}^{t}\boldsymbol{Z}^{t}\boldsymbol{W}_{p}) - \boldsymbol{Z}^{t}. \tag{25}$$

□

B. Proof of Lemma 3.2.

Proof. A coupled first-order ODE can be directly obtained from Equation 23 and Equation 24. Further, we can augment the node state representation matrix by $\boldsymbol{F}^{t} = [\boldsymbol{Z}^{t}, \dot{\boldsymbol{Z}}^{t}]$ and $\dot{\boldsymbol{F}}^{t} = [\dot{\boldsymbol{Z}}^{t}, \ddot{\boldsymbol{Z}}^{t}]$[2], resulting in a first-order ODE with the variable $\boldsymbol{F}^{t}$. □

C. Proof of Lemma 3.3.

We first introduce a theorem and then finish the proof.

Theorem C.1. *(Picard–Lindelöf theorem) $D \subseteq \mathsf{R} \times \mathsf{R}^{n}$ denotes a closed rectangle containing the point, i.e., $(t_0, y_0) \in D$. $f : D \to \mathsf{R}^{n}$ denotes a function which is continuous with respect to t and Lipschitz continuous with respect to y. Following this, some positive $\varepsilon > 0$ exists such that the initial value problem, i.e.,*

$$y^{0}(t) = f(t, y(t)), \quad y\left(t_0\right) = y_0. \tag{26}$$

has a unique solution $y(t)$ on the interval $\left[t_0 - \varepsilon, t_0 + \varepsilon\right]$.

Proof. Let the message passing matrix be

$$\boldsymbol{M}^{t} = (\hat{\boldsymbol{D}}^{t})^{\ 1}\hat{\boldsymbol{A}}^{t} = \begin{pmatrix} M_{11}^{t} & \dots & M_{1N}^{t} \\ \vdots & \ddots & \vdots \\ M_{N1}^{t} & \dots & M_{NN}^{t} \end{pmatrix}, \tag{27}$$

the weight matrix is

$$\boldsymbol{W}_{p} = \begin{pmatrix} W_{11} & \dots & W_{1d} \\ \vdots & \ddots & \vdots \\ W_{d1} & \dots & W_{dd} \end{pmatrix}. \tag{28}$$

Then, we define the transformation function

$$S\colon \mathsf{R}^{N\ \ d} \to \mathsf{R}^{Nd\ \ 1} \tag{29}$$

$$\mathbf{Z}^{t} = \begin{pmatrix} \mathbf{z}_{1}^{t} \\ \vdots \\ \mathbf{z}_{N}^{t} \end{pmatrix} \mapsto \begin{pmatrix} (\mathbf{z}_{1}^{t})^{T} \\ \vdots \\ (\mathbf{z}_{N}^{t})^{T} \end{pmatrix}, \tag{30}$$

[2] $\dot{\boldsymbol{Z}}^{t} := \frac{d\boldsymbol{Z}^{t}}{dt}$ and $\ddot{\boldsymbol{Z}}^{t} := \frac{d^{2}\boldsymbol{Z}^{t}}{dt^{2}}$

and transform the augmented ODE Equations 23 and 24 to be

$$\begin{cases} \frac{d}{dt}S(\boldsymbol{Z}^t) = S(\boldsymbol{R}^t), \\ \frac{d}{dt}S(\boldsymbol{R}^t) = -\gamma S(\boldsymbol{R}^t) - S(\boldsymbol{Z}^t) + \textbf{ReLU}(S(\boldsymbol{M}^t\boldsymbol{Z}^t\boldsymbol{W}_p)) \end{cases} \tag{31}$$

where $(\boldsymbol{M}^t\boldsymbol{Z}^t\boldsymbol{W}_p)_{ij} = \sum_{b=1}^{d}\sum_{a=1}^{N} M_{ia}^t Z_{ab}^t W_{bj}$.

Now, let $\boldsymbol{Y}^t = \begin{pmatrix} S(\boldsymbol{Z}^t) \\ S(\boldsymbol{R}^t) \end{pmatrix} \in \mathsf{R}^{2Nd\ \ 1}$, we get the ODE equation $\frac{d\boldsymbol{Y}^t}{dt} = f(t, \boldsymbol{Y}^t)$. It is obvious that f is continuous w.r.t t.

For any two solutions $\boldsymbol{Y}_1^t, \boldsymbol{Y}_2^t$, denote $\Delta R_i^t = S(\boldsymbol{R}^t)_{1i} - S(\boldsymbol{R}^t)_{2i}$, $\Delta Z_i^t = S(\boldsymbol{Z}^t)_{1i} - S(\boldsymbol{Z}^t)_{2i}$. Note $|\textbf{ReLU}(x) - \textbf{ReLU}(y)| = |x^+ - y^+| \le |x - y|$. Therefore, we have

$$\begin{aligned} \|f(\boldsymbol{Y}_1^t) - f(\boldsymbol{Y}_2^t)\|_2^2 &\le \sum_{i=1}^{Nd}(\Delta R_i^t)^2 && (32)\\ &+ 3\sum_{i=1}^{Nd}\big[\gamma^2(\Delta R_i^t)^2 + (\Delta Z_i^t)^2 && (33)\\ &+ N^2d^2\sum_{a,b}(M_{ka}^t)^2 W_{bj}^2(\boldsymbol{Z}_{ab,1}^t - \boldsymbol{Z}_{ab,2}^t)^2\big] && (34)\\ &\le \sum_{i=1}^{Nd}\big[(1+3\gamma^2)(\Delta R_i^t)^2 && (35)\\ &\quad + (3 + 3N^3d^3M^2W^2)(\Delta Z_i^t)^2\big] && (36)\\ &\le L^2\|\boldsymbol{Y}_1^t - \boldsymbol{Y}_2^t\|_2^2 && (37) \end{aligned}$$

where $M = \max_{i,j}|M_{ij}^t|$, $W = \max_{ij}|W_{ij}|$, $L = \max(\sqrt{1+3\gamma^2}, \sqrt{3+3N^3d^3M^2W^2})$. Hence, we prove that f is Lipschitz-continuous in y, then by Picard–Lindelöf theorem, we prove the uniqueness of the solution. □

D. Algorithm

We summarize the overall framework of HOPE in Algorithm 1.

Algorithm 1 Learning Algorithm of HOPE

Input: Object trajectory sequence $\mathcal{X}$, adjacency matrix sequence $\mathcal{A}$
Output: The parameters in both the encoder and the decoder.

```
Initialize model parameters.
while not convergence do
    Select a batch of training sequences;
    for each training sequence do
        Construct temporal graph using Equation 3;
        Feed the sequence into the twin encoder to obtain the posterior distribution Equation 12;
        Initialize the node and edge latent states using Equation 13 and Equation 14;
        Solve the second-order ODE system in Equation 15, Equation 16 and Equation 17;
        Feed the latent states into a decoder and obtain the prediction using Equation 19;
    end for
    Calculate ELBO in Equation 21;
    Update parameters using back propagation;
end while
```

E. More Related Work

Graph Neural Networks. Graph neural networks (GNNs) have been shown an effective tool for representation learning in relational data (Wang et al., 2021; Fan et al., 2019; Zhao et al., 2022; Ju et al., 2023a;b). They have been extensively

explored in a range of applications including node classification (Wu et al., 2020; Zhao et al., 2021), link prediction (Qu et al., 2020; Benson & Kleinberg, 2019), and recommendation systems (He et al., 2020; Wang et al., 2019). GNNs can be roughly categorized into spectral methods (Defferrard et al., 2016; Bruna et al., 2013; Henaff et al., 2015; Zhao et al., 2022) and spatial methods (Kipf & Welling, 2017; Veličković et al., 2018; Xu et al., 2019). Spectral methods seek to filter graph signals using graph Laplacian, which extends the convolution theory to graphs. Typically, these algorithms construct localized filters in the spectral domain using graph Fourier transform. In contrast, spatial methods adhere to the paradigm of message passing, where each node collects information from its neighborhood, followed by an aggregation operation to iteratively update the node representation. Further research investigates adaptive aggregation operators to enhance the model performance (Hamilton et al., 2017; Veličković et al., 2018; Xu et al., 2019; Kim & Oh, 2022). For instance, Graph Attention Network (Veličković et al., 2018) (GAT) uses the attention mechanism to determine the significance of neighbors to the central node. However, these methods could fail to capture the high-order non-neighborhood information in the spectral domain (Balcilar et al., 2021). In this paper, we combine the advantages of both worlds. In particular, our twin graph encoder leverages both spatial and spectral graph convolution to investigate complex high-order correlations in dynamical interacting systems.

F. Details of Datasets

We utilize the COVID-19 data from the Johns Hopkins University (JHU) Center for Systems Science and Engineering (Dong et al., 2020) to build our node feature data. We selected five dynamic features along with one static feature as object attributes. Detailed information about these features is introduced below.

- **Population**: The number of population in each state.
- **Confirmed-Number**: The number of state increased confirmed cases in each day.
- **Deaths-Number**: The number of state increased deaths in each day.
- **Recoverd-Number**: The number of state increased recovered cases in each day.
- **Mortality-Rate**: The number of state cumulative deaths / the number of state cumulative confirmed cases each day.
- **Testing-Rate**: The number of state cumulative test results per 100,000 persons in each day.

Then we manage to generate training and testing samples. To capture dynamic spatial correlations between each object, the Dynamic Time Warping (DTW) algorithm (Berndt & Clifford, 1994) is employed for similarity measurement of states. To be specific, for each time t the edge weight of two nodes is measured with their feature series in $[t-\Delta, t]$ by the DTW algorithm. **Social Network** models opinions migrating from individuals to individuals in a social network. Following (Huang et al., 2021), the number of individuals and the noise parameter is set to 80 and 0.2, respectively. The sparsity parameter is set to $e^{-0.4}$. For **Spring Ocsillator**, we set the number of balls to 50 and simulate the data for total 240 timestamps. The side length of the box is set to 2 and the initial locations of these balls follow uniform distribution in the area inside the box. Every two balls have a probability of 0.5 to be connected together with a spring, and we also ensure that every ball has at least one spring on it. As we discussed before, each training or testing sample is a continuous time series composed of the condition part and prediction part. The conditional part is model input and the prediction part is used for supervising or evaluating. As a result, it is sufficient to ensure no overlapping between the training sample and the testing sample.

To be specific, we split feature data in COVID-19, a 266-days time series, into a 233-day part and 31-day part. The training samples and validating samples are extracted from 233-day part, and testing samples are extracted from 31-day part. The similar procedure is deployed on Social Network and Spring Oscillator. Socail Network is divided into a 320-day part and a 80-day part, and Spring Oscillator is divided into a 500-stamp part and a 50-stamp part.

To evaluate our HOPE and baselines, we select several testing samples and adopt the average performance among the samples. For example, on the 2-week-ahead prediction task in COVID-19, we select Dec.02-Dec.15, Dec.10-Dec.22, Dec.17-Dec.29 as testing samples. The three metrics, i.e., MAE, RMSE and MAPE are then computed on each sample. Finally, we take the average on these samples to report.

G. Details of Baselines

The details of baselines are elaborated as follows:

- **LSTM** (Hochreiter & Schmidhuber, 1997): It is a classic recurrent neural network (RNN) that learns the dynamics of the sequence, without considering the interaction between nodes.
- **GRU** (Cho et al., 2014): It is another variant of RNNs, which involves two gates to model temporal evolution.
- **NODE** (Chen et al., 2018): It is the first continuous-depth neural network model which is solved by the back-propagatable ODE solver.
- **HBNODE** (Xia et al., 2021): It employs a heavy ball ODE to accelerate the forward and back propagation of the ODE model.
- **DGCRN** (Li et al., 2022): It constructs the dynamic graph and utilizes a graph convolution recurrent unit to capture the real-time spatial-temporal dependencies in the dynamical system.
- **MPNODE** (Gupta et al., 2022): It combines augmented ODE with message passing mechanism.
- **CG-ODE** (Huang et al., 2021): It is a graph ODE model that integrates the evolution of both edges and nodes into a holistic ODE system.

H. Additional Experiments

Figure 5 reports the performance sensitivity of the number of the convolution layer K and the balance coefficient γ in terms of MAPE. We can observe a similar tendency that the performance is optimal with K set to 1 and γ around -3.

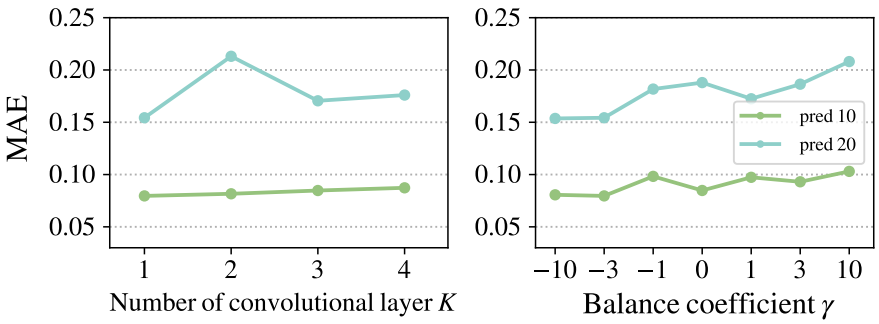

Figure 5. Performance sensitivity on Social Network in terms of MAPE.