UNIVERSITY OF CALIFORNIA

Los Angeles

Neural Dynamics for Science: The Symbiosis of Deep Graph Learning and Differential Equations

A dissertation submitted in partial satisfaction
of the requirements for the degree
Doctor of Philosophy in Computer Science

by

Zijie Huang

2024

ABSTRACT OF THE DISSERTATION

Neural Dynamics for Science: The Symbiosis of Deep Graph Learning and Differential Equations

by

Zijie Huang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Yizhou Sun, Co-Chair
Professor Wei Wang, Co-Chair

Many scientific problems require a deep understanding of internal structures and complex dynamics, spanning physical interactions within molecules, brain networks, and beyond. These problems can be formulated as modeling interacting dynamical systems using graphs, which represent entities as nodes and their relationship as edges. Traditionally, the dynamics of interacting systems are described by ordinary differential equations (ODEs), offering continuous and interpretable solutions but requiring significant domain expertise. Recent data-driven approaches such as Graph Neural Networks (GNNs) learn system dynamics from observational data, which however, struggle with long-term predictions and irregular observations due to their discrete dynamics.

My research aims to develop novel frameworks that bridge these two worlds, i.e. combining the learning power of neural networks (NNs) with the symbolic knowledge encoded in ODEs. In contrast with discrete models, such methods provide a principled approach to model continuous dynamical systems, from synthetic simulations to real-world scenarios like brain network analysis and COVID-19 prediction. Building upon this, I have further strengthened its power in three key areas: 1.) integrating data-driven inductive biases like energy conservation law; 2.) enhancing generalization ability; 3.) enabling causal decision-making. By merging deep graph learning with differential equations, I believe my research will pave the way for breakthroughs in symbolic deep learning for scientific discovery.

The dissertation of Zijie Huang is approved.

Cho-Jui Hsieh

Kai-Wei Chang

Wei Wang, Committee Co-Chair

Yizhou Sun, Committee Co-Chair

University of California, Los Angeles

2024

To all the love and support

throughout my journey —

which finally made who I am

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I have been extremely fortunate and grateful throughout my entire Ph.D. journey to receive a lot of guidance and support from so many great collaborators.

I want to express my deepest and sincere thanks to my two amazing Ph.D. advisors, Prof.Yizhou Sun and Prof.Wei Wang. They are two excellent female researchers who I always consider as role models. What impressed me the most was not the valuable and supportive guidance for my research, but the way they treated people, their efforts in creating an inclusive research community, and their positive attitude towards life.

I sincerely appreciate my thesis committee members, Prof. Kai-Wei Chang and Prof.Cho-Jui Hsieh. Their insightful feedback, engaging suggestions, and constant encouragement were indispensable. I want to thank Prof. Dominik Wodarz and Prof. Mathieu Bauchy for their valuable suggestions especially for AI4Science.

Beyond my committee members, I am fortunate to work and learn from a lot of brilliant collaborators during my internships. These unique experiences have broadened my research vision and better shaped my research goals. I want to thank Zheng Li, Haoming Jiang, Bing Yin, Daheng Wang, Binxuan Huang, Xian Li, Chenwei Zhang, Zhengyang Wang, Yan Liang, Tianyu Cao, Hanqing Lu, Karthik Subbian, Christos Faloutsos, Jingbo Shang, at the Amazon Search Team. I also want to thank Anne Cocos, Hafez Asgharzadeh, Lingyi Liu, Evan Cox, Colby Wise, Sudarshan Lamkhede, at Netflix. I also learned and received a lot of support during my internship at Nvidia, and special thanks to Sanjay Choudhry, Mohammad Amin Nabian for their endless help.

During my five wonderful Ph.D. years, I work with a lot of friends at UCLA, who makes me feel like a big family, including: Ziniu Hu, Xiao Luo, Song Jiang, Shichang Zhang, Roshni Iyer, Zhiping Xiao, Yewen Wang, Kewei Cheng, Fred Xu, Fang Sun, Yanqiao Zhu, Jingru Gan, Xiaoxuan Wang, Chenchen Ye, Xiusi Chen, Yihe Deng, Junkai Zhang, Jeeyun Hwang, Yadi Cao, Wanjia Zhao, Yuanzhou Chen, Jingdong Gao, Mingyu Ma, Yanna Ding, Derek Xu, Atefeh Sohrabizadeh, Yichao Zhou, Yunsheng Bai, Junheng Hao, Han Liu, Ruoyan Li. I also want to sincerely thank my friends for supporting my Ph.D. career: Fan Yin, Hejie Cui, Zhe Zeng, Yuanhao Xiong, Yihang Guo, Yangsibo Huang, Tao Zhang, Baolin Li, Yu Wang, Zhankui He.

Finally, I would like to give my heartfelt thanks to my parents, my boyfriend, my family. Their encouragement and support have been my rock during the hardest time of my life. Wish all the bests to those I love.

VITA

2015–2019	B.S. in Electrical Engineering, Shanghai Jiao Tong University, Shanghai, China.
2018	Visting Student Researcher, University of Illinois at Urbana-Champaign (UIUC) , IL, US.
2021	Applied Scientist Intern, Amazon Search, CA, US.
2022	Applied Scientist Intern, Amazon Search, WA, US.
2023	Research Scientist Intern, Netflix, CA, US.
2024	Research Scientist Intern, Nvidia, CA, US.

PUBLICATIONS

Fred Xu, Song Jiang, *Zijie Huang*, Xiao Luo, Shichang Zhang, Yuanzhou Chen, Yizhou Sun. "FUSE: Measure-Theoretic Compact Fuzzy Set Representation for Taxonomy Expansion." In Annual Meeting of the Association for Computational Linguistics (**ACL**) 2024 [XJH24].

*Zijie Huang**, Wanjia Zhao*, Jingdong Gao, Ziniu Hu, Xiao Luo, Yadi Cao, Yuanzhou Chen, Yizhou Sun, Wei Wang. "TANGO: Time-Reversal Latent GraphODE for Multi-Agent Dynamical Systems." In DLDE workshop at **NeurIPS** 2023. (Best Paper Award) [HZGrd].

Zijie Huang, Jeehyun Hwang, Junkai Zhang, Jinwoo Baik, Weitong Zhang, Quanquan Gu, Wei Wang. "Causal Graph ODE: Continuous Treatment Effect Modeling in Multi-agent Dynamical Systems." In The Web Conference (**WWW**) 2024 & DLDE workshop at **NeurIPS** 2023 [HHZ24].

Xiao Luo, Haixin Wang, *Zijie Huang*, Huiyu Jiang, Abhijeet Sadashiv Gangan, Song Jiang, Yizhou Sun. "CARE: Modeling Interacting Dynamics Under Temporal Distribution Shift." In Proceeding of Thirty-seventh Conference on Neural Information Processing Systems (**NeurIPS**), 2023 [LWH23].

Zijie Huang, Yizhou Sun, Wei Wang. "Generalizing Graph ODE for Learning Complex System Dynamics across Environments". In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2023 [HSW23a].

Zijie Huang, Daheng Wang, Binxuan Huang, Chenwei Zhang, Jingbo Shang, Yan Liang, Zhengyang Wang, Xian Li, Christos Faloutsos, Yizhou Sun and Wei Wang . "Concept2Box: Joint Geometric Embeddings for Learning Two-View Knowledge Graphs ". In Proceedings of the 2023 Annual Meeting of the Association for Computational Linguistics (**ACL**), 2023 [ZWH23].

Song Jiang, *Zijie Huang*, Xiao Luo, Yizhou Sun. "CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems". In Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2023 [JHL23a].

Xiao Luo, Jingyang Yuan, *Zijie Huang*, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, Yizhou Sun. "HOPE: High-order Graph ODE For Modeling Interacting Dynamics". In Proceddings of the Fortieth International Conference on Machine Learning (**ICML**), 2023 [LYH23].

Han Liu, *Zijie Huang*, Samuel S. Schoenholz, Ekin D. Cubuk, Morten M. Smedskjaer, Yizhou Sun, Wei Wang, Mathieu Bauchy. "Watching to Simulate Glass Dynamics from Their Static Structure by Machine Learning". In Proceddings of **Materials Horizons**, 2023 [LHS23].

Zijie Huang, Zheng Li, Haoming Jiang, Tianyu Cao, Hanqing Lu, Bing Yin, Karthik Subbian, Yizhou Sun, Wei Wang. "Multilingual Knowledge Graph Completion with Self-Supervised Adaptive Graph Alignment". In Proceedings of the 2022 Annual Meeting of the Association for Computational Linguistics (**ACL**), 2022 [HLJ22].

Zijie Huang, Yizhou Sun, Wei Wang. "Coupled Graph ODE for Learning Interacting System Dynamics". In Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (**KDD**), 2021 [HSW21a].

Ruijie Wang, *Zijie Huang*, Shengzhong Liu, Huajie Shao, Dongxin Liu, Jinyang Li, Tianshi Wang, Dachun Sun, Shuochao Yao, Tarek Abdelzaher. "DyDiff-VAE: A Dynamic Variational Framework for Information Diffusion Prediction". In Proceeding of Fourty-fourth International ACM SIGIR Conference on Research and Development in Information Retrieval (**SIGIR**), 2021 [WHL21a].

Zijie Huang, Yizhou Sun, Wei Wang. "Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations". In Proceeding of Thirty-fourth Conference on Neural Information Processing Systems (**NeurIPS**), 2020 [HSW20a].

Amin Javari, Zhankui He, *Zijie Huang*, Raj Jeetu, Kevin Chen-Chuan Chang. "Weakly Supervised Attention for Hashtag Recommendation using Graph Data". In The Web Conference (**WWW**), 2020 [JHH20].

CHAPTER 1

Introduction

1.1 Motivation

Many real-world scientific problems demand a profound grasp of internal structures and dynamics, encompassing fundamental physical interactions and intricate data patterns within molecules, brain networks, and beyond. These problems can be formulated as modeling interacting (multi-agent) dynamical systems using *graphs*, a versatile data structure representing entities as nodes and their relationship as edges. For example, in molecular dynamics, atoms interact and evolve over time, forming trajectories within a 3D space and ever-changing graph structures. Artificial Intelligence (AI) is fueling a new paradigm of research discoveries across diverse natural sciences. Nonetheless, existing models tailored for static graph-structured data fall short in capturing the dynamic nature of systems across scientific domains, which play vital roles for scientific discovery. For example, 9 out of the world's top 10 supercomputers are used for simulations, *i.e.* predicting trajectories in the future, spanning the field of cosmology, geophysics, and fluid dynamics [Lab19]. Building a surrogate neural simulator to accelerate these simulations, offers significant time and space efficiency for real-world applications.

Traditionally, the dynamics of a system are defined by the symbolic knowledge in the form of ordinary differential equations (ODEs). For example, molecular dynamics can be described by Newton's law of motion: $\frac{d^2\mathbf{x}_i}{dt^2} = -\nabla U(\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N)$, in which $\mathbf{x}_i$ is the position of each atom and $U(\cdot)$ is the potential energy of the system. Despite their theoretical elegance, the practical application of ODEs to complex, non-linear systems is challenging. Often, the precise ODEs are either unknown, or require significant efforts from domain experts. Thus, people resort to data-driven approaches by learning a dynamic model (*e.g.*, spatio-temporal NNs) from observational data. Nonetheless, deep learning models have fundamental limitations: 1) they discretize systems that are continuous in nature such as the spread of COVID-19 [HSW21b], thus struggling with making stable long-term predictions; 2) discrete models are restricted to learn a fixed-step state transition function, therefore, they cannot handle irregular and

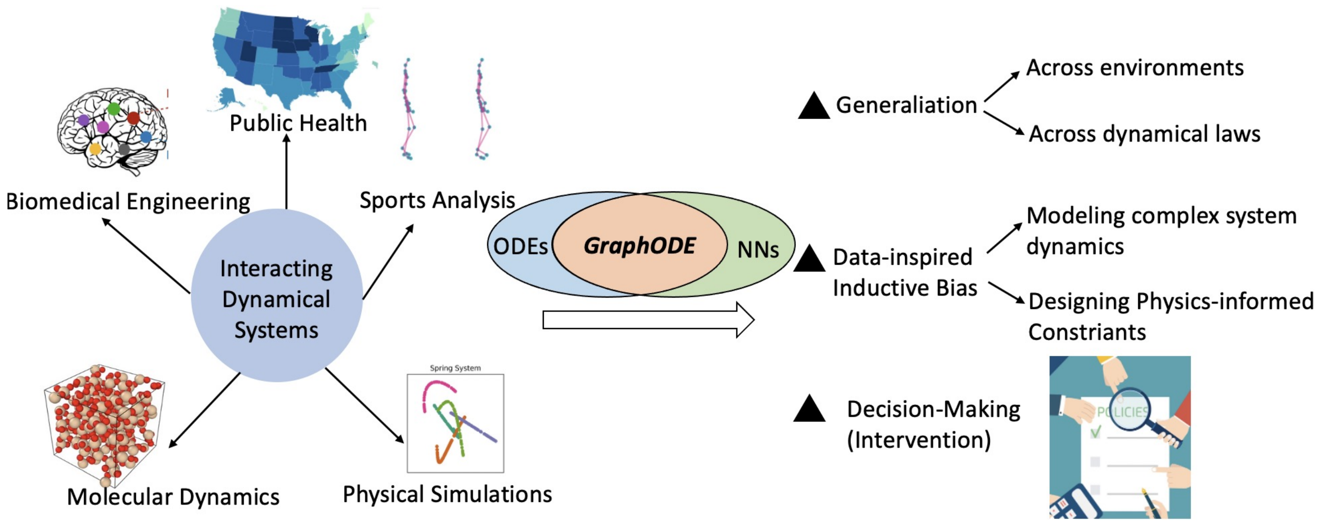


Figure 1.1: I pioneered **GraphODE**, a general framework for modeling interacting dynamical systems across many scientific domains.

incomplete observational data caused by imperfect data collectors. **My research goal is to develop neural networks that bridge deep graph learning with symbolic ODEs. Such models are built to have better understanding of dynamical laws encoded from data, so as to make rapid, accurate, and long-term predictions, and to support causal decision-making for scientific discovery.**

1.2 Research Overview

My past research has pioneered a novel framework called ***GraphODE*** [HSW20a] which opens up a new chapter to bridge the expressive learning power of GNNs with the symbolic knowledge preserved by ODEs to model interacting systems. Specifically, it employs a Graph Neural Networks (GNNs) as the ODE function and encodes initial states via another neural network, which are jointly trained from data. With GraphODE, long-term predictions are made by calling any black-box ODE solvers. In contrast with discrete models, GraphODE serves as a principled way to model real-world continuous dynamical systems. It can learn from irregular and incomplete observations and shows superior prediction accuracy in the long-range.

Chapter 2 describes the first initiative of GraphODE. Building upon that, I have further strengthened its power in three key areas illustrated in Figure 1.1, which arises from challenges at the *data*, *model*, and *application* aspects in dynamical system modeling:

- **Data: Integrating data-driven inductive biases.** Real-world dynamical systems often exhibit complex dynamical patterns, such as following higher-order dynamics [LYH23] or adhering to some physical principles [HZGrd]. Injecting different priors can enhance model performance and improve the robustness of predictions.

  – Chapter 3 presents *Coupled GraphODE (**CG-ODE**)* [HSW21b], that can capture the joint evolution of graph nodes and edges in a continuous manner.

  – Chapter 4 presents *Time-Reversal Latent GraphODE (**TREAT**)* [HZGrd] that numerically achieves high-precision modeling across dynamical systems.

- **Model: Building generalized GraphODEs.** Model generalization is an important topic in dynamical system modeling due to the huge cost of traditional simulations. By enabling joint learning from multiple systems that share similarities, a general-purpose model can improve modeling accuracy and save computational cost.

  – Chapter 5 presents *Generalized GraphODE (**GG-ODE**)* [HSW23a] that jointly learn dynamical systems across different environmental factors such as temperatures.

  – Chapter 6 presents an initial step towards a more challenging question: how to generalize across systems following different laws, such as the mixture of waters (which are liquid) and sands (which are solid). The *Self-Supervised Adaptive Graph Alignment Model (**SS-AGA**)* [HLJ22] fuses multiple knowledge graphs (KGs) together, where each KG can represent the domain knowledge of one system category. This serves as an initial step of incorporating multiple external sources to facilitate generalization across dynamical laws.

- **Applications: From Predictions to Interventions through Causal Decision-making.** The ultimate goal of dynamical system modeling is not only about making predictions. We want to understand the systems, and further conduct interventions and controls so that we can change what will happen in the future. Chapter 7 presents *Causal GraphODE (**CAG-ODE**)* [HHZ24] that enables causal decision-making in continuous dynamical systems and can consider the combined effects of multiple treatments such as analyzing multiple COVID-19 policies over time.

1.3 Research Contributions

By collaborating with domain experts in material science, physics, biomedical engineering and healthcare, we successfully deployed the proposed GraphODEs to many real-world scientific problems, from *synthetic* physical simulations [LYH23] and molecular dynamics [LZS23] to more complex *real-world* systems including brain network analysis in biomedical engineering [HYH23] and COVID-19 prediction in public health [HSW21b]. Many researchers over the world also use my work for electromagnetic simulations [YKR22], robotics [GVK22], sports analysis [XW23], and recommendations [GZL22]. The GraphODE series work has been generously acknowledged and supported by the NSF #2312501 award, and the **Best Paper Award** at the Deep Learning and Differential Equation workshop in Neurisp 2023.

CHAPTER 2

LG-ODE: Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations

Many real-world systems, such as moving planets, can be considered as multi-agent dynamic systems, where objects interact with each other and co-evolve along with the time. Such dynamics is usually difficult to capture, and understanding and predicting the dynamics based on observed trajectories of objects become a critical research problem in many domains. Most existing algorithms, however, assume the observations are regularly sampled and all the objects can be fully observed at each sampling time, which is impractical for many applications. In this paper, we propose to learn system dynamics from irregularly-sampled partial observations with underlying graph structure for the first time. To tackle the above challenge, we present LG-ODE, a latent ordinary differential equation generative model for modeling multi-agent dynamic system with known graph structure. It can simultaneously learn the embedding of high dimensional trajectories and infer continuous latent system dynamics. Our model employs a novel encoder parameterized by a graph neural network that can infer initial states in an unsupervised way from irregularly-sampled partial observations of structural objects and utilizes neural ODE to infer arbitrarily complex continuous-time latent dynamics. Experiments on motion capture, spring system, and charged particle datasets demonstrate the effectiveness of our approach.

2.1 Introduction

Learning system dynamics is a crucial task in a variety of domains, such as planning and control in robotics [LWZ19a], predicting future movements of planets in physics [KFW18a], etc. Recently, with the rapid development of deep learning techniques, researchers have started building neural-based simulators, aiming to approximate complex system interactions with neural networks [LWZ19a, BPL16, KFW18a, CUT16, SsF16] which can be learned automatically from data. Existing models, such as Interaction Networks (IN) [BPL16], usually decompose the system into distinct objects and relations and learn to reason about the consequences of their interactions and dynamics based on graph neural networks (GNNs). However, one major limitation is that they only work for fully observable systems, where the individual trajectory of each object can be accessed at every sampling time. In reality, many applications have to deal with partial observable states, meaning that the observations for different agents are not temporally aligned. For example, when a robot wants to push a set of blocks into a target configuration, only the blocks in the top layer are visible to the camera [LWZ19a]. More challengingly, the visibility of a specific object might change over time, meaning that observations can happen at non-uniform intervals for each agent, i.e. irregularly-sampled observations. Such data can be caused by various reasons such as broken sensors, failed data transmissions, or damaged storage [TYS20]. How to learn an accurate multi-agent dynamic system simulator with irregular-sampled partial observations remains a fundamental challenge.

Tang et al. [TYS20] recently have studied a seemingly similar problem which is predicting the missing values for multivariate time series (MTS) as we can view the trajectory of each object as a time series. They assumed there exist some close temporal patterns in many MTS snippets and proposed to jointly model local and global temporal dynamics for MTS forecasting with missing values, where the global dynamics is captured by a memory module. However, it differs from multi-agent dynamic systems in that the model does not assume a continuous interaction among each variable, i.e. the underlying graph structure is not considered. Such interaction plays a very important role in multi-agent dynamic systems which drives the system to move forward.

Recently Rubanova et al. [RCD19b] has proposed a VAE-based latent ODE model for modeling irregularly-sampled time series, which is a special case of the multi-agent dynamic system where it only handles one object. They assume there exists a latent continuous-time system dynamics and model the state evolution using a neural ordinary differential equation (neural ODE) [CRB18a]. The initial state is drawn from an approximated posterior distribution which is parameterized by a neural network and is learned from observations.

Inspired by this, we propose a novel model for learning continuous multi-agent system dynamics under the same framework with GNN as the ODE function to model continuous interaction among objects. However, the main challenge lies in how to approximate the posterior distributions of latent initial states for the whole system, as now the initial states of agents are closely coupled and related to each other. We handle this challenge by firstly aggregating information from observations of neighborhood nodes, obtaining a contextualized representation for each observation, then employ a temporal self-attention mechanism to capture the temporal pattern of the observation sequence for each object. The benefits of joint learning of initial states is twofold: First, it captures the complex interaction among objects. Second, when an object only has few observations, borrowing the information from its neighbors would facilitate the learning of its initial state. We conduct extensive experiments on both simulated and real datasets over interpolation and extrapolation tasks. Experiment results verify the effectiveness of our proposed method.

2.2 Problem Formulation and Preliminaries

Consider a multi-agent dynamic system as a graph $G = \langle O, R \rangle$, where vertices $O = \{o_1, o_2 \cdots o_N\}$ represent a set of N interacting objects, $R = \{\langle i, j \rangle\}$ represents relations. For each object, we have a series of observations $o_i = \{\boldsymbol{o}_i^t\}$ at times $\{t_i^j\}_{j=0}^{T_i}$,where $\boldsymbol{o}_i^t \in \mathbb{R}^D$ denotes the feature vector of object i at time t, and $\{t_i^j\}_{j=0}^{T_i}$ can be of variable length and values for each object. Observations are often at discrete spacings with non-uniform intervals and for different objects, observations may not be temporally aligned. We assume there exists a latent generative continuous-time dynamic system, which we aim to uncover. Our goal is to learn latent representations $\boldsymbol{z}_i^t \in \mathbb{R}^d$ for each object at any given time, and utilize it to reconstruct missing observations and forecast trajectories in the future.

Ordinary differential equations (ODE) for multi-agent dynamic system. In continuous multi-agent dynamic system, the dynamic nature of state is described for continuous values of t over a set of dependent variables. The state evolution is governed by a series of first-order ordinary differential equations: $\dot{\boldsymbol{z}}_i^t := \frac{d\boldsymbol{z}_i^t}{dt} = g_i(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t)$ that drive the system states forward in infinitesimal steps over time. Given the latent initial states $\boldsymbol{z}_0^0, \boldsymbol{z}_1^0 \cdots \boldsymbol{z}_N^0 \in \mathbb{R}^d$ for every object, $\boldsymbol{z}_i^t$ is the solution to an ODE initial-value problem (IVP), which can be evaluated at any desired times using a numerical ODE solver such as Runge-Kutta [MSH19]:

$$\boldsymbol{z}_i^T = \boldsymbol{z}_i^0 + \int_{t=0}^{T} g_i(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t) dt \tag{2.1}$$

The ODE function g_i specifies the dynamics of latent state and recent works [CRB18a, RCD19b, YHL19] have proposed to parameterize it with a neural network, which can be learned automatically from data. Different from single-agent dynamic system, g_i should be able to model interaction among objects. Existing works [BPL16, LWZ19a, KFW18a, SRB17] in discrete multi-agent dynamic system employ a shared graph neural network (GNN) as the state transition function. It defines an object function f_O and a relation function f_R to model objects and their relations in a compositional way. By adding residual connection and let the stepsize go to infinitesimal, we can generalize such transition function to the continuous setting as shown in Eqn 2.2, where $\mathcal{N}_i$ is the set of immediate neighbors of object o_i.

$$\dot{\boldsymbol{z}}_i^t := \frac{d\boldsymbol{z}_i^t}{dt} = g_i(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t) = f_O(\sum_{j \in \mathcal{N}_i} f_R([\boldsymbol{z}_i^t, \boldsymbol{z}_j^t])) \tag{2.2}$$

Given the ODE function, the latent initial state $\boldsymbol{z}_i^0$ for each object determine the whole trajectories.

Latent ODE model for single-agent dynamic system. Continuous single-agent dynamic system is a special case in our setting. Recent work [RCD19b] has proposed a latent ODE model following the framework of variational autoencoder [KW14], where they assume a posterior distribution over the latent initial state $\boldsymbol{z}_0$. The encoder computes the posterior distribution $q\left(\boldsymbol{z}_0 \middle| \{\boldsymbol{o}_i, t_i\}_{i=0}^N\right)$ for the single object with an autoregressive model such as RNN, and sample latent initial state $\boldsymbol{z}_0$ from it. Then the entire trajectory is determined by the initial state z_0 and the generative model defined by ODE. Finally the decoder recovers the whole trajectory based on the latent state at each timestamp by sampling from the decoding likelihood $p(\boldsymbol{o}_i|\boldsymbol{z}_i)$.

We model multi-agent dynamic system under the same framework with GNN as the ODE function to model continuous interaction among objects. Since the latent initial states of each object are tightly coupled, we introduce a novel recognition network in the encoder to infer the initial states of all objects simultaneously.

2.3 Related Work

Neural Physical Simulator. Existing works have developed various neural-based physical simulators that learn the system dynamics from data [LWZ19a, BPL16]. In particular, Kipf et al. [KFW18a] and Battaglia et al. [BPL16] have explored learning a simulator by approximating pair-wise object interactions with graph neural networks. These approaches restrict themselves to learn a fixed-step state transition function that takes the system state at time t as input to predict the state at time $t + 1$. However, they can not be applied to the scenarios where system observations are irregularly sampled. Our model handles such issue by combining a neural ODE [CRB18a] to model continuous system dynamics and a temporal-aware graph neural network followed by a temporal self-attention module to estimate system initial states. Another issue lies in that they need to observe the full states of a system; but in reality, system states are often partially observed where number and set of observable objects vary over time. A recent work [LWZ19a] tackled this issue where system is partially observed but observations are regularly sampled by learning the dynamics over a latent global representation for the system, which is for example an average over the sets of object states. However it cannot directly learn the dynamic state for each object. In our work, we design a dynamic model that explicitly operates on the latent dynamic representations over each object. This allows us to define object-centric dynamics, which can better capture system dynamics compared to the coarse global system representation.

Dynamic Graph Representation. Our model takes the form of variational auto-encoder. The encoder which is used to infer latent initial states is closely related to dynamic graph representation. Most existing methods [SWG20a, HHN19, ZGY16] learn the dynamic representations of nodes by splitting the input graph into snapshot sequence based on timestamps [HDW20b]. Each snapshot is passed through a graph neural network to capture structural dependency among neighbors and then a recurrent network is utilized to capture temporal dependency by summarizing historical snapshots. However recurrent methods scale poorly with the increase in number of time-steps [SWG20a]. Moreover, when system states are partially observed, each timestamp may only contain a small portion of objects and abundant structural information across different snapshots is ignored. A recent work [HDW20b] proposed to maintain all the edges happening in different times as a whole and introduced relative temporal encoding strategy (RTE) to model structural temporal dependencies with any duration length. RTE utilizes a linear transformation of the sender node with regard to a given timestamp based on positional encoding [VSP17]. In our work, we explored a more complex nonlinear transformation to capture complex temporal dependency in dynamic physical system.

2.4 Method

In this section, we present Latent Graph ODE (LG-ODE) for learning continuous multi-agent system dynamics. Following the structure of VAE, LG-ODE consists of three parts that are trained jointly: 1.) An encoder that infers the latent initial states of all objects simultaneously given their partially-observed trajectories; 2.) a generative model defined by an ODE function that learns the latent dynamics given the sampled initial states. 3.) a decoder that recovers the trajectory based on the decoding likelihood $p(\boldsymbol{o}_i^t|\boldsymbol{z}_i^t)$. The overall framework is depicted in Figure 7.1. In the following, we describe the three components in

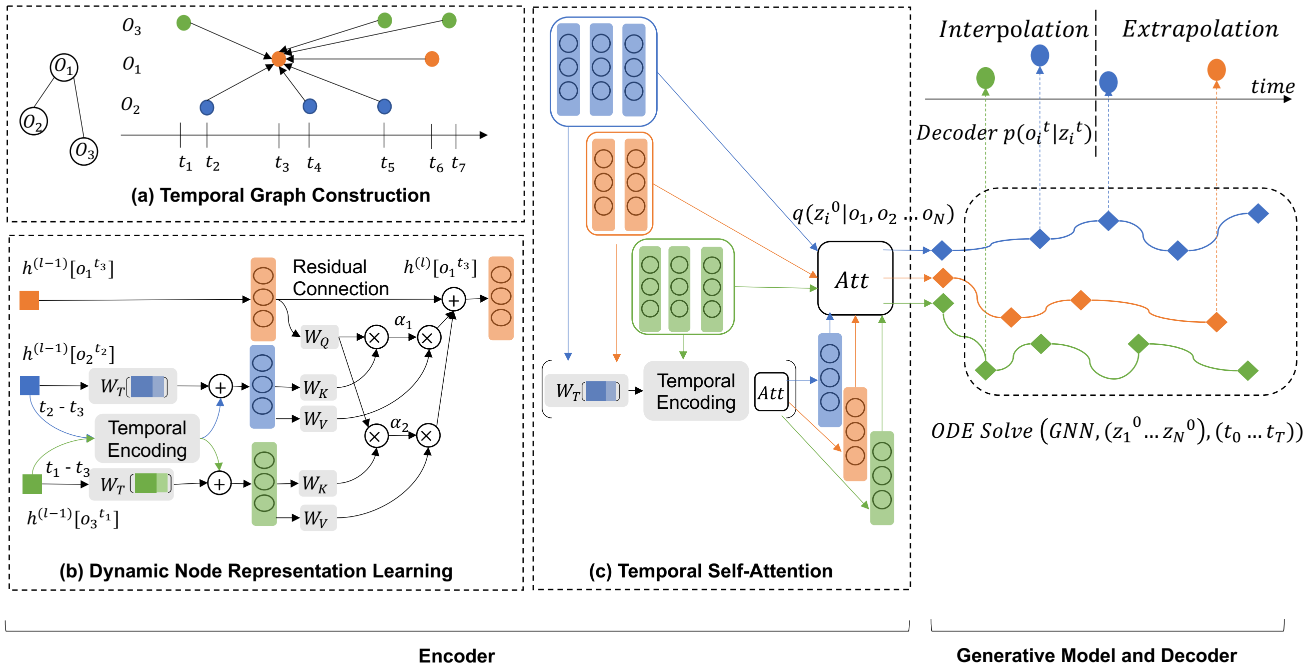

(a) Temporal Graph Construction
(b) Dynamic Node Representation Learning
(c) Temporal Self-Attention
Residual Connection
Temporal Encoding
Interpolation
Extrapolation
time
Decoder $p(o_i^t|z_i^t)$
$q(z_i^0|o_1, o_2 \ldots o_N)$
Att
ODE Solve $(GNN, (z_1^0 \ldots z_N^0), (t_0 \ldots t_T))$
Encoder
Generative Model and Decoder

Figure 2.1: Overall framework.

detail.

2.4.1 Encoder

Let $\boldsymbol{Z}^t \in \mathbb{R}^{N\times d}$ denotes the latent state matrix of all N objects at time t. The encoder returns a factorized distribution of initial states: $q_\phi(\boldsymbol{Z}^0|o_1, o_2 \cdots o_N) = \prod_{i=1}^{N} q_\phi(\boldsymbol{z}_i^0|o_1, o_2 \cdots o_N)$. In multi-agent dynamic system, objects are highly-coupled and related. Instead of encoding temporal pattern for each observation sequence $o_i = \{\boldsymbol{o}_i^t\}_{t=t_i^0}^{t_i^{T_i}}$ independently using an RNN [RCD19b], we incorporate structural information by first aggregating information from neighbors' observations, then employ a temporal self-attention mechanism to encode observation sequence for each object. Such process can be decomposed into two steps: 1.) **Dynamic Node Representation Learning**, where we aim to learn an encoding function f_{update} that outputs structural contextualized representation $\boldsymbol{h}_i^t$ for each observation $\boldsymbol{o}_i^t$. 2.) **Temporal Self-Attention**, where we learn an function f_{aggre} that aggregates the structural observation representations into a fixed-dimensional sequence representation $\boldsymbol{u}_i$ for each object. $\boldsymbol{u}_i$ is then utilized to approximate the posterior distribution for each latent initial state $\boldsymbol{z}_i^0$.

$$\boldsymbol{h}_i^t = f_{\text{update}}(o_i, \{o_j | \text{if } j \in \mathcal{N}_i\}), \quad \boldsymbol{u}_i = f_{\text{aggre}}(\boldsymbol{h}_i^{t_1}, \boldsymbol{h}_i^{t_2} \cdots \boldsymbol{h}_i^{t_{T_i}}) \tag{2.3}$$

Dynamic Node Representation Learning. One naive way to incorporate structural information is to construct a graph snapshot at each timestamp [SWG20a, ZGY16]. However, when system is partially observed, each snapshot may only contain a small portion of objects. For example in Figure 2.1 (a), 6 out of 7 timestamps only contains one object thus abundant structural information across different snapshots is ignored. We therefore preserve temporal edges and nodes across times to form a temporal graph, where every node is an observation, every edge exists when two objects are connected via a relation $r \in R\{\langle i, j \rangle\}$. Suppose on average every object has K observations, and there are E relations among objects. The constructed temporal graph has $\mathcal{O}(EK^2 + (K-1)KN)$ edges, which grows rapidly with the increase of average observation number K. We therefore set a slicing time window that filters out edges when the relative temporal gap is larger than a preset threshold.

To learn a structural representation for each observation, we propose a temporal-aware graph neural network characterized by the information propagation equation in Eqn 2.4, where $\boldsymbol{h}_t^{l-1}, \boldsymbol{h}_s^{l-1}$ are the representations of target and source node from layer $l-1$ respectively.

$$\boldsymbol{h}_t^l = \boldsymbol{h}_t^{l-1} + \sigma(\sum_{s \in \mathcal{N}_t} (\textbf{Attention}(\boldsymbol{h}_s^{l-1}, \boldsymbol{h}_t^{l-1}) \cdot \textbf{Message}(\boldsymbol{h}_s^{l-1})) \tag{2.4}$$

To model temporal dependencies among nodes, a simple alternative is to introduce a time-dependent attention score [VSP17] multiply by a linear transformation of the sender node $\boldsymbol{W}_v \boldsymbol{h}_s^{l-1}$. However, the information loss of a sender node w.r.t different temporal gap is linear, as the **Message** of a sender node is time-independent. Recently Transformer [VSP17] has proposed to add positional encoding to the sender node $\boldsymbol{h}_s^{l-1}$ and obtain a time-dependent **Message**. As adding is a linear operator, we hypothesize that taking a nonlinear transformation of the sender node as time-dependent **Message** would be more sufficient to capture the complex nature of information loss caused by temporal gap between nodes. We define the nonlinear transformation w.r.t the temporal gap $\Delta t(s,t)$ as follows:

$$\textbf{Message}(\boldsymbol{h}_s^{l-1}, \Delta t(s,t)) = \boldsymbol{W}_v \widehat{\boldsymbol{h}}_s^{l-1}, \quad \widehat{\boldsymbol{h}}_s^{l-1} = \sigma(\boldsymbol{W}_t[\boldsymbol{h}_s^{l-1} || \Delta t_{st}]) + \text{TE}(\Delta t_{st})$$

$$\text{TE}(\Delta t)_{2i+1} = cos(\Delta t / 10000^{2i/d}), \quad \text{TE}(\Delta t)_{2i} = sin(\Delta t / 10000^{2i/d}) \tag{2.5}$$

$$\textbf{Attention}(\boldsymbol{h}_s^{l-1}, \boldsymbol{h}_t^{l-1}, \Delta t(s,t)) = (\boldsymbol{W}_k \widehat{\boldsymbol{h}}_s^{l-1})^T (\boldsymbol{W}_q \boldsymbol{h}_t^{l-1}) \cdot \tfrac{1}{\sqrt{d}}$$

where $||$ is the concatenation operation and $\sigma(\cdot)$ is a non-linear activation function. d is the dimension of node embeddings and $\boldsymbol{W}_t$ is a linear transformation applied to the concatenation of the sender node and temporal gap. We adopt the dot-product form of attention where $\boldsymbol{W}_v, \boldsymbol{W}_k, \boldsymbol{W}_q$ projects input node representations into values, keys and queries. The learned attention coefficient is normalized via softmax across all neighbors. Additionally, to distinguish the sender from observations of the object itself, and observations from its neighbors, we learn two sets of projection matrices $\boldsymbol{W}_k, \boldsymbol{W}_v$ for each of these two types. Finally, we stack L layers to get the final representation for each observation as $\boldsymbol{h}_i^t = \boldsymbol{h}^L(\boldsymbol{o}_i^t)$. The overall process is depicted in Figure 2.1 (b).

Temporal Self-Attention. To encode temporal pattern for each observation sequence, we design a non-recurrent temporal self-attention layer that aggregates variable-length sequences into fixed-dimensional sequence representations $\boldsymbol{u}_i$, which is then utilized to approximate the posterior distribution for each latent initial state $\boldsymbol{z}_i^0$. Compared with traditional recurrent models such as RNN,LSTM, self-attention mechanism can be better parallelized for speeding up training process and alleviate the vanishing/exploding gradient problem in these models [SWG20a, VSP17]. Note that we have introduced inter-time edges when creating temporal graph, the observation representations $\boldsymbol{h}_i^t$ already preserve temporal dependency across timestamps. To encode the whole sequence, we introduce a global sequence vector $\boldsymbol{a}_i$ to calculate a weighted sum of observations as the sequence representation :

$$\boldsymbol{a}_i = \tanh\left(\left(\frac{1}{N}\sum_t \widehat{\boldsymbol{h}}_i^t\right)\boldsymbol{W}_a\right), \quad \boldsymbol{u}_i = \frac{1}{N}\sum_t \sigma(\boldsymbol{a}_i^T \widehat{\boldsymbol{h}}_i^t)\widehat{\boldsymbol{h}}_i^t \tag{2.6}$$

where a_i is a simple average of node representations with nonlinear transformation towards the system initial time t_{start} followed by a linear projection $\boldsymbol{W}_a$. The nonlinear transformation is defined as $\widehat{\boldsymbol{h}}_i^t = \sigma(\boldsymbol{W}_t[\boldsymbol{h}_i^t||\Delta t]) + \text{TE}(\Delta t)$ with $\Delta t = (t - t_{\text{start}})$, which is analogous to the **Message** calculation in step 1. Note that if we directly use the observation representation $\boldsymbol{h}_i^t$ from step 1, the sequence representation $\boldsymbol{u}_i$ would be the same when we shift the timestamp for each observation by ΔT, as we only utilize the relative temporal gap between observations. By taking the nonlinear transformation, we actually view each observation has an underlying inter-time edge connected to the virtual initial node at system initial time t_{start}. In this way, the sequence representation $\boldsymbol{u}_i$ reflects latent initial state towards a given time t_{start}, and varies when the initial time changes. The process is depicted in Figure 2.1 (c). Finally, we have the approximated posterior distribution as in Eqn2.7 where f is a neural network translating the sequential representation into the mean and variance of $\boldsymbol{z}_i^0$.

$$q_\phi(\boldsymbol{z}_i^0|o_1, o_2 \cdots o_N) = \mathcal{N}\left(\boldsymbol{\mu}_{z_i^0}, \boldsymbol{\sigma}_{z_i^0}\right), \quad \text{where } \boldsymbol{\mu}_{z_i^0},\ \boldsymbol{\sigma}_{z_i^0} = f(\boldsymbol{u}_i) \tag{2.7}$$

2.4.2 Generative model and decoder

We consider a generative model defined by an ODE whose latent initial state $\boldsymbol{z}_i^0$ is sampled from the approximated posterior distribution $q_\phi(\boldsymbol{z}_i^0|o_1, o_2 \cdots o_N)$ from the encoder. We employ a graph neural network (GNN) in Eqn 2.2 as the ODE function g_i to model the continuous interaction of objects. A decoder is then utilized to recover trajectory from the decoding probability $p(\boldsymbol{o}_i^t|\boldsymbol{z}_i^t)$, characterized by a neural network.

$$\begin{gathered}
\boldsymbol{z}_i^0 \sim p(\boldsymbol{z}_i^0) \approx q_\phi(\boldsymbol{z}_i^0|o_1, o_2 \cdots o_N) \\
\boldsymbol{z}_i^0, \boldsymbol{z}_i^1 \cdots \boldsymbol{z}_i^T = \text{ODESolve}(g_i, [\boldsymbol{z}_1^0, \boldsymbol{z}_2^0 \cdots \boldsymbol{z}_N^0], (t_0, t_1 \cdots t_T)) \\
\boldsymbol{o}_i^t \sim p(\boldsymbol{o}_i^t|\boldsymbol{z}_i^t)
\end{gathered} \tag{2.8}$$

2.4.3 Training

We jointly train the encoder, decoder and generative model by maximizing the evidence lower bound (ELBO) as shown below. As observations for each object are not temporally aligned in a minibatch, we take the union of these timestamps and output the solution of the ODE at them.

$$\begin{aligned}
&ELBO(\theta, \phi) \\
&= \mathbb{E}_{\boldsymbol{Z}^0 \sim q_\phi(\boldsymbol{Z}^0|o_1, \cdots o_N)}[\log p_\theta(o_1, \ldots, o_N)] - \text{KL}[q_\phi(\boldsymbol{Z}^0|o_1, \cdots, o_N) \| p(\boldsymbol{Z}^0)] \\
&= \mathbb{E}_{\boldsymbol{Z}^0 \sim \prod_{i=1}^N q_\phi(\boldsymbol{z}_i^0|o_1, \cdots, o_N)}[\log p_\theta(o_1, \cdots, o_N)] - \text{KL}[\textstyle\prod_{i=1}^N q_\phi(\boldsymbol{z}_i^0|o_1, \cdots, o_N) \| p(\boldsymbol{Z}^0)]
\end{aligned} \tag{2.9}$$

2.5 Experiments

2.5.1 Datasets

We illustrate the performance of our model on three different datasets: particles connected by springs, charged particles [KFW18a] and motion capture data [CMU03]. The first two are simulated datasets, where each sample contains 5 interacting particles in a 2D box with no external forces (but possible collisions with the box). The trajectories are simulated by solving two types of motion PDE for spring system and charged system respectively [KFW18a] with the same number of forward steps 6000 and then subsampling each 100 steps. To generate irregularly-sampled partial observations, for each particle we sample the number of observations n from $\mathcal{U}(40, 52)$ and draw the n observations uniformly from the PDE steps to get the training trajectories. To evaluate extrapolation task, we additionally sample 40 observations following the same procedure from PDE steps $[6000, 12000]$ for testing. The above sampling procedure is conducted independently for each object. We generate 20k training samples and 5k testing samples for these two datasets respectively. For motion capture data, we select the walking sequences of subject 35. Every sample is in the form of 31 trajectories, each tracking a single joint. Similar as simulated datasets, for each joint we sample the number of observations n from $\mathcal{U}(30, 42)$ and draw the n observations uniformly from first 50 frames for training trajectories. For testing, we additionally sampled 40 observations from frames $[51, 99]$. We split the different walking trials into non-overlapping training (15 trials) and test sets (7 trials).

We conduct experiment on both interpolation and extrapolation tasks as proposed in [RCD19b]. For all experiments, we report the mean squared error (MSE) on the test set. For all datasets, we rescale the time range to be in $[0, 1]$. Our implementation is available online[1]. More details can be found in the supplementary materials.

2.5.2 Baselines and Model Variants

Baselines. To the best of our knowledge, existing works on modeling multi-agent dynamic system with underlying graph structure cannot handle irregularly-sampled partial observations, in which these models require full observation at timestamp t in order to make prediction at timestamp $t + 1$ [KFW18a, BPL16]. Therefore, we firstly compare our model with different encoder structures to infer the initial states. Specifically, we consider Latent-ODE [RCD19b] which has shown to be successful for encoding single irregularly-sampled time series without considering graph interaction among agents. Edge-GNN [GC19] incorporates temporal information by viewing time gap as an edge attribute. Weight-Decay considers a simple exponential decay function for time gap as similar in [CWL18], which models $\boldsymbol{h}(t + \Delta t) = exp\{-\tau \Delta t\} \cdot \boldsymbol{h}(t)$ with a learnable decay parameter τ. The sequence representation of Edge-GNN and Weight-Decay is the weighted sum of observations within a sequence. We additionally compare LG-ODE against an RNN-based MTS model for handling irregularly-sampled missing values [CPC18] where the graph structure is not considered. It jointly imputes missing values for all agents by simple concatenation

[1] https://github.com/ZijieH/LG-ODE.git

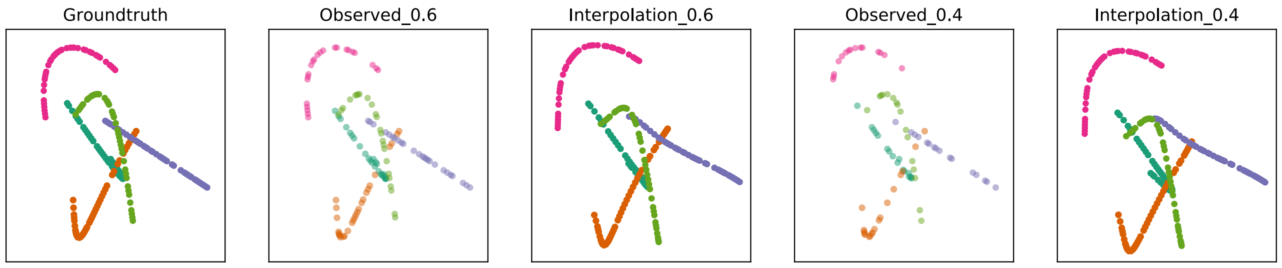


Figure 2.2: Visualization of interpolation results for spring system.

of their feature vectors. We compare it in the Interpolation Task which is to imputes missing values within the observed sequences. After imputation, we employ NRI [KFW18a] which is a multi-agent dynamic system model with regular observations and graph input to predict future sequences. We refer to this task as Extrapolation Task. In what follows, we refer to the combination of these two models as RNN-NRI.

Model Variants. Our proposed encoder contains two modules: dynamic node representation network followed by a temporal self-attention. To further analyze the components within each module, we conduct an ablation study by considering five model variants. Firstly, module one contains two core components: attention mechanism and learnable positional encoding within GNN for capturing temporal and spatial dependency among nodes. We therefore remove them separately and get LG-ODE-no att, LG-ODE-no PE respectively. We additionally compare our learnable positional encoding with manually-designed positional encoding [VSP17] denoted as LG-ODE-fixed PE. Secondly, we apply various sequence representation methods to test the efficiency of module two: LG-ODE-first takes the first observation in a sequence as sequence representation and LG-ODE-mean uses the mean pooling of all observations as sequence representation.

2.5.3 Results on Interpolation Task

Set up. In this task, we condition on a subset of observations $(40\%, 60\%, 80\%)$ from time (t_0, t_N) and aim to reconstruct the full trajectories in the same time range. We subsample timepoints for each object independently.

Table 2.1 shows the interpolation MSE across different datasets and methods. Latent-ODE performs well on encoding single timeseries but fails to consider the interaction among objects, resulting in its poor performance in the multi-agent dynamic system setting. Weight-Decay and Edge-GNN utilize fixed linear transformation of sender node to model information loss across timestamps, which is not

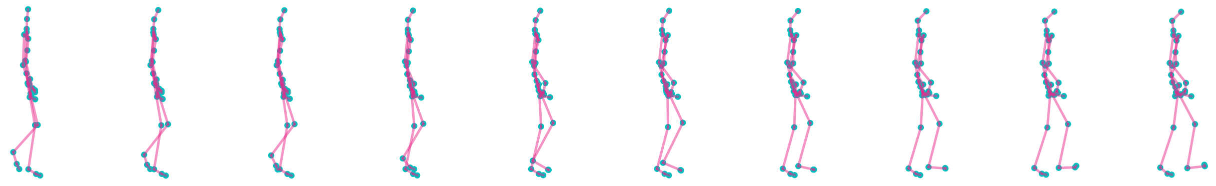
Table 2.1: Mean Squared Error(MSE) $\times 10^{-2}$ on Interpolation task.

	Springs			Charged			Motion		
Observed ratio	40%	60%	80%	40%	60%	80%	40%	60%	80%
Latent-ODE	0.5454	0.5036	0.4290	1.1799	1.1198	0.8332	0.7709	0.4826	0.3603
Weight-Decay	1.1634	1.1377	1.6217	2.8419	2.2547	1.5390	1.9007	2.0023	1.6894
Edge-GNN	1.3370	1.2786	0.8188	1.5795	1.5618	1.1420	2.7670	2.6582	1.8485
NRI + RNN	0.5225	0.4049	0.3548	1.3913	1.1659	1.0344	0.5845	0.5395	0.5204
LG-ODE	**0.3350**	**0.3170**	**0.2641**	**0.9234**	**0.8277**	**0.8046**	0.4515	**0.2870**	**0.3414**
LG-ODE-first	1.3017	1.1918	1.0796	2.5105	2.6714	2.3208	1.4904	1.3702	1.2107
LG-ODE-mean	0.3896	0.3901	0.3268	1.1246	1.0050	0.9133	0.6415	0.5834	0.5549
LG-ODE-no att	0.5145	0.4198	0.4510	0.9372	0.9503	0.9752	0.6991	0.6998	0.7452
LG-ODE-no PE	0.4431	0.4278	0.3879	1.0450	1.0350	0.9621	0.4677	0.4808	0.4799
LG-ODE-fixed PE	0.4285	0.4445	0.4083	0.9838	0.9775	0.9524	**0.4215**	0.4371	0.4313

sufficient to capture the complex temporal dependency. RNN-NRI though handles the irregular temporal information by a specially designed decay function, it conducts imputation without considering the graph interaction among objects and thus obtaining a poor performance. By comparing model variants for temporal self-attention module, we notice that taking the first observation as sequence representation produces high reconstruction error, which is expected as the first observable time for each sequence may not be the same so the inferred latent initial states are not aligned. Averaging over observations assumes equal contribution for each observation and ignores the temporal dependency, resulting in its poor performance. For module one, experiment results on model variants suggest that distinguishing the importance of nodes w.r.t time and incorporating temporal information via learnable positional encoding would benefit model performance. Notably, the performance gap between LG-ODE and other methods increases when the observation percentage gets smaller, which indicates the effectiveness of LG-ODE on sparse data. When observation percentage increases, the reconstruction loss of all models tends to be smaller, which is expected. Figure 2.2 visualizes the interpolation results of our model under different observation percentage for the spring system. Figure 2.3 visualizes the interpolation results for motion capture data with 60% observation percentage.

2.5.4 Results on Extrapolation Task

Set up. In this task we split the time into two parts: (t_0, t_{N_1}) and (t_{N_1}, t_N). We condition on the first half of observations and reconstruct the second half. For training, we condition on observations from (t_1, t_2)


(a) Groundtruth.

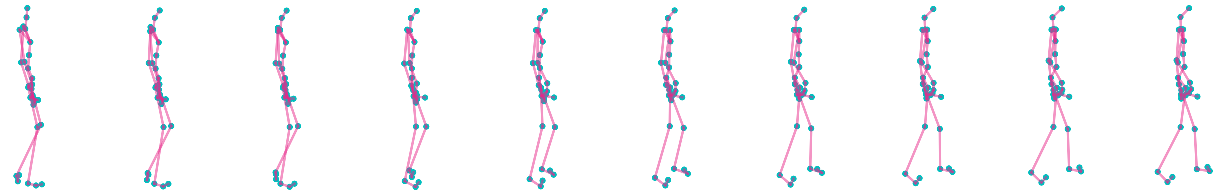

(b) Predictions with 0.6 observation ratio.

Figure 2.3: Visualization of interpolation results for walking motion data.

and reconstruct the trajectories in (t_2, t_3). For testing, we condition on the observations from (t_1, t_3) but tries to reconstruct future trajectories within (t_3, t_4). Similar to interpolation task, we experiment on conditioning only on a subset of observations in the first half and run the encoder on the subset to estimate the latent initial states. We evaluate model's performance on reconstructing the full trajectories in the second half.

Table 2.2: Mean Squared Error(MSE) $\times 10^{-2}$ on Extrapolation task.

Extrapolation	Springs			Charged			Motion		
Observed ratio	40%	60%	80%	40%	60%	80%	40%	60%	80%
Latent-ODE	6.6923	4.2478	4.3192	13.5852	12.7874	20.5501	2.4186	2.9061	2.6590
Weight-Decay	6.1559	5.7416	5.3712	9.4764	9.1008	9.0886	16.8031	13.6696	13.6796
Edge-GNN	6.0417	4.9220	3.2281	9.2124	9.1410	8.8341	13.2991	13.9676	9.8669
NRI + RNN	2.6638	2.4003	2.5550	7.1776	6.9882	6.6736	3.5380	3.0119	2.6006
LG-ODE	**1.7839**	1.8084	**1.7139**	6.5320	**6.4338**	**6.2448**	**1.2843**	**1.2435**	1.2010
LG-ODE-first	6.5742	6.3243	5.7788	9.3782	9.2107	8.4765	3.8864	3.2849	3.0001
LG-ODE-mean	2.2499	2.1165	2.2516	9.1355	8.7820	8.4422	1.3169	1.3008	1.2534
LG-ODE-no att	2.3847	2.1216	1.9634	7.2958	7.3609	6.7026	3.4510	3.2178	3.9917
LG-ODE-no PE	1.7943	1.8172	1.7332	6.9961	6.7208	6.5852	1.5054	1.2997	1.2029
LG-ODE-fixed PE	1.7905	**1.7634**	1.7545	**6.4520**	6.4706	6.3543	1.4624	1.2517	**1.1992**

Table 2.2 shows the MSE on extrapolation task. The average MSE in extrapolation task is greater

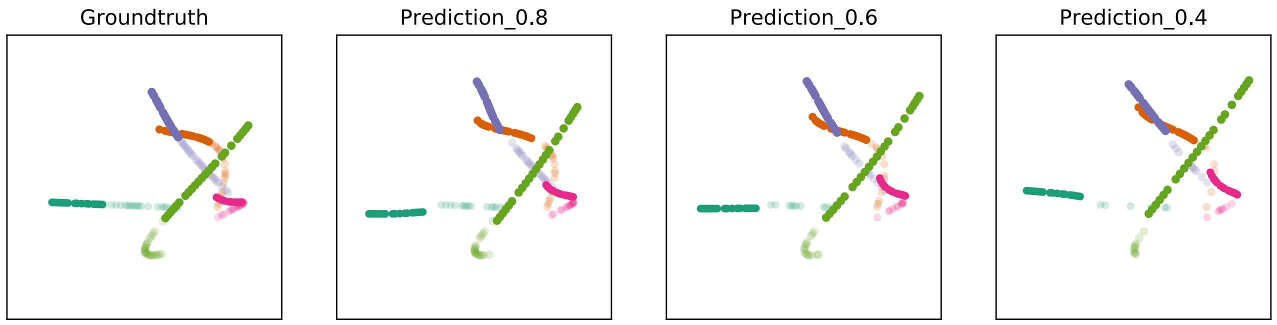


Figure 2.4: Visualization of extrapolation results for spring system. Semi-transparent paths denote observations from first-half of time, from which the latent initial states are estimated. Solid paths denote model predictions.

than interpolation task, which is expected as predicting the future is a more challenging task. Similar as in interpolation task, when observation percentage increases, the prediction error of all models tends to become smaller. LG-ODE achieves better results across different datasets and settings, which verifies the effectiveness of our design to capture structural dependency among objects, and temporal dependency within observation sequence. Specifically, RNN-NRI is a two-step model that first imputes each time series into regular-sampled one to make it a valid input for NRI, and then predict trajectories with the graph structure. LG-ODE instead is an end-to-end framework. The prediction error for RNN-NRI is large and one possible reason is that we use estimated imputation values for missing data which would add noise to NRI. We also notice that the performance drop due to the sparsity of observations is small in LG-ODE compared with other baselines, which shows our model is more powerful especially when data is sparse. We illustrate the predicted trajectories of spring system under different observation percentage as shown in Figure 2.4.

2.6 Discussion and Conclusion

In this paper, we propose LG-ODE for learning continuous multi-agent system dynamics from irregularly-sampled partial observations. We model system dynamics through a neural ordinary differential equation and draw the latent initial states for each object simultaneously through a novel encoder that is able to capture the interaction among objects. The joint learning of initial states not only captures interaction among objects but can benefit the learning when an object only has few observations. We achieve state-of-the-art performance in both interpolating missing values and extrapolating future trajectories. An limitation of current model is that we assume the underlying interaction graph is fixed over time. In the future, we plan to learn the system dynamics when the underlying interaction graph is evolving.

Part I

Injecting Data-Inspired Inductive Bias

CHAPTER 3

CG-ODE: Coupled Graph ODE for Learning Interacting System Dynamics

Many real-world systems such as social networks and moving planets are dynamic in nature, where a set of coupled objects are connected via the interaction graph and exhibit complex behavior along the time. For example, the COVID-19 pandemic can be considered as a dynamical system, where objects represent geographical locations (e.g., states) whose daily confirmed cases of infection evolve over time. Outbreak at one location may influence another location as people travel between these locations, forming a graph. Thus, how to model and predict the complex dynamics for these systems becomes a critical research problem. Existing work on modeling graph-structured data mostly assumes a static setting. How to handle dynamic graphs remains to be further explored. On one hand, features of objects change over time, influenced by the linked objects in the interaction graph. On the other hand, the graph itself can also evolve, where new interactions (links) may form and existing links may drop, which may in turn be affected by the dynamic features of objects. In this paper, we propose *coupled graph ODE*: a novel latent ordinary differential equation (ODE) generative model that learns the coupled dynamics of nodes and edges with a graph neural network (GNN) based ODE in a continuous manner. Our model consists of two coupled ODE functions for modeling the dynamics of edges and nodes based on their latent representations respectively. It employs a novel encoder parameterized by a GNN for inferring the initial states from historical data, which serves as the starting point of the predicted latent trajectories. Experiment results on the COVID-19 dataset and the simulated social network dataset demonstrate the effectiveness of our proposed method.

3.1 Introduction

Real-world systems in various domains such as physics, biology, robotics can be viewed as dynamic interacting systems, where a set of objects interact with each other and demonstrate complex behavior longitudinally. Learning the underlying dynamics of an interacting system is essential in many real-world applications. For example, learning the movement of robotics can improve planning and control in future design [LWZ19b]; studying the trajectories of moving planets can discover potential new physical laws [CSB20]; understanding the spread of COVID-19 can help governments develop disease prevention and intervention plans [CPK21], etc. With the recent advances in deep learning techniques, researchers have started building neural-based simulators, aiming to approximate complex system interactions with neural networks [KFW18a, LWZ19b, BPL16, CUT16, HSW20b]. As interacting systems contain multiple objects and are thus graph structured data, existing work [BPL16, KFW18a] usually employs graph neural networks (GNN) to reason how objects interact and to predict object (node) features in the future. However, a fundamental assumption behind a vast majority of work is that the interaction graphs among objects are static [lDH17], such as particles connected by springs where the spring structure remains unchanged. Nonetheless, the dynamic nature of many real-world systems does not only exhibit in the evolution of node features, but may also manifest as the dynamic changes in the graph structure. One example is the spread of COVID-19 within U.S., where nodes are 50 states and the interaction graph represents the population travel patterns between states. Both the daily outbreak statistics (such as the number of new cases of each state) and the mobility patterns between states (such as the number of people traveling from one state to another) evolve over time [CPK21].

Even though the graph structure and node features are two distinct data representations, they are inherently correlated [lDH17]. On one hand, node features are likely to be affected by other nodes whom they interact with in the graph. In the aforementioned COVID-19 example, New Jersey's daily confirmed cases are more likely to be affected by states with large population inflow (such as New York, Pennsylvania) than by others such as California. This is shown in Figure 3.1 where the daily death counts of New Jersey is more correlated with that of New York than California. Similar phenomena can also be observed in social networks where individuals are likely to be influenced by their friends [GSG17, JHH20, WWW20, WHL21a, WYW18]. On the other hand, the dynamics of node features may also affect the interaction. For example, the states' severity of the epidemic situation may, in short term, impact the population flow between them as shown in Figure 3.2. Inspired by these observations, we propose a novel ordinary differential equation (ODE) based generative model: coupled graph ODE, for predicting the dynamics of node features by jointly considering the evolution of nodes and edges.

In order to model the co-evolution of nodes and edges, we design two coupled ODE functions to

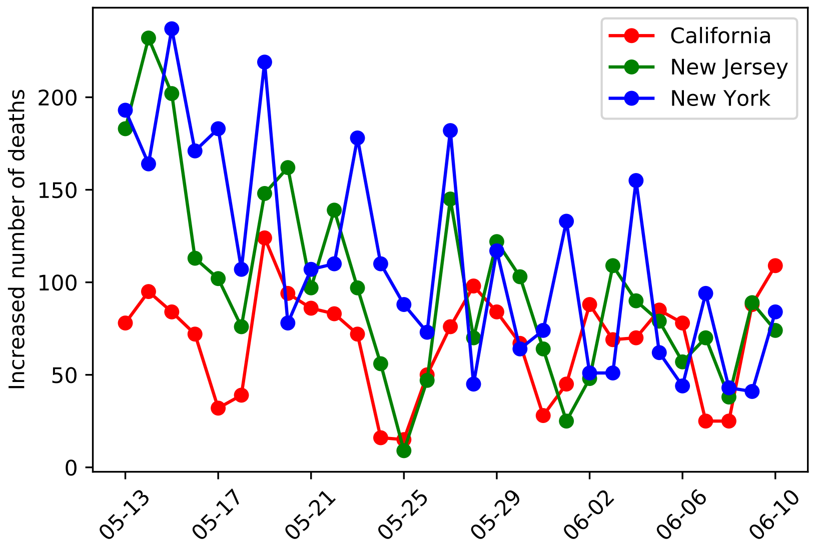
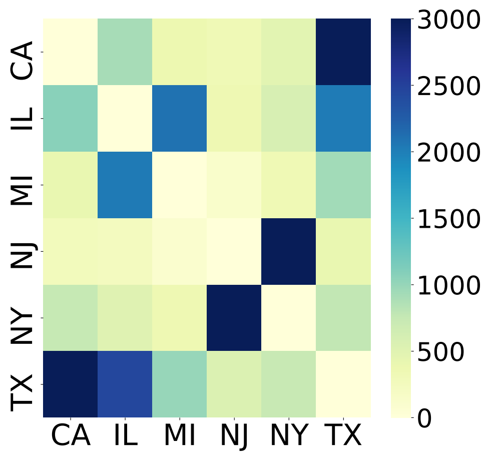
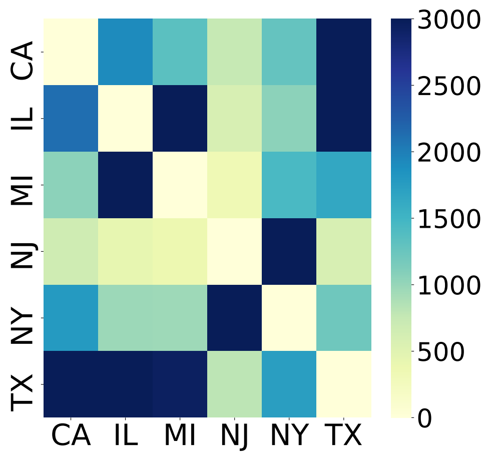


Figure 3.1: COVID-19 death count time series of three states in U.S. Correlation is higher between two states that have higher population flow.



(a) Population flow on May.11



(b) Population flow on Aug.8

Figure 3.2: Population flow in May and August with self-loop flow excluded (Diagonal entries). May has less population flow due to the "close border" policies in many states.

model the continuous evolution of nodes and edges in the latent space respectively, considering the mutual influence between them. The continuous nature of our model allows it to track the evolution of the underlying system from irregular observations, and is expected to offer improved performance compared to using discrete methods to model a continuous dynamical system such as the spread of COVID-19 [PMP19b, CPC18]. For the edge ODE, the widely-used generative process assumes that the new edges are completely determined by the features of source and target nodes [GSG17, HHN19]. However, we add an additional term to model the self-evolution of edges. Such self-evolution is widely observed in many real-world systems. For example, the population flow between two states will change naturally due to some seasonal factors (e.g. holidays), which is not necessarily related to the node features (severity of the epidemic situation). Likewise, for the node ODE, we consider the self-evolution of nodes, as well as the potential influence received from neighbors in the interaction graph.

Since we propose to learn continuous system dynamics using ODEs, a fundamental challenge lies in how to estimate the latent initial states for the whole system. We borrow a similar idea from [RCD19b, HSW20b] where a VAE-based latent ODE model is proposed to estimate the latent initial states with uncertainty. As objects are highly-coupled in interacting systems, we propose a novel GNN as the encoder which infers the latent initial states for all objects simultaneously. Overall, our model consists of three parts that are jointly trained together: (1) An encoder that infers the latent initial states for all objects and edges simultaneously considering their interaction; (2) A generative model parameterized by two coupled ODEs that learns the evolution pattern for edges and nodes respectively. (3) Two decoders for nodes and edges respectively which project the latent states for nodes and edges to the original input spaces. We conduct extensive experiments on the COVID-19 dataset and one simulated social network dataset. Experiment results verify the effectiveness of our proposed method, especially for long-range predictions. We also conduct case studies on how travel-related policies could affect the number of confirmed cases in the future on the COVID-19 dataset, by adding intervention to the interaction graph, which has demonstrated that our model is a promising tool for policymakers.

3.2 Problem Formulation

We consider a dynamical system with N interacting objects. Our input consists of the trajectories (features) of these objects and the directed weighted interaction graph among them which changes over time. We denote the snapshots of the interaction graph as $\mathcal{G} = \left\{G^1, G^2, \ldots, G^T\right\}$, where $G^t = (\mathcal{V}, \mathcal{E}^t)$ is the interaction graph at timestamp t with $\mathcal{V}$ denoting the set of N interacting objects and $\mathcal{E}^t$ being the set of directed weighted edges, respectively. For every pair of connected nodes $i, j \in \mathcal{V}$ at timestamp t, $w^t_{i \to j} \in \mathbb{R}$ denotes the weight of the directed edge linking them. The edge weight can be asymmetric, i.e., $w_{i \to j}^t \in \mathbb{R}$ may not necessarily hold the same value as $w_{i \to j}^t \in \mathbb{R}$. We use $\mathcal{A} = \left\{A^1, A^2, \ldots, A^T\right\}$ to denote the weighted adjacency matrix sequence.

We denote the node trajectory sequence as $\mathcal{X} = \{X^1, X^2, \ldots, X^T\}$, where X^t is the feature matrix of all N objects at timestamp t. We use x_i^t to denote the feature vector of object i at timestamp t. Based on the observed coupled trajectories of a dynamical system, i.e. $\mathcal{X}, \mathcal{A}$, our goal is to learn the underlying dynamics which is built upon the latent representations for nodes $\boldsymbol{z}_i^t \in \mathbb{R}^d$ and edges $\boldsymbol{z}_{i \to j}^t \in \mathbb{R}^d$, and to utilize them to forecast trajectories $X^t (t > T)$ in the future.

3.3 Related Work and Preliminaries

Ordinary Differential Equations (ODE) for Multi-agent Dynamical Systems. The dynamic nature of a multi-agent dynamical system can be captured by a series of first-order ordinary differential equations (ODE), which describes the state evolution for a set of M latent dependent variables over continuous time $t \in \mathbb{R}$. Existing work [RCD19b, CRB18a] usually associates each object with a latent state variable $z_i^t \in \mathbb{R}^d, \ i = 1, 2 \cdots N$, with the corresponding ODE: $\dot{\boldsymbol{z}}_i^t := \frac{d\boldsymbol{z}_i^t}{dt} = g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right)$, which describes how the trajectory of each object changes over time. The ODE function g is usually hand-crafted by domain experts in the past and some recent studies [RCD19b, HSW20b] have proposed to parameterize it as a neural network which can be learned from data. To capture the continuous interaction among objects, graph neural network (GNN) is employed to parameterize the ODE function g in a recent study [HSW20b]. Given the latent initial states $\boldsymbol{z}_1^0, \cdots \boldsymbol{z}_N^0 \in \mathbb{R}^d$ for each object, z_i^t is the solution to an ODE initial-value problem (IVP), which can be evaluated at any desired time as shown in Eqn 7.1 using a numerical ODE solver such as Runge-Kuttais [MSH19]. The latent state z_i^t is further decoded to generate the predicted trajectory at timestamp t: $x_i^t = f_{\text{dec}}(z_i^t)$. Given the ODE function, the latent initial states z_i^0 for each object determine the whole trajectory.

$$\boldsymbol{z}_i^T = \boldsymbol{z}_i^0 + \int_{t=0}^{T} g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right) dt \tag{3.1}$$

However, one major limitation of these methods is that, they assume the interaction graph among agents is static. Therefore, the set of M latent state variables z_i^t are only for nodes, i.e. $M = N$. In reality, the network structure may change over time, which requires the modeling of latent edge state $z_{i \to j}^t$ as well. Moreover, the evolution of latent node and edge states are highly-coupled. Taking the spread of

COVID-19 as an example, the number of cases for each state x_i^t can be affected by other states x_j^t via the (past) population flow between them. On the other hand, the (future) population flow between two states may change in response to the varying severity of states' epidemic situation x_i^t, x_j^t.

GNN for Static Graphs. GNN is a class of neural networks that operate directly on graph-structured data by passing local messages[KW17, VCC18, XHL19]. It has been widely used for approximating pair-wise object interactions in multi-agent dynamical systems[KFW18a, CUT16]. The majority of work on GNN mainly focus on static graphs and is designed for tasks such as node classification [KW17, VCC18], graph clustering and matching [BDB19], etc. While various architecture exists, the update procedure for a single GNN layer can be characterized by two major operations: (1) Extracting information. For example, graph convolution network (GCN) [KW17] utilizes the normalized Laplacian as the attention weight for attending each sender node with a linear transformation. It could be regarded as an approximation of spectral domain convolution of the graph signals. (2) Aggregating information from neighbors. Basic aggregation operators including mean, sum and max, while sophisticated pooling and normalization functions are also been proposed. In multi-agent dynamical systems where edges are static and only node attributes evolve, static GNNs are often employed as neural physical simulators to capture the complex interaction among objects, which reveals how system changes from timestamp t to timestamp $t+1$ [BPL16]. However, discrete GNNs may have inferior performance compared with continuous graph ODE-based methods when the system is continuous by nature, such as the spread of COVID-19. They also fail to handle irregularity and partial observations in multi-agent dynamical systems, as opposed to the aforementioned ODE-based methods [HSW20b].

GNN for Dynamic Graphs. In many real-world applications, both nodes and edges are dynamic such as the traffic network [PMP19b]. In order to learn hidden patterns from those dynamic graphs, spatial-temporal GNNs are proposed which is able to consider spatial and temporal dependency at the same time. To achieve this, existing approaches integrate static graph convolutions to capture spatial dependency with RNNs,CNNs or self-attention mechanism to model temporal dependency [SWG20b, HHN19, KFW18a].

The learned node representations can be utilized for downstream tasks such as link prediction [HHN19, SWG20b, GSG17, HDW20b]. However, they usually assume the new edges are solely determined by the end nodes, while in many real-life scenarios like the spread COVID-19, the self-evolution of edges also exists. Also, they are discrete models and may fail to model dynamical systems that are continuous by nature, compared to ODE-based methods.

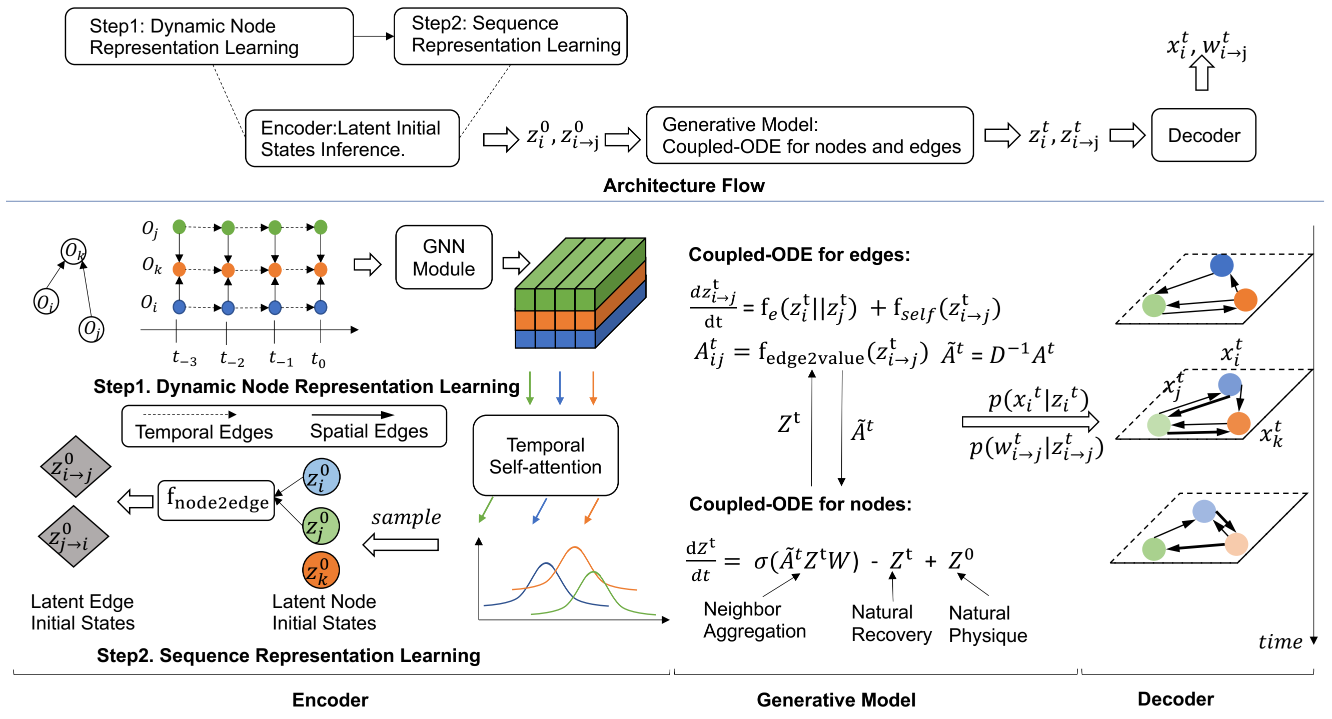
Latent Graph ODE model for Dynamical Systems. Dynamical systems with static interaction graph is a special case in our setting. As mentioned in Sec 3.1, [HSW20b] employed GNN as the ODE function and it follows the framework of variational autoencoder (VAE) [KW14], where an approximate posterior distribution $q_\phi\left(\boldsymbol{z}_i^0 \mid \mathcal{X}, \mathcal{A}\right)$ is computed over each latent initial state for an object from the encoder. The prior distribution $p(z_i^0)$, which is a standard normal distribution, adds significant regularization over how latent distribution looks like via the Kullback–Leibler divergence term in the loss function, which differs VAE from other autoencoder frameworks. z_i^0 is then sampled from the posterior distribution and the entire trajectory is determined by z_i^0 and the generative model defined by the ODE function g for all objects. Finally, the decoder outputs the predicted trajectories by mapping z_i^t to the original feature space: $x_i^t = f_{\text{dec}}(z_i^t)$. We model dynamical systems with evolving interaction graph under the same framework, where in addition to modeling latent states for nodes, we also incorporate latent states for edges. Then the challenges lie in: (1) How can we infer the initial states for both edges and nodes considering their mutual influence? (2) How to specify the ODE functions for guiding the co-evolution for node and edge latent states respectively?

3.4 Model

In this section, we present Coupled Graph ODE (CG-ODE) for learning continuous multi-agent dynamical systems with evolving interaction graph. The overall framework is depicted in Figure 7.1. Following the framework of VAE, CG-ODE consists of three parts that are trained jointly: (1) An encoder that infers the latent initial states for nodes and edges considering the interaction among objects. (2) A generative model characterized by two coupled ODE functions for edges and nodes respectively, with the goal of learning the latent dynamics of the system. (3) Two decoders that generate the predicted nodes and edges based on the decoding likelihood determined by the latent states $p\left(\boldsymbol{x}_i^t \mid \boldsymbol{z}_i^t\right)$ and $p\left(w_{i\to j}^t \mid \boldsymbol{z}_{i\to j}^t\right)$.

3.4.1 Encoder for Initial States

Given the trajectory sequence $\mathcal{X}$ and the snapshots of the interaction graph among objects, the encoder firstly computes a posterior distribution of latent initial state for each object: $q_\phi\left(\boldsymbol{z}_i^0 \mid \mathcal{X}, \mathcal{A}\right)$, from which z_i^0 is sampled. As in multi-agent dynamical systems, objects are highly-coupled and their mutual influence is propagated through the directed weighted edges, we compute the distributions for all objects simultaneously by considering both their trajectories and the dynamic interaction graph among them.



Step1: Dynamic Node Representation Learning
Step2: Sequence Representation Learning
Encoder:Latent Initial States Inference.
$z_i^0, z_{i\to j}^0$
Generative Model: Coupled-ODE for nodes and edges
$z_i^t, z_{i\to j}^t$
Decoder
$x_i^t, w_{i\to j}^t$
Architecture Flow
GNN Module
Step1. Dynamic Node Representation Learning
Temporal Edges
Spatial Edges
Temporal Self-attention
$f_{node2edge}$
sample
Latent Edge Initial States
Latent Node Initial States
Step2. Sequence Representation Learning
Coupled-ODE for edges:
$\frac{dz_{i\to j}^t}{dt} = f_e(z_i^t || z_j^t) + f_{self}(z_{i\to j}^t)$
$A_{ij}^t = f_{edge2value}(z_{i\to j}^t)\ \tilde{A}^t = D^{-1}A^t$
Z^t
$\tilde{A}^t$
$p(x_i^t | z_i^t)$
$p(w_{i\to j}^t | z_{i\to j}^t)$
Coupled-ODE for nodes:
$\frac{dZ^t}{dt} = \sigma(\tilde{A}^t Z^t W) - Z^t + Z^0$
Neighbor Aggregation
Natural Recovery
Natural Physique
time
Encoder
Generative Model
Decoder

Figure 3.3: The Overall framework of Coupled Graph ODE: Firstly, the encoder computes the latent initial states for edges and nodes respectively based on the observed sequence of node attributes and adjacency matrix sequence so far with two steps: Step1: Dynamic node representation learning over the constructed temporal graph. Step2: Sequence representation learning for summarizing over each observation sequence. Then the generative model calls the ODE solver to solve the two coupled ODEs for nodes and edges, which outputs the predicted latent states for nodes and edges in the future. Finally, decoders generate the predicted nodes and edges based on their respective decoding likelihood determined by the latent states.

After inferring the latent initial states for all nodes, we generate the latent initial states for edges based on the inferred node initial states.

3.4.1.1 Latent initial states for nodes

We now present how to infer the latent initial states for each object. Instead of encoding the temporal pattern for each object independently using an RNN [RCD19b], we incorporate the structural pattern by constructing a temporal graph as shown in Figure 7.1 Step 1, where each node is an observation of an object at a specific timestamp. For edges, we firstly construct spatial edges between two objects at each timestamp t based on the corresponding weighted adjacency matrix A^t, where the edge weight is naturally given by $w^t_{i\to j}$. Then, to preserve the autoregressive nature of each trajectory, we only introduce directed temporal edges $w_{i(t)\to i(t')}$ where $t < t'$ are the timestamps of two consecutive observations of i. We use $w_{i(t)\to j(t')}$ as a uniform expression for both spatial and temporal edges, i.e. for spatial edges (when $t = t'$), $w_{i(t)\to j(t')}$ is equivalent to $w^t_{i\to j}$; for temporal edges (when $i = j$, $t < t'$), $w_{i(t)\to i(t')}$ becomes $w^t_{i\to i}$. By introducing temporal edges and stacking multiple layers of GNN, we can capture the influence from historical observations to the current observation.

Based on the constructed temporal graph, we infer the latent initial states for objects via a two-step process similar as in [RCD19b]: 1.) **Dynamic Node Representation Learning**, where we aim to learn a structural representation $h_{i(t)}$ for each observation x^t_i. 2.) **Sequence Representation Learning**, where we employ a self-attention mechanism to summarize each observation sequence into a fixed-dimensional vector u_i. The sequence representation u_i is then utilized to generate the mean and variance for the Gaussian posterior distribution for the latent initial state of object z^0_i.

To learn a structural representation for each observation over the weighted, directed temporal graph, we propose an attention-based spatial-temporal GNN that attends over the immediate neighbors of a node as defined in Eqn 3.2. Here $h^{l-1}_{j(t')}$ is the representation of object j at timestamp t' from layer $l-1$, $\sigma(\cdot)$ is a non-linear activation function and d is the dimension of the latent node representations. The attention score $e^l_{j(t')\to i(t)}$ for both spatial edges (where $t = t'$) and temporal edges (where $i = j$ and $t < t'$), is defined as the multiplication of the corresponding edge weight, and the computed affinity score based on representations of sender node and target node. We adopt the dot-product to compute the affinity score where W_v, W_k, W_q projects input node representations into values, keys and queries. The learned attention coefficient is normalized via softmax across all neighbors. As the temporal graph contains spatial and temporal edges, we add temporal encoding [VSP17, RCD19b] to the sender node representation in order to distinguish them. Finally, we stack L layers to get the final representation for each node: $h_{i(t)} = h_{i(t)}^L$

$$\begin{gathered}
h_{i(t)}^l = h_{i(t)}^l + \sigma\left(\sum_{j(t') \in \mathcal{N}_{i(t)}} e_{j(t') \to i(t)}^l \times W_v \widehat{h}_{j(t')}^{l-1}\right) \\
e_{j(t') \to i(t)}^l = w_{j(t') \to i(t)} \times \alpha_{j(t') \to i(t)}^l \\
\alpha_{j(t') \to i(t)}^l = \left(W_k \widehat{h}_{j(t')}^{l-1}\right)^T \left(W_q h_{i(t)}^{l-1}\right) \cdot \frac{1}{\sqrt{d}} \\
\widehat{h}_{j(t')}^{l-1} = h_{j(t')}^{l-1} + \mathrm{TE}(t' - t) \\
\mathrm{TE}(\Delta t)_{2i} = \sin\left(\frac{\Delta t}{10000^{2i/d}}\right), \ \mathrm{TE}(\Delta t)_{2i+1} = \cos\left(\frac{\Delta t}{10000^{2i/d}}\right)
\end{gathered} \tag{3.2}$$

Next, we employ a self-attention mechanism to generate sequence representation for each object, which is then utilized to compute the posterior distribution for the latent node initial state. Compared with traditional recurrent models that encode temporal pattern within each sequence such as RNN, LSTM, self-attention mechanism can be better parallelized for speeding up the training process and alleviate the vanishing/exploding gradient problem in these models [SWG20b]. We introduce a global sequence vector a_i to calculate a weighted sum of observations as the sequence representation, where a_i is the average of node representations with a nonlinear transformation W_a. The process is shown in Figure 7.1 Step 2 and Eqn 5.3, where $\widehat{h}_{i(t)} = h_{i(t)} + \mathrm{TE}(t)$.

$$u_i = \frac{1}{N} \sum_t \sigma\left(a_i^T \widehat{h}_{i(t)} \widehat{h}_{i(t)}\right), \ a_i = \tanh\left(\left(\frac{1}{N} \sum_t \widehat{h}_{i(t)}\right) W_a\right) \tag{3.3}$$

Finally, we compute the mean and variance of the approximated posterior distribution from the sequence representation u_i, and sample z_i^0 from it.

$$\begin{gathered}
q_\phi\left(z_i^0 \mid \mathcal{X}, \mathcal{A}\right) = \mathcal{N}\left(\mu_{z_i^0}, \boldsymbol{\sigma}_{z_i^0}\right), \ \mu_{z_i^0}, \sigma_{z_i^0} = f_{\text{trans}}\left(u_i\right) \\
z_i^0 \sim p\left(z_i^0\right) \approx q_\phi\left(z_i^0 \mid \mathcal{X}, \mathcal{A}\right)
\end{gathered} \tag{3.4}$$

3.4.1.2 Latent initial states for edges

Given the latent initial states for a pair of nodes z_i^0, z_j^0, the latent initial state for each edge is given by Eqn 3.5, where $||$ denotes the concatenation operation.

$$z_{i\to j}^0 = f_{\text{edge}}\left([z_i^0 || z_j^0]\right) \tag{3.5}$$

3.4.2 ODE Generative Model and Decoder

After computing the latent initial states for nodes and edges, we now define the ODE function that drives the system to move forward. In multi-agent dynamical systems, the latent node and edge states are co-evolving along with time. We therefore propose the coupled ODE functions for edge and nodes respectively as shown in Eqn 3.6, where $Z^t \in \mathbb{R}^{N\times d}$ denotes the latent state matrix for all N objects, $W \in \mathbb{R}^{d\times d}$ is a linear feature transformation matrix. The node ODE function consists of three parts and can be understood from an epidemic modeling perspective [XQT20]. If we view Z^t as the infection conditions for all states in the U.S at timestamp t, the first term accounts for the infection from neighbors; the second term $-Z^t$ can be viewed as natural recovery and the third term Z^0 is for natural physique [BBH19]. Note that we use the normalized adjacency matrix $\widetilde{A} = D^{-1}A$ to compute message passing from neighbors, where D is the degree matrix of A defined as $D_{ii} = \sum_j A_{ij}$. This is because when solving the ODE using a numerical solver, it is equivalent to stack multiple GNN layers as time progresses. Using an unnormalized adjacency matrix would therefore cause the potential gradient exploding problem. As our interaction graph is asymmetric, we normalize it to $D^{-1}A$ instead of $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ for symmetric adjacency matrix. The edge ODE function consists of two parts. $f_e : \mathbb{R}^{2d} \to \mathbb{R}^d$ is a mapping function that transforms the concatenation of two nodes to the latent state of their corresponding edge. $f_{\text{self}} : \mathbb{R}^d \to \mathbb{R}^d$ accounts for the self-evolution of edges. For example, the population flow between California and Washington may change over time due to factors like holidays and quarantine policies, which may not be driven by the severity of COVID-19 at these two locations, i.e. z_i^t and z_j^t. $f_{\text{edge2value}} : \mathbb{R}^d \to \mathbb{R}$ transforms the latent edge states to a scalar, which is then utilized in the node ODE function.

$$\begin{gathered}
\frac{dZ^t}{dt} = \sigma\left(\widetilde{A}^t Z^t W\right) - Z^t + Z^0 \\
\frac{dz_{i\to j}^t}{dt} = f_e\left([z_i^t || z_j^t]\right) + f_{\text{self}}\left(z_{i\to j}^t\right) \\
A_{ij}^t = f_{\text{edge2value}}\left(z_{i\to j}^t\right), \ \widetilde{A}^t = D^{-1}A^t
\end{gathered} \tag{3.6}$$

Given the coupled ODE functions and the initial states for nodes and edges, the trajectories for all objects are determined. We compute the predicted trajectories for each object and the graph based on the decoding likelihood $p\left(\boldsymbol{x}_i^t \mid \boldsymbol{z}_i^t\right)$ and $p\left(w_{i\to j}^t \mid \boldsymbol{z}_{i\to j}^t\right)$ with two decoding functions f_{decN}, f_{decE} respectively.

3.4.3 Training

Now that we have described all the elements, the overall training process goes as follows: Each training sample is separated into two halves along the time, where we condition on the first half $[T_0, T_1]$ in order to predict/reconstruct the second half $[T_1, T_2]$. Given the trajectory sequence $\mathcal{X}$ and weighted adjacency matrix sequence $\mathcal{A}$, we firstly run the encoder to compute the posterior distribution $q_\phi\left(z_i^0 \mid X, \mathcal{A}\right)$ for each object, based on the first half. Then we sample the latent node initial states z_i^0 from it for all objects, and compute the latent initial states for edges as $z_{i\to j}^0 = f_{\text{edge}}\left(\left[z_i^0 \| z_j^0\right]\right)$. We then run the generative model defined by two coupled ODE functions to compute latent states for predicted nodes and edges in the future. Next, we run the decoder to compute the mean of each decoding distribution as :$\mu_i^t = f_{\text{decN}}(z_i^t),\ \mu_{i\to j}^t = f_{\text{decE}}(z_{i\to j}^t)$, which is treated as the predicted value for edges and nodes. Finally, we jointly train the encoder, generative model and decoder by maximizing the evidence lower bound (ELBO) as shown below, where the first term is the reconstruction loss for nodes and edges, and the second term is the KL divergence. We additionally introduce a hyperparameter λ_{edge} for balancing the reconstruction loss of edges and nodes.

$$\begin{aligned}\text{ELBO}\,(\theta, \phi) &= \mathbb{E}_{Z^0 \sim \prod_{i=1}^N q_\phi\left(z_i^0 \mid \mathcal{X}, \mathcal{A}\right)}\left[\log p_\theta\left(\mathcal{X}, \mathcal{A}\right)\right] - \text{KL}[\prod_{i=1}^N q_\phi\left(z_i^0 \mid \mathcal{X}, \mathcal{A}\right) \| p(Z^0)] \\ &= \left(1 - \lambda_{edge}\right)\mathcal{L}_{node} + \lambda_{edge}\mathcal{L}_{edge} - \text{KL}[\prod_{i=1}^N q_\phi\left(z_i^0 \mid \mathcal{X}, \mathcal{A}\right) \| p(Z^0)]\end{aligned} \tag{3.7}$$

The reconstruction loss is estimated as below where the constant σ is the standard derivation of each prior distribution. The overall pipeline is illustrated in Algo 1.

$$\begin{aligned}\mathcal{L}_{node} &= -\sum_i \sum_t \frac{\left\|\mathbf{x}_i^t - \mu_i^t\right\|^2}{2\sigma^2} \\ \mathcal{L}_{edge} &= -\sum_i \sum_j \sum_t \frac{\left\|\mathbf{w}_{i\to j}^t - \mu_{i\to j}^t\right\|^2}{2\sigma^2}\end{aligned} \tag{3.8}$$

Algorithm 1: Coupled Graph ODE training procedure.

```
Input: Adjacency matrix sequence $\mathcal{A} = \{A^1, A^2, \ldots, A^T\}$;
Node feature sequences $\mathcal{X} = \{X^{(1)}, X^{(2)}, \ldots, X^{(T)}\}$.
Output: Model parameters $\phi$ and $\theta$.
1  while model not converged do
2      for Each training sample do
3          Separate the sequence into observed half $[T_0, T_1]$ and predicted half $[T_1, T_2]$;
4          //For the encoder:
5          Construct the temporal graph as shown in Figure 7.1 Step 1 based on the observed data in
             the first half;
6          Conduct dynamic node representation learning on the temporal graph according to
             Eqn 3.2;
7          Generate sequence representation for each object according to Eqn 5.3, then sample latent
             initial states $z_i^0$ for each object according to Eqn 5.4;
8          Generate latent initial state $z_{i\to j}^0$ for each edge according to Eqn 3.5;
9          //For the generative model:
10         Given initial nodes, edges state, and timestamps to predict $[T_1, T_2]$, solve the coupled
             ODE in Eqn 3.6;
11         //For the decoder:
12         Compute predicted nodes and edges based on the decoding likelihood $p\left(\boldsymbol{x}_i^t \mid \boldsymbol{z}_i^t\right)$ and
             $p\left(w_{i\to j}^t \mid \boldsymbol{z}_{i\to j}^t\right)$ respectively;
13     end
14     Update the parameters $\phi$ and $\theta$ by optimizing ELBO loss in Eq. 3.7;
15 end
```

3.5 Experiments

In this section, we present the evaluation results over our model. We first introduce the dataset we used, followed by our experimental results and analysis.

3.5.1 Experiment Setup

Dataset. We conduct experiments on the COVID-19 data as well as the simulated social network data from [GSG17]. For the COVID-19 dataset, we utilize the daily trendency data [DDG20] from the Johns Hopkins University (JHU) Center for Systems Science and Engineering[1] to train our model for the United States. More specifically, we focus on predicting the state-level daily cumulative deaths. For node features, we choose five out of ten dynamic features provided by JHU , which are: $\#$Confirmed, $\#$Deaths, $\#$Recovered, Mortality-Rate and Testing-Rate. Details about their semantic meaning and preprocessing can be found in Appendix. Additionally, we utilize the population for each state as one static feature, which has been widely used in many existing disease prediction models [Het00, ZWX20, WXW20, YWG20]. We use the mobility data provided by Safegraph[2] to construct the interaction graph. SafeGraph is a company that aggregates anonymized location data from numerous mobile applications [CPK21]. The mobility data captures the movement of people between census block groups (CBGs) and we group them by states to form the daily population flow including in-state flow (See Appendix for details). We additionally use the in-state flow as the sixth node dynamic feature, thus each node has seven features in total. The social network data simulates the opinion migration of individuals in a social network over time [GSG17]. We set the number of nodes as 80 and generate 399 timestamps. The initial positions (opinions) of individuals follow uniform distribution in a 2-d space. We set the noise parameter as 0.2, the sparsity parameter as $e^{-0.4}$.

Data Split and Task. We train our model in a sequence to sequence setting where we split the time of each training sample into two parts $[T_0, T_1]$ and $[T_1, T_2]$. We condition on the first half of observations and reconstruct the second half. To achieve this, we generate training samples by setting three hyperparameters: prediction length, condition length and interval, where prediction length is the size of the second half; condition length is the size of the first half, and interval is the overlap between two

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

[2]https://www.safegraph.com/covid-19-data-consortium

consecutive training samples. We generate different training samples to train our model when predicting at different horizons. For the COVID-19 dataset, we utilize data from April.12.2020 to Nov.30.2020 to train our model and test the performance on data from Dec.01.2020 to Dec.31.2020. For the social network dataset, we utilize data from the first 320 timestamps to make predictions in timestamps 321-399.

Table 3.1: Mean Absolute Percentage Error (MAPE) for Cumulative Deaths

Step Length	Pred Date	UCLA-SuEIR	UT-Mobility	Columbia	IHME	LSTM	NRI	VGRNN	CG-ODE
1-week -ahead	Nov.29-Dec.05	0.03297	0.02707	**0.02001**	-	0.08094	0.07784	0.06807	0.02144
	Dec.07-Dec.12	0.02283	0.03736	0.02455	0.02458	0.08363	0.07448	0.06086	0.02653
	Dec.14-Dec.19	0.01946	0.04178	**0.01443**	-	0.07144	0.06462	0.06102	0.01997
	Dec.21-Dec.26	0.01851	0.05460	0.02595	-	0.04912	0.04616	0.04297	**0.01849**
	Average	0.02344	0.04020	**0.02124**	0.02458	0.07128	0.06578	0.05823	0.02161
2-weeks -ahead	Nov.29-Dec.12	0.11036	0.07119	0.08194	-	0.15922	0.15004	0.13791	**0.04341**
	Dec.07-Dec.19	0.07951	0.05830	0.09248	0.06252	0.14873	0.13782	0.12812	**0.04702**
	Dec.14-Dec.26	0.06356	0.04112	0.05174	-	0.13012	0.11423	0.10712	**0.03709**
	Average	0.08448	0.05687	0.07539	0.06252	0.14602	0.13403	0.12438	**0.04251**
3-weeks -ahead	Nov.29-Dec.19	0.17361	0.13255	0.13721	-	0.11793	0.10752	0.10624	**0.04513**
	Dec.06-Dec.26	0.13116	0.09570	0.14445	0.10671	0.19561	0.18088	0.17322	**0.09832**
	Average	0.15239	0.11413	0.14083	0.10671	0.15677	0.14420	0.13973	**0.07173**

Table 3.2: Mean Absolute Percentage Error (MAPE) for Social Data.

Pred Length	10	20	40
LSTM	0.12419	0.37031	0.69579
NRI	0.28879	0.41980	0.68417
VGRNN	**0.11312**	0.27789	0.56763
CG-ODE	0.12359	**0.26340**	**0.45434**

3.5.2 Baselines

For both datasets, we compare with the following three discrete neural network-based methods.

- **LSTM** [SJ97]: A classic recurrent neural network (RNN) that learns the dynamics of each node independently.
- **NRI** [KFW18a]: A VAE-based relation inference model. The encoder infers the static graph structure among nodes and the GNN-based decoder uses the inferred graph to generate the node features in the future.

- **VGRNN** [HHN19]: A VAE-based graph recurrent neural network that jointly learns the evolution of network topology and node attribute changes.

For the COVID-19 dataset, we additionally considers traditional statistical models which learn the dynamic for each state (node) independently. We choose the following four baselines developed by different institutions and obtain their predictions from the forecast hub[3] which is officially used by the centers for disease control and prevention (CDC)[4].

- **UCLA-SuEIR** [ZWX20]: A SuEIR model which is a variant of the SEIR [Het00] model considering both untested and unreported cases. The model considers reopening and assumes susceptible population will increase after the reopening. Parameters are learned via machine learning algorithms.
- **IHME** [tM20]: A non-linear curve-fitting method with the assumption that current interventions remain unchanged.
- **UT-Mobility** [WTD20]: A non-linear curve-fitting method where mobility data within each state is utilized to quantify the changing impact of social-distancing.
- **Columbia** [WXW20]: A survival-convolution model with piece-wise transmission rates that incorporates incubation period and provides a time-varying effective reproductive number.

3.5.3 Performance Evaluation

We evaluate the performance of our model based on Mean Absolute Percentage Error (MAPE) as shown in Table 3.1 and Table 3.2. For the COVID-19 dataset, as the prediction results for statistical baselines are obtained from their weekly official submissions to CDC, we compare the performance across all models using the same weekly prediction periods. Specifically, Pred Date denotes the targeted prediction period while Step Length denotes the number of days before the prediction is made. For example, for Pred Date of Nov.29 - Dec.05, the prediction is made for Dec.05 by using the data up to Nov.29. Therefore it is a 1-week-ahead prediction.

[3]https://github.com/reichlab/covid19-forecast-hub/tree/master/data-processed

[4]https://www.cdc.gov/coronavirus/2019-ncov/covid-data/forecasting-us.html

We first observe that CG-ODE is able to outperform all baselines in long-term predictions by a big margin while achieves similar short-term prediction performance. In both datasets, there is a wider performance gap between CG-ODE and other baselines (e.g. LSTM) that do not consider the interaction among objects, when predicting longer-range node attributes. This indicates that the interaction graph plays a more important role in facilitating long-term predictions. Similar observation can be found in some dynamic physical systems such as particles connected by springs [HSW20b]: to predict the location for each object in a spring system, usually the object's own velocity can be a good approximation for predicting the location at the next timestamp, while it fails to predict locations in the longer-range without considering its interaction among objects. Secondly, neural network-based baselines fail to produce accurate predictions for the COVID19-dataset, which is expected as they are discrete models and may fail to capture the underlying dynamics for a continuous interacting system. Among three neural network-based models, by comparing LSTM with NRI and VGRNN, where the latter two consider underlying interaction among objects and LSTM only models the trajectory for each state independently, we found that by jointly modeling the evolution of graph and node attributes, models can achieve better prediction results. However, NRI performs bad on the social network dataset. This is because the topology change in the social network dataset is more sharp than that of the COVID-19 dataset, and the static network topology assumption in NRI would no longer holds. Among four statistical methods, we observe that UT-Mobility shows better performance in long-term predictions than others. This indicates that the mobility data can serve as a useful signal for predicting the spread of COVID-19. However, UT-Mobility only utilizes the mobility data for each state independently, instead of utilizing it to model the interaction among states as in our model, thus it achieves worse prediction results compared to CG-ODE.

Hyperparameter Study. We then study two important hyperparameters in CG-ODE in the COVID-19 dataset, which are λ_{edge} in Eqn 3.7 for balancing the reconstruction loss for nodes and edges, and the condition length for different prediction horizons. Figure 3.4 shows the MAPE changes as a function of λ_{edge} for three prediction horizons respectively. First, we can see that the optimal λ_{edge} for 1-week-, 2-week- and 3-week-ahead predictions are 0.3, 0.4, 0.5 respectively, which increases when predicting node attributes in the longer range. This is consistent with the prediction errors illustrated in Table 3.1, where the performance gap between our models and other baselines that do not consider graph interaction increases, when predicting longer-range node attributes. They indicate that the interaction graph plays a more important role in facilitating long-term predictions. Second, for all of the three prediction horizons,

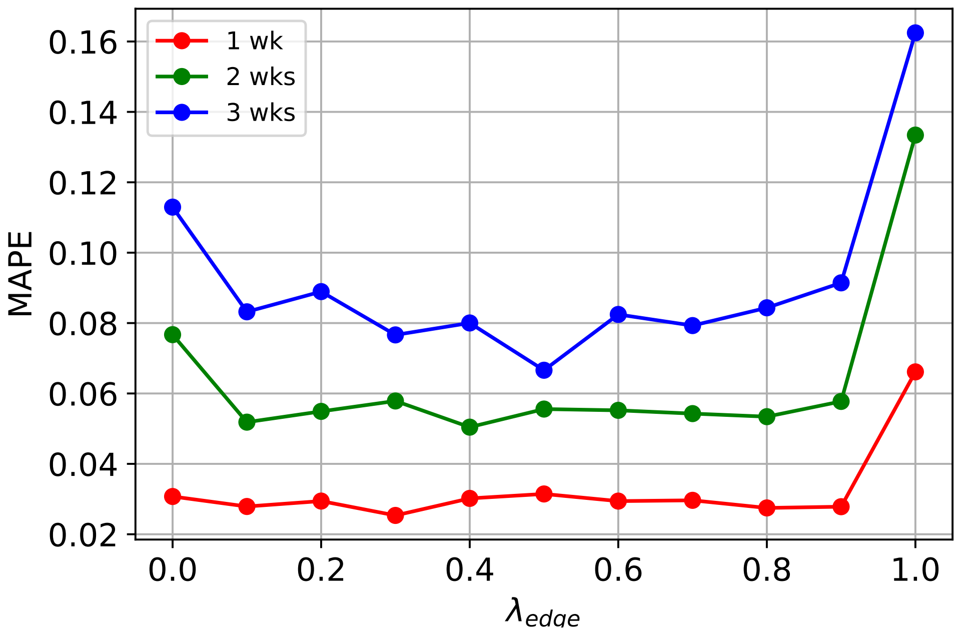


Figure 3.4: MAPE as a function of λ_{edge} on the COVID-19 dataset

when $\lambda_{edge} = 0$, the MAPE increases sharply as the model is only trained to recover the node attributes, without supervision from the dynamic interaction graph. In this case, our model has degenerated to a relation inference model where the ground truth graph is not known during training and is learned in an unsupervised way. Notably, CG-ODE is able to achieve comparable results for the 3-week-ahead predictions compared with statistical baselines even when $\lambda_{edge} = 0$, which verifies the effectiveness of our co-evolution model and the importance of introducing interaction graph for long-term predictions. Last, when $\lambda_{edge} = 1$, our model is only trained for recovering the dynamic interaction graph, and learns the dynamic node attributes in an unsupervised way. Therefore, the MAPE increases as expected. However, the prediction error is still comparable with some statistical baselines especially in the long-term prediction. For example in the 3-week-ahead prediction, UCLA-SuEIR has MAPE of 0.15239 while CG-ODE has MAPE of 0.16245 when $\lambda_{edge} = 1$. This shows the capability of CG-ODE of learning on semi-supervised data or sparse data.

Figure 3.5 shows the MAPE changes as a function of condition length for three different prediction horizons. The optimal condition length for 1-week-, 2-week- and 3-week-ahead predictions are 2 weeks, 3 weeks, 4 weeks respectively, which increases when predicting node attributes in the longer range. This is expected as long-term prediction would usually depend more on the dynamic pattern in the historical

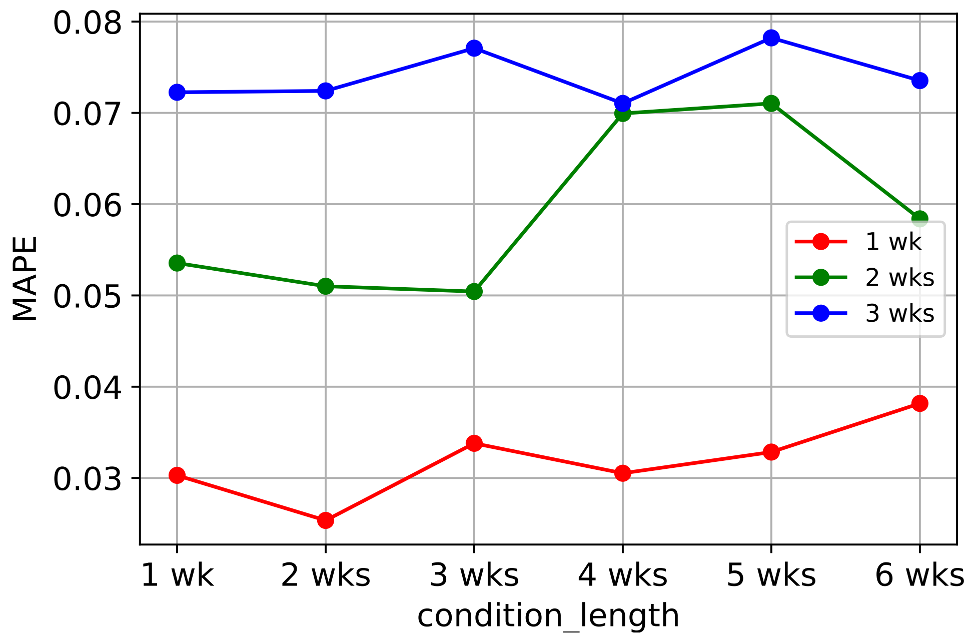


Figure 3.5: MAPE as a function of condition length on the COVID-19 dataset

data, i.e. require longer range dependency from the past. This is similar to a spring system: the location of an object at next timestamp can be well-approximated by its current location and velocity, while the locations in the longer future should depend on its historical trajectories, instead of a single point.

3.5.4 Case Studies

We conduct a case study by adding three different interventions to the interaction graph. Table 3.3 shows the summation of the number of deaths reduced for all states on Dec.26 compared to the ground truth, when adding different interventions. We set the duration of each intervention as 2 weeks and study the effect of adding the same intervention at different times. For example, 1-wk-ahead means the intervention period is one week ago, i.e. the intervention ends at Dec.20 and starts at Dec.7. The first intervention adds 20% reduction to all in-state population flows. The second intervention adds 20% reduction to all between-state population flows. The third one removes the same amount of population flow as in the second intervention, but reduces the population flow in descending order of states' original outflow. Specifically, we rank states by their daily population outflow in descending order and set the outflow for each state to zero starting from the state with the largest population outflow, until the total amount of outflow reduction equals to that of the second intervention.

We firstly observe that reducing the in-state flow will decrease the number of deaths the most. This is expected as the value of in-state population is much larger than that of the between-state population. Secondly, compared with evenly reducing between-state flow for all states, reducing the population outflow from core states will result in a larger drop in the number of deaths. This can be due to the fact that states with larger population outflow are likely to have larger in-state flow as well, due to the loose control over traveling. Thus the severity of these states are likely to be higher. Finally, by comparing the number of reduced deaths for the same intervention happened at different times, we notice that the effect of all interventions tends to decrease day by day.

Table 3.3: Number of Deaths Reduced on Dec.26

	1-wk-ahead	2-wk-ahead	3-wk-ahead
In-state flow deduction (20%)	-8973	- 7084	-6824
Between-state flow deduction (20%)	-2465	-2215	-2197
Flow deducted from core states	-3854	-3625	-3517

3.6 Conclusion

In this paper, we investigate the problem of learning the dynamics of interacting systems by jointly modeling the evolution of nodes and edges. We model system dynamics in a continuous fashion through two coupled neural ordinary differential equations. Specifically, the evolution of a node would depend on its self-evolution and influence received from the interaction graph; the evolution of an edge would depend on its end node's attributes and the edge's self-evolution. We infer the latent initial states for the two ODEs through a novel encoder, which is a VAE-based graph neural network (GNN) that infers the initial states for all objects simultaneously with uncertainty. The proposed model, coupled graph ODE (CG-ODE) is able to achieve accurate prediction for the cumulative deaths of COVID-19 in the United States as well as the simulated social network dataset, especially for long-term predictions. We also conduct an ablation study where we add intervention to the interaction graph and provide insights on how to make efficient intervention policies to control the population flow between the 50 states within the United States. There are some limitations though. Our current model tries to learn latent edge representations by assuming a fully-connected graph, which is time-consuming especially for large dataset. In the future, we plan to design efficient sampling methods for the edge ODEs to balance model efficiency and performance.

CHAPTER 4

TREAT: Physics-Informed Regularization for Domain-Agnostic Dynamical System Modeling

Learning complex physical dynamics purely from data is challenging due to the intrinsic properties of systems to be satisfied. Incorporating physics-informed priors, such as in Hamiltonian Neural Networks (HNNs), achieves high-precision modeling for energy-conservative systems. However, real-world systems often deviate from strict energy conservation and follow different physical priors. To address this, we present a framework that achieves high-precision modeling for a wide range of dynamical systems from the numerical aspect, by enforcing *Time-Reversal Symmetry* (TRS) via a novel regularization term. It helps preserve energies for conservative systems while serving as a strong inductive bias for non-conservative, reversible systems. While TRS is a domain-specific physical prior, we present the *first* theoretical proof that TRS loss can universally improve modeling accuracy by minimizing higher-order Taylor terms in ODE integration, which is numerically beneficial to various systems regardless of their properties, even for irreversible systems. By integrating the TRS loss within neural ordinary differential equation models, the proposed model TREAT demonstrates superior performance on diverse physical systems. It achieves a significant 11.5% MSE improvement in a challenging chaotic triple-pendulum scenario, underscoring TREAT's broad applicability and effectiveness. Code and further details are available at here.

4.1 Introduction

Dynamical systems, spanning applications from physical simulations [KFW18a, WKM20, LLC22] to robotic control [LXM22, NQ22], are challenging to model due to intricate dynamic patterns and potential interactions under multi-agent settings. Traditional numerical simulators require extensive domain knowledge for design, which is sometimes unknown [SGP20a], and can consume significant computational resources. Therefore, directly learning dynamics from the observational data becomes an attractive alternative.

Existing deep learning approaches [SGP20a, PFS21] usually learn a fixed-step transition function to predict system dynamics from timestamp t to timestamp $t + 1$ and rollout trajectories recursively. The transition function can have different inductive biases, such as Graph Neural Networks (GNNs) [LSW23] for capturing pair-wise interactions among agents through message passing. Most recently, neural ordinary differential equations (ODEs) [CRB18b, RCD19a] have emerged as a potent solution for modeling system dynamics in a continuous manner, which offer superior prediction accuracy over discrete models in the long-range, and can handle systems with partial observations. In particular, GraphODEs [HSW20a, LYH23, ZW20, JHL23a] extend NeuralODEs to model interacting (multi-agent) dynamical systems, where agents co-evolve and form trajectories jointly.

However, the complexity of dynamical systems necessitates large amounts of data. Models trained on limited data risk violating fundamental physical principles such as energy conservation. A promising strategy to improve modeling accuracy involves incorporating physical inductive biases [RPK19, CGH20]. Existing models like Hamiltonian Neural Networks (HNNs) [GDY19, SBC19] strictly enforce energy conservation, yielding more accurate predictions for energy-conservative systems. However, not all real-world systems strictly adhere to energy conservation, and they may adhere to various physical priors. Such system diversity largely limits the usage of existing models which are designed for individual physical prior.

To address this, we present a framework that achieves high-precision modeling for a wide range of dynamical systems from the numerical aspect, by enforcing Time-Reversal Symmetry (TRS) via a novel regularization term. Specifically, TRS posits that a system's dynamics should remain invariant when time is reversed [LR98]. To incorporate TRS, we propose a simple-yet-effective self-supervised regularization term that acts as a soft constraint. This term aligns *forward and backward trajectories* predicted by a neural network and we use GraphODE as the backbone. We theoretically prove that the TRS loss effectively minimizes higher-order Taylor expansion terms during ODE integration, offering a general numerical advantage for improving modeling accuracy across a wide array of systems, regardless of their physical properties. It forces the model to capture fine-grained physical properties such as jerk (the derivatives of accelerations) and provides more regularization for long-term prediction. We also justify our TRS design choice, showing case its superior performance both analytically and empirically. We name the model as TREAT (**T**ime-**Re**vers**a**l Symme**t**ry ODE).

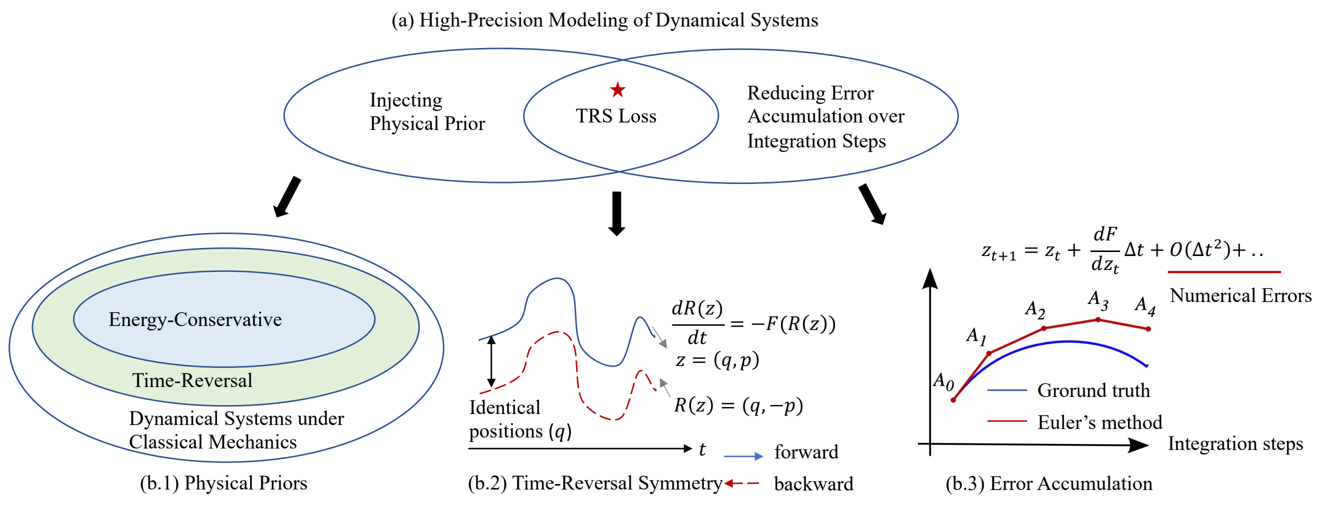

(a) High-Precision Modeling of Dynamical Systems
Injecting Physical Prior
TRS Loss
Reducing Error Accumulation over Integration Steps
Energy-Conservative
Time-Reversal
Dynamical Systems under Classical Mechanics
(b.1) Physical Priors
$\frac{dR(z)}{dt} = -F(R(z))$
$z = (q, p)$
$R(z) = (q, -p)$
Identical positions (q)
t
forward
backward
(b.2) Time-Reversal Symmetry
$z_{t+1} = z_t + \frac{dF}{dz_t}\Delta t + O(\Delta t^2) + ..$
Numerical Errors
A_0
A_1
A_2
A_3
A_4
Grorund truth
Euler's method
Integration steps
(b.3) Error Accumulation

Figure 4.1: (a) High-precision modeling for dynamical systems; (b.1) Classification of classical mechanical systems based on [Tol38, LR98];(b.2) Tim-Reversal Symmetry illustration;(b.3) Error accumulation in numerical solvers.

Note that TRS itself is a physical prior, that is broader than energy conservation as depicted in Figure 4.1(b.1). It covers classical energy-conservative systems such as Newtonian mechanics, and also non-conservative, reversible systems like Stokes flow [Poz01], commonly encountered in microfluidics [KK13, CL18, CGL19]. Therefore, TRS loss achieves high-precision modeling from both the physical aspect, and the numerical aspect as shown in Figure 4.1(a), making it domain-agnostic and widely applicable to various dynamical systems. We systematically conduct experiments across 9 diverse datasets spanning across 1.) single-agent, multi-agent systems; 2.) simulated and real-world systems; and 3.) systems with different physical priors. TREAT consistently outperforms state-of-the-art baselines, affirming its effectiveness and versatility across various dynamic scenarios.

Our primary contributions can be summarized as follows:

- We introduce TREAT, a powerful framework that achieves high-precision modeling for a wide range of systems from the numerical aspect, by enforcing Time-Reversal Symmetry (TRS) via a regularization term.

- We establish the *first* theoretical proof that the time-reversal symmetry loss could in general help learn more fine-grained and long-context system dynamics from the numerical aspect, regardless of systems' physical properties (even irreversible systems). This bridges the specific physical implication and the general numerical benefits of the physical prior -TRS.

- We present empirical evidence of TREAT's state-of-the-art performance in a variety of systems over 9 datasets, including real-world & simulated systems, etc. It yields a significant MSE improvement of 11.5% on the challenging chaotic triple-pendulum system.

4.2 Preliminaries and Related Work

We represent a dynamical system as a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the node set of N agents[1] and $\mathcal{E}$ denotes the set of edges representing their physical interactions. For simplicity, we assumed $\mathcal{G}$ to be static over time. Single-agent dynamical system is a special case where the graph only has one node. In the following, we use the multi-agent setting by default to illustrate our model. We denote $\boldsymbol{X}(t) \in \mathbb{R}^{N \times d}$ as the feature matrix at timestamp t for all agents, with d as the feature dimension. Model input consists of trajectories of feature matrices over M historical timestamps $X(t_{-M:-1}) = \{\boldsymbol{X}(t_{-M}), \ldots, \boldsymbol{X}(t_{-1})\}$ and $\mathcal{G}$. The timestamps $t_{-1}, \cdots, t_{-M} < 0$ can have non-uniform intervals and take any continuous values. Our goal is to learn a neural simulator $f_\theta(\cdot) : \left[X(t_{-M:-1}), \mathcal{G}\right] \rightarrow Y(t_{0:K})$, which predicts node dynamics $\boldsymbol{Y}(t)$ in the future on timestamps $0 = t_0 < \cdots < t_K = T$ sampled within $[0, T]$. We use $\boldsymbol{y}_i(t)$ to denote the targeted dynamic vector of agent i at time t. In some cases when we are only predicting system feature trajectories, $\boldsymbol{Y}(\cdot) \equiv \boldsymbol{X}(\cdot)$.

4.2.1 NeuralODE for Dynamical Systems

NeuralODEs [CRB18b, RCD19a] are a family of continuous models that define the evolution of dynamical systems by ordinary differential equations (ODEs). The state evolution can be described as: $\dot{\boldsymbol{z}}_i(t) := \frac{d\boldsymbol{z}_i(t)}{dt} = g\left(\boldsymbol{z}_1(t), \boldsymbol{z}_2(t) \cdots \boldsymbol{z}_N(t)\right)$, where $\boldsymbol{z}_i(t) \in \mathbb{R}^d$ denotes the latent state variable for agent i at timestamp t. The ODE function g is parameterized by a neural network such as Multi-Layer Perception (MLP), which is automatically learned from data. GraphODEs [PMP19a, HSW20a, LYH23, WWM22] are special cases of NeuralODEs, where g is a Graph Neural Network (GNN) to capture the continuous interaction among agents.

GraphODEs have been shown to achieve superior performance, especially in long-range predictions and can handle data irregularity issues. They usually follow the encoder-processor-decoder architecture,

[1]Following [KFW18a], we use "agents" to denote "objects" in dynamical systems, which is different from "intelligent agent" in AI.

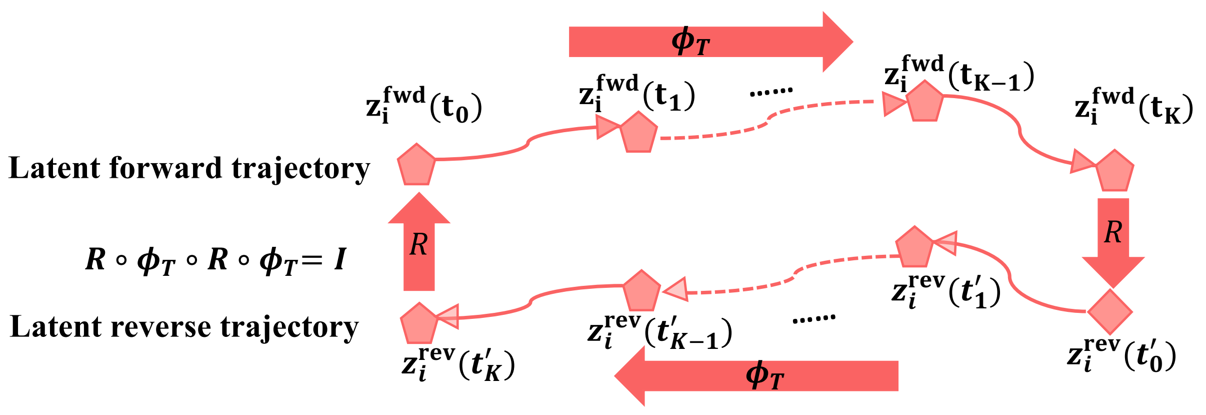


Figure 4.2: Illustration of time-reversal symmetry based on Lemma 1.The total length of the trajectory is $t_K - t_0 = T$. t'_k is the time index in the reverse trajectory, which points to the same time as t_{K-k} in the forward trajectory.

where an encoder first computes the latent initial states $\boldsymbol{z}_1(t_0), \cdots \boldsymbol{z}_N(t_0)$ for all agents simultaneously based on their historical observations as in Eqn 4.1.

$$\boldsymbol{z}_1(t_0), \boldsymbol{z}_2(t_0), ..., \boldsymbol{z}_N(t_0) = f_{\text{ENC}}\big(X(t_{-M:-1}), \mathcal{G}\big) \tag{4.1}$$

Then the GNN-based ODE predicts the latent trajectories starting from the learned initial states. The latent state $\boldsymbol{z}_i(t)$ can be computed at any desired time using a numerical solver such as Runge-Kuttais [MSH19] as:

$$\boldsymbol{z}_i(t) = \text{ODE-Solver}\big(g, [\boldsymbol{z}_1(t_0), ...\boldsymbol{z}_N(t_0)], t\big) = \boldsymbol{z}_i(t_0) + \int_{t_0}^{t} g\left(\boldsymbol{z}_1(t), \boldsymbol{z}_2(t) \cdots \boldsymbol{z}_N(t)\right) dt. \tag{4.2}$$

Finally, a decoder extracts the predicted dynamics $\widehat{\boldsymbol{y}}_i(t)$ based on the latent states $\boldsymbol{z}_i(t)$ for any timestamp t:

$$\widehat{\boldsymbol{y}}_i(t) = f_{\text{DEC}}(\boldsymbol{z}_i(t)). \tag{4.3}$$

However, vanilla GraphODEs can violate physical properties of a system, resulting in unrealistic predictions. We therefore propose to inject physics-informed regularization term to make more accurate predictions.

4.2.2 Time-Reversal Symmetry (TRS)

Consider a dynamical system described in the form of $\frac{d\boldsymbol{x}(t)}{dt} = F(\boldsymbol{x}(t))$, where $\boldsymbol{x}(t) \in \Omega$ is the observed states such as positions. The system is said to follow the *Time-Reversal Symmetry* if there exists a reversing operator $R : \Omega \mapsto \Omega$ such that [LR98]:

$$\frac{d\big(R \circ \boldsymbol{x}(t)\big)}{dt} = -F\big(R \circ \boldsymbol{x}(t)\big), \tag{4.4}$$

where $\circ$ denote the action of functional R on the function $\boldsymbol{x}$.

Intuitively, we can assume $\boldsymbol{x}(t)$ is the position of a flying ball and the conventional reversing operator is defined as $R : \boldsymbol{x} \mapsto R \circ \boldsymbol{x}, R \circ \boldsymbol{x}(t) = \boldsymbol{x}(-t)$. This implies when $\boldsymbol{x}(t)$ is a forward trajectory position with initial position $\boldsymbol{x}(0)$, $\boldsymbol{x}(-t)$ is then a position in the time-reversal trajectory, where $\boldsymbol{x}(-t)$ is calculated using the same function F, but with the integration time reversed, i.e. $dt \mapsto d(-t)$. Eqn 4.4 shows how to create the reverse trajectory of a flying ball: at each position, the velocity (i.e., the derivative of position with respect to time) should be the opposite. In neural networks, we usually model trajectories in the latent space via $\boldsymbol{z}$ [SGP20a], which can be decoded back to real observation state i.e. positions. Therefore, we apply the reversal operator for $\boldsymbol{z}$.

Now we introduce a time evolution operator ϕ_τ such that $\phi_\tau \circ \boldsymbol{z}(t) = \boldsymbol{z}(t + \tau)$ for arbitrary $t, \tau \in \mathbb{R}$. It satisfies $\phi_{\tau_1} \circ \phi_{\tau_2} = \phi_{\tau_1 + \tau_2}$, where $\circ$ denotes composition. The time evolution operator helps us to move forward (when $\tau > 0$) or backward (when $\tau < 0$) through time, thus forming a trajectory. Based on [LR98], in terms of the evolution operator, Eqn 4.4 implies:

$$R \circ \phi_t = \phi_{-t} \circ R = \phi_t^{-1} \circ R, \tag{4.5}$$

which means that moving forward t steps and then turning to the opposite direction is equivalent to firstly turning to the opposite direction and then moving backwards t steps[2]. Eqn 4.5 has been widely used to describe time-reversal symmetry in existing literature [HYH20, VWT22]. Nevertheless, we propose the following lemma, which is more intuitive to understand and straightforward to guide the design of our time-reversal regularizer.

Lemma 1. Eqn 4.5 is equivalent to $R \circ \phi_t \circ R \circ \phi_t = I$, where I denotes identity mapping.

Lemma 1 means if we move t steps forward, then turn to the opposite direction, and then move forward for t more steps, it shall restore back to the same state. This is illustrated in Figure 4.2 where

[2]Time-reversal symmetry is a property of physical systems, which requires the forward and reverse trajectories to be generated by the same mechanism $F(\cdot)$. It differs from reversibility of neural networks [CMH18, LKB19], which is a property of machine learning models and ensures the recovery of input from output via a reversed operator $f^{-1}(\cdot)$. We highlight the detailed discussions in Appendix.

the reverse trajectory should be the same as the forward trajectory.[3] It can be understood as rewinding a video to the very beginning. The proof of Lemma 1 is in Appendix.

4.3 Method: TREAT

We present a novel framework TREAT that achieves high-precision modeling for a wide range of systems from the numerical aspect, by enforcing Time-Reversal Symmetry (TRS) via a regularization term. It improves modeling accuracy regardless of systems' physical properties. We first introduce our architecture design, followed by theoretical analysis to explain its numerical benefits.

TREAT uses GraphODE [HSW20a] as the backbone and flexibly incorporates TRS as a regularization term based on Lemma 1. This term aligns model forward and reverse trajectories. In practice, our model predicts the forward trajectories at a series of timestamps $\{t_k\}_{k=0}^{K}$ as ground truth observations are discrete, where $0 = t_0 < t_1 < \cdots < t_K = T$. The reverse trajectories are also at the same series of K timestamps so as to be aligned with the forward one, which we denote as $\{t'_k\}_{k=0}^{K}$ satisfying $0 = t'_0 < t'_1 < \cdots < t'_K = T$. It's important to note that the values of the time variable t'_k in the reverse trajectories do not represent real time, but serve as indexes of reverse trajectories. This leads to the relation $t'_{K-k} = T - t_k$, which means the reverse trajectories at timestamp t'_{K-k} correspond to the forward trajectories at time t_k. For example, $t'_0 = T - t_K = 0$. It indicates t'_0 and t_K are both pointing to the same real time T, which is the ending point of the forward trajectory as shown in Figure 4.3. Based on Lemma 1, the difference of the two trajectories at any observed time should be small, i.e. $\boldsymbol{z}^{\text{fwd}}(t_k) \approx \boldsymbol{z}^{\text{rev}}(t'_{K-k})$. This serves as the guideline for our regularizer design. The weight of the regularizer is also adjustable to adapt different systems. The overall framework is depicted in Figure 4.3.

4.3.1 Time-Reversal Symmetry Loss and Training

Forward Trajectory Prediction and Reconstruction Loss. For multi-agent systems, we utilize the GNN operator described in [KFW18a] as our ODE function $g(\cdot)$, which drives the system to move forward and output the forward trajectories for latent states $\boldsymbol{z}_i^{\text{fwd}}(t)$ at each continuous time $t \in [0, T]$ and each agent i.We then employ a Multilayer Perceptron (MLP) as a decoder to predict output trajectories

[3]We explain Figure 4.2 with implementation in Appendix.

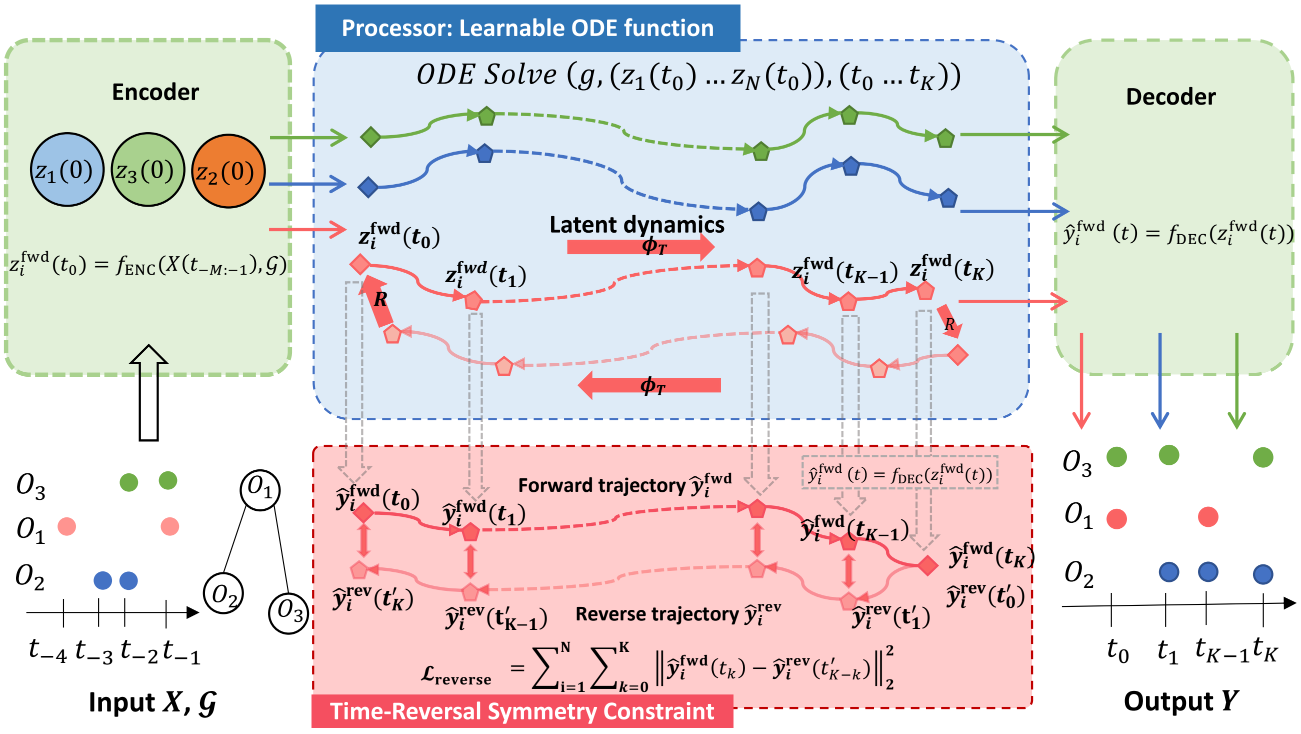

Processor: Learnable ODE function
$ODE\ Solve\ (g, (z_1(t_0) \dots z_N(t_0)), (t_0 \dots t_K))$
Encoder
$z_1(0)$
$z_3(0)$
$z_2(0)$
$z_i^{\text{fwd}}(t_0) = f_{\text{ENC}}(X(t_{-M:-1}), \mathcal{G})$
Decoder
$\hat{y}_i^{\text{fwd}}(t) = f_{\text{DEC}}(z_i^{\text{fwd}}(t))$
Latent dynamics
ϕ_T
$z_i^{\text{fwd}}(t_0)$
$z_i^{\text{fwd}}(t_1)$
$z_i^{\text{fwd}}(t_{K-1})$
$z_i^{\text{fwd}}(t_K)$
R
Forward trajectory $\hat{y}_i^{\text{fwd}}$
Reverse trajectory $\hat{y}_i^{\text{rev}}$
$\hat{y}_i^{\text{fwd}}(t_0)$
$\hat{y}_i^{\text{fwd}}(t_1)$
$\hat{y}_i^{\text{fwd}}(t_{K-1})$
$\hat{y}_i^{\text{fwd}}(t_K)$
$\hat{y}_i^{\text{rev}}(t'_K)$
$\hat{y}_i^{\text{rev}}(t'_{K-1})$
$\hat{y}_i^{\text{rev}}(t'_1)$
$\hat{y}_i^{\text{rev}}(t'_0)$
$\mathcal{L}_{\text{reverse}} = \sum_{i=1}^{N} \sum_{k=0}^{K} \left\| \hat{y}_i^{\text{fwd}}(t_k) - \hat{y}_i^{\text{rev}}(t'_{K-k}) \right\|_2^2$
Time-Reversal Symmetry Constraint
O_3
O_1
O_2
$t_{-4}\ t_{-3}\ t_{-2}\ t_{-1}$
Input $X, \mathcal{G}$
$t_0\ \ t_1\ \ t_{K-1}\ t_K$
Output Y

Figure 4.3: Overall framework of TREAT. O_1, O_2, O_3 are connected agents. It follows the encoder-processor-decoder architecture introduced in Sec 4.2.1. A novel TRS loss is incorporated to improve modeling accuracy across systems from the numerical aspect, regardless of their physical properties.

$\widehat{\boldsymbol{y}}_i^{\text{fwd}}(t)$ based on the latent states. We summarize the whole procedure as:

$$\begin{aligned} \dot{\boldsymbol{z}}_i^{\text{fwd}}(t) &:= \frac{d\boldsymbol{z}_i^{\text{fwd}}(t)}{dt} = g(\boldsymbol{z}_1^{\text{fwd}}(t), \boldsymbol{z}_2^{\text{fwd}}(t), \cdots \boldsymbol{z}_N^{\text{fwd}}(t)), \\ \boldsymbol{z}_i^{\text{fwd}}(t_0) &= f_{\text{ENC}}(X(t_{-M:-1}), \mathcal{G}), \quad \widehat{\boldsymbol{y}}_i^{\text{fwd}}(t) = f_{\text{DEC}}(\boldsymbol{z}_i^{\text{fwd}}(t)). \end{aligned} \tag{4.6}$$

To train the model, we use the reconstruction loss that minimizes the L2 distance between predicted forward trajectories $\{\widehat{\boldsymbol{y}}_i^{\text{fwd}}(t_k)\}_{k=0}^K$ and the ground truth trajectories $\{\boldsymbol{y}_i(t_k)\}_{k=0}^K$ as :

$$\mathcal{L}_{pred} = \sum_{i=1}^N \sum_{k=0}^K ||\boldsymbol{y}_i(t_k) - \widehat{\boldsymbol{y}}_i^{\text{fwd}}(t_k)||_2^2. \tag{4.7}$$

Reverse Trajectory Prediction and Regularization Loss. We design a novel time-reversal symmetry loss as a soft constraint to flexibly regulate systems' behavior based on Lemma 1. Specifically, we first compute the latent reverse trajectories $\boldsymbol{z}^{\text{rev}}(t)$ by starting from the ending state of the forward one, traversed back over time. We then employ the decoder to output dynamic trajectories $\boldsymbol{y}^{\text{rev}}(t)$.

$$\begin{aligned} \dot{\boldsymbol{z}}_i^{\text{rev}}(t) &:= \frac{d\boldsymbol{z}_i^{\text{rev}}(t)}{dt} = -g(\boldsymbol{z}_1^{\text{rev}}(t), \boldsymbol{z}_2^{\text{rev}}(t), \cdots \boldsymbol{z}_N^{\text{rev}}(t)), \\ \boldsymbol{z}_i^{\text{rev}}(t'_0) &= \boldsymbol{z}_i^{\text{fwd}}(t_K), \quad \widehat{\boldsymbol{y}}_i^{\text{rev}}(t) = f_{\text{DEC}}(\boldsymbol{z}_i^{\text{rev}}(t)). \end{aligned} \tag{4.8}$$

Next, based on Lemma 1, if the system follows *Time-Reversal Symmetry*, the forward and backward trajectories shall be exactly overlap. We thus design the reversal loss by minimizing the L2 distances between model forward and backward trajectories decoded from the latent trajectories:

$$\mathcal{L}_{reverse} = \sum_{i=1}^N \sum_{k=0}^K ||\widehat{\boldsymbol{y}}_i^{\text{fwd}}(t_k) - \widehat{\boldsymbol{y}}_i^{\text{rev}}(t'_{K-k})||_2^2. \tag{4.9}$$

Finally, we jointly train TREAT as a weighted combination of the two losses:

$$\mathcal{L} = \mathcal{L}_{pred} + \alpha\mathcal{L}_{reverse} = \sum_{i=1}^N \sum_{k=0}^K ||\boldsymbol{y}_i(t_k) - \widehat{\boldsymbol{y}}_i^{\text{fwd}}(t_k)||_2^2 + \alpha \sum_{i=1}^N \sum_{k=0}^K ||\widehat{\boldsymbol{y}}_i^{\text{fwd}}(t_k) - \widehat{\boldsymbol{y}}_i^{\text{rev}}(t'_{K-k})||_2^2, \tag{4.10}$$

where α is a positive coefficient to balance the two losses based on different targeted systems.

Remark. The computational time of $\mathcal{L}_{reverse}$ is of the same scale as the reconstruction loss $\mathcal{L}_{pred}$. As the computation process of the reversal loss is to first use the ODE solver to generate the reverse trajectories, which has the same computational overhead as computing the forward trajectories, and then compute the L2 distances.

4.3.2 Theoretical Analysis of Time-Reversal Symmetry Loss

We next theoretically show that the time-reversal symmetry loss numerically helps to improve prediction accuracy in general, regardless of systems' physical properties. Specifically, we show that it minimizes higher-order Taylor expansion terms during the ODE integration steps.

Theorem 1. Let Δt denote the integration step size in an ODE solver and T be the prediction length. The reconstruction loss $\mathcal{L}_{pred}$ defined in Eqn 4.7 is $\mathcal{O}(T^3 \Delta t^2)$. The time-reversal loss $\mathcal{L}_{reverse}$ defined in Eqn 4.9 is $\mathcal{O}(T^5 \Delta t^4)$.

We prove Theorem 1 in Appendix. From Theorem 1, we can see two nice properties of our proposed time-reversal loss: 1) Regarding the relationship to Δt, $\mathcal{L}_{reverse}$ is optimizing a high-order term Δt^4, which forces the model to predict fine-grained physical properties such as jerk (the derivatives of accelerations). In comparison, the reconstruction loss optimizes Δt^2, which mainly guides the model to predict the locations/velocities accurately. Therefore, the combined loss enables our model to be more noise-tolerable; 2) Regarding the relationship to T, $\mathcal{L}_{reverse}$ is more sensitive to total sequence length (T^5), thus it provides more regularization for long-context prediction, a key challenge for dynamic modeling.

TRS Loss Design Choice. We define $\mathcal{L}_{reverse}$ as the distance between model forward trajectories and backward trajectories. Based on the definition of TRS, there are other implementation choices. One prior work TRS-ODE [HYH20] designed a TRS loss based on Eqn 4.5, where a reverse trajectory shares the same starting point as the forward one. However, we show that our implementation based on Lemma 1 to approximate time-reversal symmetry has a lower maximum error compared to their implementation below, supported by empirical experiments.

Lemma 2. Let $\mathcal{L}_{reverse}$ be the TRS implementation of TREAT based on Lemma 1, $\mathcal{L}_{reverse2}$ be the one in [HYH20] based on Eqn 4.5. When the reconstruction loss defined in Eqn 4.7 of both methods are equal, and the two TRS losses are equal, i.e. $\mathcal{L}_{reverse} = \mathcal{L}_{reverse2}$, the maximum error between the reversal and ground truth trajectory for each agent, i.e. $MaxError_{gt\_rev} = \max_{k \in [K]} \|\boldsymbol{y}_i(t_k) - \widehat{\boldsymbol{y}}_i^{\text{rev}}(t'_{K-k})\|_2$ for $i = 1, 2 \cdots N$, made by TREAT is smaller.

We prove Lemma 2 in Appendix. Another implementation is to minimize the distances between model backward trajectories and ground truth trajectories. When both forward and backward trajectories are close to ground-truth, they are implicitly symmetric. The major drawback is that at the early stage of learning when the forward is far away from ground truth ($\mathcal{L}_{pred}$), such implicit regularization does not force time-reversal symmetry, but introduces more noise.

4.4 Experiments

Datasets. We conduct systematic evaluations over five multi-agent systems including three 5-body spring systems [KFW18a], a complex chaotic pendulum system and a real-world motion capture dataset [CMU03]; and four single-agent systems including three spring systems (with only one node) and a chaotic strange attractors system [HYH20].

The settings of spring systems include: 1) conservative, i.e. no interactions with the environments, we call it *Simple Spring*; 2) non-conservative with frictions, we call it *Damped Spring*; 3) non-conservative with periodic external forces, we call it *Forced Spring*. The *Pendulum* system contains three connected sticks in a 2D plane. It is highly sensitive to initial states, with minor disturbances leading to significantly different trajectories [SGW92, AKW08]. The real-world motion capture dataset [CMU03] describes the walking trajectories of a person, each tracking a single joint. We call it *Human Motion*. The strange attractor consists of symmetric attractor/repellor force pairs and is chaotic [SC15]. It is also highly sensitive to the initial states [KTK19]. We call it *Attractor*.

Towards physical properties, *Simple Spring* and *Pendulum* are conservative and reversible; *Force Spring* and *Attractor* are reversible but non-conservative; *Damped Spring* are irreversible and non-conservative. For *Human Motion*, it does not adhere to specific physical laws since it is a real-world dataset. Details of the datasets and generation pipelines can be found in Appendix.

Task Setup. We conduct evaluation by splitting trajectories into two halves: $[t_1, t_M]$, $[t_{M+1}, t_K]$ where timestamps can be irregular. We condition the first half of observations to make predictions for the second half as in [RCD19a]. For spring datasets and *Pendulum*, we generate irregular-sampled trajectories and set the training samples to be 20,000 and testing samples to be 5,000 respectively. For *Attractor*, We generate 1,000 and 50 trajectories for training and testing respectively following [HYH20]. 10% of training samples are used as validation sets and the maximum trajectory prediction length is 60. Details can be found in Appendix.

Baselines. We compare TREAT against three baseline types: 1) pure data-driven approaches including LG-ODE [HSW20a] and LatentODE [RCD19a], where the first one is a multi-agent approach considering pair-wise interactions, and the second one is a single-agent approach that predicts each trajectory independently; 2) energy-preserving HODEN [GDY19]; and 3) time-reversal TRS-ODEN [HYH20].

The latter two are single-agent approaches and require initial states as given input. To handle missing initial states in our dataset, we approximate the initial states for the two methods via linear spline interpolation [End03]. In addition, we substitute the ODE network in TRS-ODEN with a GNN [KFW18a] as TRS-ODEN$_{\text{GNN}}$, which serves as a new multi-agent approach for fair comparison. HODEN cannot be easily extended to the multi-agent setting as replacing the ODE function with a GNN can violate energy conservation of the original HODEN. For running LGODE and TREAT on single-agent datasets, we only include self-loop edges in the graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, which makes the ODE function g a simple MLP. Implementation details can be found in Appendix.

Table 4.1: Evaluation results on MSE (10^{-2}). Best results are in **bold** numbers and second-best results are in <u>underline</u> numbers. *Human Motion* is a real-world dataset and all others are simulated datasets.

	Multi-Agent Systems					**Single-Agent Systems**			
Dataset	*Simple Spring*	*Forced Spring*	*Damped Spring*	*Pendulum*	*Human Motion*	*Simple Spring*	*Forced Spring*	*Damped Spring*	*Attractor*
LatentODE	5.2622	5.0277	3.3419	2.6894	2.9061	5.7957	0.4563	1.3012	<u>0.58394</u>
HODEN	3.0039	4.0668	8.7950	741.2296	1.9855	3.2119	4.004	1.5675	54.2912
TRS-ODEN	3.6785	4.4465	1.7595	741.4988	0.5400	3.0271	0.4056	1.5667	2.2683
TRS-ODEN$_{\text{GNN}}$	<u>1.4115</u>	2.1102	<u>0.5951</u>	596.0319	<u>0.2609</u>	/	/	/	/
LG-ODE	1.7429	<u>1.8929</u>	0.9718	<u>1.4156</u>	0.7610	<u>1.6156</u>	<u>0.1465</u>	<u>1.1223</u>	0.6942
TREAT	**1.1178**	**1.4525**	**0.5944**	**1.2527**	**0.2192**	**1.6026**	**0.0960**	**1.0750**	**0.5581**
(—-Ablation of our method with different implementation of $L_{reverse}$—-)									
TREAT$_{\mathcal{L}_{rev}=\text{gt-rev}}$	1.1313	1.5254	0.6171	1.6158	0.2495	1.6190	0.1104	1.1205	0.6364
TREAT$_{\mathcal{L}_{rev}=\text{rev2}}$	1.6786	1.9786	0.9692	1.5631	0.8785	1.6901	0.0983	1.0952	0.7286

4.4.1 Main Results

Table 4.1 shows the prediction performance on both multi-agent systems and single-agent systems measured by mean squared error (MSE). We can see that TREAT consistently surpasses other models, highlighting its generalizability and the efficacy of the proposed TRS loss.

For multi-agent systems, approaches that consider interactions among agents (LG-ODE, TRS-

ODEN$_{\text{GNN}}$, TREAT) consistently outperform single-agent baselines (LatentODE, HODEN, TRS-ODEN), and TREAT achieves the best performance across datasets.

The chaotic nature of the *Pendulum* system and the *Attractor* system, with their sensitivity to initial states [4], poses extreme challenges for dynamic modeling. This leads to highly unstable predictions for models like HODEN and TRS-ODEN, as they estimate initial states via inaccurate linear spline interpolation [End03]. In contrast, LatentODE, LG-ODE, and TREAT employ advanced encoders that infer latent states from observed data and demonstrate superior accuracy. Among them, TREAT achieves the most accurate predictions, further showing its robust generalization capabilities.

We observe that misapplied inductive biases can degrade results, which limits the usage of physics-informed methods that are designed for individual physical prior such as HODEN. HODEN only excels on energy-conservative systems, such as *Simple Spring* compared with LatentODE and TRS-ODEN in the multi-agent setting. Its performance drop dramatically on *Force Spring*, *Damped Spring*, and *Attractor*. Note that HODEN naively forces each agent to be energy-conservative, instead of the whole system. Therefore, it performs poorly than LG-ODE, TREAT in the multi-agent settings.

For the *Human Motion* dataset, characterized by its dynamic ambiguity as it does not adhere to specific physical laws, we cannot directly determine whether it is conservative or time-reversal. For such a system with an unknown nature, TREAT outperforms other purely data-driven methods significantly, showcasing its strong numerical benefits in improving prediction accuracy across diverse system types. This is also shown by its superior performance on *Damped Spring*, which is irreversible.

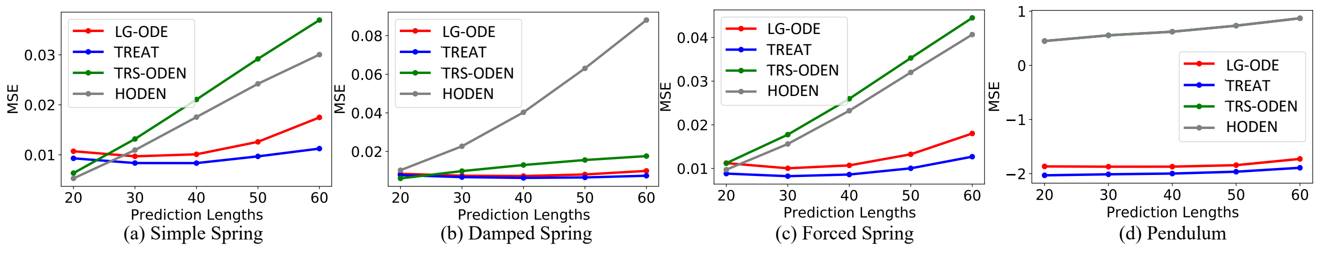


Figure 4.4: Varying prediction lengths across multi-agent datasets (Pendulum MSE is in log values).

[4]Video to show *Pendulum* is highly sensitive to initial states.

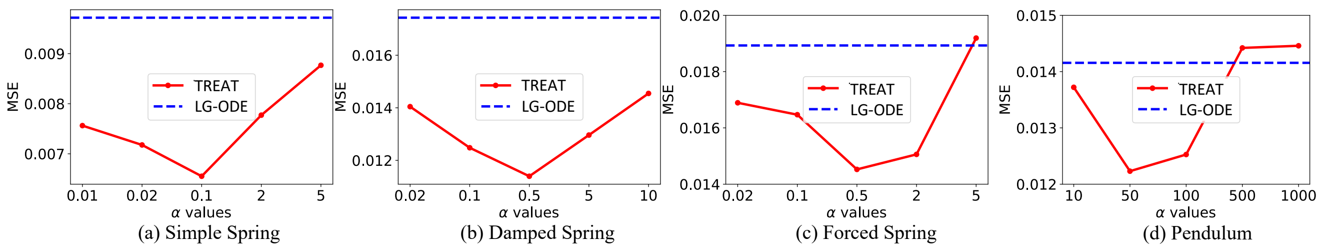


Figure 4.5: Varying α values across multi-agent datasets.

4.4.2 Ablation and Sensitivity Analysis

Ablation on implementation of $\mathcal{L}_{reverse}$. We conduct two ablation by changing the implementation of $\mathcal{L}_{reverse}$ discussed in Sec. 4.3.2: 1) TREAT$_{\mathcal{L}_{rev}=\text{gt-rev}}$, which computes the reversal loss as the L2 distance between ground truth trajectories to model backward trajectories; 2) TREAT$_{\mathcal{L}_{rev}=\text{rev2}}$, which implements the TRS loss based on Eqn 4.5 as in TRS-ODEN [HYH20]. From the last block of Table 4.1, we can clearly see that our implementation achieves the best performance against the two.

Evaluation across prediction lengths. We vary the maximum prediction lengths from 20 to 60 and report model performance as shown in Figure 4.4. As the prediction step increases, TREAT consistently maintains optimal prediction performance, while other baselines exhibit significant error accumulations. The performance gap between TREAT and baselines widens when making long-range predictions, highlighting the superior predictive capability of TREAT.

Evaluation across different α. We vary the values of the coefficient α defined in Eqn 4.10, which balances the reconstruction loss and the TRS loss. Figure 4.5 demonstrates that the optimal α values being neither too high nor too low. This is because when α is too small, the model tends to neglect the TRS physical bias, resulting in error accumulations. Conversely, when α becomes too large, the model can emphasize TRS at the cost of accuracy. Nonetheless, across different α values, TREAT consistently surpasses the purely data-driven LG-ODE, showcasing its superiority and flexibility in modeling diverse dynamical systems.

Finally, we study its sensitivity towards solver choice and observation ratios in Appendix.

4.4.3 Visualizations

Trajectory Visualizations. Model predictions and ground truth are visualized in Figure 4.6. As HODEN is a single-agent baseline that individually forces every agent's energy to be constant over time which is

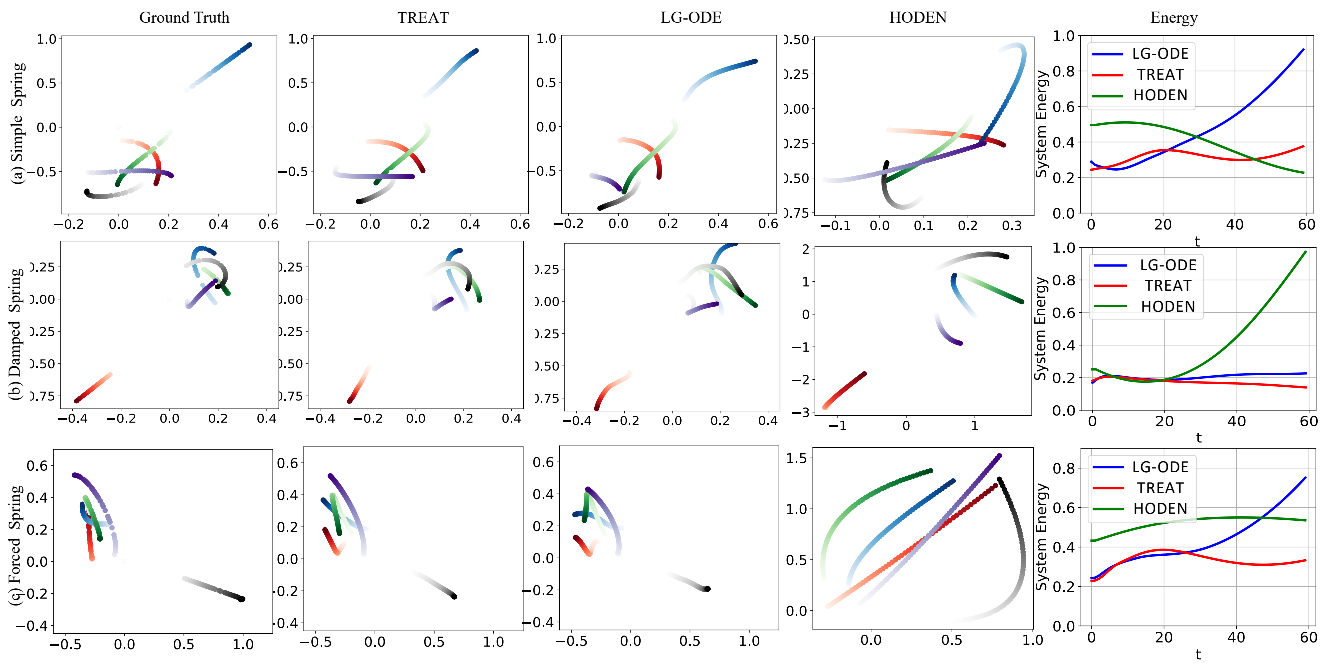


Figure 4.6: Visualization for 5-body spring systems (trajectory starts from light to dark colors).

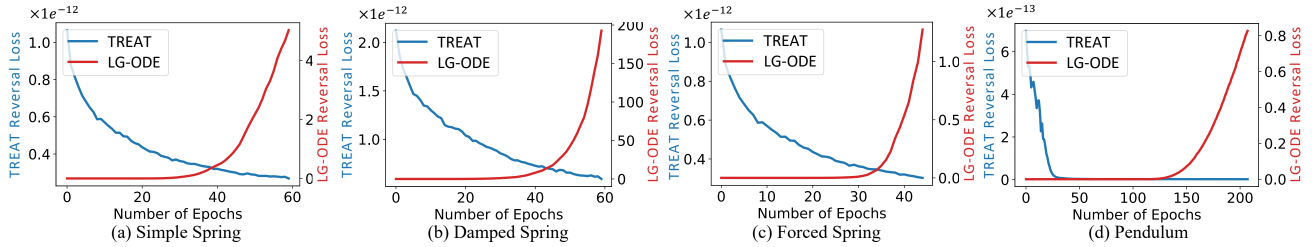


Figure 4.7: TRS loss visualization across multi-agent datasets (scales of two y-axes are different).

not valid, the predicted trajectories is having the largest errors and systems' total energy is not conserved for all datasets. The purely data-driven LG-ODE exhibits unrealistic energy patterns, as seen in the energy spikes in *Simple Spring* and *Force Spring*. In contrast, TREAT, incorporating reversal loss, generates realistic energy trends, and consistently produces trajectories closest to the ground truth, showing its superior performance.

Reversal Loss Visualizations To illustrate the issue of energy explosion from the purely data-driven LG-ODE, we visualize the TRS loss over training epochs from LG-ODE[5] and TREAT in Figure 4.7. As results suggest, LG-ODE has increased TRS loss over training epochs, meaning it is violating the

[5]There is no reversal loss backpropagation in LG-ODE, we just compute its value along training.

time-reversal symmetry sharply, in contrast to TREAT which has decreased reversal loss over epochs.

4.5 Conclusions

We propose TREAT, a deep learning framework that achieves high-precision modeling for a wide range of dynamical systems by injecting time-reversal symmetry as an inductive bias. TREAT features a novel regularization term to softly enforce time-reversal symmetry by aligning predicted forward and reverse trajectories from a GraphODE model. Notably, we theoretically prove that the regularization term effectively minimizes higher-order Taylor expansion terms during the ODE integration, which serves as a general numerical benefit widely applicable to various systems (even irreversible systems) regardless of their physical properties. Empirical evaluations on different kinds of datasets illustrate TREAT's superior efficacy in accurately capturing real-world system dynamics.

Part II

Towards Generalizable GraphODEs

CHAPTER 5

GG-ODE: Generalizing Graph ODE for Learning Complex System Dynamics across Environments

Learning multi-agent system dynamics has been extensively studied for various real-world applications, such as molecular dynamics in biology, multi-body system in physics, and particle dynamics in material science. Most of the existing models are built to learn single system dynamics, which learn the dynamics from observed historical data and predict the future trajectory. In practice, however, we might observe multiple systems that are generated across different environments, which differ in latent exogenous factors such as temperature and gravity. One simple solution is to learn multiple environment-specific models, but it fails to exploit the potential commonalities among the dynamics across environments and offers poor prediction results where per-environment data is sparse or limited. Here, we present GG-ODE (**G**eneralized **G**raph **O**rdinary **D**ifferential **E**quations), a machine learning framework for learning continuous multi-agent system dynamics across environments. Our model learns system dynamics using neural ordinary differential equations (ODE) parameterized by Graph Neural Networks (GNNs) to capture the continuous interaction among agents. We achieve the model generalization by assuming the dynamics across different environments are governed by common physics laws that can be captured via learning a shared ODE function. The distinct latent exogenous factors learned for each environment are incorporated into the ODE function to account for their differences. To improve model performance, we additionally design two regularization losses to (1) enforce the orthogonality between the learned initial states and exogenous factors via mutual information minimization; and (2) reduce the temporal variance of learned exogenous factors within the same system via contrastive learning. Experiments over various physical simulations show that our model can accurately predict system dynamics, especially in the long range, and can generalize well to new systems with few observations.

5.1 Introduction

Building a simulator that can understand and predict multi-agent system dynamics is a crucial research topic spanning over a variety of domains such as planning and control in robotics [LWZ19a], where the goal is to generate future trajectories of agents based on what has been seen in the past. Traditional simulators can be very expensive to create and use [SGP20b] as it requires sufficient domain knowledge and tremendous computational resources to generate high-quality results[1]. Therefore, learning a neural-based simulator directly from data that can approximate the behavior of traditional simulators becomes an attractive alternative.

As the trajectories of agents are usually coupled with each other and co-evolve along with the time, existing studies on learning system dynamics from data usually view the system as a graph and employ Graph Neural Networks (GNNs) to approximate pair-wise node (agent) interaction to impose strong inductive bias [BPL16]. As a pioneering work, Interaction Networks (IN) [BPL16] decompose the system into distinct objects and relations, and learn to reason about the consequences of their interactions and dynamics. Later work incorporates domain knowledge [LWZ19c], graph structure variances [PFS20], and equivariant representation learning [SLX21, GBG21] into learning from discrete GNNs, achieving state-of-the-art performance in various domains including mesh-based physical simulation [PFS20] and molecular prediction [GG22]. However, these *discrete* models usually suffer from low accuracy in long-range predictions as (1) they approximate the system by discretizing observations into some fixed timestamps and are trained to make a single forward-step prediction and (2) their discrete nature fails to adequately capture systems that are continuous in nature such as the spread of COVID-19 [HSW21b] and the movements of an n-body system [KFW18a, HSW20b].

Recently, researchers propose to combine ordinary differential equations (ODEs) - the principled way for modeling dynamical systems in a *continuous* manner in the past, with GNNs to learn continuous-time dynamics on complex networks in a data-driven way [HSW20b, HSW21b, ZW20, LYH23, JHL23b]. These Graph-ODE methods have demonstrated the power of capturing long-range dynamics, and are capable of learning from irregular-sampled partial observations [HSW20b]. They usually assume all the data are generated from one single system, and the goal is to learn the system dynamics from his-

[1]To date, out of the 10 most powerful supercomputers in the world, 9 of them are used for simulations, spanning the fields of cosmology, geophysics and fluid dynamics [jur19]

torical trajectories to predict the future. In practice, however, we might observe data that are generated from multiple systems, which can differ in their environments. For example, we may observe particle trajectories from systems that are with different temperatures, which we call exogenous factors. These exogenous factors can span over a wide range of settings such as particle mass, gravity, and temperature [SGP20b, BNM20a, BNM20b] across environments. One simple solution is to learn multiple environment-specific models, but it can fail to exploit the potential commonalities across environments and make accurate predictions for environments with sparse or zero observations. In many useful contexts, the dynamics in multiple environments share some similarities, yet being distinct reflected by the (substantial) differences in the observed trajectories. For example, considering the movements of water particles within multiple containers of varying shapes, the trajectories are driven by both the shared pair-wise physical interaction among particles (i.e. fluid dynamics) and the different shapes of the containers where collisions can happen when particles hit the boundaries. Also, the computational cost for training multiple environment-specific models would be huge. More challengingly, the exogenous factors within each environment can be latent, such as we only know the water trajectories are from different containers, without knowing the exact shape for each of them. Therefore, how to learn a single efficient model that can generalize across environments by considering both their commonalities and the distinct effect of per-environment latent exogenous factors remains unsolved. This model, if developed, may help us predict dynamics for systems under new environments with very few observed trajectories.

Inspired by these observations, in this paper, we propose Generalized Graph ODE (GG-ODE), a general-purpose continuous neural simulator that learns multi-agent system dynamics across environments. Our key idea is to assume the dynamics across environments are governed by common physics laws that can be captured via learning a shared ODE function. We introduce in the ODE function a learnable vector representing the distinct latent exogenous factors for each environment to account for their differences. We learn the representations for the latent exogenous factors from systems' historical trajectories through an encoder by optimizing the prediction goal. In this way, different environments share the same ODE function framework while incorporating environment-specific factors in the ODE function to distinguish them.

However, there are two main challenges in learning such latent exogenous factor representations. Firstly, since both the latent initial states for agents and the latent exogenous factors are learned through the historical trajectory data, how can we differentiate them to guarantee they have different semantic meanings? Secondly, when inferring from different time windows from the same trajectory, how can we guarantee the learned exogenous factors are for the same environment?

Towards the first challenge, we enforce the orthogonality between the initial state encoder and the exogenous factor encoder via mutual information minimization. For the second challenge, we reduce the variance of learned exogenous factors within the same environment via a contrastive learning loss. We train our model in a multi-task learning paradigm where we mix the training data from multiple systems with different environments. In this way, the model is expected to fast adapt to other unseen systems with a few data points. We conduct extensive experiments over a wide range of physical systems, which show that our GG-ODE is able to accurately predict system dynamics, especially in the long range.

The main contributions of this paper are summarized as follows:

- We investigate the problem of learning continuous multi-agent system dynamics across environments. We propose a novel framework, known as GG-ODE, which describes the dynamics for each system with a shared ODE function and an environment-specific vector for the latent exogenous factors to capture the commonalities and discrepancies across environments respectively.
- We design two regularization losses to guide the learning process of the latent exogenous factors, which is crucial for making precise predictions in the future.
- Extensive experiments verify the effectiveness of GG-ODE to accurately predict system dynamics, especially in the long range prediction tasks. GG-ODE also generalizes well to unseen or low-resource systems that have very few training samples.

5.2 Problem Definition

We aim to build a neural simulator to learn continuous multi-agent system dynamics automatically from data that can be generalized across environments. Throughout this paper, we use boldface uppercase letters to denote matrices or vectors, and regular lowercase letters to represent the values of variables.

We consider a multi-agent dynamical system of N interacting agents as an evolving interaction graph $\mathcal{G}^t = \{\mathcal{V}, \mathcal{E}^t\}$, where nodes are agents and edges are interactions between agents that can change over time. For each dynamical system, we denote $e \in E$ as the environment from which the data is acquired. We denote $\boldsymbol{X}^{t,e} \in \mathcal{X}$ as the feature matrix for all N agents and $\boldsymbol{x}_i^{t,e}$ as the feature vector of agent i at time

t under environment e. The edges between agents are assigned if two agents are within a connectivity radius R based on their current locations $\boldsymbol{p}_i^{t,e}$ which is part of the node feature vector, i.e. $\boldsymbol{p}_i^{t,e} \in \boldsymbol{x}_i^{t,e}$. They reflect the local interactions of agents and the radius is kept constant over time [SGP20b].

Our model input consists of the trajectories of N agents over K timestamps and we denote them as $X^{t_{1:K},e} = \{\boldsymbol{X}^{t_1,e}, \boldsymbol{X}^{t_2,e}, \ldots, \boldsymbol{X}^{t_K,e}\}$, where the timestamps $t_1, t_2 \cdots t_K$ can have non-uniform intervals and be of any continuous values. Our goal is to learn a generalized simulator $s_\theta : X^{t_{1:K},e} \to Y^{t_{K+1:T},e}$ that predicts node dynamics in the future for any environment e. Here $\boldsymbol{Y}^{t,e} \in \mathcal{Y}$ represents the targeted node dynamic information at time t, and can be a subset of the input features. We use $\boldsymbol{y}_i^{t,e}$ to denote the targeted node dynamic vector of agent i at time t under environment e.

5.3 Preliminaries and Related Work

Dynamical System Simulations with Graph Neural Networks (GNNs). Graph Neural Networks (GNNs) are a class of neural networks that operate on graph-structured data by passing local messages[KW17, VCC18, XHL19, HSW21b, HLJ22]. They have been extensively employed in various applications such as node classification [WPZ20, ZZW21], link prediction [QZD20, BK19], and recommendation systems [HDW20a, WHW19, WHL21b, JHH20]. By viewing each agent as a node and interaction among agents as edges, GNNs have shown to be efficient for approximating pair-wise node interactions and achieved accurate predictions for multi-agent dynamical systems [KFW18a, CUT16, SGP20b]. The majority of existing studies propose discrete GNN-based simulators where they take the node features at time t as input to predict the node features at time t+1. To further capture the long-term temporal dependency for predicting future trajectories, some work utilizes recurrent neural networks such as RNN, LSTM or self-attention mechanism to make prediction at time t +1 based on the historical trajectory sequence within a time window [HHN19, SWG20b, GSG17, HDW20b]. However, they all restrict themselves to learn a one-step state transition function. Therefore, when successively apply these one-step simulators to previous predictions in order to generate the rollout trajectories, error accumulates and impairs the prediction accuracy, especially for long-range prediction. Also, when applying most discrete GNNs to learn over multiple systems under different dynamical laws (environments), they usually retrain the GNNs individually for dealing with each specific system environment [SGP20b, KFW18a], which yields a large computational cost.

Ordinary Differential Equations (ODEs) for Multi-agent Dynamical Systems. The dynamic nature of a multi-agent system can be captured by a series of nonlinear first-order ordinary differential equations (ODEs), which describe the co-evolution of states for a set of N dependent variables (agents) over continuous time $t \in \mathbb{R}$ as [BB00, RCD19b]: $\dot{\boldsymbol{z}}_i^t := \frac{d\boldsymbol{z}_i^t}{dt} = g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right)$. Here $\boldsymbol{z}_i^t \in \mathbb{R}^d$ denotes the state variable for agent i at timestamp t and g denotes the ODE function that drives the system move forward. Given the initial states $\boldsymbol{z}_1^0, \cdots \boldsymbol{z}_N^0$ for all agents and the ODE function g, any black box numerical ODE solver such as Runge-Kuttais [MSH19] can solve the ODE initial-value problem (IVP), of which the solution $\boldsymbol{z}_i^T$ can be evaluated at any desired time as shown in Eqn 7.1.

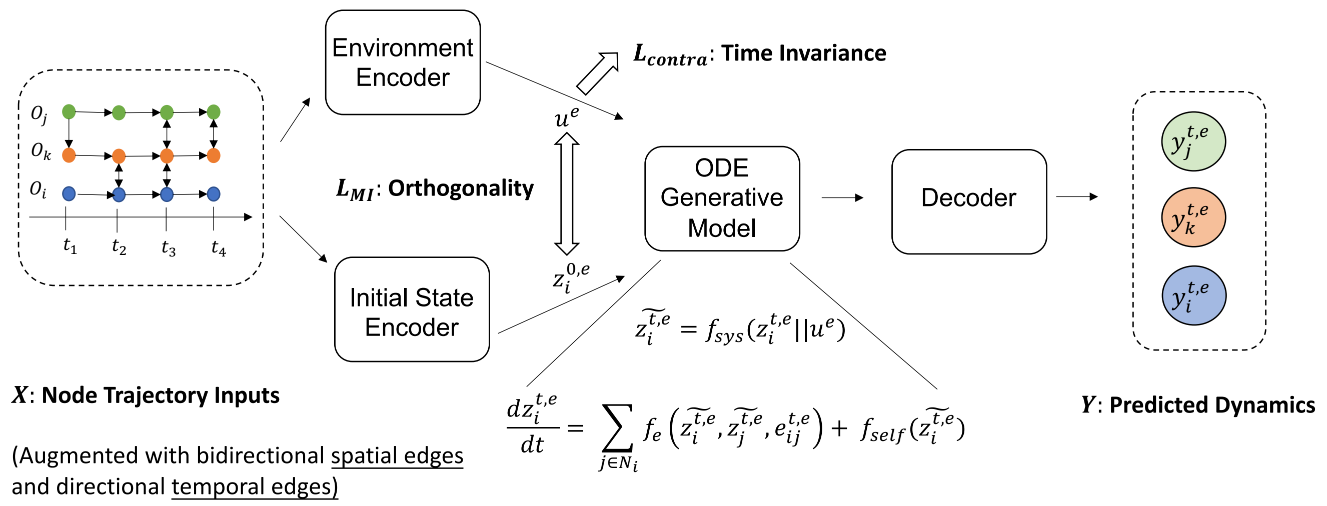
$$\boldsymbol{z}_i^T = \boldsymbol{z}_i^0 + \int_{t=0}^{T} g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right) dt \tag{5.1}$$

Traditionally, the ODE function g is usually hand-crafted based on some domain knowledge such as in robot motion control [SPA00] and fluid dynamics [LPZ02], which is hard to specify without knowing too much about the underlying principles. Even if the exact ODE functions are given, they are usually hard to scale as they require complicated numerical integration [MMD16, SGP20b]. Some recent studies [RCD19b, HSW20b, HSW21b] propose to parameterize it with a neural network and learn it in a data-driven way. They combine the expressive power of neural networks along with the principled modeling of ODEs for dynamical systems, which have achieved promising results in various applications [HSW20b, HSW21b, RCD19b].

GraphODE for Dynamical Systems. To model the complex interplay among agents in a dynamical system, researchers have recently proposed to combine ODE with GNNs, which has been shown to achieve superior performance in long-range predictions [HSW20b, HSW21b, ZW20]. In [ZW20], an encoder-processor-decoder architecture is proposed, where an encoder first computes the latent initial states for all agents individually based on their first observations. Then an ODE function parameterized by a GNN predicts the latent trajectories starting from the learned initial states. Finally, a decoder extracts the predicted dynamic features based on a decoding function that takes the predicted latent states as input. Later on, a Graph-ODE framework has been proposed [HSW20b, HSW21b] which follows the structure of variational autoencoder [KW14]. They assume an approximated posterior distribution over the latent initial state for each agent, which is learned based on the whole historical trajectories instead of a single point as in [ZW20]. The encoder computes the approximated posterior distributions for all agents simultaneously considering their mutual influence and then sample the initial states from them. Compared with [ZW20], they are able to achieve better prediction performance, especially in the long range, and are also capable of handling the dynamic evolution of graph structures [HSW21b] which is assumed to be static in [ZW20].

We follow a similar framework to this line but aim at generalizing GraphODE to model multiple systems across environments.

5.4 Method



o_j
o_k
o_i
t_1 t_2 t_3 t_4
X: Node Trajectory Inputs
(Augmented with bidirectional spatial edges and directional temporal edges)
Environment Encoder
Initial State Encoder
L_{contra}: Time Invariance
L_{MI}: Orthogonality
u^e
$z_i^{0,e}$
ODE Generative Model
Decoder
$\widetilde{z_i^{t,e}} = f_{sys}(z_i^{t,e} || u^e)$
$\frac{dz_i^{t,e}}{dt} = \sum_{j \in N_i} f_e\left(\widetilde{z_i^{t,e}}, \widetilde{z_j^{t,e}}, e_{ij}^{t,e}\right) + f_{self}(\widetilde{z_i^{t,e}})$
$y_j^{t,e}$
$y_k^{t,e}$
$y_i^{t,e}$
Y: Predicted Dynamics

Figure 5.1: The overall framework of GG-ODE consists of four modules. First, an initial state encoder computes the latent initial states for all agents simultaneously by constructing a temporal graph from the input trajectories. Additionally, an environment encoder computes the latent representations for exogenous factors that are distinct for each environment. Then, the generative model defined by a GNN-based ODE function calls the solver to output the predicted latent states for agents in the future, where the learned exogenous factors are incorporated into the ODE function. Finally, a decoder generates the predicted dynamics for each agent based on the decoding likelihood determined by the latent states. Two regularization terms are added to preserve the orthogonality of two encoders and the time-invariant property of the environment encoder.

In this section, we present Generalized Graph ODE (GG-ODE) for learning complex system dynamics across environments. As depicted in Figure 5.1, GG-ODE consists of four main components that are trained jointly: (1) an initial state encoder for inferring the latent initial states for all agents simultaneously; (2) an environment encoder which learns the latent representations for exogenous factors; (3) a generative model defined by a GNN-based ODE function that is shared across environments for modeling the continuous interaction among agents in the latent space. The distinct latent exogenous factors learned for each environment are incorporated into the ODE function to account for their discrepancies, and (4) a decoder that extracts the predicted dynamic features based on a decoding function. We now introduce each component in detail.

5.4.1 Initial State Encoder

Given the observed trajectories $X^{t_{1:K},e}$, the initial state encoder computes a posterior distribution of latent initial state $q_\phi\left(\boldsymbol{z}_i^{0,e} \mid X^{t_{1:K},e}\right)$ for each agent, from which $\boldsymbol{z}_i^{0,e}$ is sampled. The latent initial state $\boldsymbol{z}_i^{0,e}$ for each agent determines the starting point for the predicted trajectory. We assume the prior distribution $p(\boldsymbol{z}_i^{0,e})$ is a standard normal distribution, and use Kullback–Leibler divergence term in the loss function to add significant regularization towards how the learned distributions look like, which differs VAE from other autoencoder frameworks [KFW18a, HSW21b, RCD19b]. In multi-agent dynamical systems, agents are highly-coupled and influence each other. Instead of learning such distribution separately for each agent, such as using an RNN [RCD19b] to encode the temporal pattern for each individual trajectory, we compute the posterior distributions for all agents simultaneously (similar to [HSW21b]). Specifically, we fuse all trajectories as a whole into a temporal graph to consider both the temporal patterns of individual agents and the mutual interaction among them, where each node is an observation of an agent at a specific timestamp. Two types of edges are constructed, which are (1) spatial edges $\mathcal{V}^t$ that are among observations of interacting agents at each timestamp if the Euclidean distance between the agents' positions $r_{ij}^{t,e} = ||\boldsymbol{p}_i^{t,e} - \boldsymbol{p}_j^{t,e}||_2$ is within a (small) connectivity radius R; and (2) temporal edges that preserve the autoregressive nature of each trajectory, defined between two consecutive observations of the same agent. Note that spatial edges are bidirectional while temporal edges are directional to preserve the autoregressive nature of each trajectory, as shown in Figure 5.1. Based on the constructed temporal graph, we learn the latent initial states for all agents through a two-step procedure: (1) dynamic node representation learning that learns the representation $\boldsymbol{h}_i^{t,e}$ for each observation node whose feature vector is $\boldsymbol{x}_i^{t,e}$. (2) sequence representation learning that summarizes each observation sequence (trajectory) into a fixed-dimensional vector through a self-attention mechanism.

5.4.1.1 Dynamic Node Representation Learning.

We first conduct dynamic node representation learning on the temporal graph through an attention-based spatial-temporal GNN defined as follows:

$$\begin{gathered}
\boldsymbol{h}_j^{l+1(t,e)} = \boldsymbol{h}_j^{l(t,e)} + \sigma\left(\sum_{i^{(t',e)} \in \mathcal{N}_j^{(t,e)}} \alpha_i^{l(t',e)\to j(t,e)} \times \boldsymbol{W}_v \widehat{\boldsymbol{h}}_i^{l(t',e)}\right) \\
\alpha_i^{l(t',e)\to j(t,e)} = \left(\boldsymbol{W}_k \widehat{\boldsymbol{h}}_i^{l(t',e)}\right)^T \left(\boldsymbol{W}_q \boldsymbol{h}_j^{l(t,e)}\right) \cdot \frac{1}{\sqrt{d}} \\
\widehat{\boldsymbol{h}}_i^{l(t',e)} = \boldsymbol{h}_i^{l(t',e)} + \mathrm{TE}(t'-t) \\
\mathrm{TE}(\Delta t)_{2i} = \sin\left(\frac{\Delta t}{10000^{2i/d}}\right),\ \mathrm{TE}(\Delta t)_{2i+1} = \cos\left(\frac{\Delta t}{10000^{2i/d}}\right)
\end{gathered} \tag{5.2}$$

where $\sigma(\cdot)$ is a non-linear activation function; d is the dimension of node embeddings. The node representation is computed as a weighted summation over its neighbors plus residual connection where the attention score is a transformer-based [VSP17] dot-product of node representations by the use of value, key, query projection matrices $\boldsymbol{W}_v, \boldsymbol{W}_k, \boldsymbol{W}_q$. The learned attention scores are normalized via softmax across all neighbors. Here $\boldsymbol{h}_j^{l(t,e)}$ is the representation of agent j at time t in the l-th layer. $\boldsymbol{h}_i^{l(t',e)}$ is the general representation for a neighbor which is connected either by a temporal edge (where $t' < t$ and $i = j$) or a spatial edge (where $t = t'$ and $i \neq j$) to the observation $\boldsymbol{h}_j^{l(t,e)}$. We add temporal encoding [VSP17, RCD19b] to each neighborhood node representation in order to distinguish the message delivered via spatial and temporal edges respectively. Finally, we stack L layers to get the final representation for each observation node as : $\boldsymbol{h}_i^{t,e} = \boldsymbol{h}_i^{L(t,e)}$.

5.4.1.2 Sequence Representation Learning

We then employ a self-attention mechanism to generate the sequence representation $\boldsymbol{m}_i^e$ for each agent, which is used to compute the mean $\boldsymbol{\mu}_i^{0,e}$ and variance $\boldsymbol{\sigma}_i^{0,e}$ of the approximated posterior distribution of the agent's initial state. Compared with recurrent models such as RNN, LSTM [SJ97], it offers better parallelization for accelerating training speed and in the meanwhile alleviates the vanishing/exploding gradient problem brought by long sequences [SWG20b].

We follow [HSW21b] and compute the sequence representation $\boldsymbol{m}_i^e$ as a weighted sum of observations for agent i:

$$\boldsymbol{m}_i^e = \frac{1}{K}\sum_t \sigma\left((\boldsymbol{a}_i^e)^T \widehat{\boldsymbol{h}}_i^{t,e} \widehat{\boldsymbol{h}}_i^{t,e}\right),\ \boldsymbol{a}_i^e = \tanh\left(\left(\frac{1}{K}\sum_t \widehat{\boldsymbol{h}}_i^{t,e}\right)\boldsymbol{W}_a\right), \tag{5.3}$$

where $\boldsymbol{a}_i^e$ is the average of observation representations with a nonlinear transformation $\boldsymbol{W}_a$ and $\widehat{\boldsymbol{h}}_i^{t,e} = \boldsymbol{h}_i^{t,e} + \mathrm{TE}(t)$. K is the number of observations for each trajectory. Then the initial state is drawn from the approximated posterior distribution as:

$$\begin{gathered} q_\phi\left(\boldsymbol{z}_i^{0,e} \mid X^{t_{1:K},e}\right) = \mathcal{N}\left(\boldsymbol{\mu}_i^{0,e}, \boldsymbol{\sigma}_i^{0,e}\right),\ \boldsymbol{\mu}_i^{0,e}, \boldsymbol{\sigma}_i^{0,e} = f_{\text{trans}}\left(\boldsymbol{m}_i^e\right) \\ \boldsymbol{z}_i^{0,e} \sim p\left(\boldsymbol{z}_i^{0,e}\right) \approx q_\phi\left(\boldsymbol{z}_i^{0,e} \mid X^{t_{1:K},e}\right) \end{gathered} \tag{5.4}$$

where f_{trans} is a simple Multilayer Perceptron (MLP) whose output vector is equally split into two halves to represent the mean and variance respectively.

5.4.2 Environment Encoder

The dynamic nature of a multi-agent system can be largely affected by some exogenous factors from its environment such as gravity, temperature, etc. These exogenous factors can span over a wide range of settings and are sometimes latent and not observable. To make our model generalize across environments, we design an environment encoder to learn the effect of the exogenous factors automatically from data to account for the discrepancies across environments. Specifically, we use the environment encoder to learn the representations of exogenous factors from observed trajectories and then incorporate the learned vector into the ODE function which is shared across environments and defines how the system evolves over time. In this way, we use a shared ODE function framework to capture the commonalities across environments while preserving the differences among them with the environment-specific latent representation, to improve model generalization performance. It also allows us to learn the exogenous factors of an unseen environment based on only its leading observations. We now introduce the environment encoder in detail.

The exogenous factors would pose influence on all agents within a system. On the one hand, they will influence the self-evolution of each individual agent. For example, temperatures would affect the velocities of agents. On the other hand, they will influence the pair-wise interaction among agents. For example, temperatures would also change the energy when two particles collide with each other. The environment encoder $f_{\text{enc}}^{\text{env}}$ therefore learns the latent representation of exogenous factors $\boldsymbol{u}^e$ by jointly consider the trajectories from all agents, i.e. $f_{\text{enc}}^{\text{env}} : X^{t_{1:K},e} \rightarrow \boldsymbol{u}^e$. Specifically, we learn an environment-specific latent vector from the aforementioned temporal graph in Sec 5.4.1 that is constructed from observed trajectories. The temporal graph contains both the information for each individual trajectory and the mutual interaction among agents through temporal and spatial edges. To summarize the whole temporal graph into a vector $\boldsymbol{u}^e$, we attend over the sequence representation $\boldsymbol{m}_i^e$ for each trajectory introduced in Sec 5.4.1 as:

$$\boldsymbol{u}^e = \frac{1}{N}\sum_i \sigma\left((\boldsymbol{b}^e)^T \boldsymbol{m}_i^e \boldsymbol{m}_i^e\right),\ \boldsymbol{b}^e = \tanh\left(\left(\frac{1}{N}\sum_i \boldsymbol{m}_i^e\right)\boldsymbol{W}_b\right), \tag{5.5}$$

where $\boldsymbol{W}_b$ is a transformation matrix and the attention weight is computed based on the average sequence representation with nonlinear transformation similar as in Eqn (5.3). Note that we use different parameters to compute the sequence representation $\boldsymbol{m}_i^e$ as opposed to the initial state encoder. The reason is that the semantic meanings of the two sequence representations are different: one is for the latent initial states and another is for the exogenous factors.

5.4.2.1 Time Invariance.

A desired property of the learned representation for exogenous factors $\boldsymbol{u}^e$ is that it should be time-invariant towards the input trajectory time window. In other words, for the same environment, if we chunk the whole trajectories into several pieces, the inferred representations should be similar to each other as they are describing the same environment.

To achieve this, we design a contrastive learning loss to guide the learning process of the exogenous factors. As shown in Figure 5.2, we force the learned exogenous factor representations to be similar if they are generated based on the trajectories from the same environment (positive pairs), and to be apart from each other if they are from different environments (negative pairs). Specifically, we define the contrastive leanring loss as follows:

$$\mathcal{L}_{\text{contra}} = -\log \frac{\exp\left(\text{sim}\left(f_{\text{enc}}^{\text{env}}\left(X^{t_1:t_2,e}\right), f_{\text{enc}}^{\text{env}}\left(X^{t_3:t_4,e}\right)\right)/\tau\right)}{\sum_{e'\neq e}\exp\left(\text{sim}\left(f_{\text{enc}}^{\text{env}}\left(X^{t_1:t_2,e}, f_{\text{enc}}^{\text{env}}\left(X^{t_5:t_6,e'}\right)/\tau\right)\right.\right.} \tag{5.6}$$

where τ is a temperature scalar and $\text{sim}(\cdot,\cdot)$ is cosine similarity between two vectors. Note that the lengths of the observation sequences can vary. The detailed generation process for positive and negative pairs can be found in Appendix.

5.4.2.2 Orthogonality.

GG-ODE features two encoders that take the input of observed trajectories $X^{t_{1:K},e}$ for learning the latent initial states and the latent exogenous factors respectively. As they are designed for different purposes but are both learned from the same input, we disentangle the learned representations from them via a regularization loss defined via mutual information minimization.

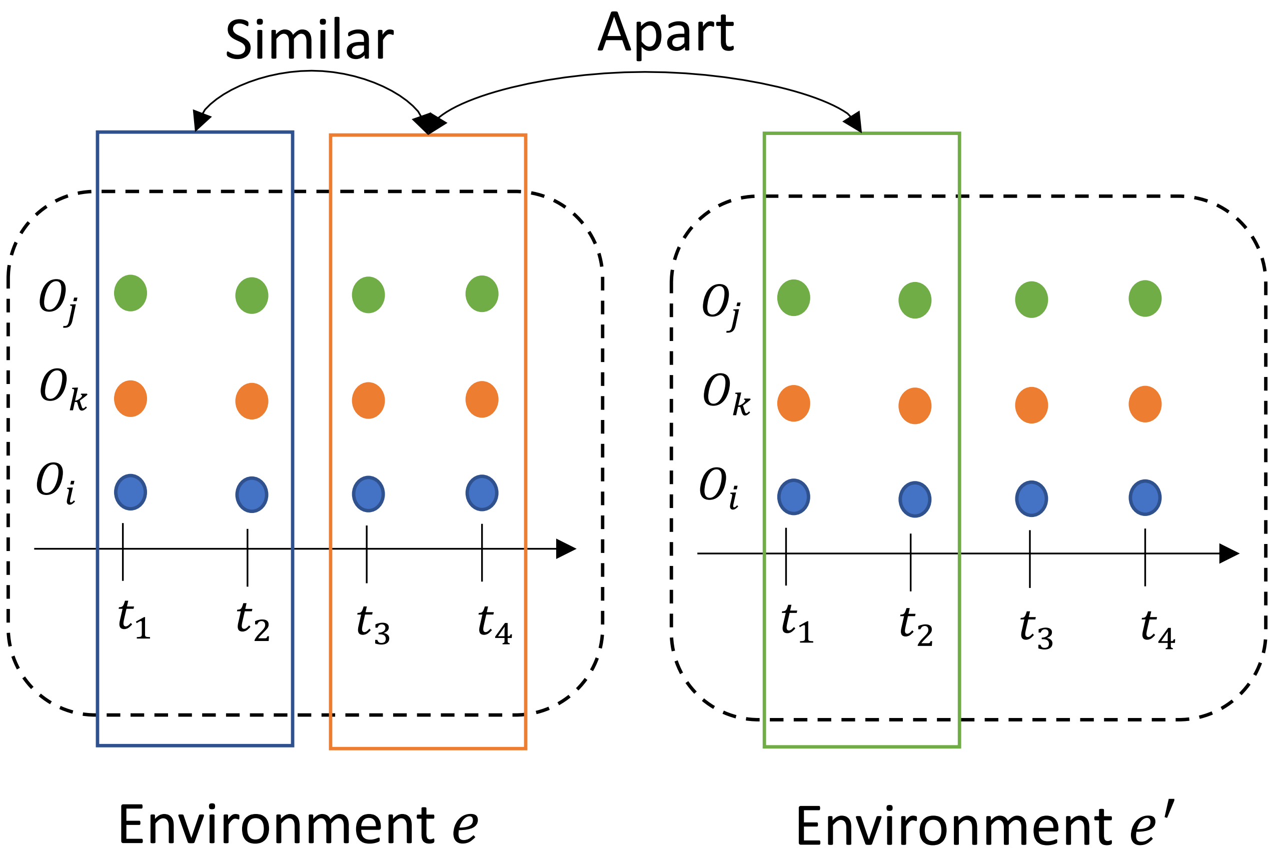


Figure 5.2: Temporal properties of the environment encoder. We use contrastive learning loss to force the latent exogenous factors learned from different windows within the same environment to be close to each other, and from different environments to be apart from each other.

Mutual information measures the dependency between two random variables X, Z [ZZL21]. Since we are not interested in the exact value of the mutual information, a lower bound derived from Jensen Shannon Divergence [HFL18] could be formulated as

$$I_{\text{JSD}}(X, Z) = E_{P_{XZ}}[-\text{sp}(-M(x, z))] - E_{P_X P_Z}[\text{sp}(M(x, z))], \tag{5.7}$$

where $P_X P_Z$ is the product of the marginal distributions and P_{XZ} is the joint distribution. $sp(w) = log(1 + e^w)$ and M is a discriminator modeled by a neural network to compute the score for measuring their mutual information.

According to recent literature [ZZL21, HFL18, SSO20], the sample pair (positive pairs) (x, z) drawn from the joint distribution P_{XZ} are different representations of the same data sample, and the sample pair (negative pairs) drawn from $P_X P_Z$ are different representations from different data samples. We therefore attempt to minimize the mutual information from the two encoders as follows

$$\mathcal{L}_{\text{MI}} = \mathbb{E}_{e \in E, i}[-sp(-\Psi(\boldsymbol{z}_i^{0,e}, \boldsymbol{u}^e))] - \mathbb{E}_{e \in E \times e' \in E, i}[sp(\Psi(\boldsymbol{z}_i^{0,e}, \boldsymbol{u}^{e'}))] \tag{5.8}$$

where Ψ is a MLP-based discriminator. Specifically, we force the latent initial states $\boldsymbol{z}_i^{0,e}$ for all agents from environment e to be dissimilar to the learned exogenous factors $\boldsymbol{u}^e$. And construct negative pairs by replacing the learned exogenous factors from another environment as $\boldsymbol{u}^{e'}$. The generation process for positive and negative pairs can be found in Appendix.

5.4.3 ODE Generative Model and Decoder

5.4.3.1 ODE Generative Model

After describing the initial state encoder and the environment encoder, we now define the ODE function that drives the system to move forward. The future trajectory of each agent can be determined by two important factors: the potential influence received from its neighbors in the interaction graph and the self-evolution of each agent. For example, in the n-body system, the position of each agent can be affected both by the force from its connected neighbors and its current velocity which can be inferred from its historical trajectories. Therefore, our ODE function consists of two parts: a GNN that captures the continuous interaction among agents and the self-evolution of the node itself. One issue here is how can we decide the neighbors for each agent in the ODE function as the interaction graph is evolving, the neighbors for each agent are dynamically changing based on their current positions, which are implicitly encoded in their latent state representations $\boldsymbol{z}_i^{t,e}, \boldsymbol{z}_j^{t,e}$. We propose to first decode the latent node representations $\boldsymbol{z}_i^{t,e}, \boldsymbol{z}_j^{t,e}$ with a decoding function f_{dec} to obtain their predicted positions $\boldsymbol{p}_i^{t,e}, \boldsymbol{p}_j^{t,e}$ at current timestamp. Then we determine their connectivity based on whether their Euclidean distance $r_{ij}^{t,e} = ||\boldsymbol{p}_i^{t,e} - p_j^{t,e}||_2$ is within the predefined radius R. This can be computed efficiently by using a multi-dimensional index structure such as the k-d tree. The decoding function f_{dec} is the same one that we will use in the decoder.

To incorporate the influence of exogenous factors, we further incorporate $\boldsymbol{u}^e$ into the general ODE function to improve model generalization ability as:

$$\begin{gathered}\frac{d\boldsymbol{z}_i^{t,e}}{dt} = g\left(\boldsymbol{z}_1^{t,e}, \boldsymbol{z}_2^{t,e} \cdots \boldsymbol{z}_N^{t,e}\right) = \sum_{j \in \mathcal{N}_i} f_{\text{GNN}}(\widetilde{\boldsymbol{z}}_i^{t,e}, \widetilde{\boldsymbol{z}}_j^{t,e}) + f_{\text{self}}(\widetilde{\boldsymbol{z}}_i^{t,e}) \\ \widetilde{\boldsymbol{z}}_i^{t,e} = f_{\text{env}}(\boldsymbol{z}_i^{t,e} || \boldsymbol{u}^e)\end{gathered} \tag{5.9}$$

where $||$ denotes concatenation and f_{GNN} can be any GNN that conducts message passing among agents. $f_{\text{self}}, f_{\text{env}}$ are implemented as two MLPs respectively. In this way, we learn the effect of latent exogenous factors from data without supervision where the latent representation $\boldsymbol{u}^e$ is trained end-to-end by optimizing the prediction loss.

5.4.3.2 Decoder

Given the ODE function g and agents' initial states $\boldsymbol{z}_i^{0,e}$ for $i = 1, 2 \cdots N$, the latent trajectories for all agents are determined, which can be solved via any black-box ODE solver. Finally, a decoder generates the predicted dynamic features based on the decoding probability $p(\boldsymbol{y}_i^{t,e}|\boldsymbol{z}_i^{t,e})$ computed from the decoding function f_{dec} as shown in Eqn 7.7. We implement f_{dec} as a simple two-layer MLP with nonlinear activation. It outputs the mean of the normal distribution $p(\boldsymbol{y}_i^{t,e}|\boldsymbol{z}_i^{t,e})$, which we treat as the predicted value for each agent.

$$\begin{gathered}
\boldsymbol{z}_i^{t_1,e} \cdots \boldsymbol{z}_i^{t_T,e} = \text{ODESolve}(g, [\boldsymbol{z}_1^{0,e}, \boldsymbol{z}_2^{0,e} \cdots \boldsymbol{z}_N^{0,e}], (t_1 \cdots t_T)) \\
\boldsymbol{y}_i^{t,e} \sim p(\boldsymbol{y}_i^{t,e}|\boldsymbol{z}_i^{t,e}) = f_{\text{dec}}(\boldsymbol{z}_i^{t,e})
\end{gathered} \tag{5.10}$$

5.4.4 Training

We now introduce the overall training procedure of GG-ODE . For each training sample, we split it into two halves along the time, where we condition on the first half $[t_1, t_K]$ in order to predict dynamics in the second half $[t_{K+1}, t_T]$. Given the observed trajectories $X^{t_{1:K},e}$, we first run the initial state encoder to compute the latent initial state $\boldsymbol{z}_i^{0,e}$ for each agent, which is sampled from the approximated posterior distribution $q_\phi\left(\boldsymbol{z}_i^{0,e} \mid X^{t_{1:K},e}\right)$. We then generate the latent representations of exogenous factors $\boldsymbol{u}^e$ from the environment e via the environment encoder. Next, we run the ODE generative model that incorporates the latent exogenous factors to compute the latent states for all agents in the future. Finally, the decoder outputs the predicted dynamics for each agent.

We jointly train the encoders, ODE generative model, and decoder in an end-to-end manner. The loss function consists of three parts: (1) the evidence lower bound (ELBO) which is the addition of the reconstruction loss for node trajectories and the KL divergence term for adding regularization to the inferred latent initial states for all agents. We use $\boldsymbol{Z}^{0,e}$ to denote the latent initial state matrix of all N agents. The standard VAE framework is trained to maximize ELBO so we take the negative as the ELBO loss; (2) the contrastive learning loss for preserving the time invariance properties of the learned exogenous factors; (3) the mutual information loss that disentangles the learned representations from the two encoders. λ_1, λ_2 are two hyperparameters for balancing the three terms. We summarize the whole procedure in Appendix.

$$\mathcal{L} = \mathcal{L}_{\text{ELBO}} + \lambda_1 \mathcal{L}_{\text{contra}} + \lambda_2 \mathcal{L}_{MI} \tag{5.11}$$

$$\begin{aligned}\mathcal{L}_{\text{ELBO}(\theta,\phi)} = -\mathbb{E}_{\boldsymbol{Z}^{0,e} \sim \prod_{i=1}^{N} q_\phi\left(\boldsymbol{z}_i^{0,e} | X^{t_{1:K},e}\right)}[\log p_\theta(Y^{t_{K+1:T},e})] \\ + \text{KL}[\prod_{i=1}^{N} q_\phi(\boldsymbol{z}_i^{0,e} | X^{t_{1:K},e}) \| p(\boldsymbol{Z}^{0,e})]\end{aligned} \tag{5.12}$$

5.5 Experiments

Table 5.1: Mean Square Error (MSE) of rollout trajectories with varying prediction lengths. The transductive setting evaluates the testing sequences whose environments are seen during training. The inductive setting evaluates new systems with unseen environments during training. The best results are bold-faced.

Dataset	Lennard-Jones potential Transductive MSE (10^{-2})			Lennard-Jones potential Inductive MSE (10^{-1})			Water Transductive MSE (10^{-3})			Water Inductive MSE (10^{-2})		
Rollout Percentage	30%	60%	100%	30%	60%	100%	30%	60%	100%	30%	60%	100%
LSTM	6.73	20.69	31.88	1.64	8.82	18.01	4.87	23.09	30.44	1.01	6.72	14.79
NRI	5.83	17.99	28.18	1.33	4.34	13.97	3.87	19.64	26.34	0.83	3.84	10.59
NDCN	5.99	17.54	27.06	1.35	4.27	12.37	3.95	18.76	24.33	0.85	3.79	10.11
CG-ODE	5.43	17.01	26.01	1.32	4.25	12.03	3.41	18.13	23.62	0.80	3.64	9.91
SocialODE	5.62	17.23	26.89	1.34	4.26	12.44	3.68	18.42	23.77	0.84	3.70	10.01
GNS	**5.03**	16.28	25.44	1.28	4.23	11.88	**3.17**	17.88	23.14	0.76	3.45	9.78
GG-ODE	5.18	**16.03**	**24.97**	**1.10**	**3.98**	**10.77**	3.20	**16.94**	**22.58**	**0.63**	**3.11**	**8.02**
-w/o $\mathcal{L}_{\text{contra}}$	5.32	17.03	26.53	1.30	4.25	12.13	3.32	18.03	23.01	0.75	3.58	10.03
-w/o$\mathcal{L}_{MI}$	5.45	17.25	26.11	1.32	4.11	11.76	3.43	18.32	22.95	0.78	3.51	9.88
shared encoders	5.66	17.44	26.79	1.33	4.46	12.22	3.55	18.57	23.55	0.81	3.66	10.08

5.5.1 Experiment Setup

5.5.1.1 Datasets

We illustrate the performance of our model across two physical simulations that exhibit different system dynamics over time: (1) The Water dataset [SGP20b], which describes the fluid dynamics of water within a container. Containers can have different shapes and numbers of ramps with random positions inside them, which we view as different environments. The dataset is simulated using the material point method (MPM), which is suitable for simulating the behavior of interacting, deformable materials such as solids, liquids, gases [2]. For each data sample, the number of particles can vary but the trajectory lengths are kept the same as 600. The input node features are 2-D positions of particles, and we calculate the velocities and accelerations as additional node features using finite differences of these positions. The total number of data samples (trajectories) is 1200 and the number of environments is 68, where each environment can have multiple data samples with different particle initializations such as positions, velocities, and accelerations. (2) The Lennard-Jones potential dataset [Jon24], which describes the soft repulsive and attractive interactions between simple atoms and molecules [3]. We generate data samples with different temperatures, which could affect the potential energy preserved within the whole system thus affecting the dynamics. We view temperatures as different environments. The total number of data samples (trajectories) is 6500 and the number of environments is 65. Under each environment, we generate 100 trajectories with different initializations. The trajectory lengths are kept the same as 100. The number of particles is 1000 for all data samples. More details about datasets can be found in Appendix.

5.5.1.2 Task Evaluation and Data Split

We predict trajectory rollouts across varying lengths and use Mean Square Error (MSE) as the evaluation metric.

Task Evaluation. The trajectory prediction task is conducted under two settings: (1) Transductive setting, where we evaluate the test sequences whose environments are seen during training; (2) Inductive setting, where we evaluate the test sequences whose environments are not observed during training. It helps to test the model's generalization ability to brand-new systems.

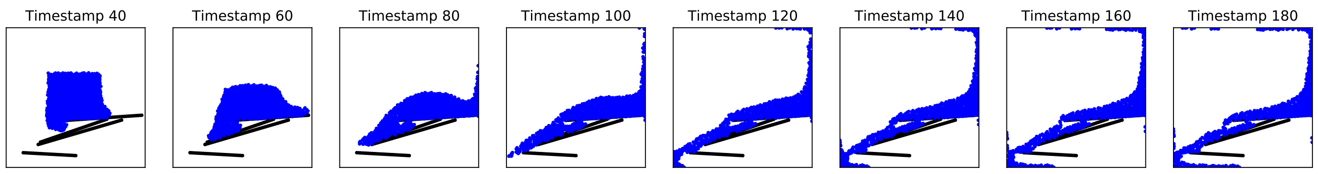
[2] `https://en.wikipedia.org/wiki/Material_point_method`

[3] `https://en.wikipedia.org/wiki/Lennard-Jones_potential`

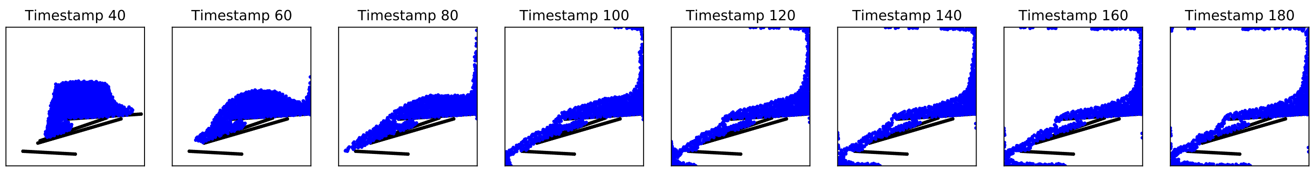
Data Split. We train our model in a sequence-to-sequence setting where we split the trajectory of each training sample into two parts $[t_1, t_K]$ and $[t_{K+1}, t_T]$. We condition on the first part of observations to predict the second part. To conduct data split, we first randomly select 20% environments whose trajectories are all used to construct the testing set $X_{\text{test}}^{\text{Induct}}$ in the inductive setting. For the remaining trajectories that cover the 80% environments, we randomly split them into three partitions: 80% for the training set X_{train}, 10% for the validation set X_{val} and 10% for the testing set in the transductive setting $X_{\text{test}}^{\text{trans}}$. In other words, we have two test sets for the inductive and transductive settings respectively, one training set and one validation set. To fully utilize the data points within each trajectory, we generate training and validation samples by splitting each trajectory into several chunks that can overlap with each other, using a sliding window. The sliding window has three hyperparameters: the observation length and prediction length for each sample, and the interval between two consecutive chunks (samples). Specifically, for the Water dataset, we set the observation length as 50 and the prediction length as 150. We obtain samples from each trajectory by using a sliding window of size 200 and setting the sliding interval as 50. For the Lennard-Jones potential dataset, we set the observation length as 20, the prediction length as 50, and the interval as 10. The procedure is summarized in Appendix. During evaluations for both settings, we ask the model to roll out over the whole trajectories without further splitting, whose prediction lengths are larger than the ones during training. The observation lengths during testing are set as 20 for the Lennard-Jones potential dataset and 50 for the Water dataset across the two settings.

5.5.2 Baselines

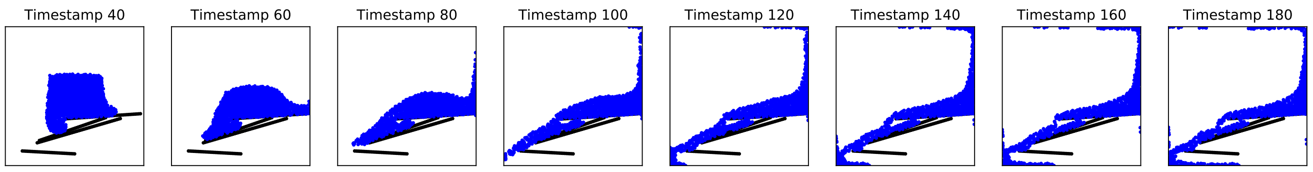
We compare both discrete neural models as well as continuous neural models where they do not have special treatment for modeling the influence from different environments. For discrete ones we choose: NRI [KFW18a] which is a discrete GNN model that uses VAE to infer the interaction type among pairs of agents and is trained via one-step predictions; GNS [SGP20b], a discrete GNN model that uses multiple rounds of message passing to predict every single step; LSTM [SJ97], a classic recurrent neural network (RNN) that learns the dynamics of each agent independently. For the continuous models, we compare with NDCN [ZW20] and Social ODE [WWM22], two ODE-based methods that follow the encoder-processor-decoder structure with GNN as the ODE function. The initial state for each agent is drawn from a single data point instead of a leading sequence. CG-ODE [HSW21b] which has the same architecture as our model, but with two coupled ODE functions to guide the evolution of systems.



(a) Ground Truth



(b) Predictions of GNS



(c) Predictions of GG-ODE

Figure 5.3: Visualization of the transductive prediction results for the Water dataset. Black lines are ramps within the container. The length of the observation sequence is set as 20. GNS makes less accurate predictions compared with GG-ODE.

5.5.3 Performance Evaluation

We evaluate the performance of our model based on Mean Square Error (MSE) as shown in Table 5.1. As data samples have varying trajectory lengths, we report the MSEs over three rollout percentages regarding different prediction horizons: $30\%, 60\%, 100\%$ where 100% means the model conditions on the observation sequence and predicts all the remaining timestamps.

Firstly, we can observe that GG-ODE consistently outperforms all baselines across different settings when making long-range predictions, while achieving competitive results when making short-range predictions. This demonstrates the effectiveness of GG-ODE in learning continuous multi-agent system dynamics across environments. By comparing the performance of LSTM with other methods, we can see that modeling the latent interaction among agents can indeed improve the prediction performance compared with predicting trajectories for each agent independently. Also, we can observe the performance gap between GG-ODE and other baselines increase when we generate longer rollouts, showing its expressive power when making long-term predictions. This may be due to the fact that GG-ODE is a continuous model trained in a sequence-to-sequence paradigm whereas discrete GNN methods are only trained to make a fixed-step prediction. Another continuous model NDCN only conditions a single data point to make predictions for the whole trajectory in the future, resulting in suboptimal performance. Finally, we can see that GG-ODE has a larger performance gain over existing methods in the inductive setting than in the transductive setting, which shows its generalization ability to fast adapt to other unseen systems with a few data points. Figure 5.3 visualizes the prediction results under the transductive setting for the Water dataset.

5.5.3.1 Ablation Studies

To further analyze the rationality behind our model design, we conduct an ablation study by considering three model variants: (1) We remove the contrastive learning loss which forces the learned exogenous factors to satisfy the time invariance property, denoted as $-w/o\mathcal{L}_{\text{contra}}$; (2) We remove the mutual information minimization loss which reduces the variance of the learned exogenous factors from the same environment, denoted as $-w/o\mathcal{L}_{MI}$. (3) We share the parameters of the two encoders for computing the latent representation $\boldsymbol{m}_i^e$ for each observation sequence in the temporal graph, denoted as shared encoders. As shown in Table 5.1, all three variants have inferior performance compared to GG-ODE , verifying the rationality of the three key designs. Notably, when making long-range predictions, removing $\mathcal{L}_{MI}$ would cause more harm to the model than removing $\mathcal{L}_{\text{contra}}$. This can be understood as the latent initial states are more important for making short-term predictions, while the disentangled latent initial states and exogenous factors are both important for making long-range predictions.

5.5.3.2 Hyperparameter Study

We study the effect of λ_1/λ_2, which are the hyperparameters for balancing the two regularization terms that guide the learning of the two encoders, towards making predictions under different horizons. As illustrated in Figure 5.4, the optimal ratio for making $30\%, 60\%, 100\%$ rollout predictions are 2, 1,0.5 respectively, under both the transductive and inductive settings. They indicate that the exogenous factors modeling plays a more important role in facilitating long-term predictions, which is consistent with the prediction errors illustrated in Table 5.1 when comparing $-w/o\mathcal{L}_{MI}$ with $-w/o\mathcal{L}_{\text{contra}}$. However, overly elevating $\mathcal{L}_{MI}$ would also harm the model performance, as the time invariance property achieved by $\mathcal{L}_{\text{contra}}$ is also important to guarantee the correctness of the learned latent initial states, which determines the starting point of the predicted trajectories in the future.

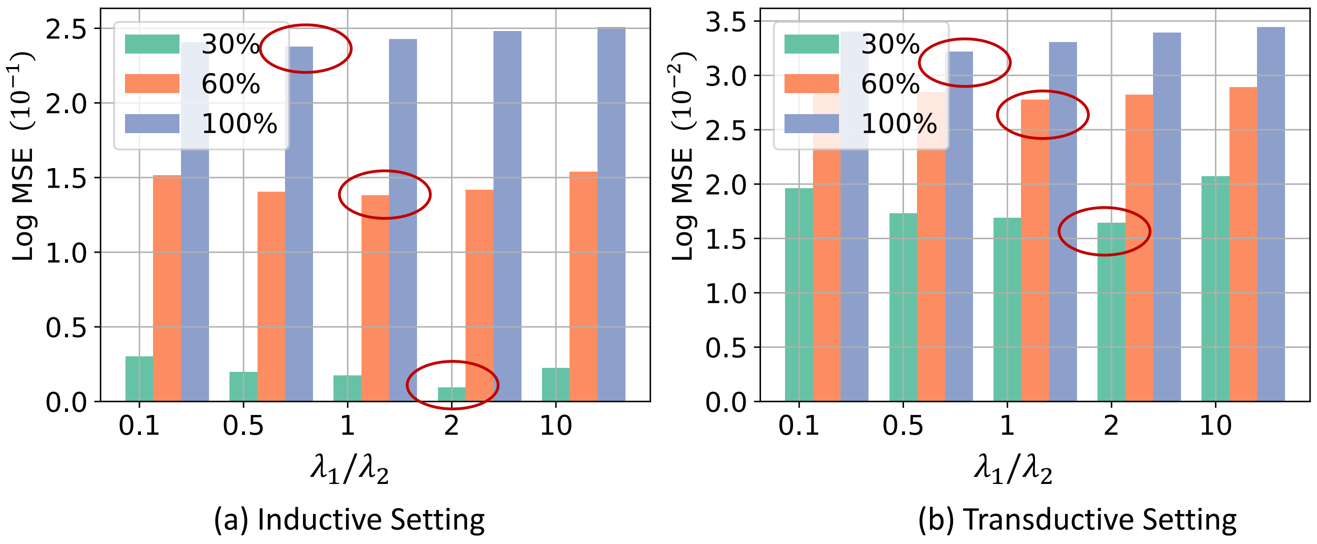


Figure 5.4: Effect of λ_1/λ_2 on the Lennard-Jones potential dataset. Best results are circled in red for each setting.

5.5.3.3 Sensitivity Analysis.

GG-ODE can take arbitrary observation lengths to make trajectory predictions, as opposed to existing baselines that only condition on observations with fixed lengths. It allows the model to fully utilize all the information in the past. We then study the effect of observation lengths on making predictions in different horizons. As shown in Figure 5.5, the optimal observation lengths for predicting the rollouts with 20, 40, and 50 steps are 20, 25, 35 in the inductive setting, and 15, 25, 30 in the transductive setting. When predicting long-range trajectories, our model typically requires a longer observation sequence to get more accurate results. Also, for making predictions at the same lengths, the inductive setting requires a longer observation length compared with the transductive setting.

5.5.4 Case Study

We conduct a case study to examine the learned representations of the latent exogenous factors on the Lennard-Jones potential dataset. We first randomly choose one data sample for each of the 65 temperatures and visualize the learned representations of exogenous factors. As shown in Figure 7.2 (a), the representations of higher temperatures are closer to each other on the right half of the figure, whereas the lower temperatures are mostly distributed on the left half. Among the 65 temperatures, 20% of them are not seen during training which we circled in black. We can see those unseen temperatures

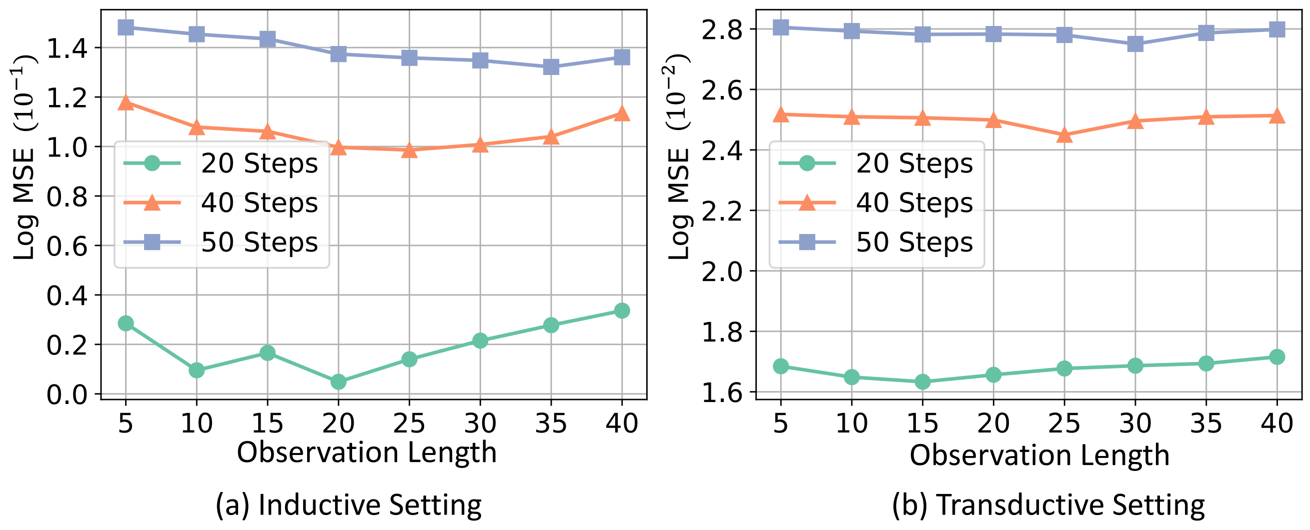
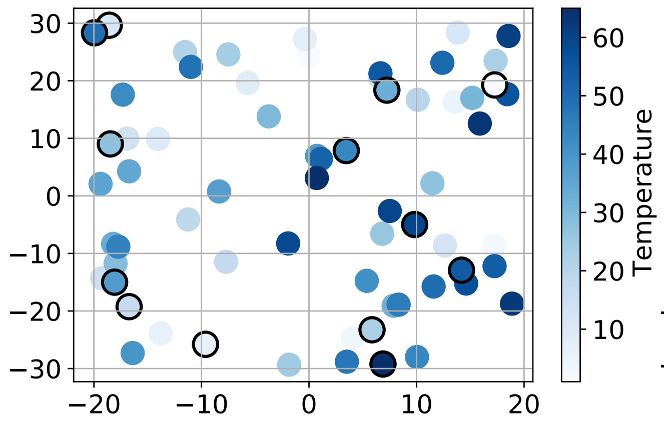


Figure 5.5: Effect of observation length on the Lennard-Jones potential dataset.

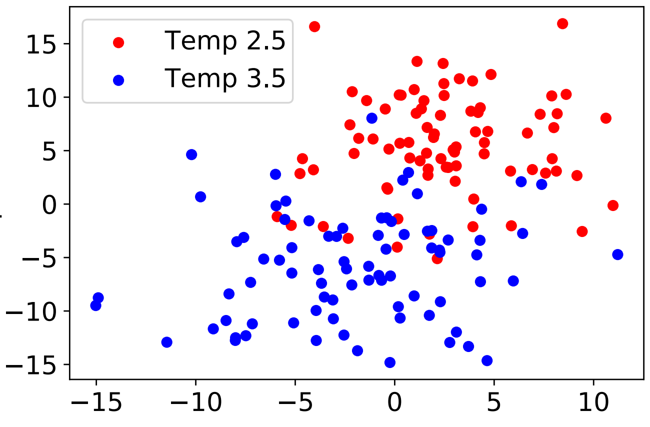
are also properly distributed, indicating the great generalization ability of our model. We next plot the representations for all data samples under temperatures 2.5 and 3.5 respectively as shown in Figure 7.2 (b). We can see that the learned representations are clustered within the two temperatures, indicating our contrastive learning loss is indeed beneficial to guide the learning process of exogenous factors.

5.6 Conclusion

In this paper, we investigate the problem of learning the dynamics of continuous interacting systems across environments. We model system dynamics in a continuous fashion through graph neural ordinary differential equations. To achieve model generalization, we learn a shared ODE function that captures the commonalities of the dynamics among environments while design an environment encoder that learns environment-specific representations for exogenous factors automatically from observed trajectories. To disentangle the representations from the initial state encoder and the environment encoder, we propose a regularization loss via mutual information minimization to guide the learning process. We additionally design a contrastive learning loss to reduce the variance of learned exogenous factors across time windows under the same environment. The proposed model is able to achieve accurate predictions for varying



(a) Exogenous Factors Across Environments



(b) Exogenous Factors from two Environments

Figure 5.6: T-SNE visualization of the learned exogenous factors on the Lennard-Jones potential dataset. (a) We randomly pick one data sample per temperature, where temperatures tested in the inductive setting are circled in black. (b) Visualization of data samples from two temperatures.

physical systems under different environments, especially for long-term predictions. There are some limitations though. Our current model only learns one static environment-specific variable to achieve model generalization. However, the environment can change over time such as temperatures. How to capture the dynamic influence of those evolving environments remain challenging.

CHAPTER 6

SS-AGA: Multilingual Knowledge Graph Completion with Self-Supervised Adaptive Graph Alignment

Predicting missing facts in a knowledge graph (KG) is crucial as modern KGs are far from complete. Due to labor-intensive human labeling, this phenomenon deteriorates when handling knowledge represented in various languages. In this paper, we explore multilingual KG completion, which leverages limited seed alignment as a bridge, to embrace the collective knowledge from multiple languages. However, language alignment used in prior works is still not fully exploited: (1) alignment pairs are treated equally to maximally push parallel entities to be close, which ignores KG capacity inconsistency; (2) seed alignment is scarce and new alignment identification is usually in a noisily unsupervised manner. To tackle these issues, we propose a novel self-supervised adaptive graph alignment (SS-AGA) method. Specifically, SS-AGA fuses all KGs as a whole graph by regarding alignment as a new edge type. As such, information propagation and noise influence across KGs can be adaptively controlled via relation-aware attention weights. Meanwhile, SS-AGA features a new pair generator that dynamically captures potential alignment pairs in a self-supervised paradigm. Extensive experiments on both the public multilingual DBPedia KG and newly-created industrial multilingual E-commerce KG empirically demonstrate the effectiveness of SS-AGA[1].

6.1 Introduction

Knowledge graphs (KGs) like Freebase [BEP08] and DBPedia [LIJ15] are essential for various knowledge-driven applications such as question answering [YRB21] and commonsense reasoning [LSD21]. A KG contains structured and semantic information among entities and relations, where prior knowledge can be instantiated as factual triples (head entity, relation, tail entity), e.g., (*Apple Inc.*, *Founded by*, *Steven Jobs*). As new facts are continually emerging, modern KGs are still far from being complete due to the high cost

[1]Code and data are open-source and available at `https://github.com/amzn/ss-aga-kgc`

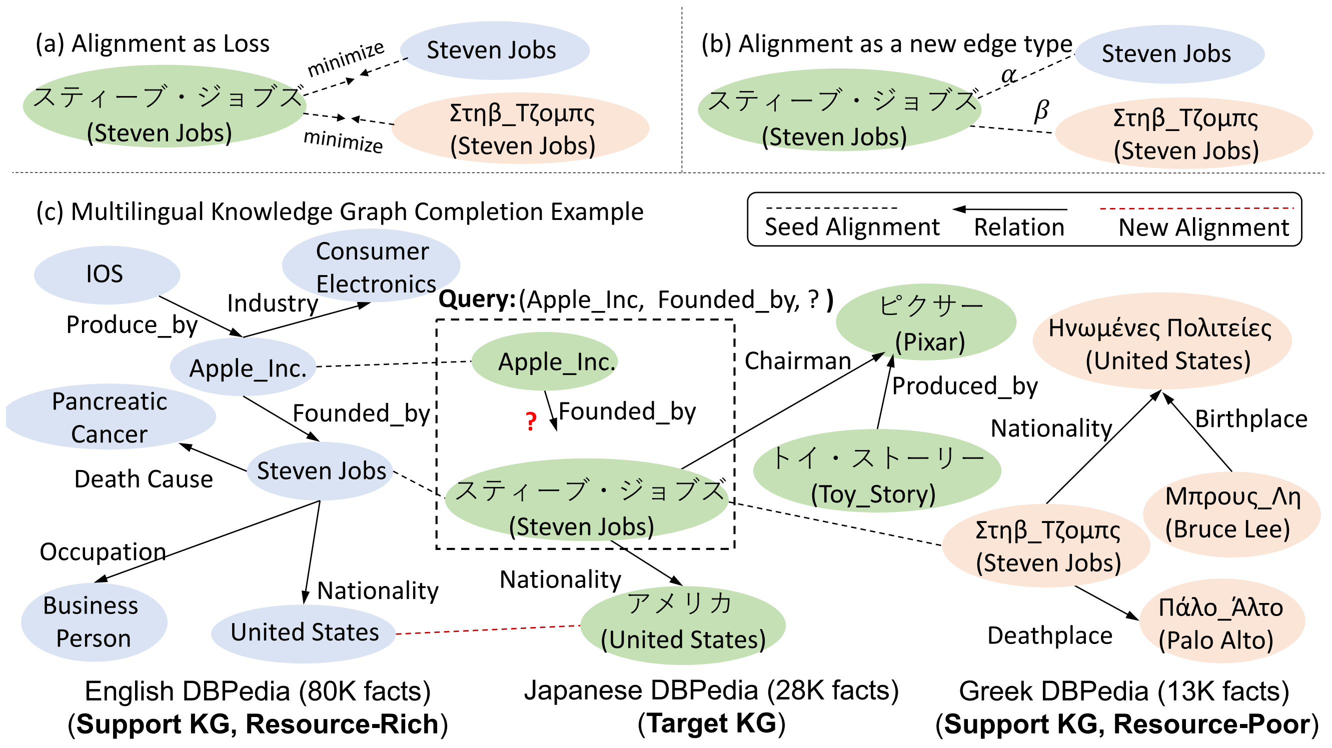

(a) Alignment as Loss
Steven Jobs
minimize
スティーブ・ジョブズ (Steven Jobs)
minimize
Στηβ_Τζομπς (Steven Jobs)
(b) Alignment as a new edge type
Steven Jobs
α
β
スティーブ・ジョブズ (Steven Jobs)
Στηβ_Τζομπς (Steven Jobs)
(c) Multilingual Knowledge Graph Completion Example
Seed Alignment
Relation
New Alignment
IOS
Consumer Electronics
Industry
Produce_by
Apple_Inc.
Pancreatic Cancer
Founded_by
Death Cause
Steven Jobs
Occupation
Business Person
Nationality
United States
Query:(Apple_Inc, Founded_by, ?)
Apple_Inc.
?
Founded_by
スティーブ・ジョブズ (Steven Jobs)
Nationality
アメリカ (United States)
Chairman
ピクサー (Pixar)
Produced_by
トイ・ストーリー (Toy_Story)
Ηνωμένες Πολιτείες (United States)
Nationality
Birthplace
Μπρους_Λη (Bruce Lee)
Στηβ_Τζομπς (Steven Jobs)
Deathplace
Πάλο_Άλτο (Palo Alto)
English DBPedia (80K facts)
(Support KG, Resource-Rich)
Japanese DBPedia (28K facts)
(Target KG)
Greek DBPedia (13K facts)
(Support KG, Resource-Poor)

Figure 6.1: (a) Existing methods treat alignment pairs equally as a loss, which maximally ensures the same entity from different languages to be as similar as possible. (b) Our method differentiates alignment pairs as a new type edge with dynamic attention weights such as α and β, which control the influence and information propagation from other support KGs. (c) An example of MKGC task answering the query in the Japanese KG.

of human annotation, which spurs on the Knowledge Graph Completion (KGC) task to automatically predict missing triples to complete the knowledge graph.

The KG incompletion circumstance is exacerbated in the multilingual setting, as human annotations are rare and difficult to gather, especially for low-resource languages. Unfortunately, most efforts for KGC have been devoted to learning each monolingual KG separately [PCL21, XZK21, LYL21, CJL21, LNV21], which usually underperform in low-resource language KGs that suffer from the sparseness [CTY17, CCF20, SZH20]. In contrast, KGs from multiple languages are not naturally isolated, which usually share some real-world entities and relations. The transferable knowledge can be treated as a bridge to align different KGs, which not only facilitates the knowledge propagation to low-resource KGs but also alleviates costly manual labeling for all languages.

In this paper, we explore multilingual KG completion (MKGC) [CCF20] with limited seed alignment across languages. To mitigate language gaps, some efforts have been initiated on multilingual KG embedding methods, which leverage a KG embedding module (e.g., TransE [BUG13]) to encode each language-specific KG independently and then employ an alignment loss to force pairs of aligned entities to be close maximally [CCF20, ZSH19, SZH20]. However, such approaches mainly involve two limitations: (1) the KG inconsistency issue among different languages is neglected due to the equal treatment for parallel entities; (2) the scarcity of seed alignment hinders the efficient knowledge transfer across languages.

Concretely, prior methods treat all alignment pairs equally by forcing all parallel entities to be maximally close to each other [CTC18, SHZ18, CTY17]. This ignores potentially negative effects from the KG inconsistency due to the language diversity. For example, as shown in Figure 6.1, the support English KG in DBP-5L [CCF20] has much more enriched knowledge (80K facts) than the Greek one (13K facts). In order to complete the query (*Apple Inc.*, *Founded by*, ?) in the resource-poor Japanese KG (28K facts), we can transfer more knowledge from resource-rich English KG through the alignment link of *Steven Jobs* than that of the low-data Greek. However, if roughly pushing *Steven Jobs* to be equally close to that English KG and Greek KG, the learned embeddings for *Steven Jobs* will be similar even though they have different structures, KG capacity, coverage and quality. As such, it will bring in irrelevant information regarding this query and may cause the model to get the wrong answer. Thus, we encourage the model to automatically distinguish the underlying inconsistency and transfer knowledge from suitable support KGs[2] for better language-specific KGC performance.

One the other hand, seed alignment is critical for cross-lingual transfer [CCF20, SZH20], while acquisition of such parallel entities across languages is costly and often noisy. To mitigate such issue, some recent works [CTC18, CCF20] propose to generate new alignment pairs based on the entity embedding similarity during the training process. The generated new pairs can increase the inter-connectivity between KGs to facilitate knowledge transfer. However, simple usage of correlations between entities without any supervision may increase the noise during training, and inhibit the effectiveness of realistic language alignment in KGs [SZH20].

Motivated by these observations, we propose a **S**elf-**S**upervised **A**daptive **G**raph **A**lignment (**SS-AGA**) framework for MKGC. To tackle the knowledge inconsistency issue, SS-AGA regards alignment as a new edge type between parallel entities instead of a loss constrain, which fuses KGs from different

[2]We regard the remaining KGs as the support KGs when conducting the KGC task in the target one.

languages as a whole graph. Based on such unified modeling, we propose a novel GNN encoder with a relation-aware attention mechanism, which aggregates local neighborhood information with learnable attention weights and differs the influence received from multiple alignment pairs for the same entity as shown in Figure 6.1(b). To alleviate the scarcity of seed alignment, SS-AGA exploits a new pair generator that iteratively identifies new alignment pairs in a self-supervised manner. This is achieved by masking some seed alignment in the fused KG before GNN encoding and teaching the generation module to recover them. Empirically, SS-AGA outperforms popular baselines in both public and industrial datasets. For the public dataset, we use the multilingual DBPedia KG [CCF20] and for the industrial dataset, we create a multilingual E-commerce Product KG called E-PKG.

Our contributions are as follows: (1) We handle the knowledge inconsistency issue for MKGC by treating entity alignment as a new edge type and introducing a relation-aware attention mechanism to control the knowledge propagation; (2) We propose a new alignment pair generation mechanism with self-supervision to alleviate the scarcity of seed alignment; (3) We constructed a new industrial-level multilingual E-commerce KG dataset; (4) Extensive experiments verify the effectiveness of SS-AGA in both public and industrial datasets.

6.2 Preliminaries

6.2.1 Knowledge Graph Completion

A knowledge graph $G = (\mathcal{E}, \mathcal{R}, \mathcal{T})$ consists of a set of entities $\mathcal{E}$, relations $\mathcal{R}$, and relational facts $\mathcal{T} = \{(e_h, r, e_t)\}$, where $e_h, e_t \in \mathcal{E}$ are head and tail entities, and $r \in \mathcal{R}$ is a relation. Entities and relations are represented by their text descriptions. The KG completion task seeks to impute the missing head or tail entity of a triple given the relation and the other entity. Without loss of generality, we hereafter discuss the case of predicting missing tails, which we also refer to as a query $q = (e_h, r, ?e_t)$.

Multilingual KG completion (MKGC) utilizes KGs across multiple languages to achieve more accurate KG completion task on each individual KG [CCF20]. Formally, we are given M different language-specific KGs as $G_1, G_2, \cdots, G_M$, and only limited entity alignment pairs $\Gamma_{G_i \leftrightarrow G_j} \subseteq \{(e_i, e_j) : e_i \in \mathcal{E}_i, e_j \in \mathcal{E}_j\}$ between G_i and G_j. We also call $\Gamma_{G_i \leftrightarrow G_j}$ the *seed* alignment pairs to distinguish it from the new or pseudo alignment. Each KG G_i has their own relation set $\mathcal{R}_i$. We denote the union of relation sets from all KGs as a unified relation set $\mathcal{R} = \mathcal{R}_1 \cup \mathcal{R}_2 \cup \cdot \cdot \mathcal{R}_M$. MKGC is related to but different from the

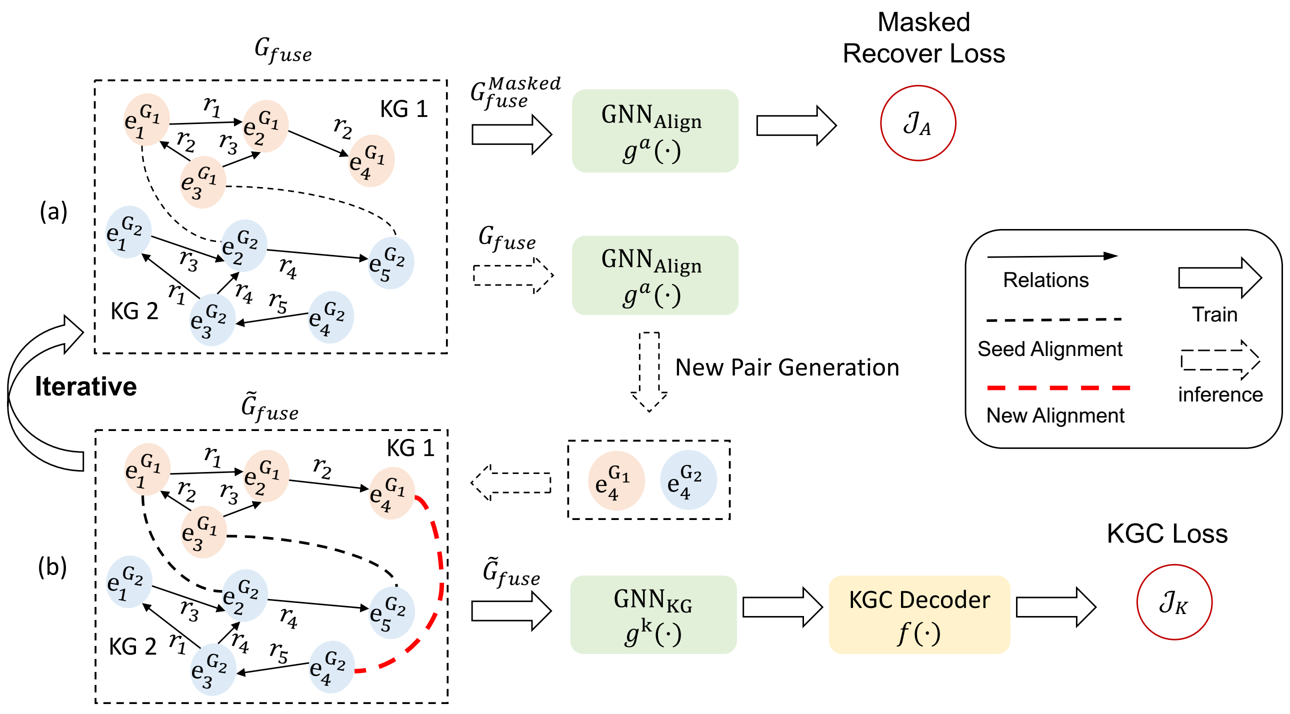


Figure 6.2: The overall framework of the Self-Supervised Adaptive Graph Alignment (SS-AGA).

entity alignment (EA) task [CLL19, SZH20]. In MKGC, seed alignment is not direct supervision while the auxiliary input features, all used in the training stage for cross-lingual transfer to boost the KGC results.

6.2.2 KG Embedding Models

KG embedding models aim to learn latent low-dimensional representations for entities $\{\boldsymbol{e}\}_{e\in\mathcal{E}}$ and relations $\{\boldsymbol{r}\}_{r\in\mathcal{R}}$. A naive implementation is an embedding lookup table [BUG13, SDN19]. Recently, Graph Neural Networks (GNN) have been explored to aggregate neighborhood information in KGs, where each triple is no longer considered independent of each other [HCY19]. Mathematically, these methods employ a GNN-based encoder g that embeds entities considering the neighborhood information,

$$\{\boldsymbol{e}\}_{e\in\mathcal{E}} = g(G).$$

Then, the plausibility of a relational fact (e_h, r, e_t) can be measured by the triple score:

$$f(\boldsymbol{e}_h, \boldsymbol{r}, \boldsymbol{e}_t),$$

where f can be any scoring function such as TransE [BUG13], RotatE [SDN19]. We also refer it to as the KGC decoder.

6.3 Method

We introduce SS-AGA for MKGC, consisting of two alternating training components (a) and (b) in Figure 6.2: (a) A new alignment pair generation module for alleviating the limited seed alignment in G_{fuse}. Specifically, we mask some seed alignment in the fuse KG to obtain $G_{\text{fuse}}^{\text{Masked}}$ and train the generator $g^a(\cdot)$ to recover them. Then, the trained generator will propose new edges based on the learned entity embeddings, which will be incorporated to G_{fuse} as $\widetilde{G}_{\text{fuse}}$ for MKG embedding model $g^k(\cdot)$ in the next iteration; (b) A novel relation-aware MKG embedding model $g^k(\cdot)$ for addressing the knowledge inconsistency across multilingual KGs. Specifically, we fuse different KGs as a whole graph G_{fuse} by treating alignment as a new edge type. Then $g^k(\cdot)$ computes the contextualized embeddings for each node with learnable relation-aware attention weights that differ the influence received from multiple alignment pairs. Finally, a KGC decoder $f(\cdot)$ computes the triple scores.

6.3.1 Relation-aware MKG Embedding

As mentioned before, the knowledge transfer is inefficient in existing MKGC methods, as they encode each KG separately and transfer knowledge by forcing aligned entities to share the same embedding. To handle the knowledge inconsistency, we first fuse all KGs as a whole, which relaxes the entity alignment to relational facts. We then design an attention-based relation-aware GNN to learn the contextualized MKG embeddings for entities, which can differ the influence from multiple alignment sources with learnable attention weights. Afterwards, we apply a KGC decoder on the contextualized embedding to get the triple scores for relational facts.

More specifically, we create the fused KG by preserving triples within each KG and converting each cross-KG alignment pair (e_i, e_j) to two relational facts $(e_i, r_{\text{align}}, e_j)$ and $(e_j, r_{\text{align}}, e_i)$ with the alignment edge as a newly introduced relation r_{align}. In this way, we enable direct message passing among entities from different KGs, where the attention weight can be learned automatically from data to differ the influence from multiple alignment pairs. We denote the fused knowledge graph as $G_{\text{fuse}} = (\mathcal{E}_{\text{fuse}}, \mathcal{R}_{\text{fuse}}, \mathcal{T}_{\text{fuse}})$, where $\mathcal{E}_{\text{fuse}} = \bigcup_{i=1}^{M} \mathcal{E}_i$, $\mathcal{R}_{\text{fuse}} = (\bigcup_{i=1}^{M} \mathcal{R}_i) \cup \{r_{\text{align}}\}$ and $\mathcal{T}_{\text{fuse}} = (\bigcup_{i=1}^{M} \mathcal{T}_i) \cup (\bigcup_{i,j}\{(e_h, r_{\text{align}}, e_t) : (e_h, e_t) \text{ or } (e_t, e_h) \in \Gamma_{G_i \leftrightarrow G_j}\})$.

Given the fused KG G_{fuse}, we propose an attention-based relation-aware GNN encoder $g^k(\cdot)$ to learn contextualized embeddings for entities following a multi-layer message passing architecture.

At the l-th layer of GNN, we first compute the relation-aware message delivered by the entity e_i in a relational fact (e_i, r, e_j) as follows:

$$\boldsymbol{h}_{i(r)}^l = Msg\left(\boldsymbol{h}_i^l, r\right) := \boldsymbol{W}_v^l Concat(\boldsymbol{h}_i^l, \boldsymbol{r}), \tag{6.1}$$

where $\boldsymbol{h}_i^l$ is the latent representation of e_i at the l-th layer, $Concat(\cdot, \cdot)$ is the vector concatenation function, and $\boldsymbol{W}_v^l$ is a transformation matrix. Then, we propose a relation-aware scaled dot product attention mechanism to characterize the importance of each entity's neighbor e_i to itself e_j, which is computed as follows:

$$\begin{aligned} Att\left(\boldsymbol{h}_{i(r)}^l, \boldsymbol{h}_j^l\right) &= \frac{\exp(\alpha_{ij}^r)}{\sum_{(e_{i'}, r) \in \mathcal{N}(e_j)} \exp\left(\alpha_{i'j}^r\right)} \\ \alpha_{ij}^r &= \left(\boldsymbol{W}_k^l \boldsymbol{h}_{i(r)}^l\right)^T \cdot \left(\boldsymbol{W}_q^l \boldsymbol{h}_j^l\right) \cdot \frac{1}{\sqrt{d}} \cdot \beta_r, \end{aligned} \tag{6.2}$$

where d is the dimension of the entity embeddings, $\boldsymbol{W}_k^l, \boldsymbol{W}_q^l$ are two transformation matrices, and β_r is a learnable relation factor. Different from the traditional attention mechanism [VCC18, BDB19], we introduce β_r to characterize the general significance of each relation r. It is essential as not all the relationships contribute equally to the query entity. We also remark that the neighborhood is bidirectional, i.e. $\mathcal{N}(e_j) := \{(e_{i'}, r) : (e_{i'}, r, e_j) \in \mathcal{T}_{\text{fuse}} \ or \ (e_j, r, e_{i'}) \in \mathcal{T}_{\text{fuse}}\}$ as the tail entity will also influence the head entity.

We then update the hidden representation of entities by aggregating the message from their neighborhoods based on the attention score:

$$\boldsymbol{h}_j^{l+1} = \boldsymbol{h}_j^l + \sigma\left(\sum_{(e_{i'}, r) \in \mathcal{N}(e_j)} Att\left(\boldsymbol{h}_{i'(r)}^l, \boldsymbol{h}_j^l\right) \cdot \boldsymbol{h}_{i'(r)}^l\right),$$

where $\sigma(\cdot)$ is a non-linear activation function, and the residual connection is used to improve the stability of GNN [HZR15].

Finally, we stack L layers to aggregate information from multi-hop neighbors and obtain the contextualized embedding for each entity e_j as: $\boldsymbol{e}_j = \boldsymbol{h}_j^L$. Given the contextualized entity embeddings, the KGC decoder computes the triple score for each relational fact: $f(\boldsymbol{e}_h, \boldsymbol{r}, \boldsymbol{e}_t)$. The learning object is to minimize the following hinge loss:

$$\mathcal{J}_K = \sum_{\substack{(e_h,r,e_t)\in\mathcal{T}_m \\ (e_{h'},r,e_{t'})\notin\mathcal{T}_m \\ m=1,\ldots,M}} \left[f\left(\boldsymbol{e}_h{}', \boldsymbol{r}, \boldsymbol{e}_t'\right) - f\left(\boldsymbol{e}_h, \boldsymbol{r}, \boldsymbol{e}_t\right) + \gamma \right]_+ , \tag{6.3}$$

where $\gamma > 0$ is a positive margin, f is the KGC decoder, $(e_{h'}, r, e_{t'})$ is a negative sampled triple obtained by replacing either head or tail entity of the true triple (e_h, r, e_t) randomly by other entities in the same language-specific KG.

Remark 1. Our method views cross-KG alignment as a relation r_{align} in the fused KG. The knowledge transfer cross KGs is essentially conducted via the learnable attention weight $\alpha_{ij}^{r_{align}}$, where e_i and e_j are connected through the relation r_{align}. Thanks to the power of GNN, $\alpha_{ij}^{r_{align}}$ differs the influence from multiple alignment sources, as opposed to some existing models that simply force pairs of entities to be close to each other through a pre-defined alignment loss. In this way, we properly conduct knowledge transfer among KGs with aware of their knowledge inconsistency.

Scalability issue. Since we fuse all the M KGs as a whole, and duplicate edges for head entities, the scale of the graph G_{fuse} would become very large. We therefore employ a k-hop graph sampler that samples the k-hop neighbors for each node and compute their contextualized embeddings.

6.3.2 Self-supervised New Pair Generation

In multilingual KGs, we are only provided with limited seed alignment pairs to facilitate knowledge transfer, as they are expensive to obtain and even sometimes noisy [SZH20]. To tackle such challenge, we propose a self-supervised new alignment pair generator. In each iteration, the generator identifies new alignment pairs which will be fed into the GNN encoder $g^k(\cdot)$ to produce the contextualized entity embeddings in the next iteration. The training of the generator is conducted in a self-supervised manner, where the generator is required to recover masked alignment pairs.

New Pair Generation (NPG) relies on two sets of entity embeddings: the structural embeddings and the textual embeddings. The structural embeddings are obtained by another GNN encoder g^a: $\{\boldsymbol{e}^a\}_{e\in\mathcal{E}_{fuse}} = g^a(G_{fuse})$, which shares the same architecture with $g^k(\cdot)$ in the relation-aware MKG Embedding model (Section 6.3.1). The reason we employ two GNN encoders is that the set of embeddings that generate the best alignment results may differ from those that can best achieve the KG completion task.

The textual embeddings are obtained by entities' text description and mBERT: $\boldsymbol{e}^{text} = mBERT(e)$. mBERT is a multilingual pre-trained language model [DCL19] and is particularly attractive to the new alignment pair generation due to the following merits: (1) it captures rich semantic information of the text; (2) the pre-trained BERT embeddings are also aligned across different languages [DCL19, SZH20].

We then model the pairwise similarity score between entity e_i and e_j as the maximum of the cosine similarities of their structural embeddings and textual embeddings:

$$sim(e_i, e_j) = \max\left(cos\left(\boldsymbol{e}_i^a, \boldsymbol{e}_j^a\right), cos\left(\boldsymbol{e}_i^{text}, \boldsymbol{e}_j^{text}\right)\right).$$

Then we introduce new alignment pairs if a pair of unaligned entities in two KGs are mutual nearest neighbors according to the cross-domain similarity local scaling (CSLS) measure [CLR18] as shown below,

$$\begin{aligned} CSLS(e_i, e_j) &= 2sim(e_i, e_j) - s(e_i) - s(e_j) \\ subject\ to\ s\left(e_i\right) &= \frac{1}{K} \sum_{e_{i'} \in \mathcal{N}(e_i)} sim\left(e_i, e_{i'}\right), \end{aligned}$$

where K is the number of each node's k-nearest neighbors. CSLS is able to capture the sturctural similarity between pairs of entities. The generated pairs are then utilized to update the graph structure of G_{fuse} to $\widetilde{G}_{\text{fuse}}$ in the next iteration, to alleviate the challenge of limited seed alignment.

Self-Supervised Learning (SSL) Similar to many existing works [CCF20, SZH20], the aforementioned NPG paradigm is unsupervised and may bring in unexpected noises. Inspired by masked language modeling [DCL19] which captures contextual dependencies between tokens, we propose a self-supervised learning procedure to guide and denoise the new pair generation. Specifically, we randomly mask out some alignment relational facts, $\mathcal{T}_{masked} \subseteq \{(e_h, r, e_t) \in \mathcal{T}_{fuse} : r = r_{align}\}$, and let the generator to recover them. Such masked alignment recovery in KGs can automatically identify the underlying correlations for alignment neighbors and encourage the NPG to generate high-quality alignment pairs that are real existences but hide due to the limited seed alignment.

Given the fused KG with masked alignment $G_{\text{fuse}}^{\text{Masked}} = \{\mathcal{E}_{fuse}, \mathcal{R}_{fuse}, \mathcal{T}_{fuse}/\mathcal{T}_{masked}\}$, the GNN encoder g^a embeds the entities as

$$\{\widetilde{\boldsymbol{e}}\}_{e \in \mathcal{E}_{fuse}} = g^a(G_{\text{fuse}}^{\text{Masked}}).$$

The GNN g^a is then trained via minimizing the following hinge loss $\mathcal{J}_A$,

$$\mathcal{J}_A^{G_i \leftrightarrow G_j} = \sum_{\substack{(e_h, e_t) \in \Gamma_{ij}^p \\ (e_{h'}, e_{t'}) \in \Gamma_{ij}^n}} [\|\widetilde{\boldsymbol{e}}_h^a - \widetilde{\boldsymbol{e}}_t^a\|_2 - \|\widetilde{\boldsymbol{e}}_{h'}^a - \widetilde{\boldsymbol{e}}_{t'}^a\|_2 + \gamma_a]_+$$
$$\mathcal{J}_A = \sum_{1 \le i < j \le M} \mathcal{J}_A^{G_i \leftrightarrow G_j}, \tag{6.4}$$

where $\Gamma_{ij}^p = \{(e_h \in \mathcal{E}_i, e_t \in \mathcal{E}_j) : (e_h, r_{align}, e_t) \in \mathcal{T}_{masked}\}$ is the masked alignment set, $\Gamma_{ij}^n = \{(e_h \in \mathcal{E}_i, e_t \in \mathcal{E}_j) : (e_h, e_t) \notin \Gamma_{G_i \leftrightarrow G_j}\}$ is the unaligned entity pair set, and $\gamma_a > 0$ is a positive margin. $(e_{h'}, e_{t'})$ is randomly sampled by replacing one of the entities in the positive entity pairs.

6.3.3 Training

The overall loss function is the combination of the KG completion loss Eq. (6.3) and the self-supervised alignment loss Eq. (6.4) as shown below

$$\mathcal{J} = \mathcal{J}_K + \lambda \mathcal{J}_A, \tag{6.5}$$

where $\lambda > 0$ is a positive hyperparameter to balance between the two losses. We summarize the training process in the Appendix.

6.4 Experiments

6.4.1 Dataset

We conduct experiments over two real-world datasets. (i) **DBP-5L** [CCF20] contains five language-specific KGs from DBpedia [LIJ15], i.e., English (EN), French (FR), Spanish (ES), Japanese (JA), Greek (EL). As the original dataset only contains structural information, we additionally crawled the text information for these entities and relations based on the given URLs. (ii) **E-PKG** is a new industrial multilingual E-commerce product KG dataset, which describes phone-related product information from an E-commerce platform across six different languages: English (EN), German (DE), French (FR), Japanese (JA), Spanish (ES), Italian (IT). The statistics are shown in Table 6.1. The $\#$ Aligned Links for a specific KG G_i denotes the number of alignment pairs where one of the aligned entities belong to that KG. It is possible for an entity to have multiple alignment pairs across different KG sources. For both datasets, we randomly split the facts in each KG into three parts: 60% for training, 30% for validation, and 10% for testing. Please refer to Appendix for the details of E-PKG construction.

Dataset	#Entity	#Relation	#Triple	#Aligned Links
Multilingual Academic KG (*DBP-5L*)				
EN	13,996	831	80,167	16,916
FR	13,176	178	49,015	16,877
ES	12,382	144	54,066	16,347
JA	11,805	128	28,774	16,263
EL	5,231	111	13,839	9,042
Multilingual Industrial KG (*E-PKG*)				
EN	16,544	21	100,531	21,382
DE	17,223	21	75,870	24,696
FR	17,068	21	80,015	24,812
JA	2,642	21	16,703	5,175
ES	9,595	21	30,163	20,184
IT	15,670	21	71,292	23,827

Table 6.1: Statistics of DBP-5L and E-PKG datasets. #Aligned Links denotes the number of alignment pairs where one of the aligned entities belongs to that KG.

6.4.2 Evaluation Protocol

In the testing phase, given each query $(e_h, r, ?e_t)$, we compute the plausibility scores $f(e_h, r, \widetilde{e}_t)$ for triples formed by each possible tail entity $\widetilde{e}_t$ in the test candidate set and rank them. We report the mean reciprocal ranks (MRR), accuracy (Hits@1) and the proportion of correct answers ranked within the top 10 (Hits@10) for testing. We also adopt the filtered setting following previous works based on the premise that the candidate space has excluded the triples that have been seen in the training set [WZF14a, YYH15a].

6.4.3 Baselines

• **Monolingual Baselines.** (i) **TransE** [BUG13] models relations as translations in the Euclidean space; (ii) **RotatE** [SDN19] models relations as rotations in the complex space; (iii) **DisMult** [YYH15b] uses a simple bilinear formulation; (iv) **KG-BERT** [YML20] employs pre-trained language models for knowledge graph completion based on text information of relations and entities.

• **Multilingual Baselines.** (i) **KEnS** [CCF20] embeds all KGs in a unified space and exploits an ensemble technique to conduct knowledge transfer; (ii) **CG-MuA** [ZWS20] is a GNN-based KG alignment model with collective aggregation. We revise its loss function to conduct MKGC. (iii) **AlignKGC** [SJC21] jointly trains the KGC loss with entity and relation alignment losses. For fair comparison, we use mBERT [DCL19] to obtain initial embeddings of entities and relations from their text for all methods. We do not employ any pretrained tasks such as EA to obtain these initial text embeddings as in [SJC21].

6.4.4 Main Results

The main results are shown in Table 6.2 and Table 6.3. Firstly, by comparing multilingual and monolingual KG models, we can observe that multilingual methods can achieve better performance. This indicates that the intuition behind utilizing multiple KG sources to conduct KG completion is indeed beneficial, compared with inferring each KG independently. Notably, multilingual models tend to bring larger performance gains for those low-resource KGs such as Greek in DBP-5L, which is expected as low-resource KGs are far from complete and efficient external knowledge transfer can bring in potential benefits. Among multilingual models, our proposed method SS-AGA can achieve better performance in most cases across different metrics, languages, and datasets, which verifies the effectiveness of SS-AGA.

6.4.5 Ablation Study

To evaluate the effectiveness of our model design, we conduct ablation study by proposing the following model variants: (i) **GNN** applies the GNN encoder without relation modeling to each KG independently, and directly forces all alignment pairs to be close to each other as in prior works [CCF20, ZWS20]; (ii) **R-GNN** is the proposed relation-aware MKG embedding model (Section 6.3.1), which utilizes all seed alignment to construct G_{fused} and differs the influence from other KGs by the relation-aware attention mechanism; (iii) **R-GNN + NPG** conducts additional new pair generation for R-GNN; (iv) **R-GNN + NPG + SSL** is our proposed full model SS-AGA, which leverages SSL to guide the NPG process. We also investigate the effect of whether to share or not share the encoders $g^a(\cdot), g^k(\cdot)$ that generate the embeddings for the SSL and KGC loss, respectively.

We report the average Hits@1, Hits@10 and MRR over DBP-5L as shown in Table 6.4. As we can see, applying a GNN encoder to each KG independently would cause the performance drop as all aligned entities are being equally forced to be close to each other. Removing the new pair generation process

Method	Metric	EL	JA	ES	FR	EN
		Monolingual Baselines				
TransE	H@1	13.1	21.1	13.5	17.5	7.3
	H@10	43.7	48.5	45.0	48.8	29.3
	MRR	24.3	25.3	24.4	27.6	16.9
RotatE	H@1	14.5	26.4	21.2	23.2	12.3
	H@10	36.2	60.2	53.9	55.5	30.4
	MRR	26.2	39.8	33.8	35.1	20.7
DisMult	H@1	8.9	9.3	7.4	6.1	8.8
	H@10	11.3	27.5	22.4	23.8	30.0
	MRR	9.8	15.8	13.2	14.5	18.3
KG-BERT	H@1	17.3	26.9	21.9	23.5	12.9
	H@10	40.1	59.8	54.1	55.9	31.9
	MRR	27.3	38.7	34.0	35.4	21.0
		Multilingual Baselines				
KenS	H@1	28.1	32.1	23.6	25.5	15.1
	H@10	56.9	65.3	60.1	62.9	39.8
	MRR	-	-	-	-	-
CG-MuA	H@1	21.5	27.3	22.3	24.2	13.1
	H@10	44.8	61.1	55.4	57.1	33.5
	MRR	32.8	40.1	34.3	36.1	22.2
AlignKGC	H@1	27.6	31.6	24.2	24.1	15.5
	H@10	56.3	64.3	60.9	62.3	39.2
	MRR	33.8	41.6	35.1	37.4	22.3
SS-AGA	H@1	**30.8**	**34.6**	**25.5**	**27.1**	**16.3**
	H@10	**58.6**	**66.9**	**61.9**	**65.5**	**41.3**
	MRR	**35.3**	**42.9**	**36.6**	**38.4**	**23.1**

Table 6.2: Main results on DBP-5L.

Method	Metric	EN	DE	FR	JA	ES	IT
Monolingual Baselines							
TransE	H@1	23.2	21.2	20.8	25.1	17.2	22.0
	H@10	67.5	65.5	66.9	72.7	58.4	63.8
	MRR	39.4	37.4	37.5	43.6	33.0	37.8
RotatE	H@1	24.2	22.3	22.1	26.3	18.3	22.5
	H@10	66.8	64.3	67.1	71.9	58.9	64.0
	MRR	40.0	38.2	38.0	41.8	33.7	38.1
DisMult	H@1	23.8	21.4	20.7	25.9	17.9	22.8
	H@10	60.1	54.5	53.5	62.6	46.2	51.8
	MRR	37.2	35.4	35.1	38.0	30.9	34.8
KG-BERT	H@1	24.3	21.8	22.3	26.9	18.7	22.9
	H@10	66.4	64.7	67.2	72.4	58.8	63.7
	MRR	39.6	38.4	38.3	44.1	33.2	37.2
Multilingual Baselines							
KenS	H@1	26.2	24.3	25.4	33.5	**21.3**	**25.1**
	H@10	69.5	65.8	68.2	73.6	59.5	**64.6**
	MRR	-	-	-	-	-	-
CG-MuA	H@1	24.8	22.9	23.0	30.4	19.2	23.9
	H@10	67.9	64.9	67.5	72.9	58.8	63.8
	MRR	40.2	38.7	39.1	45.9	33.8	37.6
AlignKGC	H@1	25.6	22.1	22.8	31.2	19.4	24.2
	H@10	68.3	65.1	67.2	72.3	59.1	63.4
	MRR	40.5	38.5	38.8	46.2	34.2	37.3
SS-AGA	H@1	**26.7**	**24.6**	**25.9**	**33.9**	21.0	24.9
	H@10	**69.8**	**66.3**	**68.7**	**74.1**	**60.1**	63.8
	MRR	**41.5**	**39.4**	**40.2**	**48.3**	**36.3**	**38.4**

Table 6.3: Main results on E-PKG.

would also cause a performance degradation due to the sparsity of seed alignment, which shows that iteratively proposing new alignment is indeed helpful. If the generation process is further equipped with supervision, the performance would be enhanced, which verifies the effectiveness of the self-supervised alignment loss. Finally, sharing the parameters of two GNN encoders would harm the performance. Though MKGC and entity alignment are two close-related tasks that can potentially benefit each other, the set of embeddings that produce the best alignment result do not necessarily yield the best performance on the MKGC task.

Method	Avg H@1	Avg H@10	Avg MRR
GNN	24.1	56.3	33.2
R-GNN	25.7	57.9	34.4
R-GNN + NPG	26.2	58.3	34.9
R-GNN + NPG + SSL (SS-AGA)			
- encoder (shared)	25.8	57.7	34.1
- encoder (no shared)	**26.9**	**58.7**	**35.3**

Table 6.4: Ablation results on DBP-5L.

6.4.6 Impact of Seed Alignment

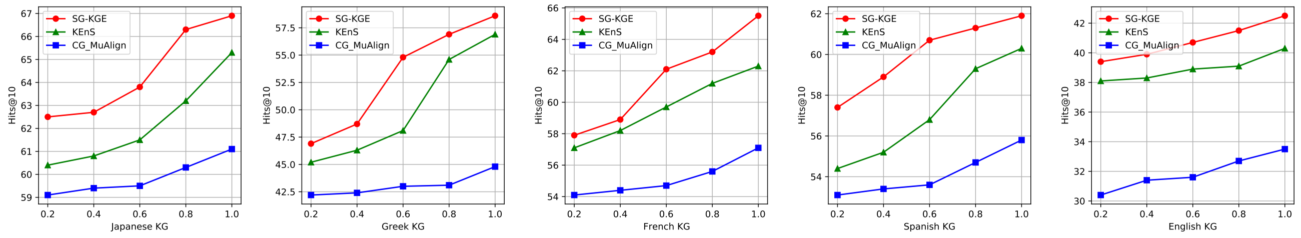


Figure 6.3: Hits@10 with respect to different sampling ratio of seed alignment pairs.

We next study the effect of seed alignment number as depicted in Figure 6.3. Firstly, we can observe that SS-AGA consistently outperforms other multilingual models on varying alignment ratios. Secondly, for low-resources KGs such as Japanese and Greek KGs, we can observe a sharp performance drop when decreasing the alignment ratio compared with those popular KGs such as English KG. This indicates that the knowledge transfer among different KGs is especially beneficial for those low-resources KGs, as popular KGs already contain relatively rich knowledge. However, such transfer process is heavily dependent on the seed alignment, which yields the necessity of new alignment generation process.

6.4.7 Case Study

To interpret the knowledge transfer across different KGs, we visualize the normalized average attention weight for each KG w.r.t. the attention score computed in Eq. (6.2) from different KG sources. We can see that for those popular KGs, they will receive the highest attention score from themselves such as English and French KGs. Although Japanese KG is low-resource, from the main results table 6.2, we can see that the gap improvement brought by multilingual methods is relatively small compared to another low-resource Greek KG. This indicates that Japanese KG may contain more reliable facts to facilitate missing triple predictions. However, for Greek KG, we can observe that the attention weights from other languages take the majority, which means that the performance boost in Greek KG is largely attributed to the efficient knowledge transfer from other KG sources.

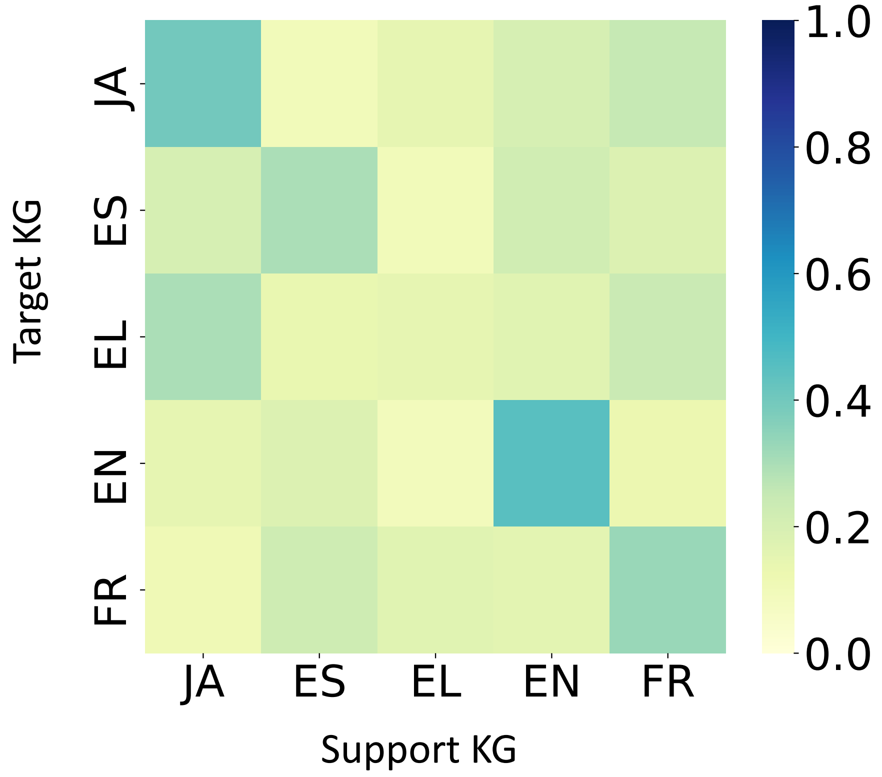


Figure 6.4: Average attention weight learned in DBP-5L.

6.5 Related Work

6.5.1 Monolingual KG Embeddings

Knowledge graph embeddings [BUG13, SDN19, Con18] achieve the state-of-the-art performance for KGC, which learn the latent low-dimensional representations of entities and relations. They measure triple plausibility based on varying score functions such as translation-based TransE [BUG13], TransH [WZF14b]; rotation-based RotatE [SDN19] and language-model-based KG-BERT [YML20]. Recently, GNN-based methods [LCH19, ZZZ20, JHH20] have been proposed to capture node neighborhood information for the KGC tasks. GNN is a class of neural networks that operate on graph-structured data by passing local messages [KW17, VCC18, XHL19, BDB19, HSW20b, HSW21b, WHL21a]. Specifically, they use GNN as an encoder to generate contextualized representation of entities by passing local messages [KW17, VCC18, XHL19, BDB19, HSW20b, HSW21b]. Then, existing score functions are employed to generate triple scores which outperform the aforementioned methods that treat each triple independently only with the scoring function.

6.5.2 Multilingual KG Embeddings

Multilingual KG embeddings are extensions of monolingual KG embeddings that consider knowledge transfer across KGs with the use of limited seed alignment [SZH20, SJC21]. Earlier work proposes different ways to reconcile KG embeddings for the **entity alignment (EA)** task: MTransE [CTY17] learns a transformation matrix between pairs of KGs. MuGNN [CLL19] reconciles structural differences via rule grounding. CG-MuA utilizes collective aggregation of confident neighborhood [ZWS20]. Others incorporate attribute information such as entity text [ZSH19, CTC18]. To tackle the sparsity of seed alignment, BootEA [SHZ18] iteratively proposes new aligned pairs via bootstrapping. [ZXL17] utilizes parameter sharing to improve alignment performance. While they focus on the EA task rather than the MKGC task that we tackle here, such techniques can be leveraged to conduct knowledge transfer among KGs. Recently, [CCF20] propose an ensemble-based approach for the MKGC task. In this paper, we view alignment as a new edge type and employ a relation-aware GNN to get the contextualized representation of entities. As such, the influence of the aligned entities is captured by the learnable attention weight, instead of assuming each alignment pair to have the same impact. We also propose a self-supervised learning task to propose new alignment pairs during each training epoch to overcome the sparsity issue of seed alignment pairs.

6.6 Discussion and Conclusion

In this paper, we propose SS-AGA for multilingual knowledge graph completion (MKGC). It addresses the knowledge inconsistency issue by fusing all KGs and utilizing a GNN encoder to learn entity embeddings with learnable attention weights that differs the influence from multiple alignment sources. It features a new pair generation conducted in a self-supervised learning manner to tackle the limited seed alignment issue. Extensive results on two real-world datasets including a newly-created E-commerce dataset verified the effectiveness of SS-AGA. Our current approach may fail to fully exploit the benefit of entity and relation texts. In the future, we plan to study more effective ways to combine text data with graph data for better model performance. We are also interested in studying MKGC where there no alignment pairs are given, which is a very practical setting and our current model is not able to deal with.

Part III

GraphODE for Causal Decision-Making

CHAPTER 7

CAG-ODE: Coupled GraphODE

Real-world multi-agent systems are often dynamic and continuous, where the agents co-evolve and undergo changes in their trajectories and interactions over time. For example, the COVID-19 transmission in the U.S. can be viewed as a multi-agent system, where states act as agents and daily population movements between them are interactions. Estimating the counterfactual outcomes in such systems enables accurate future predictions and effective decision-making, such as formulating COVID-19 policies. However, existing methods fail to model the continuous dynamic effects of treatments on the outcome, especially when multiple treatments (e.g., "stay-at-home" and "get-vaccine" policies) are applied simultaneously. To tackle this challenge, we propose Causal Graph Ordinary Differential Equations (CAG-ODE), a novel model that captures the continuous interaction among agents using a Graph Neural Network (GNN) as the ODE function. The key innovation of our model is to learn time-dependent representations of treatments and incorporate them into the ODE function, enabling precise predictions of potential outcomes. To mitigate confounding bias, we further propose two domain adversarial learning-based objectives, which enable our model to learn balanced continuous representations that are not affected by treatments or interference. Experiments on two datasets (i.e., COVID-19 and tumor growth) demonstrate the superior performance of our proposed model. [1]

7.1 Introduction

Many real-world multi-agent systems are dynamic and continuous, where agents (nodes) interact and exhibit complex behaviors over time. This results in time-evolving node trajectories and dynamic interaction edges. An example is the spread of COVID-19 in the U.S., where states act as agents and daily migration patterns across states form interaction edges [HSW21c, MDH22]. Estimating the counterfactual outcomes over time in such systems are crucial for various applications, such as formulating effective

[1]Our code implementation can be found at `https://github.com/Jun-Kai-Zhang/CAG-ODE.git`.

policies and designing medical treatment plans [SIB22, BAJ20a, BV21]. This can achieve more accurate predictions than non-causal methods by considering the influence of biased confounders. Confounders are variables that have influences on treatments and outcomes. For example, the health status of the residents in each state (confounders) can impact their level of adherence to the state's policies (treatments), which can influence future confirmed cases/deaths (outcomes). Non-causal methods only learn the statistical associations between treatments and outcomes from observational data, which can have non-uniform treatment distributions across confounder values, potentially leading to incorrect predictions such as taking vaccines can increase the number of confirmed cases for each state. Furthermore, causal inference for multi-agent dynamical systems enables effective decision-making by addressing causal questions such as "What if we remove a policy at a specific time" or "What if we change the order of different policies". Therefore, it serves as a promising tool for policymakers.

Traditionally, the standard approach for causal inference over time is randomized controlled trials (RCTs) [CSB81], which can be very costly to obtain and can raise some ethical problems [SIB22, BAJ20a]. Thus, researchers have turned to using observational data and employed methods like linear regression [RHB00], recurrent neural networks (RNNs) [Lim18, BAJ20b], and Transformers [MFF22] to estimate counterfactual outcomes with time dependencies. However, causal inference for multi-agent dynamical systems presents unique challenges.

One is that most existing methods [SIB22, BAJ20a] assume that nodes are independent, meaning their trajectories are determined solely by their own treatments. Some [JHL23c] considers the influence of neighboring nodes but only assumes static interactions among them, which fails to capture situations such as daily population travel patterns between states in the context of COVID-19.

In casual terms, influences of neighboring nodes can be categorized into two parts: 1.) time-dependent neighborhood confounding, where a node's treatment and outcome may be confounded by the covariates of its neighbors. For example, if cases in neighboring states rise (covariate), a state may implement a vaccine policy (treatment) that affects future confirmed cases/deaths (outcome). 2.) time-dependent interference, where the outcome of a node can be influenced by the treatments of its neighbors. For example, a state may have reduced future cases/deaths (outcome) if neighboring states have implemented a vaccine policy (covariates), as higher vaccination rates within the population flow network give stronger protection. As the interaction edges evolve along with node trajectories, the challenges lie in predicting the neighbors of each node (edges) and then addressing the time-dependent neighborhood confounding and interference issues.

Another challenge is that current methods lack the ability to capture the continuous and dynamic effects of multiple treatments on such systems. For instance, the impact of a "stay-at-home" policy may be most significant during its initial implementation, and when a "get-vaccine" policy is subsequently introduced, the combined effect of these policies can result in a different outcome. Existing studies often focus on a single treatment [JHL23c, SIB22] or simply append fixed multi-hot treatment representations when a node receives them. These fixed treatment representations fail to differentiate the influences of the same treatment administered at different times.

To tackle these challenges, we propose a novel causal inference framework: the **Ca**usal **G**raph **O**rdinary **D**ifferential **E**quations (CAG-ODE) to estimate the continuous counterfactual outcome of a multi-agent dynamical system in the presence of multiple treatments and time-varying confounding and interference. Building upon the recent success of graph ordinary differential equations (ODE) in capturing the continuous interaction among agents [HSW21c, LWH23, HSW20b, HZG24], our key innovation is to learn time-dependent representations of simultaneous treatments and incorporate them into the ODE function to accurately account for their casual effects on the system. As nodes and edges are jointly evolving, we utilize two coupled treatment-induced ODE functions to account for their respective dynamics. To mitigate confounding bias, we further design two adversarial learning losses, which enable our model to learn balanced continuous trajectory representations unaffected by treatments or interference. Experiments on both real and simulated datasets demonstrate the effectiveness of our proposed model. The primary contributions of this paper can be summarized as follows:

- We propose CAG-ODE to estimate continuous counterfactual outcomes in multi-agent systems with evolving interaction edges and multiple treatments.
- CAG-ODE features a novel treatment fusing module that can capture the dynamic effects of treatment over time and the combined effect of multiple treatments.
- Our method achieves the state-of-art results in counterfactual estimation across varying systems, and can serve as a promising tool for policymakers.

7.2 Preliminaries and Related Work

Graph Neural Networks (GNNs). Graph Neural Networks (GNNs) are a class of neural networks that operate on graph-structured data by passing local messages [KW17, VCC18, XHL19]. They have been extensively employed in various applications such as node classification, link prediction, and recommendation systems [HWH23, HLJ22]. GNNs have shown to be efficient for approximating pair-wise node interactions and achieved accurate predictions for multi-agent dynamical systems [KFW18a, SGP20b]. The majority of existing studies propose discrete GNN-based simulators where they take the node features at time t as input to predict the node features at time t+1. To further capture the long-term temporal dependency for predicting future trajectories, some work utilizes recurrent neural networks such as RNN, LSTM, or self-attention mechanism to make predictions at time t +1 based on the historical trajectory sequence [HHN19, SWG20b, HDW20b]. However, they restrict themselves to learning a one-step state transition function. Therefore, when we successively apply these one-step simulators to previous predictions in order to generate the rollout trajectories, error accumulates and impairs the prediction accuracy, especially for long-range prediction.

Graph Ordinary Differential Equations for Continuous Multi-agent Dynamical Systems. The dynamics of a multi-agent system can be captured by a series of nonlinear first-order ordinary differential equations (ODEs) [RCD19b, HSW20b, HSW23b, LYH23], which describe how the states of N dependent variables co-evolve over continuous time: $\dot{\boldsymbol{z}}_i^t := \frac{d\boldsymbol{z}_i^t}{dt} = g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right)$. Here $\boldsymbol{z}_i^t \in \mathbb{R}^d$ denotes the state variable for agent i at timestamp t and g denotes the ODE function that drives the system to move forward. Given the initial states $\boldsymbol{z}_1^0, \cdots \boldsymbol{z}_N^0$ for all agents and the ODE function g, a numerical ODE solver such as Runge-Kutta [MSH19] can be used to evaluate $\boldsymbol{z}_i^T$ at any desired time T using Eqn (7.1):

$$\boldsymbol{z}_i^T = \boldsymbol{z}_i^0 + \int_{t=0}^{T} g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right) \mathrm{d}t. \tag{7.1}$$

To model the interactions among agents, recent studies [HSW20b, HSW21c, ZW20, PMP19a] propose using a GNN as the ODE function g which is learned from observational data. Such GraphODE framework follows an encoder-processor-decoder architecture. The encoder computes latent initial states for all agents based on historical observations. The GNN-based ODE function then predicts the latent trajectories starting from the learned initial states. Finally, a decoder extracts the predicted dynamic features. To regularize the generated trajectories, GraphODE frameworks often adopt a variational autoencoder (VAE) structure [KW14], where the encoder samples initial states from approximated posterior distributions. GraphODEs are promising in making long-range predictions and can handle irregularly-sampled observations effectively [HSW20b, ZW20].

Causal Inference Over Time. Time-dependent causal inference methods mainly differ in how they deal with confounding. They differ from traditional statistical time series analysis [KFW18b, ZY23, BLZ22] which we do not discuss in this paper. Traditionally, many statistical tools that are applied, such as marginal structural models (MSMs) [RHB00] utilize the inverse probability of treatment weighting (IPTW). Recently, representation learning-based balancing approaches are proposed, which learn representations that are not predictable of the treatments to ensure unbiased outcome prediction [BAJ20b, MFF22]. However, one major limitation is that they are discrete methods, which can offer poor performance on continuous systems such as the spread of COVID-19. There are a series of works [SIB22, BAJ20a, GSP20, DGH22] that estimate the continuous counterfactual outcomes through neural ODEs or neural controlled differential equations (CDEs). Despite their success, they assume that nodes are independent of each other, regardless of their interactions. One recent work [JHL23c] proposed to parameterize the ODE function with a GNN for multi-agent settings. However, this model cannot handle evolving graph structures and the effect of multiple treatments.

7.3 Problem Definition

We consider a dynamical system of N agents as an evolving interaction graph $\mathcal{G}^t = \{\mathcal{V}, \mathcal{E}^t\}$, where nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ are agents and $\mathcal{E}^t$ are the weighted edges among them, denoting agents' dynamic interaction that changes over time. Each node is associated with time-varying causal characteristics, which we introduce in the following along with the casual inference framework.

We follow the longitudinal causal inference setting for predicting future potential outcomes as in [Rub78]. We denote the observational data at timestamp t as $(\mathbf{X}^t, \mathbf{W}^t, \mathbf{A}^t, \mathbf{Y}^t)$, where $\mathbf{X}^t \in \mathbb{R}^{N\times d_1}$ represents the time-varying covariates (e.g., the health status of residents) of N agents. $\mathbf{W}^t \in \mathbb{R}^{N\times N}$ represents the weighted adjacency matrix, whose element $w_{i\to j} \in \mathbb{R}$ is the weight of the directed edge that points from node i to node j and may be asymmetric. $\mathbf{A}^t \in \{0,1\}^{N\times K}$ are time-dependent treatments, where $\mathbf{A}^t_{kj} = 1$ denotes the k^{th} treatment assigned to node i at timestamp t, and K is the number of heterogeneous treatments. $\mathbf{Y}^t \in \mathbb{R}^{N\times d_2}$ is the time-dependent outcome, such as the number of confirmed cases in each state, which can be part of $\mathbf{X}^t$. The historical observations up to time t is represented as $\mathcal{H}^t = \left\{\overline{\mathbf{X}}^t, \overline{\mathbf{W}}^t, \overline{\mathbf{A}}^t, \overline{\mathbf{Y}}^t\right\}$, where $\overline{\mathbf{X}}^t, \overline{\mathbf{W}}^t, \overline{\mathbf{A}}^t, \overline{\mathbf{Y}}^t$ contain all $\mathbf{X}^{t^-}, \mathbf{W}^{t^-}, \mathbf{A}^{t^-}, \mathbf{Y}^{t^-}$ $(t^- \leq t)$. We aim to predict the unbiased potential outcomes $\mathbb{E}\big(\mathbf{Y}^{t^+}\big(\mathbf{A}^{t^+}=a\big)|\mathcal{H}^t\big)$ under any treatment assignment a[2]. Here, a is the dynamic treatment trajectory (e.g. sequences of state policies). As only one of the potential outcome trajectories is observed for each treatment assignment, we refer to the unobserved potential outcomes as counterfactuals [BAJ20b, SIB22].

To make potential outcomes identifiable from observational data, we follow three standard assumptions [BAJ20b, SIB22, JHL23c] below:

Assumption 1: Consistency. The potential outcome is equal to the observed factual outcome if $\mathbf{A}^t = a^t$: $\mathbf{Y}^{t^+}(\mathbf{A}^t = a^t) = \mathbf{Y}^{t^+}$.

Assumption 2: Overlap. At any time point t^+, there is some positive probability of treatment assignment regardless of the historical observation: $0 < P(\mathbf{A}^{t^+} = a \mid \mathcal{H}^t) < 1,\ \forall \mathcal{H}^t,\ t < t^+$.

The last assumption defines unconfoundedness (strong ignorability) in dynamical systems. We first define the interference effects caused by neighbors' treatments of node i as $\mathbf{G}_i^t = \sum_{j \in \mathcal{N}_i} \frac{1}{|N_i|}\mathbf{A}_j^t \in \mathbb{R}^K$, which is the proportion of treated nodes in node i's neighbors for each treatment type. We refer to $\mathbf{G}_i^t$ as interference summary, which assumes that a node is only influenced by treatments of its immediate neighbors as in previous studies [JHL23c, MWY22, JS22].

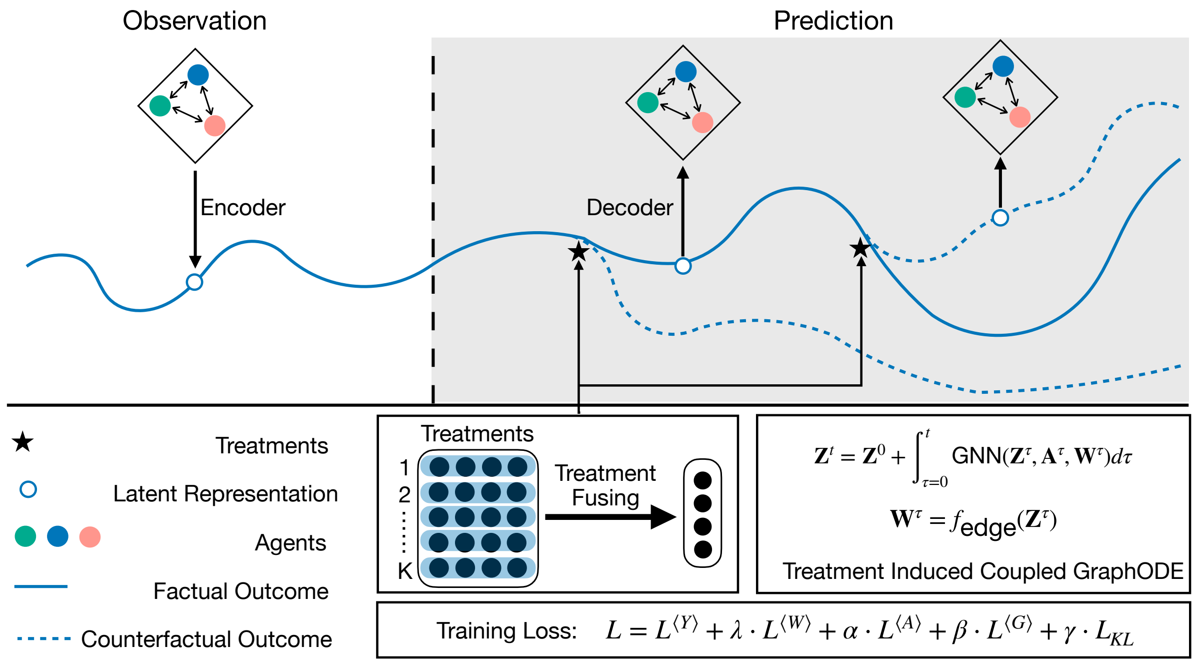
Assumption 3: Strong Ignorability for Multi-Agent Dynamical Systems. Given the historical observations, the potential outcome trajectory is independent of the treatments and interference summary: $\mathbf{Y}^{t^+}(\mathbf{A}^t = a) \perp \mathbf{A}^{t^+}, \mathbf{G}^{t^+} \mid \mathcal{H}^t,\ \forall a, t$.

It ensures that it is sufficient to only condition on the historical observations and graph sequences up to t to block all backdoor paths so as to estimate the potential outcome in the future. With these three assumptions, the potential outcome trajectory can be identified as:

$$\mathbb{E}\left(\mathbf{Y}^{t^+}(\mathbf{A}^t = a) \mid \mathcal{H}^t\right) = \mathbb{E}\left(\mathbf{Y}^{t^+} \mid \mathbf{A}^{t^+}, \mathbf{G}^{t^+}, \mathcal{H}^t\right).$$

This enables us to estimate the potential outcomes by training a machine learning model using observational data, and to use the same model to predict counterfactual outcomes given new treatment trajectories.

[2]The potential outcome can also be formalized using *do* operation [Pea09]



Observation
Prediction
Encoder
Decoder
Treatments
Latent Representation
Agents
Factual Outcome
Counterfactual Outcome
Treatments
1
2
K
Treatment Fusing
$\mathbf{Z}^t = \mathbf{Z}^0 + \int_{\tau=0}^{t} \mathrm{GNN}(\mathbf{Z}^\tau, \mathbf{A}^\tau, \mathbf{W}^\tau) d\tau$
$\mathbf{W}^\tau = f_{\mathrm{edge}}(\mathbf{Z}^\tau)$
Treatment Induced Coupled GraphODE
Training Loss: $L = L^{\langle Y \rangle} + \lambda \cdot L^{\langle W \rangle} + \alpha \cdot L^{\langle A \rangle} + \beta \cdot L^{\langle G \rangle} + \gamma \cdot L_{KL}$

Figure 7.1: Overall Framework of CAG-ODE. The encoder first computes the latent initial states. Then the treatment-induced coupled ODE functions predict the continuous trajectories over time. Treatment representations learned through the fusing module are incorporated into the ODE functions to enable counterfactual prediction. Finally, the decoder outputs the predicted dynamics. Treatment and interference balancing losses are designed to ensure unbiased counterfactual predictions.

7.4 The Proposed Model: CAG-ODE

In this section, we present Causal Graph ODE (CAG-ODE) to predict continuous counterfactual outcomes for multi-agent dynamical systems with evolving interaction edges and dynamic multi-treatment effects. Following the framework of GraphODEs [HSW20b, HSW21c, JHL23c, ZW20, PMP19a], CAG-ODE adopts the encoder-ODE generative model-decoder architecture as in [HSW21b] to capture the continuous interaction among agents. As nodes and edges are jointly evolving, we utilize two coupled ODE functions [HSW21c] for the evolution of nodes and edges respectively. Contrary to GraphODEs, CAG-ODE can perform causal reasoning by injecting treatment effects into the ODE functions, which we call *treatment-induced coupled graph ODE*. The multi-treatment effects are captured by a novel treatment fusing module that assigns temporal weights to the treatments using an attention mechanism. As time-dependent confounders can result in a biased distribution of treatment assignments and imbalanced interferences due to the evolving graph structure, CAG-ODE utilizes two adversarial learning losses to ensure unbiased estimations of counterfactual outcomes. The overall framework is depicted in Figure 7.1. We now discuss each module in detail.

7.4.1 Spatial-Temporal Initial State Encoder

The encoder of CAG-ODE infers the posterior distributions from the historical observations and then samples the latent initial states from them. It follows the architecture described in [HSW21c]. As the evolution of different nodes is mutually influenced, we calculate the initial states for all nodes simultaneously considering their interactions over time. The initial states of edges are derived from the initial states of nodes.

Dynamic Node Representation Learning. We construct a graph to represent the spatial-temporal structure of multi-agent dynamical systems, with each node corresponding to an agent's observation at a particular timestamp. There are two types of edges: spatial edges at the same timestamp and temporal edges across different timestamps. The spatial edges are formed according to the adjacency matrices, denoted as $w_{i(t)\to j(t)}$. For the temporal edges, we only consider edges from an agent's own previous observations to later observations, denoted as $w_{i(t)\to i(t')}$, where $t' = t + 1$.

The latent representations of observations are learned from this spatial-temporal graph through an attention mechanism approach. The propagation among L GNN layers is depicted in Equation(7.2).

$$
\begin{aligned}
\boldsymbol{h}_{i(t')}^{l} &= \boldsymbol{h}_{i(t')}^{l} + \sigma\left(\sum_{j(t)\in\mathcal{N}_{i(t')}} e_{j(t)\to i(t')}^{l} \times \boldsymbol{W}_v \widehat{\boldsymbol{h}}_{j(t)}^{l-1}\right), \\
e_{j(t)\to i(t')}^{l} &= w_{j(t)\to i(t')} \times \alpha_{j(t)\to i(t')}^{l}, \\
\alpha_{j(t)\to i(t')}^{l} &= \left(\boldsymbol{W}_k \widehat{\boldsymbol{h}}_{j(t)}^{l-1}\right)^{T} \left(\boldsymbol{W}_q \boldsymbol{h}_{i(t')}^{l-1}\right) \cdot \frac{1}{\sqrt{d}}, \\
\widehat{\boldsymbol{h}}_{j(t)}^{l-1} &= \boldsymbol{h}_{j(t)}^{l-1} + \mathrm{TE}\left(t-t'\right), \\
\mathrm{TE}(\Delta t)_{2i} &= \sin\left(\frac{\Delta t}{10000^{2i/d}}\right), \mathrm{TE}(\Delta t)_{2i+1} = \cos\left(\frac{\Delta t}{10000^{2i/d}}\right).
\end{aligned}
\tag{7.2}
$$

Here, $\boldsymbol{h}_{i(t)}^{l}$ represents the agent i at time t from layer l. The attention score $e_{j(t)\to i(t')}^{l}$ is defined as the product of edge weights $w_{j(t)\to i(t')}$ and affinity score $\alpha_{j(t)\to i(t')}^{l}$, which is computed using the representations of the sender and receiver nodes. Additionally, we incorporate temporal embedding, denoted as TE, into the sender node's representation to establish temporal distinction. Then, the final representation is obtained from the L layer as $\boldsymbol{h}_{i(t)} = \boldsymbol{h}_{i(t)}^{L}$.

Sequence Representation Learning. Then, we employ self-attention to compute the sequence representation of observed temporal information for each node, where $\widehat{\boldsymbol{h}}_{i(t)} = \boldsymbol{h}_{i(t)} + \mathrm{TE}(t)$.

$$
\boldsymbol{u}_i = \frac{1}{N}\sum_{t=1}^{T}(\boldsymbol{a}_i^{T}\widehat{\boldsymbol{h}}_{i(t)}\widehat{\boldsymbol{h}}_{i(t)}), \boldsymbol{a}_i = \tanh\left(\left(\frac{1}{N}\sum_{t=1}^{T}\widehat{\boldsymbol{h}}_{i(t)}\right)\boldsymbol{W}_a\right). \tag{7.3}
$$

Finally, the mean and variance of the posterior distribution is obtained through a neural network f_{ddist} from the sequence representation $\boldsymbol{u}_i$.

$$
\boldsymbol{z}_i^0 \sim q_\phi(\boldsymbol{z}_i^0|\mathcal{H}^0) = \mathcal{N}(\boldsymbol{\mu}_{z_i^0}, \sigma_{\boldsymbol{z}_i^0}^2),\ \boldsymbol{\mu}_{\boldsymbol{z}_i^0}, \sigma_{\boldsymbol{z}_i^0} = f_{\text{dist}}(\boldsymbol{u}_i).
$$

Next, the latent initial state for an edge is given by $\boldsymbol{z}_{i\to j}^0 = f_{\text{edge}}([\boldsymbol{z}_i^0, \boldsymbol{z}_j^0])$, where f_{edge} is parameterized by a neural network and $[,]$ is concatenation operation.

7.4.2 Treatment Fusing

To conduct causal inference with CAG-ODE, we propose to inject the dynamic effects of multiple treatments into the ODE function. Treatments can have time-varying effects in multi-agent dynamical systems and they can occur simultaneously, resulting in a combined effect. To model such complex behaviors, we propose a novel treatment fusing module that assigns temporal weights to multiple treatments through an attention mechanism. The temporal weight of treatment at timestamp t is dependent on both the start time of each treatment and the occurrence of other treatments as shown in Eqn (7.4). Let $\boldsymbol{e}_k \in \mathbb{R}^K$ be the one-hot representation of treatment k. We first add it with the temporal encoding TE[HSW21c, VSP17] to account for the time elapsed since the start of the treatment t'. Here $\mathbf{A}_{ik}^t \in \{0,1\}$ is an indicator showing whether treatment k would be applied to agent i at timestamp t. Therefore the computed treatment representation $\widehat{\boldsymbol{o}}_{ik}^t$ becomes zero when $\mathbf{A}_{ik}^t = 0$, to ensure computational efficiency. A contraction matrix $\mathbf{W}_q$ is then used to transform this sparse representation into a more compact form.

$$\begin{aligned}
&\widehat{\boldsymbol{o}}_{ik}^t = \mathbf{A}_{ik}^t \boldsymbol{e}_k + \mathrm{TE}\left(t-t'\right) \mathbb{1}[\mathbf{A}_{ik}^t = 1], \quad \boldsymbol{o}_{ik}^t = \mathbf{W}_q \widehat{\boldsymbol{o}}_{ik}^t, \\
&\mathrm{TE}(\Delta t)_{2i} = \sin\left(\Delta t / M^{2i/d}\right), \\
&\mathrm{TE}(\Delta t)_{2i+1} = \cos\left(\Delta t / M^{2i/d}\right), M = 10000.
\end{aligned} \tag{7.4}$$

To account for the combined effect of simultaneous treatments, we compute the combined treatment representation as a weighted sum of all in-effect treatments at timestamp t (Eqn 7.5). We first compute an attention vector m_i^t as the tanh-transformed average of all the treatment representations, $\widehat{\boldsymbol{o}}_{ij}^t$. Each treatment's weight is derived from the dot product of its representation and m_i^t, thereby integrating each treatment's influence into $\boldsymbol{o}_i^t$.

$$\boldsymbol{o}_i^t = \frac{1}{K} \sum_k \left(\boldsymbol{m}_i^{t\top} \widehat{\boldsymbol{o}}_{ik}^t \widehat{\boldsymbol{o}}_{ik}^t \right), \boldsymbol{m}_i^t = \tanh\left(\left(\frac{1}{K} \sum_k \boldsymbol{o}_{ik}^t \right) \mathbf{W}_m \right). \tag{7.5}$$

The fusing operation has a time complexity of $O(K)$ if having K treatments and therefore is able to scale up to larger systems.

7.4.3 Treatment-Induced GraphODE

We use two coupled ODEs to predict the latent trajectories for nodes and edges respectively, accounting for their co-evolution [HSW21c]. We incorporate the learned treatment representations into the ODEs to enable counterfactual predictions in the future. Specifically, the co-evolution of nodes and edges is depicted in Eqn 7.6. The co-evolution depends on all historical information implicitly as $\mathbf{Z}_t$ embeds the trajectories up to time t. $\widetilde{\mathbf{W}}_A^t = \mathbf{D}^{-1}\mathbf{W}_A^t$ is the normalized adjacency matrix and $\mathbf{D}$ is the diagonal degree matrix defined as $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{Aij}^t$. $f_e, f_{\text{self}}, f_{\text{edge2value}}$ are all implemented as Multi-Layer Perceptrons (MLPs). To incorporate the treatment effect into the function, we use a linear transformation $\mathbf{W}$ to merge the latent states of nodes $\mathbf{Z}^t$ and the treatment representation $\boldsymbol{O}^t$. In this way, the latent trajectories of agents are affected not only by their own past trajectories and treatments but also by the trajectories and treatments of their interacting agents.

$$
\begin{aligned}
&\frac{\mathrm{d}\mathbf{Z}^t}{\mathrm{d}t} = \sigma\left(\widetilde{\mathbf{W}}_A^t \mathbf{W}[\mathbf{Z}^t, \mathbf{O}^t]\right) - \mathbf{Z}^t + \mathbf{Z}^0, \\
&\frac{\mathrm{d}\boldsymbol{z}_{i\to j}^t}{\mathrm{d}t} = f_e\left(\left[\boldsymbol{z}_i^t, \boldsymbol{z}_j^t\right]\right) + f_{\text{self}}\left(\boldsymbol{z}_{i\to j}^t\right), \\
&\mathbf{W}_{Aij}^t = f_{\text{edge2value}}\left(\boldsymbol{z}_{i\to j}^t\right), \quad \widetilde{\mathbf{W}}_A^t = \mathbf{D}^{-1}\mathbf{W}_A^t.
\end{aligned}
\tag{7.6}
$$

7.4.4 Outcome Prediction

Given the treatment representations, the ODE functions, the latent initial states for nodes and edges, and the latent trajectories for all agents can be determined using any black-box ODE solver. Finally, we compute the predicted trajectories for each agent and their interactions based on the decoding likelihoods in Eqn (7.7), where f_{decN} and f_{decE} are node and edge decoding functions respectively. They output the means of the normal distributions $p(\boldsymbol{y}_i^t|\boldsymbol{z}_i^t)$ and $p(\boldsymbol{w}_{i\to j}^t|\boldsymbol{z}_i^t)$, which we treat as the predicted values from our model.

$$
\boldsymbol{y}_i^t \sim p(\boldsymbol{y}_i^t|\boldsymbol{z}_i^t) = f_{\text{decN}}(\boldsymbol{z}_i^t),\ \boldsymbol{w}_{i\to j}^t \sim p(\boldsymbol{w}_{i\to j}^t|\boldsymbol{z}_i^t) = f_{\text{decE}}(\boldsymbol{z}_i^t). \tag{7.7}
$$

We implemented all of our decoders using two-layer fully connected neural networks. The node feature decoder's input dimension matches the latent state dimension d, while the output dimension is one, reflecting our outcome of interest. The edge decoder's input dimension is $2d$ and the output dimension is 1. The treatment decoder also has an input dimension equal to the latent state's dimension d. However, its output dimension matches the number of distinct treatments, predicting the probability of each treatment being chosen. Lastly, the interference decoder's input dimension is the sum of the latent state dimension and the treatment embedding dimension, i.e. $2d$. Its output dimension mirrors the number of treatment options. For all decoders, the latent hidden dimension is half of their respective input dimensions.

We calculate the reconstruction loss of model predictions for nodes $\widehat{Y}_i^t$ and edges $\widehat{w}_{i\to j}^t$ as:

$$
L^{\langle Y\rangle} = \frac{1}{N}\frac{1}{T}\sum_t \|\mathbf{Y}^t - \widehat{\mathbf{Y}}^t\|_2^2,\ L^{\langle W\rangle} = \frac{1}{N^2}\frac{1}{T}\sum_t \|\mathbf{W}_A^t - \widehat{\mathbf{W}}_A^t\|_F^2.
$$

7.4.5 Domain Adversarial Learning

In observational data, treatment assignments are not randomized but are biased based on time-varying confounder values. This can lead to increased variance and bias in counterfactual estimation [SIB22]. In multi-agent dynamical systems, unbalanced interference from neighboring agents further exacerbates this effect and alters the state of each agent. To obtain an unbiased counterfactual prediction, we need to ensure that the distribution of latent representation trajectories is invariant to treatments and interference [JHL23c]. This guarantees that the treatments cannot be inferred from the latent trajectory representations and that the interference is not predictable when the treatment is combined with the latent representation.

To achieve this, we incorporate two adversarial learning losses into the optimization objective function and use gradient reversal layers for the implementation.

Treatment Balancing The treatment combinations $\widehat{\mathbf{A}}^t$ can be predicted using a decoder from the latent state $\boldsymbol{z}_i^t$. Formally, $\widehat{\mathbf{A}}_{i\cdot}^t = \Phi_A(r(\boldsymbol{z}_i^t))$, where Φ_A is a neural network attempting to recover treatments from the latent state $\boldsymbol{z}_i^t$, and the gradient reversal layer, denoted by r, reverses the sign of gradient during back-propagation. The treatment balancing can be expressed as the maximization of the following loss term through the construction of min-max games:

$$L^{\langle A\rangle} = -\frac{1}{N}\frac{1}{T}\frac{1}{K}\sum_{i=1}^{N}\sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{j\in\{0,1\}} \mathbb{1}[(\mathbf{A}_{ik}^t = j)]\log(\Phi_A^{j,k}(r(\boldsymbol{z}_i^t))),$$

where $\Phi_A^{j,k}$ represents the logits of $d_A(\cdot)$ for predicting j on k-th treatment. Note that we achieve treatment balancing by letting the latent representations $\boldsymbol{z}_i^t$ not be predictable for each individual treatment. This is because the representation of multiple treatments is essentially a linear combination of individual treatments. If each individual treatment is not predictable based on $\boldsymbol{z}_i^t$, then it is also impossible to use such representation to predict when multiple treatments occur together.

Interference Balancing Similar to treatment balancing, the interference prediction can be represented as $\widehat{\mathbf{G}}_i^t = \Phi_G(r([Z_i^t, A_i^t]))$, where d_G denotes a neural network designed to estimate interference. As interference is a continuous variable, we employ continuous domain adversarial training to accomplish interference balancing. By incorporating a gradient reversal layer, interference balancing can be achieved by minimizing the following loss term:

$$L^{\langle G\rangle} = \frac{1}{N}\frac{1}{T}\frac{1}{K}\sum_{i=1}^{N}\sum_{t=1}^{T}\|\Phi_G(r([\boldsymbol{z}_i^t, \boldsymbol{o}_i^t])) - \mathbf{G}_i^t\|_2^2.$$

Overall Loss The overall training objective is defined as the weighted summation of node reconstruction loss, edge reconstruction loss, treatment balancing loss, and interference balancing loss. Since

Table 7.1: Root Mean Square Error (RMSE) for factual outcome evaluation across prediction lengths (the duration for which predictions are made). For the COVID-19 dataset, we report the mean and standard deviation accuracy with multiple runs.

Dataset	Covid-19			Tumor Growth		
Prediction Length	7-days	14-days	21-days	14-days	21-days	28-days
CG-ODE	4063 ± 68	4454 ± 100	4659 ± 63	18.37	21.00	24.58
TE-CDE	7999 ± 212	7470 ± 289	6832 ± 243	55.45	55.38	71.23
COVID-POLICY	4008 ± 44	4128 ± 60	3963 ± 59	20.07	25.93	29.29
CAG-ODE	$\mathbf{3710 \pm 29}$	$\mathbf{3925 \pm 44}$	$\mathbf{3933 \pm 40}$	**10.91**	**10.82**	**14.84**
w/o $L^{\langle G \rangle}$	3800 ± 60	3987 ± 40	3990 ± 49	15.57	16.28	16.62
w/o $L^{\langle A \rangle}$	3840 ± 35	4100 ± 53	4069 ± 49	17.90	14.69	20.19
w/o $L^{\langle G \rangle}$,$L^{\langle A \rangle}$	3793 ± 23	4089 ± 79	3953 ± 38	17.28	16.72	24.36
w/o attention	3867 ± 61	3958 ± 31	4256 ± 55	18.91	17.55	34.45

we follow the VAE framework, we also incorporate a KL divergence loss to add regularization towards the sampled initial states, which is defined as: $L_{KL} = \mathrm{KL}\left[\prod_{i=1}^{N} q_\phi\left(\boldsymbol{z}_i^0 \mid \mathcal{H}^0\right) \| p\left(\mathbf{Z}^0\right)\right]$. Therefore, the overall training loss is formalized as:

$$L = L^{\langle Y \rangle} + \lambda L^{\langle W \rangle} + \alpha L^{\langle A \rangle} + \beta L^{\langle G \rangle} + \gamma L_{KL}.$$

7.5 Experiments

7.5.1 Experiment Setup

7.5.1.1 Datasets and Experiment Configuration

We evaluate the performance of our model using two datasets: 1.) The **COVID-19 dataset**, which captures the daily COVID-19 trends of U.S. states from April.12.2020 to Dec.31.2020. The daily population flows among states are represented as dynamic edges. Treatments are state-level COVID-19 policies. We ask the model to predict the daily confirmed cases in each state. 2.) The **Tumor Growth simulation dataset** [GPG17], which describes the tumor growth dynamics in different regions of patients, where they may receive differing treatments. We aim to predict the tumor volumes in each region. Additional details about the datasets can be found in the Appendix.

We predict trajectory rollouts across varying lengths and use Root Mean Square Error (RMSE) as the evaluation metric. Specifically, we train our model in a sequence-to-sequence setting where we split the trajectory of each training sample into two parts $[t_1, t_K]$ and $[t_{K+1}, t_T]$. We condition the model on the first part of observations and predict the second part. To fully utilize the data points within each trajectory, we generate training and validation samples by splitting each trajectory into several chunks using a sliding window. Details can be found in the Appendix.

7.5.1.2 Baselines and Model Variants

We conduct a comparative analysis of our model with three baseline models: one non-causal continuous multi-agent baseline CG-ODE [HSW21c], and two causal models: TE-CDE [SIB22] and COVID-POLICY [MDH22]. TE-CDE [SIB22] is a causal model that employs continuous-time differential equations to capture temporal event dependencies. COVID-Policy [MDH22] is another causal model designed specifically for assessing the impact of public health policies on COVID-19 outcomes. To further analyze the performance of our model, we also compare variants of our model. Each variant excludes a specific component to assess its individual impact on performance. The variants include models without treatment balancing, interference balancing, both components or the attention module.

7.5.1.3 Training Details

We employ the AdamW optimizer, as proposed in the study by Loshchilov et al. [LH], to train our model. The initial learning rate is set at $\eta = 0.005$, and the batch size is set as 8 to accommodate memory constraints.

The Graph Neural Network (GNN) used for the encoder has a singular layer with a hidden dimension of 64. Similarly, the GNN that parameterizes the ODE function is also comprised of a single layer. The dimension of the latent state is set at 20, and the dimension for the embedded treatments is 5. We assign a weight of 10 for both the treatment balancing term α and the interference balancing term β. Additionally, the weight designated for the edge reconstruction error λ is set at 0.5.

7.5.2 Performance Evaluation

We evaluate the performance of our model, CAG-ODE, as well as the baselines using Root Mean Square Error (RMSE) across different prediction lengths. The results are shown in Table 7.1 and Table 7.2, reporting the factual and counterfactual outcomes respectively. As the COVID-19 is a real-world dataset that does not have counterfactual outcomes, we evaluate only the Tumor Growth dataset in Table 7.2. To ensure consistent comparison, we align the prediction periods of all models with weekly intervals on the COVID-19 dataset, similar to the statistical baselines derived from their official weekly submissions to the CDC, as done in [HSW21c]. To assess the accuracy of short-term and long-term predictions, the prediction lengths for the COVID-19 and Tumor Growth datasets are set to 7, 14, 21 days and 14, 21, and 28 days, respectively. We include longer-range predictions on the Tumor-Growth dataset in the Appendix.

Factual Outcome Predictions. Table 7.1 shows that our model, CAG-ODE, consistently outperforms the baseline models across all prediction lengths for both datasets. This underscores the effectiveness of our model in capturing the dynamic interactions among objects, especially over longer time periods. Comparing our model with TE-CDE, we observe a performance gap that highlights the benefits of incorporating interference balancing and spatial correlation in the model. Additionally, our model outperforms the COVID-POLICY model, indicating its broader generalizability across different types of data due to modeling dynamic interactions. Furthermore, our model exhibits proficiency in both short-term and long-term predictions. For instance, it achieves promising results for 21-day predictions on the COVID-19 dataset and 28-day predictions on the Tumor Growth simulation dataset. The analysis of our model variants further emphasizes the importance of each component in the model. Particularly, the model variant excluding the attention module has the weakest performance, indicating the significance of our time-embedding attention module in effectively representing the treatment.

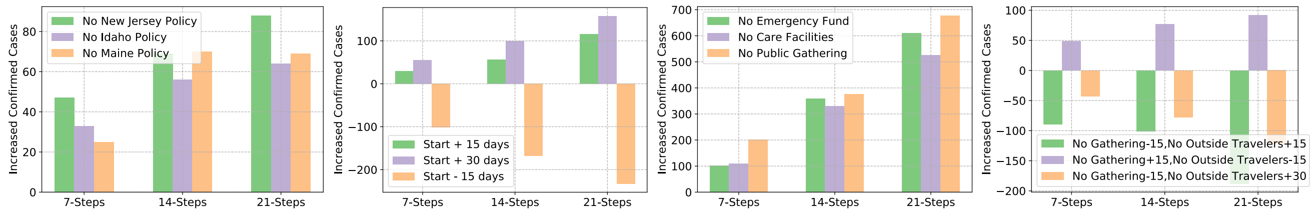
Counterfactual Outcome Predictions. In the context of a multi-agent dynamical system, the total number of possible treatments for all nodes is $\boldsymbol{O}(K \times 2N)$, making it infeasible to enumerate all treatment combinations. To assess the robustness of each model to counterfactual treatment scenarios, we perform an experiment where we randomly flip a certain percentage of observed treatments. In Table 7.2, we evaluate the performance when 25%, 50%, and 75% of all observed treatments in each experiment are randomly flipped. The purpose of this experiment is to examine the robustness of the models to counterfactual treatment scenarios, and since CG-ODE does not incorporate causal modeling, it is excluded from this experiment. CAG-ODE outperforms others by a wide margin across all settings. These findings collectively demonstrate the superiority of our proposed model, CAG-ODE, in capturing the dynamics of multi-agent systems and making accurate predictions across different time horizons. We additionally include the visualization of the learned balanced latent representations in Section 7.5.4.

Table 7.2: Root Mean Square Error (RMSE) for counterfactual Outcome evaluation on the Tumor Growth dataset with treatment flipping ratio. Treatment F.R. (**Treatment F**lipping **R**atio) represents the ratio of treatments that are flipped.

Prediction Length	14-days			21-days			28-days		
Treatment F.R.	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
TE-CDE	95.61	103.2	100.8	98.65	103.0	97.93	118.3	124.0	121.4
COVID-POLICY	21.32	22.37	23.31	26.63	26.83	27.00	32.01	32.16	32.21
CAG-ODE	**17.23**	**16.98**	**16.96**	**18.64**	**18.84**	**18.85**	**19.91**	**19.88**	**19.87**
w/o $L^{\langle G \rangle}$	20.62	20.53	20.51	19.70	19.60	19.55	21.10	21.41	21.38
w/o $L^{\langle A \rangle}$	22.17	22.35	22.35	20.19	20.10	20.09	20.83	21.14	21.15
w/o $L^{\langle G \rangle}$, $L^{\langle A \rangle}$	19.78	19.75	19.71	19.34	19.29	19.27	21.31	21.40	21.34
w/o attention	19.09	18.37	18.13	22.16	21.78	21.65	27.70	27.44	27.38

7.5.3 Case Study about COVID-19 Policies

We conduct a case study to show the impact of different treatments, e.g., COVID-19 related policies, on the COVID-19 dataset as shown in Figure 7.2. Specifically, we consider four different policy intervention methods and report the resulting average changes in the number of daily confirmed cases across all states in the U.S.



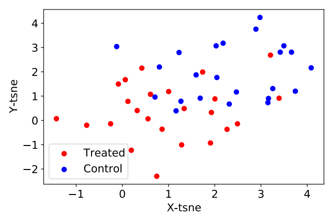
(a) Remove partial states' policy. (b) Change policy start date. (c) Remove policy across states. (d) Change relative time of policies.

Figure 7.2: Case Study for changing different policies on the COVID-19 dataset.

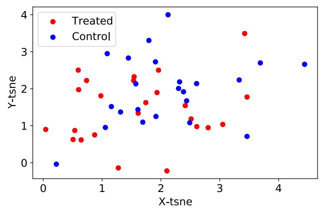
First, we focus on the removal of policies in three states that have the highest number of announced policies during the time frame of the COVID-19 dataset. By masking out these policies, we observe an increase in the average number of confirmed cases across states in the future. This increase is attributed to both in-state disease spread and population flow to other states. The removal of policies exacerbates the spread of COVID-19 over an extended period, as shown in Figure 7.2(a), indicating that our model captures the dynamic interference resulting from agents' interactions.

We then explore the effect of changing the starting time of a specific policy for all states. We changed the "No Public Gatherings" policy starting time for each state to be 15 days earlier, 15 and 30 days later respectively. As shown in Figure 7.2(b) when announcing the policy earlier, we observe a decrease in the average number of daily confirmed cases in the future, while announcing the policy later leads to an increase. This intuitive outcome highlights the capability of our model to capture the causal relationships between policy interventions and COVID-19 spread.

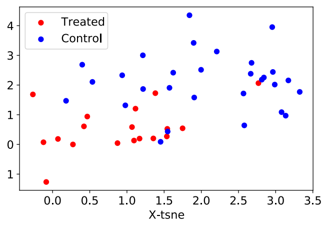
Next, we analyze the impact of the top three most frequent policies across all states by removing them separately. As shown in Figure 7.2(c), the "Public Gatherings" policy has the largest effect in reducing the spread of COVID-19, even though the most frequent policy is "Emergency Funds". This demonstrates the potential of our model in assisting policymakers to identify the relative importance of each policy over time.



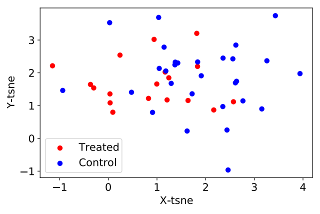
(a)
"State-of-Emergency" w/o. Treatment Balancing.



(b)
"State-of-Emergency" with Treatment Balancing.



(c)
"No-Public-Gathering" w/o. Treatment Balancing.



(d)
"No-Public-Gathering" with Treatment Balancing.

Figure 7.3: Treatment Balancing Visualization on the COVID-19 Dataset.

Finally, we study the effects of different orders in policy announcements, specifically focusing on the simultaneous or closely timed announcements of "No Public Gatherings" and "No Traveler from Outside States" policies. We change the announcement dates for the two policies in each state to mimic three scenarios shown in Figure 7.2 (d). We found that initializing the announcement of "No Public Gatherings" early generally contributes to a reduction in the spread of COVID-19 compared with "No Traveler from Outside States". We further analyzed the daily population flow during the given time frame and found that the majority of population flows are within the same states, indicating that residents of each state pose a high risk of virus transmission compared to people from other states. These insights suggest prioritizing the earlier announcement of the "No Public Gatherings" policy over the "No Traveler from Outside States" policy can better mitigate the spread of COVID-19.

These case study results demonstrate the effectiveness of our model CAG-ODE in capturing the complex interactions between treatments, disease spread, and population flow, providing valuable insights for policymakers in making informed decisions.

7.5.4 Visualization of Learned Balanced Representations

To further understand the effect of treatment balancing loss in CAG-ODE, we visualize the 2-D T-SNE projections of the latent representations of nodes on the COVID-19 dataset, i.e. $\boldsymbol{z}_i^t$ as shown in Figure 7.3. Specifically, we visualize the latent node representations under two different treatments: "State-of-Emergency" and "No-Public-Gathering". Under each treatment (policy), we use different colors to denote whether a node receives such treatment (treated) or not (control). As shown in Figure 7.3(a) and (c), the distributions of the learned representations are more distinguishable between the two groups, compared with Figure 7.3(b) and (d) which have the treatment balancing loss. This indicates that CAG-ODE indeed learns balanced latent representations by employing the treatment balancing loss.

7.6 Conclusion

In this paper, we introduce the causal graph ODE (CAG-ODE) as a model for estimating continuous counterfactual outcomes in multi-agent-dynamical systems with evolving interaction edges and dynamic multi-treatment effects. Our model builds upon existing GraphODEs and incorporates causal reasoning for multi-agent dynamical systems. We propose a novel treatment fusing module that captures the dynamic effects of multiple treatments occurring simultaneously. Through extensive experiments on both the real-world and the simulated datasets, we demonstrate the superior performance of our model across various prediction settings, validating its effectiveness. Furthermore, we leverage our model to analyze policy effects analysis on the COVID-19 dataset, providing valuable insights for policymakers.

7.7 Discussion and Future Directions

My research lies at the interdisciplinary crossroads, and I have been fortunate to collaborate with people from different backgrounds, covering material science, physics, biomedical engineering, and healthcare, which provides a basis for my future work. I aim to build an intelligent "**AI Assistant for Scientists**" that is able to facilitate scientific discovery across disciplines. Such procedure can be staged into 1.) experimental design, 2.) data analysis [WFD23]. I propose the following directions in line with the three stages.

Cost-Efficient Experimental Design: Experiment-in-the-Loop Neural Simulator Training. Neural simulators are trained on observational data generated by ground-truth simulators. The acquisition of such data can be expensive, especially when demanding high precision at time scales in the order of milliseconds (10^{-3}) or even nanoseconds (10^{-9}). In various scenarios, adjusting experiment designs dynamically becomes essential to optimize costs. For instance, in drug discovery, the extended validation period for a new drug necessitates costly testing and human analysis. In quantum physics, selecting the most effective approach for a complex experiment can be counterintuitive. Hence, optimizing experiment design stands as a pivotal challenge in advancing scientific discovery. To address this, I propose integrating experiment designs into the neural simulator's training loop. Considering training a neural simulator that can be generalized across different time scales, by utilizing intermediate outputs, such as prediction accuracy from neural networks, one can refine subsequent experiment designs, such as increasing the time precision scale (high-fidelity simulations) or opting for a lower precision (low-fidelity simulations) in ground-truth simulators for generating new data. The newly generated data will be employed to train the neural networks, with iterative generation guided by the model's performance. One potential approach involves employing reinforcement learning, allowing for iterative experiment design and receiving feedback in the process.

Unveiling Symbolic Knowledge from Scientific Data. Unraveling dynamic formulas from data lies at the core of scientific development, grounded in closed-form symbolic expressions that encapsulate laws and principles discovered by humans. These concise expressions distill insights from observational data, serving as adaptable tools in addressing related problems and fostering the generation of new knowledge. While neural networks excel in accurate predictions for dynamical systems, their black-box nature hinders interpretability and generalization across diverse systems. Symbolic regression, a supervised machine learning technique, constructs analytic functions inspired by neural networks. However, most fundamental formulas in sciences are described by differential equations such as Newton's Law of Motion, making GraphODE a perfect choice for extracting dynamical formulas from data in contrast with discrete models. In the future, I plan to develop novel methods to extract dynamical formulas learned by GraphODE, which can serve as a precise summary of the knowledge learned from data in the last stage. I am also interested in studying the relationships among formulas extracted from different systems, and seeking new insights from them in a manner akin to human learning processes.

Broader Applications via Dynamical System Modeling. Finally, I plan to explore broader applications in real-world contexts that can be modeled as dynamical systems. An interesting example is to investigate the optimization of neural network training by adopting a dynamical system perspective, where neurons are treated as nodes, and weights between neurons as edges. By modeling the training process through the lens of dynamical systems, the aim is to accelerate and enhance the efficiency of neural network training and model selection. I am also interested in extending ODEs to stochastic differential equations (SDE) and partial differential equations (PDE). The former naturally captures the stochastic nature of dynamical systems such as Brownian motion, whereas the latter is well-suited to capture the continuity in both space and time aspects such as mesh simulations and fluid dynamics. Finally, I intend to integrate external knowledge and signals into GraphODE for more precise and robust/reliable reasoning when faced with limited observational data, spanning from multi-modal learning (*e.g.* visions, languages) to knowledge graphs. By incorporating these additional dimensions, the model can provide a more comprehensive and accurate representation of complex real-world system dynamics.

Embarking on these research directions, my goal is to push the boundaries of the limits of symbolic deep learning in scientific discovery. This pursuit aims not only to enhance the precision of predictions and reasoning but also to unearth novel insights and unveil untapped potentials within dynamic relational data.

Bibliography

[AKW08] J. Awrejcewicz, G. Kudra, and G. Wasilewski. "Chaotic zones in triple pendulum dynamics observed experimentally and numerically." *Applied Mechanics and Materials*, pp. 1–17, 2008.

[BAJ20a] Ioana Bica, Ahmed M Alaa, James Jordon, and Mihaela van der Schaar. "Estimating counterfactual treatment outcomes over time through adversarially balanced representations." *International Conference on Learning Representations*, 2020.

[BAJ20b] Ioana Bica, Ahmed M. Alaa, James Jordon, and Mihaela van der Schaar. "Estimating counterfactual treatment outcomes over time through adversarially balanced representations." In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

[BB00] Cx K Batchelor and GK Batchelor. *An introduction to fluid dynamics*. Cambridge university press, 2000.

[BBH19] Maximilian Behr, Peter Benner, and Jan Heiland. "Solution formulas for differential Sylvester and Lyapunov equations." In *Calcolo*, 2019.

[BDB19] Yunsheng Bai, Hao Ding, Song Bian, Ting Chen, Yizhou Sun, and Wei Wang. "SimGNN: A Neural Network Approach to Fast Graph Similarity Computation." In *WSDM'19*, 2019.

[BEP08] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. "Freebase: A Collaboratively Created Graph Database for Structuring Human Knowledge." In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD '08, p. 1247–1250, 2008.

[BK19] Austin Benson and Jon Kleinberg. "Link prediction in networks with core-fringe data." In *The Web Conference (WWW)*, pp. 94–104, 2019.

[BLZ22] Guangji Bai, Chen Ling, and Liang Zhao. "Temporal Domain Generalization with Drift-Aware Dynamic Neural Networks." *arXiv preprint arXiv:2205.10664*, 2022.

[BNM20a] Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. "CoPhy: Counterfactual Learning of Physical Dynamics." In *ICLR*, 2020.

[BNM20b] Fabien Baradel, Natalia Neverova, Julien Mille, Greg Mori, and Christian Wolf. "CoPhy: Counterfactual Learning of Physical Dynamics." In *ICLR*, 2020.

[BPL16] Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, and koray kavukcuoglu. "Interaction Networks for Learning about Objects, Relations and Physics." In *Advances in Neural Information Processing Systems 29*, pp. 4502–4510. 2016.

[BUG13] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Durán, Jason Weston, and Oksana Yakhnenko. "Translating Embeddings for Modeling Multi-Relational Data." In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, p. 2787–2795, 2013.

[BV21] Alexis Bellot and Mihaela Van Der Schaar. "Policy analysis using synthetic controls in continuous-time." In *International Conference on Machine Learning*, pp. 759–768. PMLR, 2021.

[CCF20] Xuelu Chen, Muhao Chen, Changjun Fan, Ankith Uppunda, Yizhou Sun, and Carlo Zaniolo. "Multilingual Knowledge Graph Completion via Ensemble Knowledge Transfer." In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pp. 3227–3238, 2020.

[CGH20] M. Cranmer, S. Greydanus, S. Hoyer, P. Battaglia, D. Spergel, and S. Ho. "Lagrangian neural networks." *arXiv preprint arXiv:2003.04630*, 2020.

[CGL19] Y. Cao, X. Gao, and R. Li. "A liquid plug moving in an annular pipe–Heat transfer analysis." *International Journal of Heat and Mass Transfer*, **139**:1065–1076, 2019.

[CJL21] Yixin Cao, Xiang Ji, Xin Lv, Juanzi Li, Yonggang Wen, and Hanwang Zhang. "Are Missing Links Predictable? An Inferential Benchmark for Knowledge Graph Completion." In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 6855–6865, Online, 2021.

[CL18] Y. Cao and R. Li. "A liquid plug moving in an annular pipe—Flow analysis." *Physics of Fluids*, **30**(9), 2018.

[CLL19] Yixin Cao, Zhiyuan Liu, Chengjiang Li, Zhiyuan Liu, Juanzi Li, and Tat-Seng Chua. "Multi-Channel Graph Neural Network for Entity Alignment." In *ACL*, 2019.

[CLR18] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Herve J ´egou. "Word translation without parallel data." In *In International Conference on Learning Representations*, 2018.

[CMH18] B. Chang, L. Meng, E. Haber, L. Ruthotto, D. Begert, and E. Holtham. "Reversible architectures for arbitrarily deep residual neural networks." In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[CMU03] CMU. "Carnegie-Mellon Motion Capture Database." 2003.

[Con18] "Convolutional 2D Knowledge Graph Embeddings." In *Proceedings of the 32th AAAI Conference on Artificial Intelligence*, pp. 1811–1818, 2018.

[CPC18] Zhengping Che, Sanjay Purushotham, Kyunghyun Cho, David Sontag, and Yan. Liu. "Recurrent Neural Networks for Multivariate Time Series with Missing Values." In *Scientific Reports*, p. 6085, 2018.

[CPK21] Serina Chang, Emma Pierson, Pang Wei Koh, Jaline Gerardin, Beth Redbird, David Grusky, and Jure Leskovec. "Mobility network models of COVID-19 explain inequities and inform reopening." In *Nature*, 2021.

[CRB18a] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. "Neural Ordinary Differential Equations." In *Advances in Neural Information Processing Systems 31*, pp. 6571–6583. 2018.

[CRB18b] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud. "Neural Ordinary Differential Equations." In *Advances in Neural Information Processing Systems*, 2018.

[CSB81] Thomas C Chalmers, Harry Smith Jr, Bradley Blackburn, Bernard Silverman, Biruta Schroeder, Dinah Reitman, and Alexander Ambroz. "A method for assessing the quality of a randomized control trial." *Controlled clinical trials*, **2**(1):31–49, 1981.

[CSB20] Miles Cranmer, Alvaro Sanchez-Gonzalez, Peter Battaglia, Rui Xu, Kyle Cranme, David Spergel, and Shirley Ho. "Discovering Symbolic Models from Deep Learning with Inductive Biases." In *Neurips'20*, 2020.

[CTC18] Muhao Chen, Yingtao Tian, Kai-Wei Chang, Steven Skiena, and Zaniolo Carlo. "Co-training embeddings of knowledge graphs and entity descriptions for cross-lingual entity alignment." In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 3998–4004, 2018.

[CTY17] Muhao Chen, Yingtao Tian, Mohan Yang, and Carlo Zaniolo. "Multilingual Knowledge Graph Embeddings for Cross-lingual Knowledge Alignment." In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, 2017.

[CUT16] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. "A Compositional Object-Based Approach to Learning Physical Dynamics." *ICLR*, 2016.

[CWL18] Wei Cao, Dong Wang, Jian Li, Hao Zhou, Lei Li, and Yitan Li. "BRITS: Bidirectional Recurrent Imputation for Time Series." In *Advances in Neural Information Processing Systems 31*, pp. 6775–6785. 2018.

[DCL19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

[DDG20] Ensheng Dong, Hongru Du, and Lauren Gardner. "An interactive web-based dashboard to track COVID-19 in real time." In *The Lancet Infectious Diseases*, 2020.

[DGH22] Edward De Brouwer, Javier Gonzalez, and Stephanie Hyland. "Predicting the impact of treatments over time with uncertainty aware neural differential equations." In *International Conference on Artificial Intelligence and Statistics*, pp. 4705–4722. PMLR, 2022.

[End03] D. F. M. Endre Süli. *An Introduction to Numerical Analysis*. Cambridge University Press, 2003.

[GBG21] Johannes Gasteiger, Florian Becker, and Stephan Günnemann. "Gemnet: Universal directional graph neural networks for molecules." *Advances in Neural Information Processing Systems*, **34**:6790–6802, 2021.

[GC19] L. Gong and Q. Cheng. "Exploiting Edge Features for Graph Neural Networks." In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 9203–9211, 2019.

[GDY19] S. Greydanus, M. Dzamba, and J. Yosinski. "Hamiltonian neural networks." *Advances in Neural Information Processing Systems*, 2019.

[GG22] Nicholas Gao and Stephan Günnemann. "Ab-Initio Potential Energy Surfaces by Pairing GNNs with Neural Wave Functions." In *International Conference on Learning Representations*, 2022.

[GPG17] Changran Geng, Harald Paganetti, and Clemens Grassberger. "Prediction of treatment response for combined chemo-and radiation therapy for non-small cell lung cancer patients using a bio-mathematical model." *Scientific reports*, **7**(1):13542, 2017.

[GSG17] Yupeng Gu, Yizhou Sun, and Jianxi Gao. "The Co-Evolution Model for Social Network Evolving and Opinion Migration." In *KDD'17*, 2017.

[GSP20] Daehoon Gwak, Gyuhyeon Sim, Michael Poli, Stefano Massaroli, Jaegul Choo, and Edward Choi. "Neural ordinary differential equations for intervention modeling." *arXiv preprint arXiv:2010.08304*, 2020.

[GVK22] Jayesh Gupta, Sai Vemprala, and Ashish Kapoor. "Learning Modular Simulations for Homogeneous Systems." *Advances in Neural Information Processing Systems (Neurips)*, **35**:14852–14864, 2022.

[GZL22] Jiayan Guo, Peiyan Zhang, Chaozhuo Li, Xing Xie, Yan Zhang, and Sunghun Kim. "Evolutionary preference learning via graph nested gru ode for session-based recommendation." In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management (CIKM)*, pp. 624–634, 2022.

[HCY19] Junheng Hao, Muhao Chen, Wenchao Yu, Yizhou Sun, and Wei Wang. "Universal Representation Learning of Knowledge Bases by Jointly Embedding Instances and Ontological Concepts." In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, p. 1709–1719, 2019.

[HDW20a] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. "Lightgcn: Simplifying and powering graph convolution network for recommendation." In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pp. 639–648, 2020.

[HDW20b] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. "Heterogeneous Graph Transformer." In *Proceedings of the 2020 World Wide Web Conference*, 2020.

[Het00] H. W Hethcote. "The mathematics of infectious diseases." In *SIAM review*. 2000.

[HFL18] R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. "Learning deep representations by mutual information estimation and maximization." *arXiv preprint arXiv:1808.06670*, 2018.

[HHN19] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. "Variational Graph Recurrent Neural Networks." In *Advances in Neural Information Processing Systems 32*, pp. 10701–10711. 2019.

[HHZ24] Zijie Huang, Jeehyun Hwang, Junkai Zhang, Jinwoo Baik, Weitong Zhang, Quanquan Gu, Dominik Wodarz, Yizhou Sun, and Wei Wang. "Causal Graph ODE: Continuous Treatment Effect Modeling in Multi-agent Dynamical Systems." In *The Web Conference (WWW)*, 2024.

[HLJ22] Zijie Huang, Zheng Li, Haoming Jiang, Tianyu Cao, Hanqing Lu, Bing Yin, Karthik Subbian, Yizhou Sun, and Wei Wang. "Multilingual Knowledge Graph Completion with Self-Supervised Adaptive Graph Alignment." In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.

[HSW20a] Zijie Huang, Yizhou Sun, and Wei Wang. "Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations." In *Advances in Neural Information Processing Systems (Neurips)*, 2020.

[HSW20b] Zijie Huang, Yizhou Sun, and Wei Wang. "Learning Continuous System Dynamics from Irregularly-Sampled Partial Observations." In *Neurips'20*, 2020.

[HSW21a] Z. Huang, Y. Sun, and W. Wang. "Coupled Graph ODE for Learning Interacting System Dynamics." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2021.

[HSW21b] Zijie Huang, Yizhou Sun, and Wei Wang. "Coupled Graph ODE for Learning Interacting System Dynamics." In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2021.

[HSW21c] Zijie Huang, Yizhou Sun, and Wei Wang. "Coupled graph ode for learning interacting system dynamics." In *The 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD)*, 2021.

[HSW23a] Zijie Huang, Yizhou Sun, and Wei Wang. "Generalizing Graph ODE for Learning Complex System Dynamics across Environments." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, p. 798–809, 2023.

[HSW23b] Zijie Huang, Yizhou Sun, and Wei Wang. "Generalizing graph ode for learning complex system dynamics across environments." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 798–809, 2023.

[HWH23] Zijie Huang, Daheng Wang, Binxuan Huang, Chenwei Zhang, Jingbo Shang, Yan Liang, Zhengyang Wang, Xian Li, Christos Faloutsos, Yizhou Sun, and Wei Wang. "Concept2Box: Joint Geometric Embeddings for Learning Two-View Knowledge Graphs." In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10105–10118. Association for Computational Linguistics, 2023.

[HYH20] I. Huh, E. Yang, S. J. Hwang, and J. Shin. "Time-Reversal Symmetric ODE Network." In *Advances in Neural Information Processing Systems*, 2020.

[HYH23] Kaiqiao Han, Yi Yang, Zijie Huang, Yang Yang, Lifang He, Liang Zhan, Yizhou Sun, Wei Wang, and Carl Yang. "BrainODE: Dynamic Brain Network Analysis via Graph-Aided Neural Ordinary Differential Equations." *in submission*, 2023.

[HZG24] Zijie Huang, Wanjia Zhao, Jingdong Gao, Ziniu Hu, Xiao Luo, Yadi Cao, Yizhou Sun, and Wei Wang. "TANGO: Time-Reversal Latent GraphODE for Multi-Agent Dynamical Systems.", 2024.

[HZGrd] Zijie Huang, Wanjia Zhao, Jingdong Gao, Ziniu Hu, Xiao Luo, Yadi Cao, Yuanzhou Chen, Yizhou Sun, and Wei Wang. "TANGO: Time-Reversal Latent GraphODE for Multi-Agent Dynamical Systems." In *The Symbiosis of Deep Learning and Differential Equations III, Neurips*, 2023. Best Paper Award.

[HZR15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition." *arXiv preprint arXiv:1512.03385*, 2015.

[JHH20] Amin Javari, Zhankui He, Zijie Huang, Raj Jeetu, and Kevin Chen-Chuan Chang. "Weakly Supervised Attention for Hashtag Recommendation Using Graph Data." In *Proceedings of The Web Conference 2020*, WWW '20, 2020.

[JHL23a] Song Jiang, Zijie Huang, Xiao Luo, and Yizhou Sun. "CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*, 2023.

[JHL23b] Song Jiang, Zijie Huang, Xiao Luo, and Yizhou Sun. "CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems." In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2023.

[JHL23c] Song Jiang, Zijie Huang, Xiao Luo, and Yizhou Sun. "CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems." In *29th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[Jon24] John Edward Jones. "On the determination of molecular fields.—I. From the variation of the viscosity of a gas with temperature." *Proceedings of the Royal Society of London. Series A, Containing Papers of a Mathematical and Physical Character*, **106**(738):441–462, 1924.

[JS22] Song Jiang and Yizhou Sun. "Estimating Causal Effects on Networked Observational Data via Representation Learning." In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 852–861, 2022.

[jur19] “A Sneak Peek at 19 Science Simulations for the Summit Supercomputer in 2019.”, 2019.

[KFW18a] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. “Neural Relational Inference for Interacting Systems.” *arXiv preprint arXiv:1802.04687*, 2018.

[KFW18b] Thomas Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel. “Neural Relational Inference for Interacting Systems.” *arXiv preprint arXiv:1802.04687*, 2018.

[KK13] S. Kim and S. J. Karrila. *Microhydrodynamics: principles and selected applications*. Courier Corporation, 2013.

[KTK19] Georgia Koppe, Hazem Toutounji, Peter Kirsch, Stefanie Lis, and Daniel Durstewitz. “Identifying nonlinear dynamical systems via generative recurrent neural networks with applications to fMRI.” *PLoS computational biology*, **15**(8):e1007263, 2019.

[KW14] Diederik P. Kingma and Max Welling. “Auto-Encoding Variational Bayes.” In *ICLR*, 2014.

[KW17] Thomas N. Kipf and Max Welling. “Semi-Supervised Classification with Graph Convolutional Networks.” In *ICLR’17*, 2017.

[Lab19] The Oak Ridge National Laboratory. “A Sneak Peek at 19 Science Simulations for the Summit Supercomputer.”, 2019.

[LCH19] Chengjiang Li, Yixin Cao, Lei Hou, Jiaxin Shi, Juanzi Li, and Tat-Seng Chua. “Semi-supervised Entity Alignment via Joint Knowledge Embedding Model and Cross-graph Model.” In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 2723–2732, 2019.

[lDH17] Jundong li, Harsh Dani, Xia Hu, Jiliang Tang, Yi Chang, and Huan Liu. “Attributed Network Embedding for Learning in a Dynamic Environment.” In *CIKM’17*, 2017.

[LH] Ilya Loshchilov and Frank Hutter. “Decoupled Weight Decay Regularization.” In *International Conference on Learning Representations*.

[LHS23] Han Liu, Zijie Huang, Samuel S. Schoenholz, Ekin D. Cubuk, Morten M. Smedskjaer, Yizhou Sun, Wei Wang, and Mathieu Bauchy. “Learning molecular dynamics: predicting the dynamics of glasses by a machine learning simulator.” In *Material Horizons*, 2023.

[LIJ15] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, and Soren Auer. “Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia.” In *Semantic Web*, pp. 167–195, 2015.

[Lim18] Bryan Lim. “Forecasting treatment responses over time using recurrent marginal structural networks.” *advances in neural information processing systems*, **31**, 2018.

[LKB19] J. Liu, A. Kumar, J. Ba, J. Kiros, and K. Swersky. “Graph normalizing flows.” *Advances in Neural Information Processing Systems*, **32**, 2019.

[LLC22] Yupu Lu, Shijie Lin, Guanqi Chen, and Jia Pan. “ModLaNets: Learning Generalisable Dynamics via Modularity and Physical Inductive Bias.” In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pp. 14384–14397. PMLR, 17–23 Jul 2022.

[LNV21] Justin Lovelace, Denis Newman-Griffis, Shikhar Vashishth, Jill Fain Lehman, and Carolyn Rosé. “Robust Knowledge Graph Completion with Stacked Convolutions and a Student Re-Ranking Network.” In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1016–1029, 2021.

[LPZ02] Harvard Lomax, Thomas H Pulliam, David W Zingg, and TA Kowalewski. “Fundamentals of computational fluid dynamics.” *Appl. Mech. Rev.*, **55**(4):B61–B61, 2002.

[LR98] Jeroen SW Lamb and John AG Roberts. “Time-reversal symmetry in dynamical systems: a survey.” *Physica D: Nonlinear Phenomena*, pp. 1–39, 1998.

[LSD21] Bill Yuchen Lin, Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Xiang Ren, and William Cohen. “Differentiable Open-Ended Commonsense Reasoning.” In *Proceedings of the 2021*

Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pp. 4611–4625, 2021.

[LSW23] Remi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Ferran Alet, Suman Ravuri, Timo Ewalds, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Oriol Vinyals, Jacklynn Stott, Alexander Pritzel, Shakir Mohamed, and Peter Battaglia. "Learning skillful medium-range global weather forecasting." *Science*, **382**(6677):1416–1421, 2023.

[LWH23] Xiao Luo, Haixin Wang, Zijie Huang, Huiyu Jiang, Abhijeet Sadashiv Gangan, Song Jiang, and Yizhou Sun. "CARE: Modeling Interacting Dynamics Under Temporal Environmental Variation." In *Thirty-seventh Conference on Neural Information Processing Systems (Neurips)*, 2023.

[LWZ19a] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. "Propagation Networks for Model-Based Control Under Partial Observation." In *ICRA*, 2019.

[LWZ19b] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. "Propagation Networks for Model-Based Control Under Partial Observation." In *ICRA*, 2019.

[LWZ19c] Yunzhu Li, Jiajun Wu, Jun-Yan Zhu, Joshua B Tenenbaum, Antonio Torralba, and Russ Tedrake. "Propagation networks for model-based control under partial observation." In *2019 International Conference on Robotics and Automation (ICRA)*, pp. 1205–1211. IEEE, 2019.

[LXM22] C. Li, F. Xia, R. Martín-Martín, M. Lingelbach, S. Srivastava, B. Shen, K. E. Vainio, C. Gokmen, G. Dharan, T. Jain, A. Kurenkov, K. Liu, H. Gweon, J. Wu, L. Fei-Fei, and S. Savarese. "iGibson 2.0: Object-Centric Simulation for Robot Learning of Everyday Household Tasks." In *Proceedings of the 5th Conference on Robot Learning*, 2022.

[LYH23] Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. "HOPE: High-Order Graph ODE for Modeling Interacting Dynamics." In *Proceedings of the 40th International Conference on Machine Learning (ICML)*, ICML'23, 2023.

[LYL21] Zongwei Liang, Junan Yang, Hui Liu, and Keju Huang. "A Semantic Filter Based on Relations for Knowledge Graph Completion." In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 7920–7929, 2021.

[LZS23] Han Liu, Huang Zijie, Samuel S Schoenholz, Ekin D Cubuk, Morten M Smedskjaer, Yizhou Sun, Wei Wang, and Mathieu Bauchy. "Learning molecular dynamics: predicting the dynamics of glasses by a machine learning simulator." *Materials Horizons*, **10**(9):3416–3428, 2023.

[MDH22] Jing Ma, Yushun Dong, Zheng Huang, Daniel Mietchen, and Jundong Li. "Assessing the Causal Impact of COVID-19 Related Policies on Outbreak Dynamics: A Case Study in the US." In *Proceedings of the ACM Web Conference 2022*, WWW '22, p. 2678–2686, 2022.

[MFF22] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. "Causal Transformer for Estimating Counterfactual Outcomes." In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 15293–15329. PMLR, 2022.

[MMD16] F Mokalled, L Mangani, and M Darwish. "The Finite Volume Method on Computational Fluid Dynamics.", 2016.

[MSH19] Schober Michael, Särkkä Simo, and Philipp Hennig. "A probabilistic model for the numerical solution of initial value problems." In *Statistics and Computing*, pp. 99–122. 2019.

[MWY22] Jing Ma, Mengting Wan, Longqi Yang, Jundong Li, Brent Hecht, and Jaime Teevan. "Learning causal effects on hypergraphs." In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 1202–1212, 2022.

[NQ22] Ruiqi Ni and Ahmed H Qureshi. "Ntfields: Neural time fields for physics-informed robot motion planning." *arXiv preprint arXiv:2210.00120*, 2022.

[PCL21] Xutan Peng, Guanyi Chen, Chenghua Lin, and Mark Stevenson. "Highly Efficient Knowledge Graph Embedding Learning with Orthogonal Procrustes Analysis." In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2364–2375, 2021.

[Pea09] Judea Pearl. *Causality*. Cambridge university press, 2009.

[PFS20] Tobias Pfaff, Meire Fortunato, Alvaro Sanchez-Gonzalez, and Peter W Battaglia. "Learning mesh-based simulation with graph networks." *arXiv preprint arXiv:2010.03409*, 2020.

[PFS21] T. Pfaff, M. Fortunato, A. Sanchez-Gonzalez, and P. Battaglia. "Learning Mesh-Based Simulation with Graph Networks." In *International Conference on Learning Representations*, 2021.

[PMP19a] M. Poli, S. Massaroli, J. Park, A. Yamashita, H. Asama, and J. Park. "Graph neural ordinary differential equations." *arXiv preprint arXiv:1911.07532*, 2019.

[PMP19b] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. "Graph Neural Ordinary Differential Equations." *arXiv*, 2019.

[Poz01] C. Pozrikidis. "Interfacial dynamics for Stokes flow." *Journal of Computational Physics*, **169**(2):250–301, 2001.

[QZD20] Liang Qu, Huaisheng Zhu, Qiqi Duan, and Yuhui Shi. "Continuous-time link prediction via temporal dependent graph neural network." In *The Web Conference (WWW)*, pp. 3026–3032, 2020.

[RCD19a] Y. Rubanova, R. T. Chen, and D. K. Duvenaud. "Latent ordinary differential equations for irregularly-sampled time series." In *Advances in Neural Information Processing Systems*, 2019.

[RCD19b] Yulia Rubanova, Ricky T. Q. Chen, and David K Duvenaud. "Latent Ordinary Differential Equations for Irregularly-Sampled Time Series." In *Advances in Neural Information Processing Systems 32*, pp. 5320–5330. 2019.

[RHB00] James M Robins, Miguel Angel Hernan, and Babette Brumback. "Marginal structural models and causal inference in epidemiology.", 2000.

[RPK19] M. Raissi, P. Perdikaris, and G. E. Karniadakis. "Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations." *Journal of Computational physics*, **378**:686–707, 2019.

[Rub78] Donald B Rubin. "Bayesian inference for causal effects: The role of randomization." *The Annals of statistics*, pp. 34–58, 1978.

[SBC19] A. Sanchez-Gonzalez, V. Bapst, K. Cranmer, and P. Battaglia. "Hamiltonian Graph Networks with ODE Integrators." In *Advances in Neural Information Processing Systems*, 2019.

[SC15] Sprott and Julien Clinton. "Symmetric time-reversible flows with a strange attractor." *International Journal of Bifurcation and Chaos*, **25**(05):1550078, 2015.

[SDN19] Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space." In *International Conference on Learning Representations*, 2019.

[SGP20a] A. Sanchez-Gonzalez, J. Godwin, T. Pfaff, R. Ying, J. Leskovec, and P. W. Battaglia. "Learning to Simulate Complex Physics with Graph Networks." In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[SGP20b] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. "Learning to Simulate Complex Physics with Graph Networks." In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pp. 8459–8468, 2020.

[SGW92] T. Shinbrot, C. Grebogi, J. Wisdom, and J. A. Yorke. "Chaos in a double pendulum." *American Journal of Physics*, (6):491–499, 1992.

[SHZ18] Zequn Sun, Wei Hu, Qingheng Zhang, and Yuzhong Qu. "Bootstrapping Entity Alignment with Knowledge Graph Embedding." In *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*, p. 4396–4402, 2018.

[SIB22] Nabeel Seedat, Fergus Imrie, Alexis Bellot, Zhaozhi Qian, and Mihaela van der Schaar. "Continuous-Time Modeling of Counterfactual Outcomes Using Neural Controlled Differential Equations." In *International Conference on Machine Learning*, pp. 19497–19521. PMLR, 2022.

[SJ97] Hochreiter Sepp and Schmidhuber Jürgen. "Long Short-term Memory." *Neural computation*, 1997.

[SJC21] Harkanwar Singh, Prachi Jain, Soumen Chakrabarti, et al. "Multilingual Knowledge Graph Completion with Joint Relation and Entity Alignment." *arXiv preprint arXiv:2104.08804*, 2021.

[SLX21] Chence Shi, Shitong Luo, Minkai Xu, and Jian Tang. "Learning gradient fields for molecular conformation generation." In *International Conference on Machine Learning*, pp. 9558–9568. PMLR, 2021.

[SPA00] R.J. Spiteri, D.K. Pai, and U.M. Ascher. "Programming and control of robots by means of differential algebraic inequalities." *IEEE Transactions on Robotics and Automation*, **16**(2):135–145, 2000.

[SRB17] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. "A simple neural network module for relational reasoning." In *Advances in Neural Information Processing Systems 30*, pp. 4967–4976. 2017.

[SsF16] Sainbayar Sukhbaatar, arthur szlam, and Rob Fergus. "Learning Multiagent Communication with Backpropagation." In *Advances in Neural Information Processing Systems 29*, pp. 2244–2252. 2016.

[SSO20] Eduardo Hugo Sanchez, Mathieu Serrurier, and Mathias Ortner. "Learning disentangled representations via mutual information estimation." In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII 16*, pp. 205–221. Springer, 2020.

[SWG20a] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. "DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks." In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pp. 519–527, 2020.

[SWG20b] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. "DySAT: Deep Neural Representation Learning on Dynamic Graphs via Self-Attention Networks." In *WSDM'20*, 2020.

[SZH20] Zequn Sun, Qingheng Zhang, Wei Hu, Chengming Wang, Muhao Chen, Farahnaz Akrami, and Chengkai Li. “A Benchmarking Study of Embedding-Based Entity Alignment for Knowledge Graphs.” *Proc. VLDB Endow.*, p. 2326–2340, July 2020.

[tM20] IHME COVID-19 health service utilization forecasting team and Christopher JL Murray. “Forecasting COVID-19 impact on hospital bed-days, ICU-days, ventilator-days and deaths by US state in the next 4 months.” In *medRxiv preprint :2020.03.27.20043752*. 2020.

[Tol38] E. C. Tolman. “The Determiners of Behavior at a Choice Point.” *Psychological Review*, **45**(1):1–41, 1938.

[TYS20] Xianfeng Tang, Huaxiu Yao, Yiwei Sun, Charu Aggarwal, Prasenjit Mitra, and Suhang Wang. “Joint Modeling of Local and Global Temporal Dynamics for Multivariate Time Series Forecasting with Missing Values.” In *AAAI*. 2020.

[VCC18] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. “Graph Attention Networks.” *ICLR’18*, 2018.

[VSP17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. “Attention is All you Need.” In *Advances in Neural Information Processing Systems 30*, pp. 5998–6008. 2017.

[VWT22] R. Valperga, K. Webster, D. Turaev, V. Klein, and J. Lamb. “Learning Reversible Symplectic Dynamics.” In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, 2022.

[WFD23] Hanchen Wang, Tianfan Fu, Yuanqi Du, and et.al. “Scientific discovery in the age of artificial intelligence.” In *Nature*, 2023.

[WHL21a] Ruijie Wang, Zijie Huang, Shengzhong Liu, Huajie Shao, Dongxin Liu, Jinyang Li, Tianshi Wang, Dachun Sun, Shuochao Yao, and Tarek Abdelzaher. “DyDiff-VAE: A Dynamic Variational Framework for Information Diffusion Prediction.” In *SIGIR’21*, 2021.

[WHL21b] Ruijie Wang, Zijie Huang, Shengzhong Liu, Huajie Shao, Dongxin Liu, Jinyang Li, Tianshi Wang, Dachun Sun, Shuochao Yao, and Tarek Abdelzaher. “DyDiff-VAE: A Dynamic

Variational Framework for Information Diffusion Prediction." In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, p. 163–172, 2021.

[WHW19] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. "Neural graph collaborative filtering." In *Proceedings of the international ACM SIGIR conference on Research and development in Information Retrieval*, pp. 165–174, 2019.

[WKM20] R. Wang, K. Kashinath, M. Mustafa, A. Albert, and R. Yu. "Towards physics-informed deep learning for turbulent flow prediction." In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[WPZ20] Man Wu, Shirui Pan, Chuan Zhou, Xiaojun Chang, and Xingquan Zhu. "Unsupervised domain adaptive graph convolutional networks." In *The Web Conference (WWW)*, pp. 1457–1467, 2020.

[WTD20] Spencer Woody, Mauricio Tec, Maytal Dahan, Kelly Gaither, Michael Lachmann, Spencer J. Fox, Lauren Ancel Meyers, and James Scott. "Projections for first-wave COVID-19 deaths across the U.S. using social-distancing measures derived from mobile phones." 2020.

[WWM22] S. Wen, H. Wang, and D. Metaxas. "Social ODE: Multi-agent Trajectory Forecasting with Neural Ordinary Differential Equations." In *European Conference on Computer Vision*, 2022.

[WWW20] Haiwen Wang, Ruijie Wang, Chuan Wen, Shuhao Li, Yuting Jia, Weinan Zhang, and Xinbing Wang. "Author Name Disambiguation on Heterogeneous Information Network with Adversarial Representation Learning." In *AAAI '20*, 2020.

[WXW20] Qinxia Wang, Shanghong Xie, Yuanjia Wang, and Zeng Donglin. "Survival-Convolution Models for Predicting COVID-19 Cases and Assessing Effects of Mitigation Strategies." In *Frontiers in Public Health*. 2020.

[WYW18] Ruijie Wang, Yuchen Yan, Jialu Wang, Yuting Jia, Ye Zhang, Weinan Zhang, and Xinbing Wang. "AceKG: A Large-Scale Knowledge Graph for Academic Data Mining." In *CIKM '18*, 2018.

[WZF14a] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge Graph Embedding by Translating on Hyperplanes." In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, p. 1112–1119, 2014.

[WZF14b] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. "Knowledge Graph Embedding by Translating on Hyperplanes." *Proceedings of the 28th AAAI Conference on Artificial Intelligence*, **28**, 2014.

[XHL19] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. "How Powerful are Graph Neural Networks?" In *ICLR'19*, 2019.

[XJH24] Fred Xu, Song Jiang, Zijie Huang, Xiao Luo, Shichang Zhang, Yuanzhou Chen, and Yizhou Sun. "FUSE: Measure-Theoretic Compact Fuzzy Set Representation for Taxonomy Expansion." In *The 62nd Annual Meeting of the Association for Computational Linguistics*, 2024.

[XQT20] Louis-Pascal Xhonneux, Meng Qu, and Jian Tang. "Continuous Graph Neural Networks." In *ICML'20*, 2020.

[XW23] Yucheng Xing and Xin Wang. "HDG-ODE: A Hierarchical Continuous-Time Model for Human Pose Forecasting." In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 14700–14712, 2023.

[XZK21] Jingwen Xu, Jing Zhang, Xirui Ke, Yuxiao Dong, Hong Chen, Cuiping Li, and Yongbin Liu. "P-INT: A Path-based Interaction Model for Few-shot Knowledge Graph Completion." In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 385–394, 2021.

[YHL19] Cagatay Yildiz, Markus Heinonen, and Harri Lahdesmaki. "ODE2VAE: Deep generative second order ODEs with Bayesian neural networks." In *Advances in Neural Information Processing Systems 32*, pp. 13412–13421. 2019.

[YKR22] Çağatay Yıldız, Melih Kandemir, and Barbara Rakitsch. "Learning interacting dynamical systems with latent Gaussian process ODEs." *Advances in Neural Information Processing Systems (Neurips)*, **35**:9188–9200, 2022.

[YML20] Liang Yao, Chengsheng Mao, and Yuan Luo. "KG-BERT: BERT for Knowledge Graph Completion." *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

[YRB21] Michihiro Yasunaga, Hongyu Ren, Antoine Bosselut, Percy Liang, and Jure Leskovec. "QA-GNN: Reasoning with Language Models and Knowledge Graphs for Question Answering." In *North American Chapter of the Association for Computational Linguistics (NAACL)*, 2021.

[YWG20] Chaoqi Yang, Ruijie Wang, Fangwei Gao, Dachun Sun, Jiawei Tang, and Tarek Abdelzaher. "Analyzing the Design Space of Re-opening Policies and COVID-19 Outcomes in the US.", 2020.

[YYH15a] Bishan Yang, Wen tau Yih, Xiaodong He, Jianfeng Gao, , and Li Deng. "Embedding entities and relations for learning and inference in knowledge bases." In *Proceedings of the 3th International Conference on Learning Representations (ICLR)*, 2015.

[YYH15b] Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. "Embedding entities and relations for learning and inference in knowledge bases." In *International Conference on Learning Representations (ICLR)*, 2015.

[ZGY16] L. Zhu, D. Guo, J. Yin, G. V. Steeg, and A. Galstyan. "Scalable Temporal Latent Space Inference for Link Prediction in Dynamic Social Networks." *IEEE Transactions on Knowledge and Data Engineering*, **28**(10):2765–2777, 2016.

[ZSH19] Qingheng Zhang, Zequn Sun, Wei Hu, Muhao Chen, Lingbing Guo, and Yuzhong Qu. "Multi-view Knowledge Graph Embedding for Entity Alignment." In *Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5429–5435, 2019.

[ZW20] C. Zang and F. Wang. "Neural dynamics on complex networks." In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2020.

[ZWH23] Huang Zijie, Daheng Wang, Binxuan Huang, Chenwei Zhang, Jingbo Shang, Yan Liang, Zhengyang Wang, Xian Li, Christos Faloutsos, Yizhou Sun, and Wei Wang. "Concept2Box:

Joint Geometric Embeddings for Learning Two-View Knowledge Graphs." In *Findings of the Association for Computational Linguistics (ACL)*, pp. 10105–10118, 2023.

[ZWS20] Qi Zhu, Hao Wei, Bunyamin Sisman, Da Zheng, Christos Faloutsos, Xin Luna Dong, and Jiawei Han. "Collective Multi-Type Entity Alignment Between Knowledge Graphs." In *Proceedings of The Web Conference 2020*, 2020.

[ZWX20] Difan Zou, Lingxiao Wang, Pan Xu, Jinghui Chen, Weitong Zhang, and Quanquan Gu. "Epidemic Model Guided Machine Learning for COVID-19 Forecasts in the United States." In *medRxiv preprint :2020.05.24.20111989*. 2020.

[ZXL17] Hao Zhu, Ruobing Xie, Zhiyuan Liu, and Maosong Sun. "Iterative Entity Alignment via Joint Knowledge Embeddings." In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI)*, p. 4258–4264, 2017.

[ZY23] Yunhao Zhang and Junchi Yan. "Crossformer: Transformer Utilizing Cross-Dimension Dependency for Multivariate Time Series Forecasting." In *International Conference on Learning Representations*, 2023.

[ZZL21] Wei Zhu, Haitian Zheng, Haofu Liao, Weijian Li, and Jiebo Luo. "Learning bias-invariant representation by cross-sample mutual information minimization." In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15002–15012, 2021.

[ZZW21] Tianxiang Zhao, Xiang Zhang, and Suhang Wang. "Graphsmote: Imbalanced node classification on graphs with graph neural networks." In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pp. 833–841, 2021.

[ZZZ20] Zhao Zhang, Fuzhen Zhuang, Hengshu Zhu, Zhiping Shi, Hui Xiong, and Qing He. "Relational Graph Neural Network with Hierarchical Attention for Knowledge Graph Completion." *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 9612–9619, 2020.