UNIVERSITY OF CALIFORNIA
Los Angeles

Towards Causally-Aware Dynamical System Prediction

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

by

Song Jiang

2024

ABSTRACT OF THE DISSERTATION

Towards Causally-Aware Dynamical System Prediction

by

Song Jiang
Doctor of Philosophy in Computer Science
University of California, Los Angeles, 2024
Professor Yizhou Sun, Chair

Understanding and predicting the dynamics is one fundamental problem that supports various real-world applications. Deep learning dynamical models such as recurrent neural networks (RNNs) and Transformer show powerful expressiveness in modeling sequential data. However, pure deep learning models lack appropriate inductive bias for dynamics, which limits their potential for more accurate dynamic predictions.

This dissertation aims to enhance deep neural networks' capability of modeling dynamics. My research starts by injecting physical law as prior knowledge into deep nets, with the finding that such prior knowledge shapes the predicted trajectory desirably and therefore achieves more accurate forecasting. However, such physical law is not available for more general and complicated dynamics, such as retail time series, and energy consumption sequence. To this end, we propose to use the Fourier series instead of task-specific rules as a more general inductive bias to capture the periodicity. Unfortunately, either specific physical law or general periodic series still just learns the association between historical observations and the future series. However, answering counterfactual questions like "Would the community protection be better had a different group of people gotten vaccinated first?" is one key problem for decision-making in dynamical systems. A dynamical is naturally represented by a graph, where units are nodes and the interactions among them are edges. The second part of my research focuses on how to answer causal questions on graphs and then extend to general dynamical systems.

The dissertation of Song Jiang is approved.

Baharan Mirzasoleiman

Wei Wang

Adnan Youssef Darwiche

Yizhou Sun, Committee Chair

University of California, Los Angeles

2024

To my beloved family

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

The PhD journey has been a lifelong treasure for me. I could not have reached this point without the support and love from many people.

First of all, I extend my heartfelt gratitude to my advisor, Prof. Yizhou Sun, for her invaluable guidance and unconditional support throughout my PhD journey. Her enthusiasm and mentality have influenced my personal and professional growth. It is my luck and blessing to have worked with Prof. Sun.

Second, I would like to express my sincere thanks to my thesis committee members, Prof. Wei Wang, Prof. Adnan Darwiche, and Prof. Baharan Mirzasoleiman. Their insightful feedback and suggestions have greatly enhanced the comprehensiveness of my thesis.

Third, I sincerely appreciate the support and guidance of my mentors and collaborators from all my internships, including Zahra Shakeri, Aaron Chan, Maziar Sanjabi, Hamed Firooz, Yinglong Xia, Bugra Akyildiz, Jinchao Li, Qifan Wang, Asli Celikyilmaz, Koustuv Sinha, Yan Xie, Jiwen Ren, Yan Zhu, Tahin Syed, Xuan Zhu, Joshua Levy, Boris Aronchik, Yuxiang Xie, and Xiaolin Shi. Their assistance was instrumental not only in the success of my internship projects but also in shaping my career aspirations.

Next, I would like to thank all my labmates from the UCLA Data Mining Lab: Ting Chen, Yupeng Gu, Xuelu Shirley Chen, Yunsheng Bai, Junheng Hao, Kewei Vivian Cheng, Yewen Wang, Ziniu Hu, Zhiping Patricia Xiao, Xiusi Chen, Roshni Iyer, Zijie Huang, Shichang Zhang, Zongyue Qin, Derek Xu, Arjun Subramonian, Zeyuan Fred Xu, Xiao Luo, Yanqiao Zhu, Yijia Xiao, Xiaoxuan Mandy Wang, Weikai Li, Fang Sun, Zongyu Johnson Lin, Chenchen Ye, Haixin Wang, Qiyue Yao and Yujun Zhao. I am also deeply thankful to my collaborators, including Bernard Koch, Jacob Gates Foster, Ang Li, Hanjia Lyu, Weilin Cong and Jianfeng Chi.

Finally, I extend my deepest gratitude to my parents, and my friends: Yubing Bai, Xiaotian Han, Tianmu Gao, and Quan Zheng. Their consistent support and unconditional

love have been invaluable to me. I love you all deeply.

Chapter 2 is a version of Song Jiang Bernard Koch, Yizhou Sun, HINTS: citation time series prediction for new publications via dynamic heterogeneous information network embedding. *Proceedings of the Web Conference 2021*, 2021, doi:10.1145/3442381.3450107. Benard Koch contributes to idea brainstorming, model development and experiment design, Yizhou Sun is PI.

Chapter 3 is a version of Song Jiang Tahin Syed, Xuan Zhu, Joshua Levy, Boris Aronchik, Yizhou Sun, Bridging Self-Attention and Time Series Decomposition for Periodic Forecasting. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022, doi:10.1145/3511808.3557077. Tahin Syed, Xuan Zhu, Joshua Levy, Boris Aronchik, Yizhou Sun all contribute to the idea discussion and experiment design.

Chapter 4 is a version of Song Jiang, Yizhou Sun, Estimating Causal Effects on Networked Observational Data via Representation Learning. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022, doi:10.1145/3511808.3557311. Yizhou Sun is PI.

Chapter 5 is a version of Song Jiang, Zijie Huang, Xiao Luo, Yizhou Sun, CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, doi:10.1145/3580305.3599272. Zijie Huang and Xiao Luo contribute to model discussion and experiment analysis. Yizhou Sun is PI.

VITA

2011–2015 B.S. in Communication Engineering, Shandong University

2015–2018 M.S. in Electronic engineering, Tsinghua University

2018–2024 Graduate Student Researcher & Teaching Fellow in Department of Computer Science, University of California, Los Angeles

PUBLICATIONS

Song Jiang Bernard Koch, Yizhou Sun, HINTS: citation time series prediction for new publications via dynamic heterogeneous information network embedding. *Proceedings of the Web Conference 2021*, 2021, doi:10.1145/3442381.3450107

Song Jiang Tahin Syed, Xuan Zhu, Joshua Levy, Boris Aronchik, Yizhou Sun, Bridging Self-Attention and Time Series Decomposition for Periodic Forecasting. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022, doi:10.1145/3511808.3557077

Song Jiang, Yizhou Sun, Estimating Causal Effects on Networked Observational Data via Representation Learning. *Proceedings of the 31st ACM International Conference on Information and Knowledge Management*, 2022, doi:10.1145/3511808.3557311

Song Jiang, Zijie Huang, Xiao Luo, Yizhou Sun, CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems. *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023, doi:10.1145/3580305.3599272

Song Jiang, Qiyue Yao, Qifan Wang, Yizhou Sun, A Single Vector Is Not Enough: Taxonomy Expansion via Box Embeddings. *Proceedings of the ACM Web Conference 2023*, 2023, doi:10.1145/3543507.3583310

Song Jiang, Zahra Shakeri, Aaron Chan, Maziar Sanjabi, Hamed Firooz, Yinglong Xia, Bugra Akyildiz, Yizhou Sun, Jinchao Li, Qifan Wang, Asli Celikyilmaz, Resprompt: Residual Connection Prompting Advances Multi-Step Reasoning in Large Language Models. *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2024

CHAPTER 1

Introduction

1.1 Overview

The world is a dynamical system and it never stops evolving. Dynamical data is ubiquitous and essential in many practical applications. For example, predicting early citation patterns of scientific papers provides a guide for researchers to find the hidden gems from numerous publications, and predicting the disease diffusion process helps policymakers better allocate medical resources.

Deep learning based dynamical models, such as recurrent neural networks (Hochreiter and Schmidhuber, 1997; Chung et al., 2014a) and transformer (Vaswani et al., 2017), have been the mainstream solutions because of their powerful expressiveness on sequential-like data. However, deep neural networks are "black-box", limiting the interpretability of how and why the forecasting is made. Before the "deep learning era", transitional statistical methods, such as time series decomposition (Robert et al., 1990), model the dynamics with specific mechanistic factors. These mechanistic factors naturally capture and explain the causes of evolution. However, learning these factors usually requires strong assumptions and prior knowledge. The simple abstraction of such knowledge used in traditional methods can not fully capture the complicated dynamical pattern, and therefore leads to a low accuracy of prediction. The first part of my research aims to inject mechanistic factors into deep models as inductive bias to boost the dynamical modeling. Specifically, for dynamical processes that can be described by physical laws, we directly inject such physical laws as prior, and for more general dynamics that are difficult to explicitly be modeled by physical laws, we use the Fourier series to capture the

periodicity in dynamics. Although such prior knowledge can improve the performance, models still learn the association between historical observations and forecasting, and therefore can not answer causal questions like "Would the community protection be better had a different group of people gotten vaccinated first?" However, such causal questions are crucial to measure the impacts of certain interventions in dynamical systems. The second part of my research, therefore, focuses on causal reasoning in dynamical systems, i.e., estimating the causal effects. Since the units in a dynamical system mutually affect each other and evolve collectively, it is natural to represent a dynamical system by a graph, in which nodes are the units and edges capture their interactions. To answer causal questions in dynamical systems, we first study causal effects estimation on graphs, and then extend it into the dynamical context.

1.2 Dissertation Contributions

The primary goal of this dissertation is to study more powerful dynamical prediction, which is accurate, interpretable, and able to answer causal questions. The major contributions are listed as follows:

- **Physical law as mechanistic factors.** Using citation count prediction for newly published papers as a running example, this dissertation demonstrates that injecting physical laws as inductive bias enhances the accuracy of dynamics modeling.
- **General periodic function as mechanistic factors.** Physical law is not always available. However, we observe that many dynamics has some repeatable patterns over time, i.e., periodicity. This dissertation shows that the Fourier series can be a general alternative for dynamics that exhibit periodic patterns.
- **Counterfactual inference on static graphs.** The units of a dynamical system are mutually affected and therefore graph is naturally used to model a dynamical system. As a prerequisite of counterfactual estimation in dynamical systems, this dissertation first studies causal effects estimation on static graphs.

- **Causal effects estimation in dynamical systems.** Finally, this dissertation extends causal effects estimation from static graphs into general dynamical systems.

1.3 Dissertation Outline

The remaining of this dissertation is organized as: 1) Chapter 2 introduces injecting physical law as mechanistic factors for more accurate dynamics prediction (Jiang et al., 2021); 2) Chapter 3 studies using the Fourier series as general prior for dynamical prediction (Jiang et al., 2022); 3) Chapter 4 discusses counterfactual inference on static graphs via representation learning (Jiang and Sun, 2022); 4) Chapter 5 finally presents causal effects estimation in dynamical systems (Jiang et al., 2023); 5) Chapter 6 summarizes this dissertation and discusses thoughts on future directions.

CHAPTER 2

HINTS: Citation Time Series Prediction for New Publications via Dynamic Heterogeneous Information Network Embedding

2.1 Introduction

Predicting the “impact" of scientific research is crucial for authors to decide what to study and where to submit their research, for readers or recommender systems to identify important/relevant contributions in a vast scientific literature, and for funding agencies to identify promising young scientists and fields for future support. Because impact is hard to quantify, citation counts of scientific papers are often used as an approximation (Yan et al., 2011; Sinatra et al., 2016; Evans and Reimer, 2009).

To predict future citations, previous works (Wang et al., 2013; Shen et al., 2014; Xiao et al., 2016; Liu et al., 2017) have often relied on observed citations (i.e., leading citations values) in the first few years after publication. These leading value-based solutions have taken both parametric and machine learning approaches. (Wang et al., 2013; Shen et al., 2014; Xiao et al., 2016) propose formal models to encode assumptions and prior knowledge about how papers are cited (e.g., that citation trajectories follows a log-normal distribution). They then use leading citations for the first several years after a paper is published to infer paper-specific parameters to predict long-term citation counts. Machine learning approaches have used representation learning via recurrent neural networks (RNN) to automatically capture complex citation patterns from leading values, followed by another RNN as decoder to make predictions (Abrishami and Aliakbary, 2019; Yuan

et al., 2018).

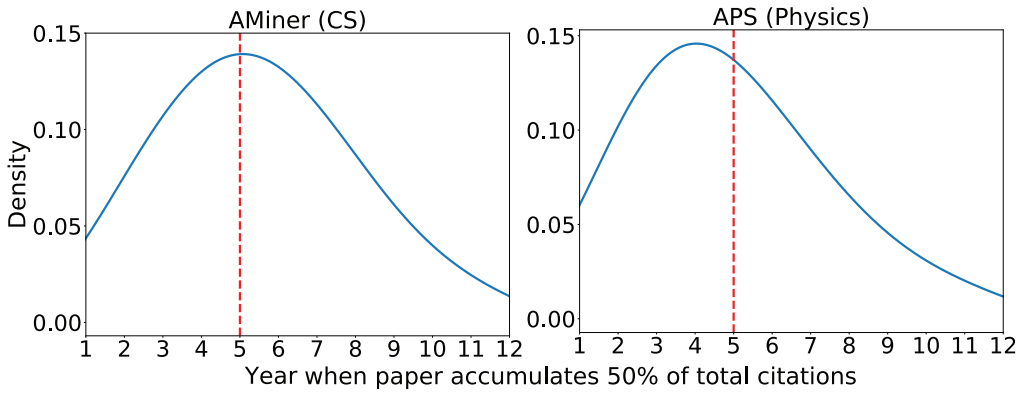

Figure 2.1: Year from publication when an academic paper has accumulated more than 50% of cumulative citations. Citations only counted up until 12 years after publication. Density of distribution shown in blue. Median shown in red. Zero citation papers removed. Left: 351,926 Computer Science papers (AMiner) published from 2000 to 2005; Right: 71,142 Physics papers (APS) published from 1995 to 2000.

A significant problem for these approaches is that many scientific papers have peak impact in the first few years after publication, when leading values are not yet available. For example in both Computer Science and Physics, we find that half of all cited papers accumulate the majority of their citations within five years of publication (Fig. 2.1). In fields with very fast publication rates (e.g., machine learning) it is not practical to wait three to five years before predicting impact. In this paper, we therefore tackle a new challenge: generating citation time series for newly-published papers *without any leading values*. To the best of our knowledge, we are the first to focus on predicting citation time series from the time of publication event. While we focus on scientific impact prediction, this “cold start" issue is common to many time series prediction tasks where earlier outcomes are more critical than later ones (e.g., in-links flow of new webpages, revenue streams for start-ups).

One way to avoid relying on leading values for impact prediction is to leverage clues visible to domain experts before they even read the paper. By reading the title, abstract, and bibliography, researchers can identify whether the paper is about a hot topic in their field. Within the author list, they can identify productive labs and reputable researchers. Papers in prestigious venues are also likely to be higher quality. To leverage

these "metadata" for long-term citation prediction, previous studies (Yan et al., 2011, 2012; Dong et al., 2016; Yu et al., 2014) have manually designed complex features. However, feature engineering is time-consuming, non-transferable and rarely complete. Moreover, these approaches rarely use the additional information encoded in the relationships between metadata, or their historical temporal trends. Using historical and relational context, domain experts can identify not just popular topics and reputable authors, but also trending topics and "rising star" researchers.

To leverage all of the predictive "hints" available to domain experts before reading the paper, we encode papers, authors, topics, and venues in a dynamic heterogeneous information network (DHIN). A DHIN captures not just a paper's metadata, but also the relationships between those metadata and their historical trends. This additional information allows us to predict citation counts without leading values, using our proposed end-to-end framework called ***HINTS*** (Heterogeneous Information Network to Time Series).

Composed of three modules, HINTS translates temporal and relational information from a DHIN before publication into a citation time series after publication. In the first module, we use a temporally-aligned GNN which concurrently learns effective embeddings for all nodes in each year's heterogeneous bibliographic network. Because static GNNs (Kipf and Welling, 2017a; Schlichtkrull et al., 2018; Hamilton et al., 2017a) do not preserve the underlying evolution of nodes in a bibliographical network, we apply a smooth regularizer to align the positions of nodes in the embedding space across time stamps. This approach allows us to capture the temporal trends of nodes (e.g., the rising star phenomenon). We show that this alignment regularization can be easily integrated into GNN models and improves predictive performance. The second module is a weighted imputation mechanism to estimate a sequence of embeddings for a new paper in the years prior to its publication. By aggregating the dynamics of the metadata, this imputation approximates the temporal trajectory of the new paper in the years before it is published. This learned temporal trajectory serves as "pseudo"-leading values for time series prediction after publication. The third module is a parametric generator based on (Wang

et al., 2013) that encodes prior assumptions about citation processes to predict long-term citation time series. Using an RNN followed by fully-connected layers, we transform the imputed paper embedding trajectory into the parameters of this generator. In conclusion, HINTS combines novel approaches for encoding DHINs (Module 1), synthesizing leading values prior to publication (Module 2), and formal modeling assumptions (Module 3) into an end-to-end framework that can predict citation times series from the time of publication. We note that this framework can be generically adapted to other "cold start" time series problems where pre-event temporal and relational data are available.

Empirically, we apply HINTS to two real-world academic datasets in Computer Science and Physics with extensive experiments. The results show that HINTS achieves significant and consistent improvements compared with baseline cold-start prediction approaches. Ablation studies on variants of HINTS also demonstrate the importance of each component of our proposed model.

Our contributions can be summarized as follows:

- We tackle a new, challenging "cold start" time series problem: predicting a new paper's citation time series without leading citation values.
- We propose a novel framework called HINTS that converts signals from a DHIN into signals for citation time series generation.
- We conduct extensive experiments on two real-world large-scale bibliographic datasets from different fields to demonstrate HINTS' effectiveness at impact prediction.

2.2 Problem Statement

In this section, we first introduce necessary definitions used throughout this paper. Then we present a formal definition of the new paper citation time series prediction problem.

2.2.1 Preliminaries

Definition 1. ***Heterogeneous information network.*** *A heterogeneous information network (HIN) (Sun et al., 2011b) is defined as a graph* $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ *with a node type mapping function* φ*:* $\mathcal{V} \rightarrow \mathcal{T}$ *and an edge type mapping function* $\phi : \mathcal{E} \rightarrow \mathcal{R}$*.* $\mathcal{T}$ *and* $\mathcal{R}$ *denote all the predefined types of node and edges, where* $|\mathcal{T}| + |\mathcal{R}| > 2$*.*

A bibliographic network (Sun et al., 2011a; Chen and Sun, 2017) is a type of heterogeneous information network. Scientific papers are the central nodes, with their metadata as neighbors. In our case, a paper's metadata includes referenced papers, authors, keywords, and a venue. Given these typed components, a network schema (Sun and Han, 2013) $\tilde{G} = (\mathcal{T}, \mathcal{R})$ can be utilized to abstract the node types and edge types at the meta level. The schema of our bibliographic network is shown in Fig. 2.2.

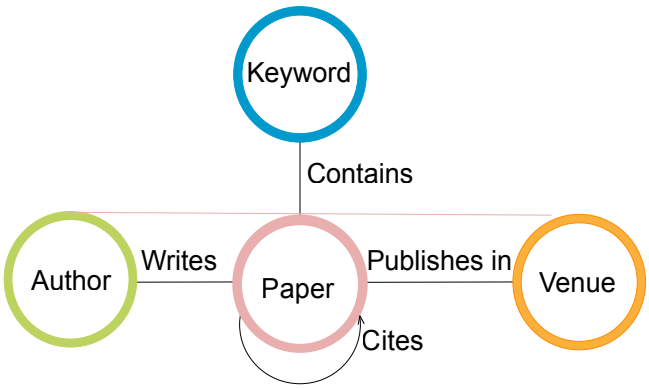

Figure 2.2: The schema of a bibliographic network. The nodes include paper, author, keyword, and venue, while their four relationships are: paper-cites-paper, author-writes-paper, paper-contains-keywords and paper-publishes in-venue.

In reality, bibliographic networks are constantly evolving. For instance, new papers will be published, new researchers will join the community, and new keywords will be created. These new entities will be added into the network, bringing new edges as well. Formally, given T timestamps, we define a dynamic heterogeneous information network as follows.

Definition 2. ***Dynamic heterogeneous information network.*** *A dynamic heterogeneous information network (DHIN) is a sequence of HIN snapshots, denoted by*

$\langle \mathcal{G}_t \rangle_{t=1}^{T} = \{\mathcal{G}^1, \mathcal{G}^2, \ldots, \mathcal{G}^T\}$, *where* $\mathcal{G}^t = (\mathcal{V}^t, \mathcal{E}^t)$ $(1 \leq t \leq T)$ *represents the heterogeneous graph snapshot and its corresponding node set and edge set at time* t.

In our case, a dynamic bibliographic network is a DHIN that consists of T sequential snapshots of the evolving bibliographic network in every calendar year. Thus $\mathcal{G}^t$ is a bibliographic graph snapshot at year t whose node and edge types are described in Fig. 2.2.

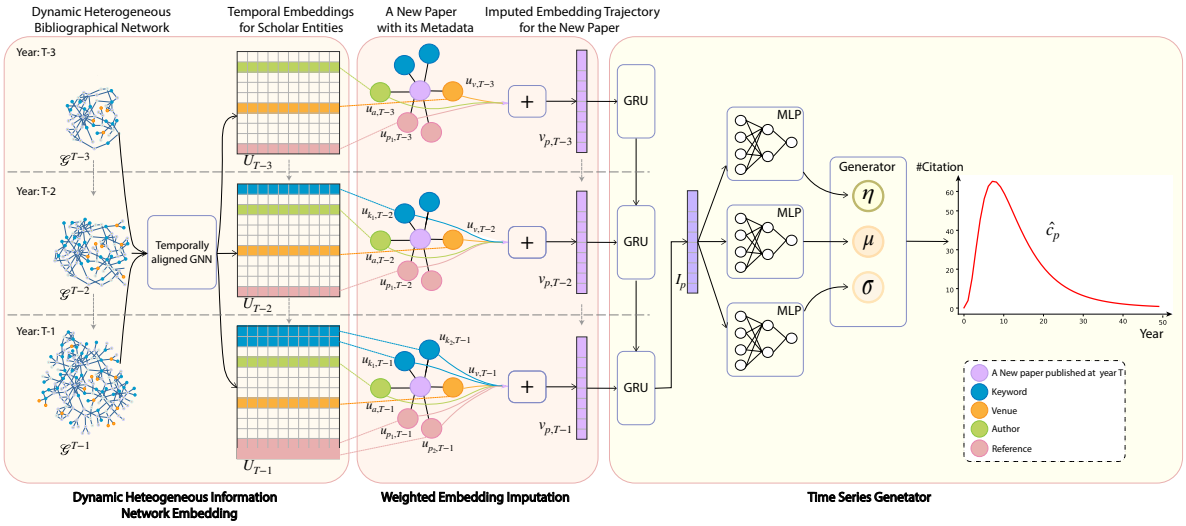

Figure 2.3: The overall architecture of HINTS. For a new paper published in year T, HINTS first learns temporally-aligned embeddings for each metadata neighbor in the dynamic heterogeneous bibliographical network that exists in the years preceding T (three in this example). An imputed embedding trajectory is built for the new paper (purple node) by computing a weighted average of the neighbors' embeddings. Note that some metadata nodes may not exist across all previous timesteps. (e.g., a new keyword proposed only one year ago). After temporal encoding by an RNN, the imputed embedding trajectory is transformed into three interpretable parameters, based on which HINTS generates the new paper's future citation time series.

2.2.2 Problem Definition

We now formalize the new paper citation time series prediction problem using a dynamic bibliographic network with network schema shown in Fig. 2.2. For each new paper p, we represent its citation counts over next L years after publication as a sequence $c_p = \{c_p^1, \ldots, c_p^l, \ldots, c_p^L\}$, where c_p^l denotes the citation count paper p will receive in the l-th year after publication.

The New Paper Citation Time Series Prediction Problem. Given a dynamic bibliographic network $\langle \mathcal{G}_t \rangle_{t=1}^{T-1}$ and a newly published paper p, our goal is to learn a function $f(\cdot)$ that maps paper p, given its context described by $\langle \mathcal{G}_t \rangle_{t=1}^{T-1}$, to its citation time series in future L years' after the publication year T, which is denoted as follows:

$$\left(\langle \mathcal{G}_t \rangle_{t=1}^{T-1}, p\right) \xrightarrow{f(\cdot)} \left\{c_p^1, \ldots, c_p^l, \ldots, c_p^L\right\}. \tag{2.1}$$

Note p is not in $\langle \mathcal{G}_t \rangle_{t=1}^{T-1}$, and no citations are received before T.

2.3 Proposed Framework: HINTS

In this section, we introduce our proposed framework, HINTS. We first describe the intuition behind why a DHIN provides key information for paper citation prediction. Then we breakdown the three modules used by HINTS to turn signals from DHIN into citation time series in detail. The overall framework of HINTS is shown in Fig. 2.3.

2.3.1 Motivation for HINTS

Although the reasons that some scientific papers achieve high impact are complicated, there are several cues identifiable by domain experts and network scientists that can predict impact. Ideally, representation learning of a DHIN should capture the following factors predictive of citation:

Topic. A paper is more likely to be cited by readers from a similar research area. Papers on hot or trending topics generally attract more attention and thus receive more citations (e.g., artificial intelligence in recent years). *Keywords* can serve as proxies for topic, because they are carefully selected by the authors to describe the new paper.

Author Status. Readers are more likely to search for papers by reputable authors, the advisees of reputable authors, or rising stars because of the high quality of their work.

Venue Status. Because of peer review, readers are more likely to assume that papers published in prestigious venues in their field are higher quality.

Bibliography. Highly influential papers do not start from scratch, instead, they stand on the shoulder of giants. (Dong et al., 2015) Citing high-impact papers is a baseline signal for relevance and potential impact (Uzzi et al., 2013).

Temporal Cues. Domain experts rely not only on the content of metadata when scanning papers, but also on knowledge of the temporal trends of these metadata. For example, a "rising star" might not only have a well-known advisor (relational context), but their network centrality will increase over time as they publish more influential papers (temporal context).

"Fitness". While domain experts can quickly identify the above cues, there are other intangible factors predictive of citation that are not encoded in metadata, such as the rigor of the work or the value of its contributions. For example, the graph convolution network (GCN) paper (Kipf and Welling, 2017a) was a milestone that allowed for new applications of deep learning to graphs. Network scientists have used the term "fitness" to generically capture these latent intangibles. (Wang et al., 2013; Ke et al., 2015)

The three modules of HINTS are designed to automatically detect these six types of information and leverage them for citation prediction. Note that the first five factors can be implicitly encoded in a DHIN connecting keywords, authors, venues and papers (i.e., metadata) as Fig. 2.2 over time. In the first module, we learn low-dimensional representation vectors of metadata across all time slices concurrently. The learned embeddings naturally capture topic, author status, venue status bibliography, and their trends. Because leading citation values do not exist for new papers, the second module in HINTS uses these node embeddings to impute embeddings in the years before a paper's publication by averaging the embeddings of it's metadata nodes in those years. Because some factors (e.g., author status) are more predictive of citation than others (e.g., bibliography), our framework learns weights to perform this averaging. The imputed embedding trajectory encodes all above factors and serves as the pseudo-leading values before publication. In the third module, we translate our imputed embeddings into the parameters of a parametric citation generator. This model, adapted from (Wang et al., 2013), encodes prior knowledge about citation processes and captures intangible factors

(i.e., "fitness") to predict citation counts in the years immediately following publication. We introduce the details of these three modules in following subsections.

2.3.2 Dynamic Heterogeneous Network Embedding via Temporally-aligned GNN

Given a static heterogeneous bibliographic network, several embedding methods (Chen and Sun, 2017; Dong et al., 2017; Gui et al., 2018; Schlichtkrull et al., 2018) have been proposed. Without loss of generality, we employ a relational graph convolution network (R-GCN) (Schlichtkrull et al., 2018) to encode nodes into low-dimensional vectors. R-GCN learns a relation-aware function that updates a node's representation by weighted aggregation of its neighbors according to the corresponding relation types. Formally, given a dynamic bibliographic network $\langle \mathcal{G}_t \rangle_{t=1}^{T}$, each $\mathcal{G}_t$ can be seen as a static network at time t. Let $h_{i,t}^{(k)} \in \mathcal{R}^{d(k)}$ be the embedding vector of node i in the k-th layer at time t, where $d(k)$ denotes the dimension for k-th layer, it will be updated via R-GCN as:

$$h_{i,t}^{(k+1)} = \sigma\Big(\sum_{r \in \mathcal{R}} \sum_{j \in N_{i,t}^{r}} \frac{1}{|N_{i,t}^{r}|} W_r^{(k)} h_{j,t}^{(k)} + W_0^{(k)} h_{i,t}^{(k)} \Big), \tag{2.2}$$

where $\mathcal{R}$ denotes the set of predefined types of edges/relations in the bibliographic network, while $N_{i,t}^{r}$ represents the set of neighbors of node i under relation $r \in \mathcal{R}$ at time t. $W_r^{(k)}$ and $W_0^{(k)}$ are the weight matrices of k-th layer, and σ is a non-linear activity function.

Temporally-aligned Graph Neural Network. R-GCN shows superior performance across many graph-related tasks, but extending it to dynamic settings is still challenging. An important contribution of HINTS is a method for aligning graph neural networks temporally. Because each individual bibliographical network describes a snapshot of the research community in the corresponding year, we first apply R-GCN annually to encode each bibliographical network separately. To ensure that each year's embeddings are in the same space (i.e., they are comparable), we make transformation weight matrices $W_r^{(k)}$ and $W_0^{(k)}$ shared across different timestamps. Second, unlike general dynamic networks where the characteristics of nodes may change rapidly (e.g., online social networks, or

protein-protein interaction networks), most entities in dynamic bibliographic information networks do not change too much within a short time frame. For example, a researcher's interests or venue's theme are likely to be similar in adjacent years. Motivated by this observation, we force the embeddings for the same entity in nearby years to be close to each other by introducing a temporal smoothing regularizer $L_{t,t+1}^{\langle time \rangle}$:

$$L_{t,t+1}^{\langle time \rangle} = \frac{1}{|V_t \cap V_{t+1}|} \sum_{i \in V_t \cap V_{t+1}} \|u_{i,t} - u_{i,t+1}\|_2^2, \tag{2.3}$$

where $u_{i,t}$ denotes entity i's embedding at year t, and V_t is the node set of $\mathcal{G}^t$. After multiple layers of R-GCN operations (we use 2 layers in practice), the final embedding matrix for $\mathcal{G}^t$ is denoted as $U_t \in \mathcal{R}^{N_t \times D}$, where N_t is the number of nodes in $\mathcal{G}^t$, and D is the dimension of the embeddings. Note that although we use R-GCN here, our HINTS framework can accommodate many graph neural networks, e.g., GCN (Kipf and Welling, 2017a), GAT (Veličković et al., 2018), HAN (Wang et al., 2019).

2.3.3 Weighted Embedding Imputation

To predict a newly published paper's long-term impact at time of publication, we need a distinct representation for it in the years before publication (i.e., pseudo-leading values). Although a paper is new, the metadata it is linked to may already exist in previous years' bibliographic networks. For example, a paper is usually published in a venue with a long track record, by co-authors with several previous publications, and with keywords that exist for quite a long time.

Using temporally-aligned GNN, we have already learned a vectorized representation for each metadata node which encodes both its historical trend and relationships with other nodes. Suppose one metadata node i appears in t-th year, and we have learned all its embeddings after t, which is a sequence denoted as $\{u_{i,t}, u_{i,t+1}, \ldots, u_{i,T}\}$. This sequence can be considered as an evolutionary trajectory for metadata i across time in the embedding space.

Given the above two preconditions, we can utilize the embedding sequences of a new paper's metadata neighbors to create an imputed embedding sequence that approximates its trajectory in the years prior to publication. One option to impute such embeddings is to directly use the same R-GCN operator defined in Eq. 2.2. However, this operator will result in an embedding in a different space due to additional transformation. Alternatively, inspired by the method in (Chen and Sun, 2017), we impute a new paper's embedding sequence by aggregating its metadata's embeddings with type-aware trainable weights to preserve the unequal contribution of different kinds of metadata.

Formally, for a new paper p, given its metadata set $N_{p,t}$ at time t and the metadata type set M, its imputed representation $v_{p,t}$ at time t would be derived from:

$$v_{p,t} = \sum_{m \in M} \sum_{i \in N_{p,t}^m} w_m * u_{i,t} / |N_{p,t}^m|. \tag{2.4}$$

By applying Eq. 2.4 in every timestamp, we can construct an imputed embedding sequence $V_p = \{v_{p,t}, v_{p,t+1}, ..., v_{p,T-1}\}$ for the new paper p, where t is the first year that p's metadata is observed. w_m is the weight for metadata type m, which will be learned in training stage. By integrating the temporal trends of its metadata, this imputed embedding sequence could be a good proxy for the hypothetical trend of the paper before publication.

2.3.4 Time Series Generator

Base on the imputed embedding sequence for paper V_p, we predict future impact by learning a function $g(\cdot) : V_p \rightarrow c_p^l$ that transforms network signals encoded in the imputation V_p to the new paper's long-term citation time series. One straightforward solution is directly "translating" the imputation sequence to a citation sequence with an encoder-decoder schema (e.g., seq2seq (Sutskever et al., 2014)). This approach has two major limitations. First it cannot generate flexible long-term citation predictions. Once learned, the decoder can only produce discrete sequences of predefined length. Furthermore, citation time series of scientific publications have been successfully modeled with some minimal assumptions (Wang et al., 2013; Sinatra et al., 2016). This solution,

however, fails to leverage this important prior knowledge in prediction.

Intuitively, a paper's impact fades gradually since new ideas and new research topics always appear and attract researchers' attention. Therefore, following (Wang et al., 2013), we model a new paper's citation trajectory as a log-normal survival function along time l, which is formalized as:

$$P_p(l) = \frac{1}{l\sqrt{2\pi}\sigma_p} \exp\left[-\frac{(\ln l - \mu_p)^2}{2\sigma_p^2} \right], \tag{2.5}$$

where μ_p describes the timestamp when paper p will reach its citation peak, and σ_p indicates the rate of decay of paper p's citations. Moreover, as discussed in 2.3.1, the "fitness" makes significant contributions to a paper's citations, so another parameter η_p is used to model it. The citation counts are thus positively related to η_p. Integrated across η_p, the predicted cumulative citation counts $\hat{C}_p^l$ of paper p at l-th year after publication can be generated by

$$\hat{C}_p^l = \alpha\big[\exp\left(\eta_p * \Phi(\frac{\ln l - \mu_p}{\sigma_p})\right) - 1\big] \tag{2.6}$$

where $\Phi(x)$ is

$$\Phi(x) = (2\pi)^{-1/2} \int_{-\infty}^{x} e^{-y^2/2}\, dy. \tag{2.7}$$

Following (Wang et al., 2013), a parameter α is added for the average number of references each new paper contains. Note that α is a global parameter that will be fixed during the model training process.

In Eq. 2.6, the three parameters η_p, μ_p and σ_p will be estimated for each new paper to generate its citation time series. In contrast with (Wang et al., 2013) who use the first several years' (e.g., 10 years) of leading citation values to infer parameters, we learn these three parameters by transforming the signals encoded in the DHIN prior to publication. Specifically, the imputed embedding sequence V_p is first temporally encoded into a single vector I_p through a recurrent neural network (RNN) with GRU (Chung et al., 2014b) to model the new paper's temporal trajectory, and then three fully connected layers decode

I_p into the three parameters respectively. (See Fig. 2.3)

It is worth noting that compared to the straightforward encoder-decoder method, our time series generator has three major advantages: (1) It can generate citation time series predictions in a flexible manner, i.e., assigning l a time length, (2) It leverages prior knowledge about citation patterns to achieve better performance, and (3) the three parameters are reasonably interpretable. We show this in detail in Sec. 2.4.

After accumulating L years citation counts for new paper p with Eq. 2.6, we transform the cumulative citation counts $\{\hat{C}_p^1, ..., \hat{C}_p^l, ..., \hat{C}_p^L\}$ into citation counts in each individual year and build the predicted citation time series as $\{\hat{c}_p^1, ..., \hat{c}_p^l, ..., \hat{c}_p^L\}$. This predicted sequence is then compared to the ground-truth with a Mean Square Error loss function. The loss function is defined as follows:

$$L^{\langle pred \rangle} = \frac{1}{P}\sum_{p=1}^{P}\frac{1}{L}\sum_{l=1}^{L}(\log(\hat{c}_p^l) - \log(c_p^l))^2, \tag{2.8}$$

where c_p^l is the ground-truth citation count of paper p at the l-th year after publication, and P is the total number of papers. Following (Cao et al., 2017; Li et al., 2017), we use log-scale for citation counts to smooth the contribution of each paper to the total loss regardless of its citation level. Note that many paper actually won't receive any citations, so we add 1 count as pseudo citation value before taking the log transformation.

2.3.5 Objective

By putting the citation time series generator objective and the temporal alignment regularizer aforementioned together, the overall objective function of HINTS $\mathcal{J}$ is defined as :

$$\mathcal{J} = L^{\langle pred \rangle} + \beta \sum_{t=1}^{T-1} L_{t,t+1}^{\langle time \rangle}. \tag{2.9}$$

For the alignment regularizer, in contrast to the method described in (Du et al., 2018) that updates embedding in chronological order, we align embeddings simultaneously across all timestamps such that the final aligned embedding U_t preserves every previous

embeddings instead of only U_{t-1}. A hyperparameter β ($\beta > 0$) is used to control the degree of alignment. All parameters in HINTS are updated by optimizing this objective.

2.4 Experiments

In this section, we evaluate HINTS' efficacy in predicting citation time series for new papers. We describe our experimental setting and then present the results compared with baselines. We also breakdown the framework in ablation studies and interpretation analyses to understand how HINTS works.

2.4.1 Experimental Setup

Datasets. We use two publicly-available bibliographic datasets in different fields for our analyses: the *AMiner* (Tang et al., 2008) Computer Science dataset[1] and the *American Physical Society (APS)* Physics dataset[2]. AMiner covers papers in major computer science venues. We use data from 2000-2009 to build the model and data from 2010-2015 for evaluation. The APS dataset covers publications in APS physics journals. Similarly, we use years 1995-2004 for training and years 2005-2010 for testing. The distribution of cumulative citation counts (the number of papers vs. citation counts) for papers in the test set is shown in Fig. 2.4. We note that a large number of papers are rarely cited after publication, so we take a down-sampling to balance the highly and lowly cited papers in model training.

We construct annual snapshots of the heterogeneous bibliographic network for both datasets in each year with the network schema shown in Fig. 2.2. Because the keywords are not explicitly provided in the original APS dataset, we generate them by combining unigrams and key phrases extracted from the title of each paper using the method proposed in (Shang et al., 2018).

[1] https://aminer.org/citation

[2] https://journals.aps.org/datasets

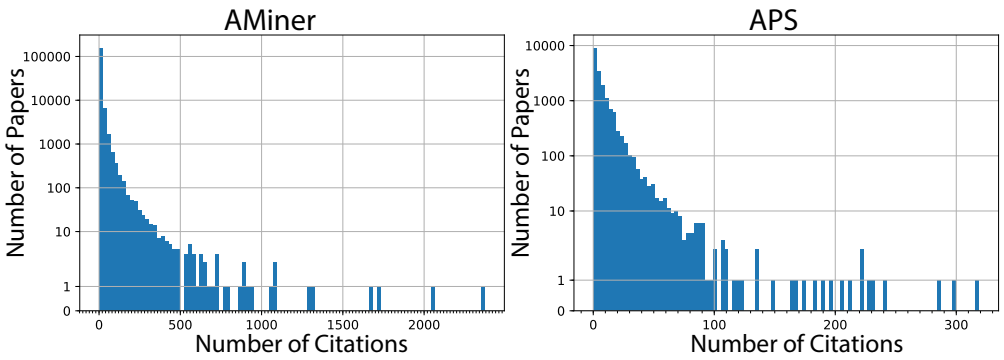

Figure 2.4: Distribution of cumulative citation counts within five years after publication for papers published in 2010 (AMiner), and 2005 (APS).

Baselines. Because the "cold start" citation time series prediction is a novel problem, there are no exact baselines for comparison, to our knowledge. Many state-of-the-art time series models (e.g., (Qin et al., 2017; Wang et al., 2018; Zhao et al., 2019; Manjunatha et al., 2003)) are not applicable for prediction immediately after publication because they require leading values. Instead, we compare HINTS to plausible alternatives. We consider three types of baselines: 1) Models that use manually constructed features 2) Models designed to predict information cascades (citation can be construed as an information cascade), and 3) two variants of HINTS. The specific approaches we consider are:

- **Gradient Boosting Machine (GBM):** We extract scientific features designed by (Yan et al., 2011; Castillo et al., 2007) except those that are not available in our problem setting or data, e.g., "first-year citation," h-index. We then use them to predict time series with XGBoost (Chen and Guestrin, 2016).

- **DeepCas (Li et al., 2017):** A state-of-the-art deep learning model for popularity prediction based on random walks across an information cascade graph. Because of the "cold start" setting, we directly use the ego network of a new paper in the snapshot of the DHIN in the publication year as the initial cascade graph.

- **HINTS-GCN:** A variant of HINTS using a homogeneous GCN (Kipf and Welling, 2017a) instead of R-GCN.

- **HINTS-Seq:** A variant of HINTS that replaces the citation generator module

with seq2seq (Sutskever et al., 2014), directly transforming imputed embedding sequences into discrete citation sequences.

- **HINTS:** Our proposed framework whose three modules are described in Sec. 2.3.

Evaluation Metrics. Following (Li et al., 2017; Cao et al., 2017), we use two log-scaled metrics to compare different models:
Mean Absolute Log-scaled Error (MALE)

$$MALE(c^l, \hat{c^l}) = \frac{1}{P}\sum_{i=1}^{P} |\log(\hat{c_p^l}) - \log(c_p^l)|. \tag{2.10}$$

Root Mean Square Log-scaled Error(RMSLE),

$$RMSLE(c^l, \hat{c^l}) = \sqrt{\frac{1}{P}\sum_{i=1}^{P}(\log(\hat{c_p^l}) - \log(c_p^l))^2}. \tag{2.11}$$

where c_p^l and $\hat{c_p^L}$ are the ground truth and predicted citation counts of paper p at the l-th year after publication respectively, and P is the total number of papers. As discussed in Sec. 2.3.4, we use log transformations because citation counts vary widely.

Implementation Details. We implement HINTS using Tensorflow 1.14. For the DHIN embedding module, we use two layers of GNN with 64 and 128 node (for both R-GCN and the variant with GCN). In the GNN layers, node features are randomly initialized in four different ranges according to the node type. The hidden dimension of HINT's RNN temporal encoder is set as 50. Finally, the hidden dimensions of three fully-connected layers are 20, 8, and 1, respectively. For HINTS-Seq, we also use GRU (Chung et al., 2014b) as the RNN decoder. The coefficient of alignment β is set as 0.5.

For training hyperparameters, we set the learning rate to 0.01 for both datasets. We train for 700 epochs with a batch size of 3000 papers for AMiner, and 500 epochs with a batch size of 1200 papers for APS. We randomly initialize all the parameters and optimize them with Adam (Kingma and Ba, 2015a). We run every experiment three times and report the average. All the experiments are conducted on a desktop machine with a 4-core

i7-5860k CPU, 40G memory, and two Nvidia Titan X GPUs. The total running time (not including data prepossessing) is around 24 minutes for AMiner and around 10 minutes on APS with the above settings.

Table 2.1: Results of effectiveness experiments on AMiner and APS datasets.

Dataset	Model	MALE						RMSLE					
		1st year	2nd year	3rd year	4 year	5 year	overall	1st year	2nd year	3rd year	4 year	5 year	overall
AMiner	GBM	**0.673**	0.971	1.069	1.383	1.332	1.085	**0.753**	**1.108**	1.283	1.624	1.685	1.291
	DeepCas	1.003	1.103	1.068	0.987	1.025	1.037	1.119	1.325	1.366	1.330	1.346	1.321
	HINTS-GCN	0.824	0.904	0.919	0.958	1.019	0.925	0.936	1.119	1.152	1.176	1.196	1.116
	HINTS-Seq	1.139	0.953	0.969	0.980	0.992	1.011	1.375	1.164	1.206	1.216	1.223	1.237
	HINTS	0.783	**0.866**	**0.879**	**0.877**	**0.865**	**0.854**	0.976	1.110	**1.146**	**1.155**	**1.154**	**1.111**
APS	GBM	0.952	0.968	0.972	0.982	1.103	0.995	1.151	1.168	1.189	1.214	1.355	1.215
	DeepCas	0.993	0.998	0.966	0.931	0.886	0.955	1.198	1.221	1.195	1.160	1.114	1.178
	HINTS-GCN	0.949	0.950	0.939	0.917	0.906	0.932	1.153	1.166	1.160	1.133	1.124	1.147
	HINTS-Seq	1.263	0.951	0.959	0.969	0.975	1.023	1.397	1.219	1.199	1.193	1.119	1.225
	HINTS	**0.934**	**0.936**	**0.923**	**0.903**	**0.875**	**0.914**	**1.135**	**1.151**	**1.142**	**1.127**	**1.102**	**1.132**

2.4.2 Numerical Comparison Results

In this part, we examine the performance of HINTS comprehensively. We compare HINTS with baselines, conduct ablation studies on components and objective of HINTS, and analyze differences in HINTS ability to predict trajectories for low-citation and high-citation papers.

Comparison with Baselines. Table. 2.1 shows the first five years of predictions errors for all models. In general, HINTS outperforms our proposed baselines in almost every time step. On AMiner, HINTS outperforms the best baseline, DeepCas, by 17.6% in terms of MALE and 15.8% in terms of RMSLE. And these numbers are 4.3% and 3.9% on APS. We speculate that DeepCas may suffer in "cold start" settings where the initial bibliographic cascade graph is quite small. The strong performance of GBM (particularly on AMiner) in early years is due to overfitting the majority papers that do not receive citations. However, GBM performance degrades sharply over time. In contrast, HINTS actually achieves better scores over time, indicating the importance of leveraging parametric assumptions for long-term citation prediction.

Ablation Study on HINTS Components. We further compare HINTS with two variants in order to evaluate the effectiveness of each module. First, we find that HINTS

consistently outperforms the HINTS-Seq variant (Table. 2.1), again indicating the value of supplementing contextualized embeddings with domain knowledge encoded in formal models. Second, while HINTS-GCN outperforms HINTS-Seq, there is still a non-trivial gap in performance between HINTS-GCN and HINTS. This result underscores the utility of modeling heterogeneous relationships between metadata for citation prediction.

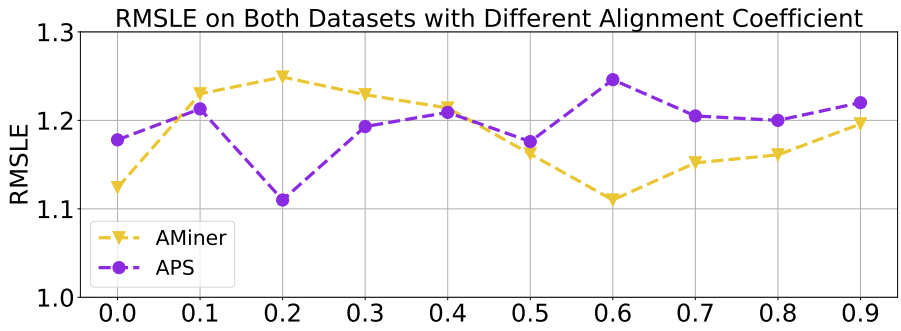

Figure 2.5: The average RMSLE across five years on AMiner and APS datasets with 10 different alignment coefficients.

Ablation Study on Alignment. The learning objective of HINTS balances citation prediction and the temporal alignment of embeddings. To measure the impact of the alignment regularizer ($\sum_{t=1}^{T-1} L_{t,t+1}^{\langle time \rangle}$), we conduct another ablation experiments with 10 different alignment coefficient β ranging from 0 to 0.9.

We compare the average RMSLE across five years on testing set of both AMiner and APS datasets. We also run HINTS three times for every β and report the mean results in Fig. 2.5. The optimal βs for AMiner and APS datasets are 0.6 and 0.2, respectively. Although the two datasets rely on the alignment regularization to varying degrees, these results show regularization improves performance for both datasets (especially APS) by learning more accurate embeddings for the DHIN. We also note that performance begins to degrade when β>0.7. This is because a larger β forces embeddings across years to be too similar. Actually, an extreme case is when β is large enough, the embeddings over time would be restricted almost the same, which makes the embeddings no longer reasonable.

Sensitivity to Paper Impact. The number of citations received by scientific papers varies widely. To evaluate HINTS' ability to accurately predict citations for papers

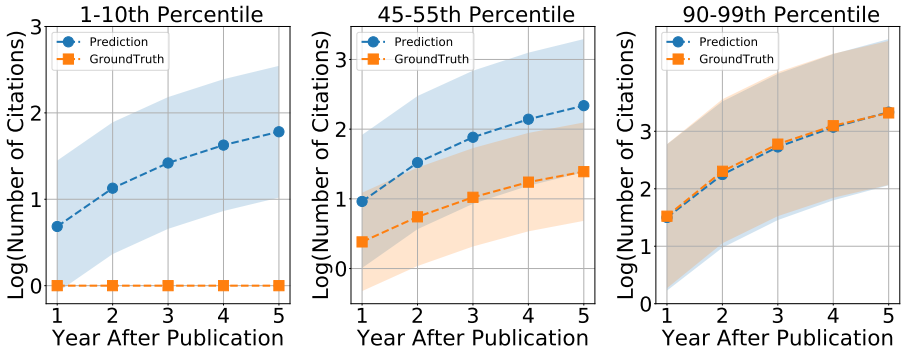

Figure 2.6: Predicted versus ground truth citations for AMiner papers published in 2010, stratified by log-scale cumulative citation count. Solid shapes indicate mean, ribbons cover 95% of data. From left: papers in 1-10th percentile, papers in 45-55th percentile, papers in the top 90th-99th percentile.

of varying impact, we stratify AMiner papers into three groups based on their ground truth log-scale cumulative citation count five years after publications: lowly cited papers (bottom decile), moderately cited papers (45-55th percentiles), and highly-cited papers (90-99th percentiles). Fig. 2.6 shows the average predicted time series for all papers within each of these groups compared with their corresponding average ground truth trajectory. HINTS seems to over-predict low-citation papers over time. This is likely because our parametric generator is designed to model the trajectories of cited papers, while many of these papers receive no citations. In future work, this could be addressed with a zero-inflation parameter. However, HINTS performs remarkably well for highly-cited papers and shows tolerable error for medium-cited papers. The strong performance on very high impact papers suggests HINTS could be useful for both scientists and funders to spot "hidden gems" in the scientific literature.

2.4.3 How HINTS Works

In this part, we perform a series of detailed analyses to better understand the performance of HINTS. We first compare how the algorithm uses metadata differently across fields. Next, we explore the imputed embeddings and learned citation time series parameters through visualization.

Importance of Metadata Types in Imputation. Not all metadata contain equal information for citation prediction. To understand how HINTS uses different types of metadata, we normalize the learned imputation weights using a softmax function (Table. 2.2). Unsurprisingly, we find that each reference contributes comparatively little information, while author, venue and keywords are more important predictors of citation time series in both Computer Science and Physics. However, these three factors play different roles between CS than Physics. The distinction possibly reflects differences in how these two communities operate (e.g., perhaps Physics communities converge more strongly on papers in top journals).

Table 2.2: Learned weights of metadata for imputation.

Field	Reference	Author	Venue	Keywords
AMiner (CS)	0.181	0.243	0.281	0.295
APS (Physics)	0.191	0.260	0.312	0.237

Visualization of Imputed Embeddings. To confirm that our imputed, temporally-encoded embeddings contribute to prediction, we randomly sampled 1000 papers from each strata described in 2.4.2 from AMiner, i.e., 1-10th percentile, 45th-55th percentile and 90-99th percentile (3000 papers in total). We use t-SNE (Maaten and Hinton, 2008) to project embeddings into a two-dimensional space (Fig. 2.7). Each point represents a paper, which is colored by its log-scale 5-year cumulative citation count after publication.

The embeddings clearly capture information about cumulative citation counts, as evidenced by the gradient from blue points (left-top) to red points (right-bottom). However, the gradient is not perfect; consistent with Fig. 2.6, the bottom 10% of papers is widely scattered, and some are mixed with the top 10%. This overlap shows that two papers with the same metadata can still have vastly different outcomes due to differences in quality or chance preferential attachment. Although metadata are strongly predictive of citation, they can't capture everything that contributes to citation for new papers.

Interpretation of Parameters in Time Series Generator. Modeled after the parameters in (Wang et al., 2013), we expect our three citation parameters "fitness" η,

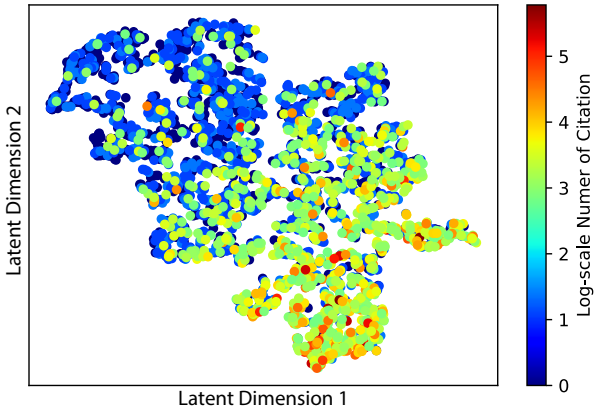

Figure 2.7: 2D t-SNE projections of imputed embeddings from AMiner sample. Embeddings color is coded by log-scale five-year cumulative citation count: blue means lower citations while red means higher.

"peak time" μ and "rate of decay" σ described in Sec. 2.3.4 to capture different aspects of the citation process. Notably Fig. 2.8 shows a strong correlation between "fitness" η and the cumulative citation counts. Furthermore, highly-cited papers have a larger σ, indicating their longer survival time due to preferential attachment. The interpretability of these parameters reinforces our conclusion from the ablation analysis with HINTS-Seq: leveraging domain-specific knowledge is crucial for accurate time series prediction.

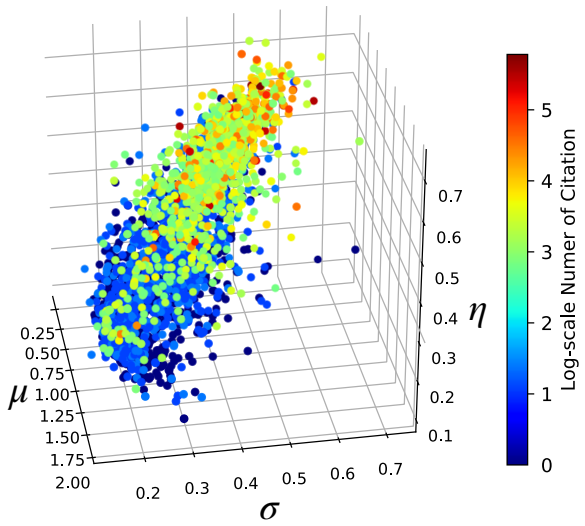

Figure 2.8: AMiner papers with respect to citation function parameters η, μ, σ. Papers are colored by log-scale five-year cumulative citation count: blue means lower citations while red means higher.

2.5 Conclusion

In this paper, we tackle a new problem: citation time series prediction from the time of publication, *without leading citation values*. We propose a novel framework, HINTS, which transforms the historical and relational signals encoded in pre-publication dynamic bibliographical networks into predicted citation trajectories for new papers. More generally, HINTS demonstrates how relational and temporal information in DHINs can be combined with interpretable, domain-specific statistical models for effective citation prediction. The novelty of HINTS comes from the framework, and future work could substitute each of the three modules with other algorithms (e.g., GAT (Veličković et al., 2018) for node embedding, point processes for parametric modeling) as needed to tackle other cold-start prediction problems. In extensive experiments, we demonstrate that HINTS is effective for citation prediction. We believe HINTS could be useful for scientists, granting organizations, and academic recommender systems seeking to identify “hidden gems" with high potential for future impact.

CHAPTER 3

Bridging Self-Attention and Time Series Decomposition for Periodic Forecasting

3.1 Introduction

Time series is a fundamental data abstract that has diverse real-world use cases, such as product sales (Bandara et al., 2019) and healthcare analytics (Caballero Barajas and Akella, 2015). With powerful expressiveness for sequential data, neural forecasting models have been introduced into time series with various attempts. For example, DeepAR (Salinas et al., 2020), based on recurrent neural networks (RNNs), outperforms the traditional statistical models by significant margins. Inspired by the success in natural language, transformer architecture (Vaswani et al., 2017) has been introduced in modeling time series data recently. (Li et al., 2019; Zhou et al., 2021) refine the vanilla canonical self-attention (Vaswani et al., 2017) by exploiting the sparsity of the attention scores. Their customized attention mechanisms lower the high computational cost in modeling long-term time series. (Wu et al., 2021) expands the embeddings aggregation from point-wise in above works to series-wise. The attention scores are computed over the similar subsequences induced by shifting the original sequence along time. Although remarkable progress has been made in applying these sequential neural networks on time series, we argue that standard deep learning models are *not* capable of learning the periodicity of time series sufficiently. Fig. 3.1 shows such a failure example. Recent study (Ziyin et al., 2020) reveals that the reason is standard neural nets do not have any modules to capture the periodicity explicitly in their architectures. However, real-world time series typically exhibit stronger periodicity than general sequential data such as

audio or text. The inability to learn periodic functions limits the potential of existing neural models for more accurate time series forecasting.

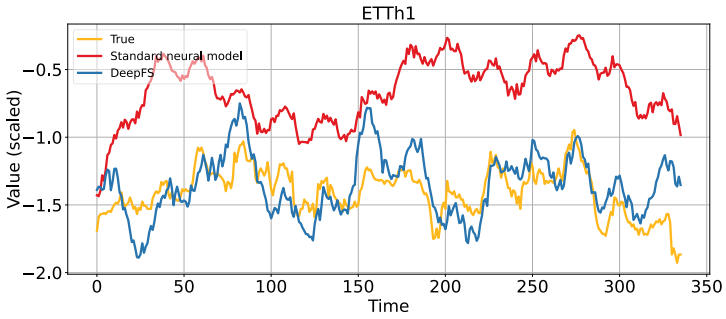

Figure 3.1: Forecasting v.s. ground-truth on the Electricity Transformer Temperature hourly (ETTh1) dataset (Zhou et al., 2021). Values are scaled. The standard neural model fails to fully learn the periodicity, aligned with the conclusion in (Ziyin et al., 2020). We propose DeepFS to capture the periodic fluctuations for more accurate real-world time series forecasting.

Unlike deep learning, classical time series analysis methods are commonly built on periodicity and other physical-mechanistic factors. A widely used mechanistic approach is time series decomposition (Hyndman and Athanasopoulos, 2018; Robert et al., 1990). It destructs a time series into seasonality, trend (or trend-cycle), and irregularity, which reflect periodicity, long-term movements and random variation, respectively. However, extrapolating based on these factors alone suffers from poor forecasting accuracy (e.g., ARMIA (Hyndman and Athanasopoulos, 2018)). This is because the simple decomposition is not fully capable of modeling practical time series where the long-term sequential patterns are complicated.

Contributions: In this paper, inspired by their complementary strengths and weaknesses, we bridge the deep sequential networks and time series decomposition through a simple yet effective encoder-decoder paradigm. The proposed model DeepFS preserves temporal patterns through a self-attention mechanism, from which a periodic inductive bias is employed to capture periodicity for more accurate time series forecasting.

DeepFS is an end-to-end encoder-decoder framework. Composed of the self-attention layers, the encoder converts the historical leading series into latent representations by

preserving the inter-dependencies of each timestamp. To capture the periodicity and trend of a sequence, we integrate the time series decomposition in the decoder. Because only an observed sequence can be decomposed, we reformulate the infeasible decomposition of the forecast output as a learning reconstruction problem, namely predicting the periodic and non-periodic components. Specifically, DeepFS uses the Fourier series to represent the periodic component and transforms the leading series embeddings into the parameters of Fourier bases. The learned Fourier series is a periodic inductive bias that explicitly discloses the periodicity of the sequence for better forecasting. Moreover, we use another projection network to generate the prediction of the non-periodic components (i.e., trend) from the leading series embeddings. The final forecasting is an additive combination of these periodic and non-periodic series.

A second benefit of DeepFS is interpretability. In contrast to the lag analysis (Wu et al., 2021) or score based interpretation (e.g., salience maps (Ismail et al., 2020), attention weights (Alaa and van der Schaar, 2019) and feature importance (Lim et al., 2021)), DeepFS learns the mechanistic factors explicitly, including periodicity and trend. For example, we show that DeepFS extracts 24-hour and 12-hour as the periodicity of the wet bulb temperature in Sec. 3.4. Such factors explicitly explain how the predictions are generated from historical observations and offer fruitful insights into practices such as business planning and decision making.

Empirically, we first justify the periodicity learning module on synthetic datasets. We then present experimental results on four real-world datasets, where DeepFS achieves accuracy gains compared to the state-of-the-art transformer models. We also provide analyses of the insightful periods learned from real-world datasets, and study why and when DeepFS can work. We summarize the main contributions of this work as follows:

- We propose DeepFS, a novel model that bridges self-attention and time series decomposition for accurate forecasting.
- We propose to inject Fourier series as a learnable periodic inductive bias in DeepFS to capture periodicity.

- DeepFS is also an explainable model that provides insightful interpretations for real-world tasks.
- We conduct comprehensive experiments and analyses on both synthetic and real-world data, demonstrating the effectiveness of DeepFS on time series forecasting.

3.2 Problem Setup

In this section, we introduce the univariate time series forecasting problem. Given a L-length observed series $Y' = [Y^{t_0-L+1}, ..., Y^{t_0}] \in \mathbb{R}^L$, where t_0 is the last timestamp with observation, we aim to predict the future H-length sequence $Y = [Y^{t_0+1}, ..., Y^{t_0+H}] \in \mathbb{R}^H$. Following the mechanistic decomposition, we hypothesize that the future time series Y is composed of periodic series (seasonality) $P = [P^{t_0+1}, ..., P^{t_0+H}] \in \mathbb{R}^H$, and non-periodic series (trend) $C = [C^{t_0+1}, ..., C^{t_0+H}] \in \mathbb{R}^H$. Note that both the periodic and non-periodic series are at every timestamp in the future, and are therefore aligned with the forecasting horizon (with same length H). Our goal is to learn a function $f(\cdot)$ that maps the past observations Y' to the periodic series P and non-periodic series C separately, to further reconstruct the future sequences Y. $f(\cdot)$ is formalized as:

$$[Y^{t_0-L+1}, ... , Y^{t_0}] \xrightarrow{f(\cdot)} [P^{t_0+1}, ..., P^{t_0+H}], [C^{t_0+1}, ..., C^{t_0+H}]. \tag{3.1}$$

Then the final forecasting is an additive combination of P and C, i.e., $[Y^{t_0+1}, ... , Y^{t_0+H}] = [P^{t_0+1}, ..., P^{t_0+H}] + [C^{t_0+1}, ..., C^{t_0+H}]$. We consider capturing the periodicity of future time series Y explicitly for better forecasting. A real-world time series is commonly affected by various causes, e.g., daytime-night, workday-weekend or seasons, and thus its periodicity may consist of multiple sub-components. For example, the product sales typically exhibit weekly and seasonal repeating patterns. We further denote $T = [T^1, ..., T^S] \in \mathbb{R}^S$ the periodicity of time series Y where S is the number of periodic components. Note that the periodic series P can be reconstructed based on the periods T.

3.3 Methodology

In this section, we introduce our framework, DeepFS. We argue that a decent design of $f(\cdot)$ in Eq. 3.1 should follow three principles: (1) simple, with no verbose modules, yet effective; (2) the architecture should be expressive to model the complicated temporal pattern of time series; (3) the periods T should be extracted as much as possible; (4) the prediction should be interpretable for practitioners.

3.3.1 Overview of DeepFS

Following the above four principles, we depict our proposed model DeepFS in Fig. 3.2. DeepFS follows the encoder-decoder paradigm, where the encoder coverts the leading values into latent representations at each timestamp with self-attention mechanism, followed by a decoder that transforms these embeddings to the predicted periodic series and the non-periodic series, which reflect periodicity and trend, respectively. The final forecast series is reconstructed by combining the periodic and non-periodic series, following the additive decomposition model (Hyndman and Athanasopoulos, 2018). We breakdown these two modules in detail in the following sub-sections.

3.3.2 Time Series Self-Attention

The RNNs and their variants (Schuster and Paliwal, 1997; Salinas et al., 2020; Qin et al., 2017; Gasthaus et al., 2019) process the time series data iteratively under the Markov property assumption, i.e., the hidden state encoded at a timestamp is only retained for the next timestamp. Therefore, the impact of a timestamp becomes trivial if the sequence is long, namely RNNs fail to capture long-distance dependency. Hence, we use the self-attention network (Vaswani et al., 2017) to model the observed leading time series as it breaks the Markov assumption and is capable of capturing the dependencies between two timestamps even if they are far apart. In this work, we follow the time series self-attention proposed in (Zhou et al., 2021) for time series data, which encodes the

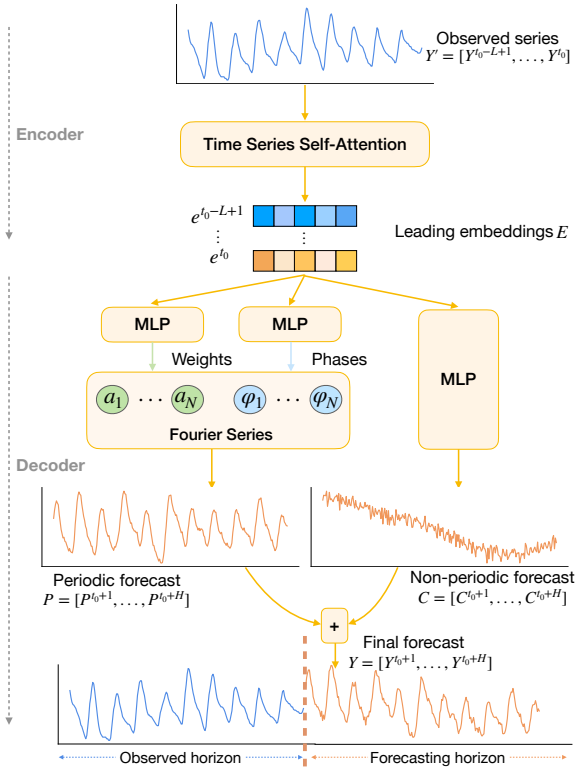

Figure 3.2: Overview of DeepFS. DeepFS first encodes an observed leading sequence to the leading embeddings E via self-attention. E is then transformed to the parameters of Fourier series (the weights and phases of sinusoidal bases) to predict the periodic series P. Separately, E is also converted to a non-periodic series C that represents the trend prediction. The final time series forecast is an additive combination of the periodic P and non-periodic C.

value at a time step into latent embeddings via computing its attention strengths with all timestamps in the sequence.

Leading Embeddings Initialization. For later purposes of the attention computation, we first project each numerical value of the leading sequence Y' to a d_u-dimensional vectorized representation via a projection operator $\phi(\cdot) : \mathbb{R} \to \mathbb{R}^{d_u}$ as in (Zhou et al., 2021). The mutual attention mechanism in self-attention inherently discards the relative position contexts, yet the temporal order has proven essential for time series (Li et al., 2019; Zhou et al., 2021). Therefore, we also include the positional encoding $\tau(\cdot) : \mathbb{R} \to \mathbb{R}^{d_u}$ and temporal encoding $\upsilon(\cdot) : \mathbb{D} \to \mathbb{R}^{d_u}$ used in (Zhou et al., 2021), which inject the local relative orders and the global contexts of timestamps (e.g., hour, week, month) into the sequence representations, respectively. We denote the global contexts of timestamp t

by $\tilde{t} \in \mathbb{D}$, where $\mathbb{D}$ is the space of global time[3]. The final initialized leading embeddings $u^t \in \mathbb{R}^{d_u}$ at timestamp t is then formalized as:

$$u^t = \phi(Y^t) + \tau(t) + \upsilon(\tilde{t}). \tag{3.2}$$

Leading Embeddings with Self-Attention. With the initialized leading embeddings sequence $U = [u^{t_0-L+1}, ..., u^{t_0}]$, we then compute the d_e-dimensional embeddings $E = [e^{t_0-L+1}, ..., e^{t_0}]$ ($e^t \in \mathbb{R}^{d_e}$) for the observed leading sequence Y' that capture its temporal patterns and the inter-dependencies between timestamps via the self-attention mechanism. The self-attention computation is conducted with three matrices: query $Q \in \mathbb{R}^{L \times d_q}$, key $K \in \mathbb{R}^{L \times d_k}$ and value $V \in \mathbb{R}^{L \times d_v}$, which are derived from the initialized embeddings U via three corresponding linear transformations $l_q : \mathbb{R}^{d_u} \to \mathbb{R}^{d_q}$, $l_k : \mathbb{R}^{d_u} \to \mathbb{R}^{d_k}$ and $l_v : \mathbb{R}^{d_u} \to \mathbb{R}^{d_v}$, respectively. d_q, d_k, d_v are the dimensions of the vectors in Q, K and V, where $d_q = d_k$ for the following attention scores computation. The attention based aggregated embeddings $o^t \in \mathbb{R}^{d_v}$ for the leading timestamp t is then formalized as:

$$o^t = \sum_i \frac{\exp(q_t k_i^\top / \sqrt{d_q})}{\sum_j \exp(q_t k_j^\top / \sqrt{d_q})} v_i, \tag{3.3}$$

where the q_t is the query vector at timestamp t from Q, and i, j are the timestamp indicators of K and V. The final leading embeddings e^t is then transformed from o^t with a further linear projector $l_e : \mathbb{R}^{d_v} \to \mathbb{R}^{d_e}$. The above embeddings aggregation is under the multi-head setting following (Vaswani et al., 2017).

With the time series self-attention module, the final leading embeddings $E = [e^{t_0-L+1}, ..., e^{t_0}]$ overcome the long distance challenge of the recursive based temporal aggregation. Note that although we use the attention mechanism from (Zhou et al., 2021), DeepFS is flexible to accommodate other time series transformer methods such as (Li et al., 2019; Wu et al., 2021; Kitaev et al., 2020).

[3]We use "year-month-day-hour-minute-second" for $\mathbb{D}$ in practice.

3.3.3 Decomposition Based Forecasting

With the leading embeddings E, a straightforward way to model sequential data is using another neural module to convert E into prediction, either a recursive structure (Sutskever et al., 2014) or self-attention (Zhou et al., 2021). Compared to general sequential data such as natural language or audio, we hypothesis that real-world time series are typically more periodic. For example, the traffic volumes exhibit daily and weekly repeating patterns. Intuitively, capturing such periods explicitly has the potential to boost the forecast of future series. However, standard neural nets can not learn the periodicity of time series sufficiently. This is because they do not contain any modules that can represent the periodic functions (Ziyin et al., 2020).

To model the periodicity of time series, we take inspiration from the time series decomposition (Hyndman and Athanasopoulos, 2018). It deconstructs an observed time series into periodicity, trend and irregularity. We refer to the latter two as non-periodic components. We propose to integrate the time series decomposition in our model, with the mind of capturing periodicity, to generate the forecast series from the leading embeddings E. However, the prerequisite of decomposition is an observed time series, yet the forecast sequences are not available until the model makes final prediction. Therefore, as Fig. 3.2 shows, we transform the decomposition into a learning based reconstruction problem, i.e., instead of decomposing an observed sequence, we predict the future periodic and non-periodic series based on the leading embeddings E, and then combine them to reconstruct the future series.

Periodic Series Prediction. A periodic sequence can be represented by the Fourier series with appropriate sinusoidal bases, parameterized by weights, periods and phases. Therefore, we inject the Fourier series as a periodic inductive bias in DeepFS to predict the periodic series $P = [P^{t_0+1}, ..., P^{t_0+H}]$, which is formalized as:

$$P = a_0 + \sum_{n=1}^{N} a_n \sin(\frac{2\pi n t}{p_0} + \varphi_n), \tag{3.4}$$

where $t \in [t_0 + 1, ..., t_0 + H]$ (the forecast horizon). N is the number of sinusoidal bases, a_n and φ_n are the weights and phases for the n-th sinusoidal basis, while a_0 is a constant bias term. The periods of the sinusoidal basis set include the basic period p_0 and all its n-th harmonics. Note that to represent an arbitrary periodic sequence, N should be theoretically infinite, which is intractable in practice. Instead, we set N as a tunable parameter and use finite sinusoidal bases to approximate the periodic sequence P. Typically, the periods of the sinusoidal bases set are uniform divisions of forecast horizon H by frequency (Oreshkin et al., 2020). Though this set of sinusoidal bases can mimic the periodicity, we argue it can not explicitly represent some meaningful periods by which H are not divisible, e.g., when predicting next month's daily electricity load ($H = 30$), which shows weekly period (7). Therefore, we set the periods of the N bases from 1 to N. A natural choice of N is the forecast horizon H. Our practical experience suggests that if using sliding window to collect training examples, the weights of the sinusoidal bases can still be learned well even if N is larger than H. Therefore, in practices we just set N according to the specific tasks. Another benefit of choosing N regardless of H is that longer period may still be captured even if the forecast horizon H for one data sample is not enough.

As mentioned, the periodic series P can not be derived from post-hoc decomposition, so we turn Eq. 3.4 into a learnable module in which the sinusoidal bases weights a_n and φ_n are learned from the leading embeddings E with non-linear mapping functions $g_a : \mathbb{R}^{d_e} \to \mathbb{R}^N$ and $g_\varphi : \mathbb{R}^{d_e} \to \mathbb{R}^N$ separately, which are instantiated by the multilayer perceptrons (MLP). This learnable Fourier series serve as the periodic inductive bias to capture the periodicity. Ideally, the weights a_n will be learned to 0 if the sequence does not contain the corresponding period, otherwise some non-trivial values. Therefore, we can infer a sequence's periodicity $T = [T^1, ..., T^S]$ by analyzing the learned weights a_n. It is worth noting that one advantage of using a learnable Fourier series is the explicit periodicity T can be induced, offering human-interpretable insights of the real-world time series compared to just a periodic sequence P as in (Oreshkin et al., 2020). We validate this design in Sec. 3.4.3.

Non-periodic Series Prediction. The non-periodic series $C = [C^{t_0+1}, ..., C^{t_0+H}]$ represents the overall trend of the future sequence. Similar to the periodic series, C is also learnable in our model, which is converted from the leading embeddings E with a non-linear projector $g_c : \mathbb{R}^{d_e} \to \mathbb{R}^H$. A typical approach to the projection from the latent embeddings is the recursive decoding as in seq2seq (Sutskever et al., 2014). Because practical time series may have various trends, here we avoid assuming too many priors and simply use a MLP as g_c in our decoder to generate the forecast of non-periodic series C.

Time series reconstruction. We follow the additive time series decomposition (Hyndman and Athanasopoulos, 2018) to reconstruct the entire future time series. Namely, the final forecast $Y = [Y^{t_0+1}, ..., Y^{t_0+H}]$ is a summation of the predicted periodic series $P = [P^{t_0+1}, ..., P^{t_0+H}]$ and non-periodic series $C = [C^{t_0+1}, ..., C^{t_0+H}]$. We use the Mean Square Error as the loss function L to compare the forecast and the ground-truth sequences, which is formalized as:

$$L = \frac{1}{H} \sum_{h=1}^{H} (Y^{t_0+h} - Y_g^{t_0+h})^2, \tag{3.5}$$

where Y_g^t is the ground-truth at timestamp t. The loss L is further averaged over the entire training sequences.

Note that our framework also accommodates for the time series in which the periodicity is insignificant. In this case, all the sinusoidal base weights are learned as some trivial values closed to 0, and DeepFS becomes a standard neural network model for time series. We justify this property empirically in Sec. 3.4.1.

Interpretaion. A second benefit of DeepFS is the learned periodic and trend series are human-interpretable. Periodicity and trend are mechanistic factors that reveal how a future time series is achieved. In particular, the learned periods T provide practitioners with insights into real-world time series, such as daily repeating patterns of electrical loads. We show the learned periods for various real-world datasets in Sec. 3.4.2 and Sec. 3.4.3. Compared to the post-hoc spectral analysis that requires the forecasting series,

our periodicity learning is performed up front, and then the learned periods are used to boost prediction accuracy.

3.4 Experiments

In this part, we present the empirical evaluation of DeepFS on univariate time series forecasting. We first justify periodicity learning on synthetic data, then compare DeepFS with diverse baselines on four real-world datasets. We further breakdown DeepFS in various ablation studies to understand why and when DeepFS works.

3.4.1 Justification on Synthetic Datasets

Since we do not know the ground-truth of either periodicity or the trend on real-world datasets, we first justify periods and non-periodic series learned by DeepFS on synthetic datasets.

Data Synthesis Protocol. We inject all the time series components, i.e., periodicity, trend and randomness, described in (Hyndman and Athanasopoulos, 2018) in the simulated data. The synthesis protocol is formalized as follows:

$$\sum_{i=1}^{V} a_i^s \sin(\frac{2\pi t}{T_i^s} + \varphi_i^s) + \sum_{j=0}^{W} w_j t^j + \epsilon. \tag{3.6}$$

The first term represents the periodicity by a combination of V sinusoidal functions, where a_i^s, T_i^s and φ_i^s are the simulated weight, period and phase φ_i^s for the i-th sine wave, respectively. The second term is for the trend. We follow (Oreshkin et al., 2020) to use a polynomial function with small degree W as the trend of a time series usually does not change severely, and w_j is the weight for j-th power function. The ϵ stands for the randomness in time series synthesis.

In experiments, we select the sinusoidal function number $V \in \{0, 10, 20, 30\}$ to examine how DeepFS works under different periodicity complexity. The periods T_i^s are

non-repeatable sampled from the range $[1, 30]$. We set the polynomial function degree W as 2. Other parameters a_i^s, φ_i^s, w_j and ϵ are all randomly generated for each experiment. We simulate 20000 data instances and split them into train/val/test with ratio 0.7/0.1/0.2.

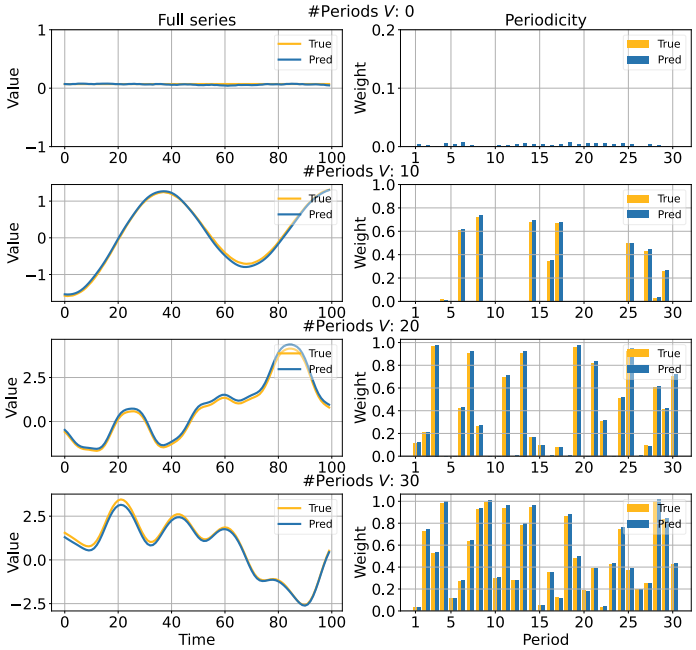

Figure 3.3: Results on synthetic datasets. Left: predicted time series v.s. ground-truth; Right: predicted periods with their corresponding weights v.s. ground-truth. Rows indicate different numbers of periodicity components V in simulation. The choices of V from top: 0, 10, 20, 30.

Results and analysis. We first report the comparison of the predicted time series and their periodicity with the simulated ground-truth in Fig. 3.3. Overall, the time series predicted by DeepFS are able to fit the simulated curves closely, while the learned periods weights are also consistent with ground-truth. We notice DeepFS still works even the time series is completely non-periodic (first row in Fig. 3.3), suggesting the effectiveness of DeepFS on learning true periods rather than just using the sine bases to approximate the sequence. We further summarize the accuracy of predicted periods weights and the non-periodic series in Table. 3.1, showing that DeepFS could learn both with very low errors. These results demonstrate the correctness of the learned periods and trend, making DeepFS trustworthy to capture periodicity on real-world datasets.

Table 3.1: Mean absolute error (MAE) of the periods weights and non-periodic series at four periodicity complexities.

#sinusoidals V	Periods weights	Non-periodic series
0	0.0127	0.0315
10	0.0121	0.0325
20	0.0127	0.0582
30	0.0199	0.0555

3.4.2 Experiments on Real-world Datasets

We then evaluate DeepFS on four real-world datasets to study the accuracy and interpretability of our model in practical scenarios.

Datasets. We use four real-world datasets that are collected by (Zhou et al., 2021). We describe the datasets as follows:

- **ETTh1, ETTh2, ETTm1:** The ETT (Electricity Transformer Temperature) datasets are metrics that can reflect the electric power deployment. We use 3 ETT datasets ETTh1, ETTh2, ETTm1 released by (Zhou et al., 2021) with granularity 1-hour, 1-hour and 15-mins, respectively. We use the exactly same data split as in (Zhou et al., 2021) (train/val/test: 12/4/4 months).

- **Weather:** The datasets is 4 years of weather records in United States with hourly granularity. We also use the same split as in (Zhou et al., 2021) (train/val/test: 28/10/10 months).

Baselines. We use three-category time series forecasting models as baselines, i.e., (1) statistical model ARIMA (Hyndman and Athanasopoulos, 2018), (2) RNN-based models LSTMa (Bahdanau et al., 2014) and DeepAR (Salinas et al., 2020), and (3) transformer based models Reformer (Kitaev et al., 2020) and state-of-the-art transformer Informer (Zhou et al., 2021).

Metrics. Because the datasets contain many zeros, we avoid the relative metrics like the mean absolute percentage error (MAPE). Instead, we use the mean square error (MSE) $\frac{1}{D}\frac{1}{H}\sum_{i=1}^{D}\sum_{h=1}^{H}(Y^{t_0+h}-Y_g^{t_0+h})^2$ and mean absolute error (MAE) $\frac{1}{D}\frac{1}{H}\sum_{i=1}^{D}\sum_{h=1}^{H}|Y^{t_0+h}-Y_g^{t_0+h}|$ to compare the DeepFS with baselines, where D is the number of data samples; H

is the prediction length; t_0 is the end of leading sequence; Y^t and Y_g^t are the ground-truth and predicted values at timestamp t, respectively.

Table 3.2: Accuracy results on ETTh1, ETTh2, ETTm1 and Weather datasets. We use same settings as in (Zhou et al., 2021). The average MSE and MAE of three runs are reported. Results of all baselines are from (Zhou et al., 2021). The best model is in boldface for each row. The percentage increases in DeepFS compared to the second-best baseline are listed. "-" symbols refer to worse accuracy.

Dataset	Prediction Length	ARIMA		DeepAR		LSTMa		Reformer		Informer		DeepFS		Gain	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	1d (24)	0.108	0.284	0.107	0.280	0.114	0.272	0.222	0.389	0.092	0.246	**0.090**	**0.234**	2.17%	4.88%
	2d (48)	0.175	0.424	0.162	0.327	0.193	0.358	0.284	0.445	0.158	0.319	**0.121**	**0.274**	23.42%	14.11%
	1w (168)	0.396	0.504	0.239	0.422	0.236	0.392	1.522	1.191	0.183	0.346	**0.130**	**0.284**	28.96%	17.92%
	2w (336)	0.468	0.593	0.445	0.552	0.590	0.698	1.860	1.124	0.215	0.369	**0.098**	**0.246**	54.42%	33.33%
	1m (720)	0.659	0.766	0.658	0.707	0.683	0.768	2.112	1.436	0.257	0.421	**0.167**	**0.329**	35.02%	21.85%
ETTh2	1d (24)	3.554	0.445	0.098	0.263	0.155	0.307	0.263	0.437	**0.093**	**0.240**	0.120	0.270	-	-
	2d (48)	3.190	0.474	0.163	0.341	0.190	0.348	0.458	0.545	0.155	0.314	**0.146**	**0.300**	5.81%	4.46%
	1w (168)	2.800	0.595	0.255	0.414	0.385	0.514	1.029	0.879	0.232	0.389	**0.191**	**0.347**	17.67%	10.69%
	2w (336)	2.753	0.738	0.604	0.607	0.558	0.606	1.668	1.228	0.263	0.417	**0.241**	**0.391**	8.37%	6.24%
	1m (720)	2.878	1.044	0.429	0.580	0.640	0.681	2.030	1.721	**0.277**	0.431	0.281	**0.425**	-	1.39%
ETTm1	6h (24)	0.090	0.206	0.091	0.243	0.121	0.233	0.095	0.228	0.030	0.137	**0.021**	**0.112**	30.00%	18.25%
	12h (48)	0.179	0.306	0.219	0.362	0.305	0.411	0.249	0.390	0.066	0.194	**0.035**	**0.146**	46.97%	24.74%
	1d (96)	0.272	0.399	0.364	0.496	0.287	0.420	0.920	0.767	**0.187**	0.384	**0.187**	**0.345**	0.00%	10.16%
	3d (288)	0.462	0.558	0.948	0.795	0.524	0.584	1.108	1.245	0.401	0.554	**0.219**	**0.386**	45.39%	30.32%
	1w (672)	0.639	0.697	2.437	1.352	1.064	0.873	1.793	1.528	0.512	0.644	**0.248**	**0.416**	51.56%	35.40%
Weather	1d (24)	0.219	0.355	0.128	0.274	0.131	0.254	0.231	0.401	0.117	0.251	**0.102**	**0.231**	12.82%	8.09%
	2d (48)	0.273	0.409	0.203	0.353	0.190	0.334	0.328	0.423	0.178	0.318	**0.139**	**0.272**	21.91%	14.47%
	1w (168)	0.503	0.599	0.293	0.451	0.341	0.448	0.654	0.634	0.266	0.398	**0.216**	**0.350**	18.80%	12.06%
	2w (336)	0.728	0.730	0.585	0.644	0.456	0.554	1.792	1.093	0.297	0.416	**0.280**	**0.413**	5.72%	0.75%
	1m (720)	1.062	0.943	0.499	0.596	0.866	0.809	2.087	1.534	**0.359**	**0.466**	0.394	0.488	-	-

Implementation Details. We use Pytorch 1.8.1 to conduct our experiments. The initial input embeddings dimension is set to 100. For the encoder, we use 2 self-attention layers with multi-head as 4 and embeddings dimension as 100. We also use layer-normalization and drop-out (0.05) for each self-attention layer. For the decoder, both the MLP modules used for periodic and non-periodic series are 3 layers with hidden embeddings size as 100. We set base number N of Fourier series as 100. We ignore the bases with period 1 and 2 because they are easily learned to be constants, causing over-fitting in practice. For the hyperparameters, we use 0.0001 for learning rate, 100 for batch size across all datasets. We use early-stopping based on the error on validation sets to avoid over training, but stopping too early may prevent DeepFS from learning meaningful periods. Therefore, we apply early-stopping after several iterations of training (20 for ETTh1, ETTh2 and Weather; 10 for ETTm1) with 5-step patience in our experiments. We initialize all the model parameters randomly and use Adam (Kingma and Ba, 2015b)

as optimizer. All experiments are done within an AWS g4dn.4xlarge instance.

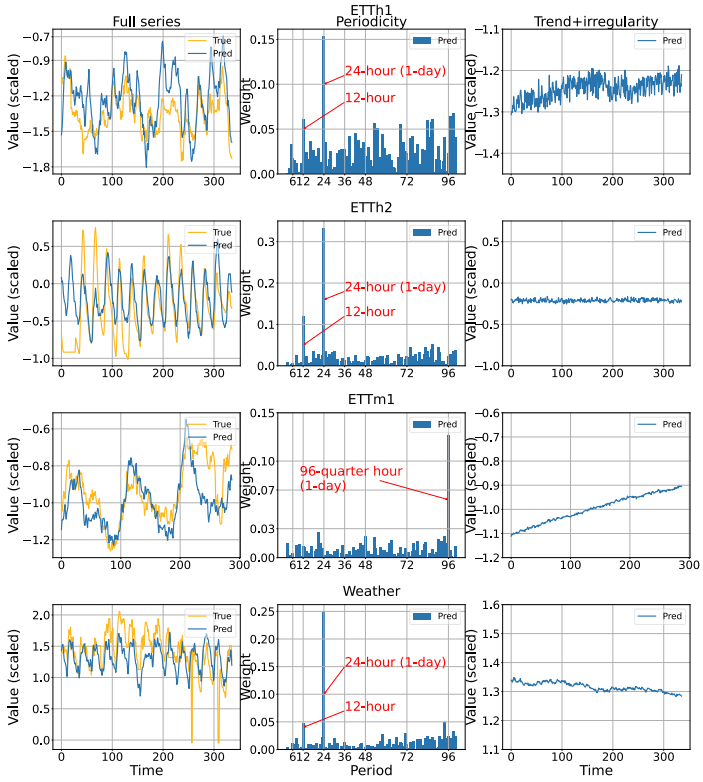

Figure 3.4: Results on real-world datasets. Left: full predicted time series v.s. ground-truth; Right: predicted trend. Data is scaled. Middle: predicted periods with weights. Striking predicted periods are highlighted with red annotation. From top: ETTh1, ETTh2, ETTm1, Weather.

Qualitative results and analysis. We show the forecast curves, the predicted periods and non-periodic series of all datasets in Fig. 3.4. The learned periodicity for ETTh1 (24-hour, 12-hour), ETTh2 (24-hour, 12-hour), ETTm1 (96-quarter hour), and Weather (24-hour, 12-hour) are all in the daily-wise, which are aligned with human's practical experience of electricity usage and the nature of weather evolution. The learned non-periodic series are also consistent with the overall movements of the ground-truths in general (e.g., a clear increase for ETTm1). Both the prediction of periods and trend contribute to and explain the observation that our predicted time series are able to capture the complicated fluctuations of the ground-truths (left column in Fig. 3.4), indicating the effectiveness of DeepFS on learning periodicity for better forecasting.

Quantitative results and analysis. We report the forecasting accuracy of DeepFS

and baselines in Table. 3.2. DeepFS outperforms all baselines on 17 out of 20 experiments. Specifically, compared to the state-of-the-art transformer based model Informer, DeepFS improves average 18.4% on MSE and 12.6% on MAE, suggesting the effectiveness of injecting periodic inductive bias into neural networks, as it captures the periodic-trend nature of time series. We notice that Informer is still strong in three cases especially in ETTh2 and Weather datasets. We speculate that the self-attention in Informer's decoder somehow captures the temporal patterns when the periodicity is dominant and clear, such as the ETTh2 and Weather datasets as shown in Fig. 3.4. However, for more complicated periodic patterns like ETTh1 and ETTm1, our model enjoys non-trivial accuracy gains, especially for long prediction length. We see this as the benefit of explicitly modeling periodicity.

Table 3.3: Accuracy comparison between DeepFS and variants. DeepFS (P) removes the non-periodic MLP, and DeepFS (NP) removes the Fourier series. The means and standard deviations of five runs are reported. (Numbers) under datasets are prediction length. The best model is in boldface for each row.

Datasets	Metrics	DeepFS (P)	DeepFS (NP)	DeepFS
ETTh1	MSE	1.990±0.002	0.380±0.175	**0.108±0.007**
(336)	MAE	1.363±0.001	0.529±0.135	**0.262±0.008**
ETTh2	MSE	1.554±0.002	0.264±0.037	**0.225±0.014**
(336)	MAE	1.118±0.001	0.416±0.033	**0.376±0.013**
ETTm1	MSE	1.917±0.001	**0.191±0.014**	0.192±0.008
(288)	MAE	1.341±0.000	0.367±0.017	**0.360±0.011**
Weather	MSE	0.989±0.002	0.316±0.012	**0.284±0.009**
(336)	MAE	0.803±0.001	0.439±0.011	**0.420±0.008**

3.4.3 Why Does DeepFS Work?

We further conduct detailed analyses of DeepFS from five aspects to study the reasons why our model can achieve better accuracy.

Ablation study on model architecture. Our intuition is that both the Fourier series and the non-periodic MLP layers contribute to prediction and are therefore indispensable. To verify this, we compare DeepFS with two variants: only with Fourier series (DeepFS (P)) and only with non-periodic MLP (DeepFS (NP)). We report the accuracy in

Table. 3.3 and draw the predicted series in Fig. 3.5. We find removing either component leads to a non-trivial accuracy drop, justifying their necessity. Fig. 3.5 shows the Fourier series alone is able to capture the fluctuations, further demonstrating the effectiveness of the learnable periodic inductive bias. We notice DeepFS (NP) just learns almost flat lines. We infer it is merely an attempt to reduce the overall loss (e.g., the good numerical error on ETTm1 in Table. 3.3) rather than preserving the inherent periodic patterns of time series. The results are aligned with the observations in (Ziyin et al., 2020) that standard neural networks can not fully learn periodicity.

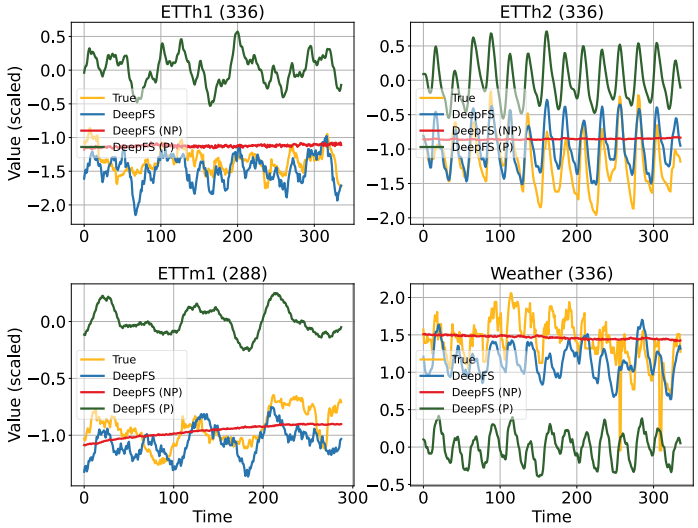

Figure 3.5: Forecasting of DeepFS, DeepFS (NP), DeepFS (P) v.s. ground-truth on four datasets. From top left: ETTh1, ETTh2, ETTm1, Weather. (Numbers) in titles are prediction lengths. Values are scaled.

Ablation study on sinusoidal bases. The sinusoidal bases are the "silver bullet" to learn periodicity of time series. To measure the impact of the number of sinusoidal bases N, we perform experiments on ETTh2 dataset with 9 different sine bases numbers $N \in \{20, 40, 60, 80, 100, 120, 140, 160, 180\}$. We set the prediction length as 2-week (336 timestamps) and report the accuracy (in MSE and MAE) and predicted periods in Fig. 3.6. The optimal basis number N in this setting is 40. This is probably because 40 bases (with periods until 40) include the principal periods of ETTh2, i.e., 24-hour and 12-hour. We also observe a rapid drop in accuracy when the bases number is greater than 100, especially 160 and 180. Combining the predicted periods weights, we think the reason is

the model fails to learn correct weights for the low-frequency bases (large periods). We infer that an optimal sinusoidal bases number choice should follow two rules: a) must include all principal periodic components, b) when a) holds, use a small number to reduce the difficulty in learning weighting for low-frequency bases. We further explore the failure reason in Sec. 3.4.4.

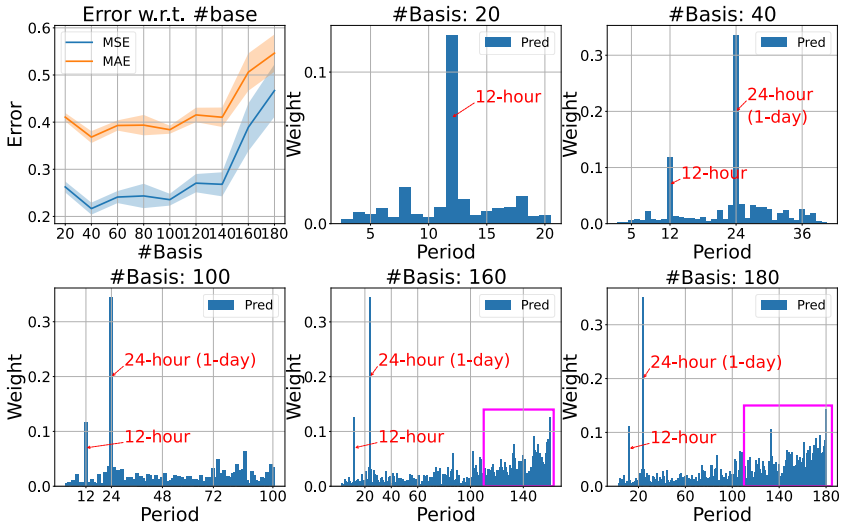

Figure 3.6: Accuracy and predicted periods weights on ETTh2 dataset with different sinusoidal basis numbers. Solid curves are average of MSE and MAE, ribbons indicate standard derivation. Striking predicted periods are highlighted with red annotation. Magenta rectangles ▭ circle the failures of learning sinusoidal bases weights.

Ablation study on leading series length. Our model learns the periodic and non-periodic series from the embeddings of leading sequence. To understand how the length of leading sequence L affect DeepFS, we conduct experiments with 6 different lengths $L \in \{48, 96, 168, 336, 540, 720\}$ on ETTh2 dataset, and report the accuracy and predicted periods in Fig 3.7. The error first decreases quickly with longer leading sequences, reaches the minimum at L of 96, and then gets worse. However, we note that the learned periods are reasonable regardless of leading series lengths. Therefore, we speculate that the non-periodic series patterns are not fully learned with a short leading sequence (pre-96), but the encoder may suffer from the cumbrous attention computation to predict the non-periodic series if the sequence is too long (e.g., post-540).

Periodicity generalization. Many real-world applications exhibit various periods,

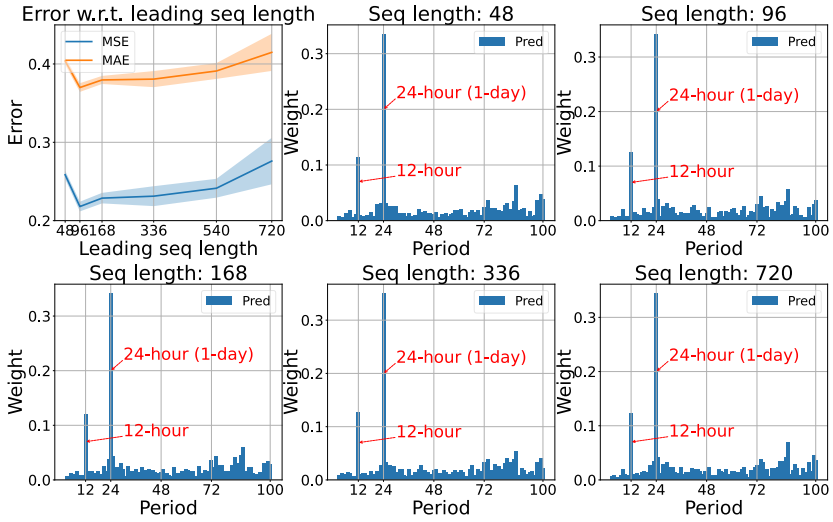

Figure 3.7: Accuracy and predicted periods on ETTh2 dataset with different leading observed sequence lengths. Solid curves are average of MSE and MAE, ribbons indicate standard derivation. Striking predicted periods are highlighted with red annotation.

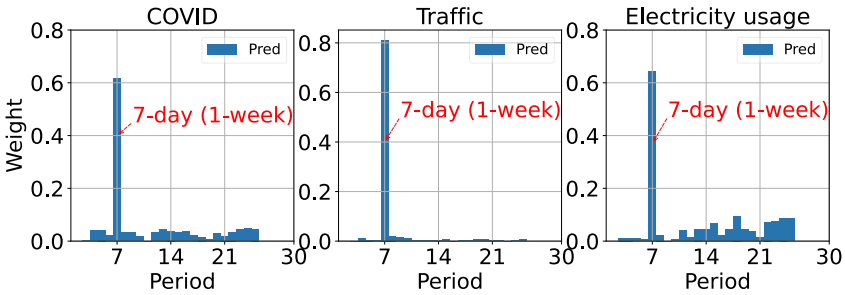

Figure 3.8: Predicted periods with corresponding weights on daily-granularity datasets. Striking predicted periods are highlighted with red annotation. From left: US COVID-19 death, freeway occupancy rates (traffic), electricity load.

e.g., weekly. We further use DeepFS to learn periods for three additional datasets: JHU CSSE COVID-19 Data (Dong et al., 2020), freeway occupancy rates (traffic)[4], and electricity load [5]. We preprocess all datasets to daily granularity and report the predicted periods in Fig. 3.8. All three datasets show weekly periods, which are consistent with the practical experience. (We believe the weekly period of COIVD-19 patients is primarily due to the fact that the cases in many states are not fully updated over the weekend.) These results confirm that DeepFS is able to capture various granularity periods, indicating the

[4] https://pems.dot.ca.gov/

[5] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

potential use of DeepFS in real scenarios.

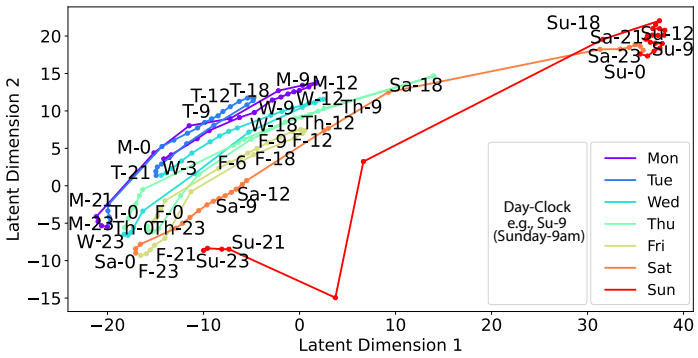

Figure 3.9: 2D t-SNE projections of leading embeddings E from ETTh2 dataset. Colors are coded by days (Monday to Sunday). "Day-clock" refers to an hour of the day, e.g., "Su-9" is Sunday 9AM. Adjacent times of the same day are connected by lines.

Leading embeddings visualization. To study whether the leading embeddings E are informative for the forecasting, we use t-sne (Van der Maaten and Hinton, 2008) to project the learned leading embeddings E into two-dimensional space in Fig. 3.9. The 2-D embeddings points exhibit two interesting patterns: (1) continuous from beginning to end of the week (left to right) while parts of weekends are dispersed, and (2) clustered at night-times (bottom left) and during day times (center) separately, with a looping shape for each day. These patterns reflect the periods (e.g., 24-hour) of the leading sequences, reinforcing our conclusion that DeepFS well captures the periodicity.

3.4.4 When Will DeepFS Work?

To further understand when it is appropriate to employ DeepFS for real-world applications, we explore why learning low-frequency sinusoidal bases fails in Sec. 3.4.3. We test the failure sinusoidal base number $N = 180$ on synthetic data. We compare the predicted periods with different numbers of synthetic data samples in Fig. 3.10. We find that enough data samples are essential to learning correct weights for low-frequency sinusoidal bases, and thus for low forecast errors. With a fixed length of the leading series, more data samples expand the entire horizon of the observed sequence, which probably explains the success of learning weight for a large period sinusoidal basis. We note that DeepFS may not work for time series with large periods but the available data for training is not

sufficient. We leave this "few-shot" problem to future work.

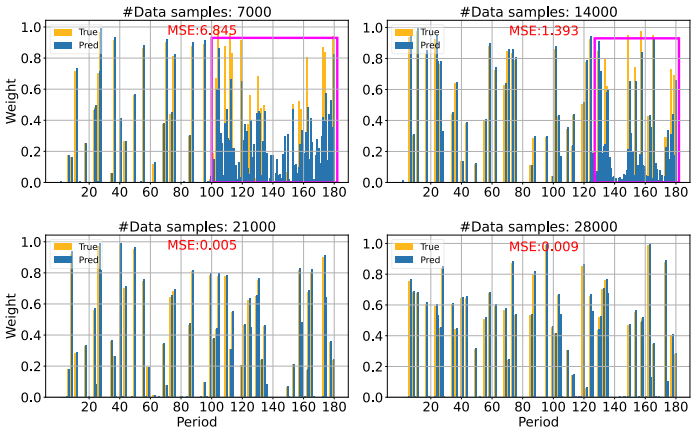

Figure 3.10: The predicted v.s. ground-truth periods weights with different numbers of synthetic data samples. The periodicity components number $V = 30$ and sinusoidal bases number $N = 180$ for all experiments. The MSE of each experiment is reported. Magenta rectangles ▭ circle the failures of learning sinusoidal bases weights.

3.5 Conclusion

In this paper, we study univariate time series forecasting by explicitly capturing the periodicity. We propose DeepFS, a novel model that combines self-attention and time series decomposition to enhance the periods preserving of neural networks. We achieve this by injecting Fourier series as an periodic inductive bias in our model. Extensive experiments demonstrate that DeepFS achieves better forecasting accuracy. However, DeepFS fails when training data is rare. Future work could study the transferablity of DeepFS for such few-shot scenario. How to learn periodicity for multi-variate time series is also an interesting and useful direction.

CHAPTER 4

Estimating Causal Effects on Networked Observational Data via Representation Learning

4.1 Introduction

Causal inference (also formalized as counterfactual reasoning (Greenland et al., 1999; Pearl, 2009)) has attracted increasing interests on networked scenarios, such as social networks (Ogburn et al., 2017, 2020), online advertisements (Nabi et al., 2020), and vaccine distribution (Barkley et al., 2020). Randomized controlled trials (RCTs) are still the "gold standard" on networked data (Gui et al., 2015; Yuan et al., 2021). However, RCTs are usually time-consuming, highly-costly and even not doable, which is especially true in the context of networks. Therefore, estimating the causal effects from networked observational data is an important yet challenging problem and the focus of this paper. We use vaccine distribution as our motivating example throughout this paper. Given the observed vaccine assignments (i.e., treatment) and the immunity level (Matrajt et al., 2020) (i.e., outcome) of a social community (i.e., network), we aim to answer the counterfactual questions like "would the community immunity level be stronger had a different group of people been vaccinated"?

The difficulties of causal inference on networked data are due to the dependency between units in a network and the need of inductive inference raised by real-world applications. First, compared with traditional independent setting (Shalit et al., 2017; Johansson et al., 2016; Yoon et al., 2018; Yao et al., 2018), the non-i.i.d. nature of networks introduces two-fold challenges to causal inference, i.e., *homophily* (McPherson et al., 2001) and *interference* (Hudgens and Halloran, 2008). Homophily describes the

phenomenon that similar units in networks tend to form social ties, which brings in new confounders (factors that affect both treatment and potential outcome) for causal effects in addition to the units' own features (a.k.a., characteristics). Interference refers to the fact that the potential outcome of a unit is caused by not only their own but the neighbors' treatments on networks, e.g., getting vaccines protects both a person and their social contacts. In other words, the traditional SUTVA (Rubin, 1980) assumption that one's potential outcome is stable regardless of the treatment assignments of others is no longer valid. Second, from the empirical view, many real-world problems require to predict the causal effects on a *new* network without any observed outcomes (known as "out-of-sample" estimation (Shalit et al., 2017), or "inductive" prediction (Hamilton et al., 2017b) in machine learning), e.g., finding out the best initial vaccine plan for a community. However, transferring the estimation from the observed networks to a new network is non-trivial because two network structures could be quite distinct.

To estimate causal effects on networked data, Forastiere *et al.* (Forastiere et al., 2021) extend the "no unobserved confounders" assumption to networks with interference, and propose a networked propensity score based method to infer the causal effects. Arbour *et al.* (Arbour et al., 2016) find out the adjustment variables on networks and estimate the treatment effects via back-door criterion (Pearl, 2009). Despite the success on networks with observed outcomes (known as "within-sample" estimation (Shalit et al., 2017)), these methods are not able to generalize the effects to a new network where we do not have any outcomes observed. Recent works propose to use network embeddings to capture unobserved confounders encoded in network structure (Veitch et al., 2019; Guo et al., 2020a,c; Ma et al., 2021; Chu et al., 2021). However, these works still follow the STUVA assumption and ignore the interference, which induces estimation bias of real-world networks.

Given that networked observational data contains features, treatments, observed outcomes and the network structure, a natural idea is to train a standard graph machine learning model, e.g., graph neural networks (GNNs) (Kipf and Welling, 2017a; Velickovic et al.; Xu et al., 2019; Wu et al., 2019), on the observed data and then to predict

the counterfactual outcomes for causal effects estimation. However, we theoretically demonstrate that such standard graph machine learning models fail in inferring the causal effects on networks, because there are two distribution mismatches between their objective functions (details in Sec. 4.3.1). In other words, standard graph machine learning models are solving a different optimization goal from estimating the causal effects on networks. To fill this gap, we further find that it is sufficient to enforce the two mismatched distributions to be uniform. These insights motivate us to propose a novel framework **NetEst**, which formulates the **Net**worked causal effects **Est**imation into a data-driven multi-task paradigm with two optimization goals: predicting the potential outcomes and bridging the distribution gaps between standard graph machine learning and networked causal inference. To facilitate this, NetEst first uses GNNs to encode the confounders that are from both a unit's own and neighbors' features into latent representations. Together with both a unit's own and their neighbors' treatments, these embeddings are then used to estimate the potential outcomes via an estimator. Meanwhile, NetEst uses two adversarial learning modules to force the mismatched distributions to follow uniform distributions based on the embeddings. NetEst is applicable to both the "out-of-sample" (Shalit et al., 2017) and the traditional "within-sample" (Johansson et al., 2016) estimation on networked data.

Our **main contributions** are summarized as follows: *First*, we theoretically prove that standard graph machine learning models can not estimate causal effects on networks due to the distribution mismatches between their objective functions. *Second*, we formalize the networked causal effects estimation to a multi-task learning problem and propose a novel framework NetEst that solves the distribution gaps and alleviates the challenges induced by the nature of neworked data. *Third*, we conduct extensive experiments on two datasets, demonstrating the effectiveness of NetEst and present empirical analyses of why and when NetEst works.

4.2 Problem Setup

We follow Arbour *et al.* (Arbour et al., 2016) to set up the causal effects estimation on networks. We first discuss the causal graph of networked data in the presence of homophily and interference. Then we present the definition of causal effects on networks and discuss its identification. We list all the notations used in this paper in Table 4.1.

Table 4.1: Notations used in this chapter.

Symbol	Description
G, A, X	graph, adjacency matrix and feature matrix
t_i, x_i, y_i	treatment, feature and potential outcome of unit i
$\{x_j\}_{j\in\mathcal{N}_i}$	features of i's neighbors in network
T, Y	treatment vector, potential outcome vector of all users
$\{t_j\}_{j\in\mathcal{N}_i}$	treatments of i's neighbors in network
$\{x_j\}_{j\in-\mathcal{N}_i}$	treatments of i's non-neighbors in network
Z	summary function of neighbors' treatments
z_i	peer exposure of unit i
$Y^i_{t_i,z_i}$	observed outcome of unit i under t_i and z_i
$Y_i\|do(t_i=t,z_i=z)$	potential outcome of unit i under t_i and z_i
$\mathcal{N}_i$, N_i	i's neighbors set, and size
τ, $\hat{\tau}$	treatment effects, estimate of treatment effects
ϕ, s_i	representation function, representation of unit i
m	outcome estimation function from representation
f	outcome estimation function from feature
d_t, d_z	discriminators
$\mathcal{J}$	loss function

4.2.1 Causal Graph on Networks

Causal graph is a directed acyclic graph (DAG) that describes the causal relations among variables (Pearl, 2009). Without loss of generality, we still use vaccination as our motivating example to depict a plausible causal graph on networks in Fig. 4.1. The social structure of a three-unit community is described on the left, and right part shows the causal relations of their features, treatments and potential outcomes. In practice, a unit's features (e.g., health condition) cause both their (1) decisions to get vaccinated (treatment) and (2) immunity to a virus (potential outcome), namely a unit's features contain confounders between treatment and potential outcome (indicated by red edges in Fig. 4.1). In addition, a unit's features may also affect the neighbors' treatments and potential outcomes as they may influence each other. For example, a person with a weakened immune system may increase their risk of infection, prompting their family members to get vaccinated. In

other words, networks introduce new confounders between the treatment and potential outcome (blue edges in Fig. 4.1). Different from the independent setting, treatment of a unit spills over to their neighbors. For example, getting vaccination protects not only oneself, but others in the community (i.e., herd immunity (Anderson and May, 1985)). This "peer effect" reflects the interference nature of the network (marked by green edges in Fig. 4.1). Following (Arbour et al., 2016), we assume that network confounders and the peer effect only exist among *1-hop* neighbors. We further assume that all treatments are carried out at the same time without any order. In other words treatments will not affect each other. The readers can refer to (Ogburn and VanderWeele, 2014) for other plausible causal graphs on networked data.

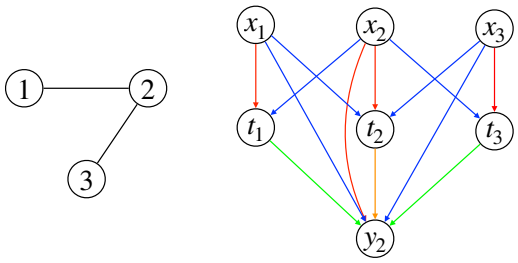

Figure 4.1: Causal graph of a network with three nodes. Left: the network connection topology. Right: causal graph. x, t, y are features, treatments and potential outcome respectively. A red edge shows confounders from a unit's own features, a blue edge means confounders brought by network, an orange edge represents the causal effects between a node's own treatment and potential outcome, and a green edge shows the peer effects of treatment. Only y_2 is shown for simplicity. We assume that peer effects occur only between 1-hop neighbors and that there are no unmeasured confounders.

4.2.2 Causal Inference on Networked Data

Formally, given a network $G = \langle A, X \rangle$, in which $A \in \mathbb{R}^{V \times V}$ is the adjacency matrix where V is the number of units (nodes) in G; $X \in \mathbb{R}^{V \times k}$ is the units' features matrix and k is the feature dimension. We use $x_i \in \mathbb{R}^k$ to represent the feature of i-th node. We denote $T = [t_1, ..., t_V]$ as the treatment vector of all V units where $t_i \in \{0, 1\}$ is the treatment of i-th unit. Following many existing works (Veitch et al., 2019; Guo et al., 2020b; Shalit et al., 2017; Johansson et al., 2016), we assume that t_i is binary (e.g., $t_i = 1$ means

getting vaccinated while 0 means not). We then denote the potential outcome vector $Y = [y_1, ..., y_V]$ where $y_i \in \mathbb{R}$ is the potential outcome of unit i. We further assume y_i is continuous (e.g., a higher value means a stronger immunity). Following (Arbour et al., 2016), we can define the causal effects $\tau(X)$ on the whole network G as the difference in the potential outcomes under two treatments vectors T' and T'', which is formalized as:

$$\tau(X) := \mathbb{E}\left[Y|do(T') - Y|do(T'')\middle|X, A\right], \tag{4.1}$$

where the *do*-calculus (Pearl, 2009) represents an intervention on treatments. In our motivating example, T' and T'' can be two vaccine distribution strategies. With Eq. (4.1), we can answer causal questions on networks, such as comparing the impacts of two vaccine plans.

To measure the overall effects $\tau(X)$ on the entire network, we need to estimate the treatment effects $\tau(x_i)$ for every unit, namely the individual treatment effects (ITE). From the individual view, a unit's potential outcome y_i is caused by their own feature x_i, treatment t_i, neighbors' features $\{x_j\}_{j\in\mathcal{N}_i} \in \mathbb{R}^{N_i\times k}$ and treatments $\{t_j\}_{j\in\mathcal{N}_i} \in \{0,1\}^{N_i}$ as in Fig. 4.1, where N_i is the number of 1-hop neighbors of unit i. To represent the interference of neighbors' treatments $\{t_j\}_{j\in\mathcal{N}_i}$, following (Forastiere et al., 2021), we define a summary function $Z : 2^T \to [0, 1]$ that reduces a set of treatments in 2^T into a scalar. We set $z_i = Z(\{t_j\}_{j\in\mathcal{N}_i})$, where z_i is defined as the *peer exposure* of unit i to neighbors' treatments $\{t_j\}_{j\in\mathcal{N}_i}$. In this paper, we define Z as a function to calculate the percentage of treated neighbors, i.e., $z_i = \sum_{j\in\mathcal{N}_i} t_j/|N_i|$, and thus z_i means the ratio of i's neighbors whose treatments are 1. Therefore, the range of z_i is $[0, 1]$. To highlight these causes, we reformulate unit i's potential outcome y_i as $Y_i|do(t_i = t, z_i = z)$, indicating the potential outcome under the treatment t and the peer exposure z. Then individual treatment effects (ITE) $\tau(x_i)$ of unit i can be formalized as:

$$\tau(x_i) := \mathbb{E}\left[Y_i|do(t_i = t', z_i = z') - Y_i|do(t_i = t'', z_i = z'')\middle|x_i, \{x_j\}_{j\in\mathcal{N}_i}\right]. \tag{4.2}$$

Given the treatment vector T and the topology A of networks, we can compute the peer exposure z_i for every unit. Therefore, the network effects can be fully represented by the peer exposure z_i and neighbors' features $\{x_j\}_{j\in\mathcal{N}_i}$ from the individual view. The presence of z_i and $\{x_j\}_{j\in\mathcal{N}_i}$ in Eq. (4.2) indicates the major difference of networked ITE compared to the general independent scenarios where potential outcomes are not affected by neighbors' treatments.

To estimate ITE $\tau(x_i)$ in Eq. (4.2), we need the two potential outcomes under different treatments and peer exposures. However, we can only observe at most one of them from observational data. For instance, we can only observe the outcomes of a community w.r.t. one vaccine distribution plan. Therefore, the core of $\tau(x_i)$ is to estimate the counterfactual outcome, namely the treatment-peer exposure-potential outcome tuples that are not observed.

Eq. (4.2) enables us to further study some interesting causal effects questions on networks. As in (Arbour et al., 2016), we focus the following three:

- *Individual effects:* $Y_i|do(t_i = 1, z_i = 0) - Y_i|do(t_i = 0, z_i = 0)$. It represents unit i's own treatment effects, e.g., how much protection would I get if it was just me and none of my friends were vaccinated?

- *Peer effects:* $Y_i|do(t_i = 0, z_i = z') - Y_i|do(t_i = 0, z_i = z'')$. It reflects the effects of treatment inference, e.g., how much protection would I get if different groups of my friends but not me were vaccinated?

- *total effects:* $Y_i|do(t_i = 1, z_i = 1) - Y_i|do(t_i = 0, z_i = 0)$. It describes the combined effects of individual treatment and the network interference, e.g., how much protect would I get if everyone is vaccinated?

4.2.3 Causal Identification on Networks

Causal inference is the estimation of causal quantities (e.g., i's potential outcome $Y_i|do(t_i = t, z_i = z)$). However, only the statistical quantities (e.g., i's observed outcome $Y^i_{t_i,z_i}$) are

available in observational data. To ensure that these statistical quantities can be used to infer the potential outcome (a.k.a., the causal identification problem), we make the following essential assumptions.

Assumptions. We make two lines of assumptions on networked data. First, we adapt the standard assumptions on independent data to the network setting following (Forastiere et al., 2021):

Assumption1: Positivity. The probability of a unit with their neighbors to receive treatment or not is always positive, i.e., $\forall x, 0 < p(t_i = 1|x_i, \{x_j\}_{j\in\mathcal{N}_i}) < 1$.

Assumption2: Consistency. The potential outcome is same as the observed outcome under the same treatment assignment and peer exposure to neighbors , i.e., $Y_i|do(t_i = t, z_i = z) = Y^i_{t,z}$.

Assumption3: Strong Ignorability. Conditional to the features x_i and neighbors' features $\{x_j\}_{j\in\mathcal{N}_i}$, potential outcome $Y_i|do(t_i = t, z_i = z)$ is independent of treatment t_i and peer exposure z_i , i.e., $Y_i|do(t_i = t, z_i = z) \perp\!\!\!\perp t_i, z_i|x_i, \{x_j\}_{j\in\mathcal{N}_i}$.

In networked data, the standard SUTVA does not hold because of the presence of interference. Therefore, to identify the causal effects, we further assume the interference has the Markov property (i.e., *1-hop*) following (Arbour et al., 2016) (here we set $T_{N_i} = \{t_j\}_{j\in\mathcal{N}_i}$ and $T_{-N_i} = \{t_j\}_{j\in-\mathcal{N}_i}$ for simplicity):

Assumption4: Markov. The potential outcome of a unit is only affected by their own and the immediate neighbors' treatments, i.e., $\forall\ T_{N_i}, T'_{N_i},\ T_{-N_i}, T'_{-N_i}$ such that $Z(T_{N_i}) = Z(T'_{N_i})$, we have $Y_i|do(t_i = t, T_{N_i}, T_{-N_i}) = Y_i|do(t_i = t, T'_{N_i}, T'_{-N_i})$.

Identification. Given these assumptions, unit i's causal effects $\tau(x_i)$ (Eq. (4.2)) is identifiable. To avoid mess, we omit the subscription and denote by $x = (x_i, \{x_j\}_{j\in\mathcal{N}_i})$ in the following proof:

Proof.

$$
\begin{aligned}
\tau(x) &= \mathbb{E}\left[Y|do(t=t', z=z') - Y|do(t=t'', z=z'')\middle|x\right] \\
&= \mathbb{E}\left[Y|do(t=t', z=z'))\middle|x\right] - \mathbb{E}\left[Y|do(t=t'', z=z'')\middle|x\right] \\
&= \mathbb{E}\left[Y|do(t=t', z=z')\middle|t=t', z=z', x\right] \\
&- \mathbb{E}\left[Y|do(t=t'', z=z'')\middle|t=t'', z=z'', x\right] \qquad (4.3) \\
&= \mathbb{E}\left[Y_{t',z'}\middle|t=t', z=z', x\right] - \mathbb{E}\left[Y_{t'',z''}\middle|t=t'', z=z'', x\right]. \qquad (4.4)
\end{aligned}
$$

Eq. (4.3) holds because of the "Strong Ignorability" assumption that given $x = (x_i, \{x_j\}_{j\in\mathcal{N}_i})$, the potential outcome $Y_i|do(t_i = t, z_i = z)$ is independent from the treatment t_i and peer exposure z_i. Eq. (4.4) is true because of the "Consistency" assumption. □

4.3 Methodology

In this section, we introduce our proposed method. We first prove why standard graph machine learning can not estimate causal effects. Then we breakdown the modules of NetEst in details.

4.3.1 Why Standard Graph Machine Learning Fails in Causal Inference?

We show that the failure of standard graph machine learning in estimating causal effects is due to two distribution mismatches between their objective functions.

We first introduce several functions with their corresponding notations. Following (Shalit et al., 2017), we define a one-to-one projection function $\phi : \mathcal{X} \times 2^{\mathcal{X}} \to \mathcal{S}$, which maps a unit's own features and the neighbors' features into representation space $\mathcal{S}$. We denote $s_i = \phi(x_i, \{x_j\}_{j\in\mathcal{N}_i})$ as unit i's representation induced by ϕ. We will introduce the motivation of using this representation projection function in Sec. 4.3.2. We further define an estimation function $m : \mathcal{S} \times \{0, 1\} \times [0, 1] \to \mathcal{Y}$ that estimates the potential outcome from feature representation s_i, treatment t_i and peer exposure

z_i. For simplicity, we also use a function $f : \mathcal{X} \times 2^{\mathcal{X}} \times \{0,1\} \times [0,1] \to \mathcal{Y}$ such that $f(x_i, t_i, z_i) = m(s_i, t_i, z_i) = m(\phi(x_i, \{x_j\}_{j\in\mathcal{N}_i}), t_i, z_i)$ to denote the whole estimation function starting from the original features. An estimation objective function needs a loss function, and we use the square loss in this paper. Now we can compare the objective functions of standard graph machine learning J_{ml} and causal effects estimation on networks J_{ce}. For simplicity, we remove the subscriptions in the following.

Objective function of machine learning J_{ml}**.** Standard graph machine learning estimates the potential outcome by optimizing the estimation loss over the networked observational data. Given network G, estimation function f, features x, treatment t, peer exposure z and outcome y, as stated in Sec. 4.2.2, the peer exposure z sufficiently represents the network effects induced by network G. Therefore, we can denote the observational data by the joint probability $p(x, t, z, y)$. Then the objective function of standard graph machine learning J_{ml} is:

$$J_{ml} = \int_{\mathcal{X}}\int_{\mathcal{T}}\int_{\mathcal{Z}}\int_{\mathcal{Y}} (f(x,t,z) - y)^2 p(x,t,z,y)\, dx\, dt\, dz\, dy \tag{4.5}$$

$$= \int_{\mathcal{X}}\int_{\mathcal{T}}\int_{\mathcal{Z}}\int_{\mathcal{Y}} (f(x,t,z) - y)^2 p(x)p(t|x)p(z|x,t)p(y|x,t,z)\, dx\, dt\, dz\, dy. \tag{4.6}$$

Note that the effects of network is encoded in the graph function f and peer exposure z, so G is not explicitly shown in Eq. (4.5). Eq. (4.6) is a chain rule expansion of Eq. (4.5). We can build a graph machine learning model f (e.g., GNNs) to predict the outcome by optimizing Eq. (4.6) on observational data.

Objective function of causal effects estimation J_{ce}**.** Causal inference is to estimate the causal effects $\tau(x)$ defined in Eq. (4.2) on network G. Therefore, given estimation model f, feature x, treatment t, peer exposure z and outcome y, the objective function of causal effects estimation J_{ce} is the estimation error of causal effects $\tau(x)$ over all units:

$$J_{ce} = \int_{\mathcal{X}} (\hat{\tau}(x) - \tau(x))^2 p(x)\, dx \tag{4.7}$$

$$\leq 8 \int_{\mathcal{X}} \int_{\mathcal{T}} \int_{\mathcal{Z}} \int_{\mathcal{Y}} \big(f(x,t,z) - y\big)^2 p(x) p(y|t,x,z)\, dx\, dt\, dz\, dy, \tag{4.8}$$

where $\hat{\tau}(x)$ is the estimated causal effects. Eq. (4.8) is an upper bound for the objective function J_{ce}. Because directly optimizing the original objective function J_{ce} (Eq. (4.7)) is difficult, this upper bound can be used as an approximated objective function and the estimation function f can be built by optimizing it on the networked observational data. We use a general format of causal effect $\tau(x) = \mathbb{E}\left[Y_{1,z'} \middle| t = 1, z = z', x\right] - \mathbb{E}\left[Y_{0,0} \middle| t = 0, z = 0, x\right]$ following (Forastiere et al., 2021), which can be further decomposed as:

$$\tau(x) = \mathbb{E}\left[Y_{1,z'} \middle| t = 1, z = z', x\right] - \mathbb{E}\left[Y_{0,0} \middle| t = 0, z = 0, x\right] \tag{4.9}$$

$$= \mathbb{E}\left[Y_{1,z'} \middle| t = 1, z = z', x\right] - \mathbb{E}\left[Y_{0,z'} \middle| t = 0, z = z', x\right] \tag{4.10}$$

$$+ \mathbb{E}\left[Y_{0,z'} \middle| t = 0, z = z', x\right] - \mathbb{E}\left[Y_{0,0} \middle| t = 0, z = 0, x\right]. \tag{4.11}$$

Eq. (4.10) captures the individual effects of treatment and Eq. (4.11) models the peer effects from network interference. If we set $z = 1$, Eq. (4.9) becomes to the total effects. Therefore, Eq. (4.9) is a general format that contains all causal effects of interest in Sec. 4.2.2.

We then show the proof of upper bound in Eq. (4.8) as follows:

Proof. Given network G, model f, feature x, treatment t and outcome y, an empirical estimate can be denoted as $\hat{\tau}(x) = f(x, t = 1, z) - f(x, t = 0, 0)$, which can be similarly decomposed as $\hat{\tau}(x) = f(x, t = 1, z) - f(x, t = 0, z) + f(x, t = 0, z) - f(x, t = 0, 0)$.

Finally the objective function of causal effects estimation J_{ce} [6] is as follows:

$$
\begin{aligned}
J_{ce} &= \int_{\mathcal{X}} (\hat{\tau}(x) - \tau(x))^2 p(x)\, dx \\
&= \int_{\mathcal{X}} \big[f(x, t=1, z) - f(x, t=0, z) \\
&- \big(\mathbb{E}(Y_{1,z} \big| t=1, z, x) - \mathbb{E}(Y_{0,z} \big| t=0, z, x) \big) \\
&+ f(x, t=0, z) - f(x, t=0, 0) \\
&- \big(\mathbb{E}(Y_{1,z} \big| t=0, z, x) - \mathbb{E}(Y_{0,z} \big| t=0, 0, x) \big) \big]^2 p(x)\, dx \qquad (4.12) \\
&\leq 2 \int_{\mathcal{X}} \big[f(x, t=1, z) - f(x, t=0, z) \\
&- \big(\mathbb{E}(Y_{1,z} \big| t=1, z, x) - \mathbb{E}(Y_{0,z} \big| t=0, z, x) \big) \big]^2 p(x)\, dx \qquad (4.13) \\
&+ 2 \int_{\mathcal{X}} \big[f(x, t=0, z) - f(x, t=0, 0) \\
&- \big(\mathbb{E}(Y_{0,z} \big| t=0, z, x) - \mathbb{E}(Y_{0,0} \big| t=0, 0, x) \big) \big]^2 p(x)\, dx, \qquad (4.14)
\end{aligned}
$$

where Eq. (4.12) is immediate with the definition of $\tau(x)$ and $\hat{\tau}(x)$. Eq. (4.13) and Eq. (4.14) are the estimation error of individual effects and peer effects, respectively. The inequality holds because $(a+b)^2 \leq 2(a^2+b^2)$. For clearness, we conduct the proof of them separately.

We first focus on the individual effects Eq. (4.13):

[6] With a square loss, J_{ce} is also know as Precision in Estimation of Heterogeneous Effects (PEHE) (Hill, 2011)

$$
\begin{aligned}
\textit{Eq. (4.13)} &= 2\int_{\mathcal{X}} \big[f(x,t=1,z) - \mathbb{E}(Y_{1,z}\big|t=1,z,x) \\
&\quad + \big(\mathbb{E}(Y_{0,z}\big|t=0,z,x) - f(x,t=0,z)\big)\big]^2 p(x)\,dx \\
&\leq 4\int_{\mathcal{X}} \big[f(x,t=1,z) - \mathbb{E}(Y_{1,z}\big|t=1,z,x)\big]^2 p(x)\,dx \\
&\quad + 4\int_{\mathcal{X}} \big[f(x,t=0,z) - \mathbb{E}(Y_{0,z}\big|t=0,z,x)\big]^2 p(x)\,dx \qquad (4.15) \\
&\leq 4\int_{\mathcal{X}} \mathbb{E}\left[\big(f(x,t=1,z) - (Y_{1,z}\big|t=1,z,x)\big)^2\right] p(x)\,dx \\
&\quad + 4\int_{\mathcal{X}} \mathbb{E}\left[\big(f(x,t=0,z) - (Y_{0,z}\big|t=0,z,x)\big)^2\right] p(x)\,dx \qquad (4.16) \\
&= 4\int_{\mathcal{X}}\int_{\mathcal{Y}} \big(f(x,t=1,z)-y\big)^2 p(x)p(y|t=1,x,z)\,dx\,dy \\
&\quad + 4\int_{\mathcal{X}}\int_{\mathcal{Y}} \big(f(x,t=0,z)-y\big)^2 p(x)p(y|t=0,x,z)\,dx\,dy \\
&= 4\int_{\mathcal{X}}\int_{\mathcal{Y}} \sum_{t\in\{0,1\}} \big(f(x,t,z)-y\big)^2 p(x)p(y|t,x,z)\,dx\,dy \qquad (4.17) \\
&\leq 4\int_{\mathcal{X}}\int_{\mathcal{T}}\int_{\mathcal{Z}}\int_{\mathcal{Y}} \big(f(x,t,z)-y\big)^2 p(x)p(y|t,x,z)\,dx\,dt\,dz\,dy. \qquad (4.18)
\end{aligned}
$$

Eq. (4.16) can be obtained by Jensen's inequality. Given T is binary, we can unify Eq. (4.17) to Eq. (4.18) with integral. With a similarly technique, we have peer effects Eq. (4.14):

$$
\begin{aligned}
\textit{Eq. (4.14)} &= 2\int_{\mathcal{X}} \big[f(x,t=0,z) - f(x,t=0,0) \\
&\quad - \big(\mathbb{E}(Y_{0,z}\big|t=0,z,x) - \mathbb{E}(Y_{0,0}\big|t=0,0,x)\big)\big]^2 p(x)\,dx \\
&= 2\int_{\mathcal{X}} \big[f(x,t=0,z) - \mathbb{E}(Y_{0,z}\big|t=0,z,x) \\
&\quad + \big(\mathbb{E}(Y_{0,0}\big|t=0,0,x) - f(x,t=0,0)\big)\big]^2 p(x)\,dx \\
&\leq 4\int_{\mathcal{X}}\int_{\mathcal{T}}\int_{\mathcal{Z}}\int_{\mathcal{Y}} \big(f(x,t,z)-y\big)^2 p(x)p(y|t,x,z)\,dx\,dt\,dz\,dy. \qquad (4.19)
\end{aligned}
$$

Finally, add the upper bounds of individual effects Eq. (4.18) and peer effects Eq. (4.19), we can obtain an upper bound of the causal effects estimation error J_{ce} (Eq. (4.7)):

$$J_{ce} \leq 8 \int_{\mathcal{X}} \int_{\mathcal{T}} \int_{\mathcal{Z}} \int_{\mathcal{Y}} \big(f(x,t,z) - y\big)^2 p(x)p(y|t,x,z)\, dx\, dt\, dz\, dy. \tag{4.20}$$

Eq. (4.20) is Eq. (4.8), concluding the proof. □

Distribution mismatch. Comparing Eq. (4.6) and Eq. (4.8), we find that standard graph machine learning actually models two more conditional probabilities $p(t|x)$ and $p(z|x,t)$ than the objective function of causal effects estimation. However, $p(t|x)$ and $p(z|x,t)$ are typically biased in observational data due to confounders (known as confounding bias (Shalit et al., 2017; Forastiere et al., 2021)). Consequently, a graph machine learning model trained on observational data will have biased estimations of the counterfactual outcomes and causal effects, because $p(t|x)$ and $p(z|x,t)$ are different in the counterfactual data. These distribution mismatches lead to the failure of applying standard graph machine learning models to estimate causal effects on networks.

How to fix it. To apply graph machine learning models for causal effects estimation on networks, the distribution gaps must be mitigated. By comparing Eq. (4.6) and Eq. (4.8), we find a sufficient (not necessary) solution is to force $p(t|x)$ and $p(z|x,t)$ as uniform distributions. In other words, causal effects estimation can be reduced into a multi-task graph machine learning problem on networked data. Namely, we can use a data-driven graph machine learning model, with some appropriate and sufficient losses that can force $p(t|x)$ and $p(z|x,t)$ to be uniformly distributed, to estimate the potential outcome. Although the original data generation can not be manipulated, we can achieve this goal by learning representations s_i for every unit i. Our model **NetEst** is motivated by these insights. For consistency, we still use $p(t|x)$ and $p(z|x,t)$ instead of s_i to denote the distributions to be uniformed throughout this paper.

Note that if the treatments are randomly assigned to units in a data collection, e.g., randomized controlled trials, these two conditional distribution $p(t|x)$ and $p(z|x,t)$ actually follow uniform distributions. In this case, it is safe to use a standard machine

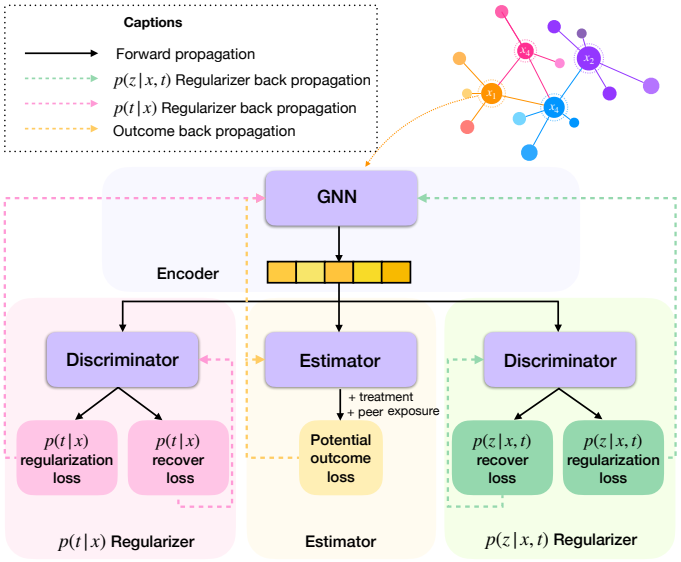

Figure 4.2: The overall framework of NetEst. NetEst is trained adversarially. The unit features and network structure are first encoded into embeddings via GNN. Then, the two discriminators in $p(t|x)$ regularizer and $p(z|x,t)$ regularizer are trained to recover treatment t and peer exposure z from embeddings by optimizing the $p(t|x)$ recover loss and $p(z|x,t)$ recover loss, respectively. With fixed parameters, the two well-trained discriminators optimize the encoder by the $p(t|x)$ regularization loss and the $p(z|x,t)$ regularization loss, together with the potential outcome loss given by the estimator. Solid lines are tensor forward propagation and dotted lines are loss back propagation. Note that the $p(t|x)$ regularization loss and $p(z|x,t)$ regularization loss are not used for the two discriminators although propagated through them.

learning model to estimate causal effects. $p(t|x)$ is usually called propensity score in existing literature (Rosenbaum and Rubin, 1983). Similarly, we refer to $p(z|x,t)$ as the *peer exposure score* in this paper.

4.3.2 NetEst

Our model NetEst follows multi-task paradigm that uses graph machine learning to estimate causal effects on networks. NetEst is composed of four modules: Encoder, $p(t|x)$ Regularizer, $p(z|x,t)$ Regularizer and Estimator. Fig. 4.2 shows the overview of NetEst.

Encoder. The bias of propensity score $p(t|x)$ and peer exposure score $p(z|x,t)$ in observational data is caused by confounders. Traditional methods like matching (Rubin, 2006; Hirano and Imbens, 2004) can partially alleviate this by augmenting counterfactual

data examples according to propensity score. However, we argue that a single scalar propensity score is not enough to capture the high dimensional confounders, especially on networked data. To capture both confounders from individual features x_i and neighbors' features $\{x_j\}_{j\in\mathcal{N}_i}$ while be flexible to later distribution regularization, we propose to learn representation for every unit on networks. In addition, because only immediate neighbors are assumed to have influences on a unit (Fig. 4.1), we just need to capture the features of i's 1-hop neighbors as $\{x_j\}_{j\in\mathcal{N}_i}$. Given this, we use Graph Convolutional Network (GCN) (Kipf and Welling, 2017a) as the representation function ϕ. A GCN layer aggregates the features of immediate neighbors according to a weight w.r.t. both the a unit's and his/her neighbor's degrees. The aggregated new features is then transformed to low-dimensional embeddings. Formally, given a network G, let $r_i^{(l)} \in \mathcal{R}^{d(l)}$ be the embedding of i in the l-th layer, where $d(l)$ is the embedding dimension of the l layer, the embeddings will be forwarded as:

$$r_i^{(l+1)} = \sigma\big(\sum_{j\in\mathcal{N}_i} \frac{1}{\sqrt{d_i d_j}} r_j^{(l)} W^{(l)}\big), \tag{4.21}$$

where $\sigma(\cdot)$ is a non-linear function, d_i and d_j are the degrees of units i and j, respectively. $W^{(l)}$ is a weight matrix of l-th layer, and $\mathcal{N}_i$ is the neighbors of node i. Note that because we need to retain the features of i, $\mathcal{N}_i$ also includes node i. Another benefit of using GCN is that it is applicable to both "inductive" and "transductive" settings, i.e., GCN could make predictions for a new network, or nodes within the same network. This property enables us to estimate both the out-of-sample and within-sample causal effects. The final GCN layer produces the embeddings $s_i = \phi(x_i, \{x_j\}_{j\in\mathcal{N}_i})$, which encodes confounders from both a unit's own and neighbors' features.

p(t|x) Regularizer. Based on the embeddings s_i, we can uniform the propensity score $p(t_i|x_i)$ for every unit i. We propose to use adversarial training paradigm (Goodfellow et al., 2014) to achieve this goal. Specifically, we first train a model (i.e., discriminator) that can recover the treatment t_i for every unit i from the fixed embeddings s_i as much as accurately. Formally, let $d_t : \mathcal{S} \to \{0, 1\}$ be the discriminator, it is trained by the $p(t|x)$

recover loss $\mathcal{J}_{rt}$ as:

$$\mathcal{J}_{rt} = -\frac{1}{V}\sum_{i=1}^{V}\big(t_i \log d_t(s_i) + (1-t_i)\log(1-d_t(s_i))\big). \tag{4.22}$$

Having the well-trained discriminator, we fix it as a "referee" to update the embeddings such that $p(t_i|x_i)$ is close to a uniform distribution. Given that treatment t_i is binary, the probability mass function of a uniformly distributed $p(t_i|x_i)$ is $p(t_i = 0|x_i) = p(t_i = 1|x_i) = 0.5$. Therefore, the $p(t|x)$ *regularization loss* $\mathcal{J}_{ut}$ used to uniform $p(t_i|x_i)$ is as:

$$\mathcal{J}_{ut} = \frac{1}{V}\sum_{i=1}^{V}(d_t(s_i) - 0.5)^2. \tag{4.23}$$

After many interactions of the "adversaries" between the encoder ϕ and discriminator d_t, the embedding s_i can finally be updated such that the discriminator d_t can not identify whether every unit receives treatment or not (both have 0.5 probability), i.e., $p(t|x)$ is forced into a uniform distribution.

p(z/x,t) Regularizer. Similar to the $p(t|x)$ Regularizer, we use another adversarial training paradigm to make the peer exposure score $p(z_i|x_i, t_i)$ uniformed for every unit i. A new discriminator $d_z : \mathcal{S} \times \{0, 1\} \to [0, 1]$ is first trained to recover the peer exposure z_i given embeddings s_i and treatment t_i via the following $p(z|x, t)$ *recover loss* $\mathcal{J}_{rz}$:

$$\mathcal{J}_{rz} = \frac{1}{V}\sum_{i=1}^{V}(d_z(s_i, t_i) - z_i)^2. \tag{4.24}$$

Then, we fix the discriminator d_z to update embeddings s_i to force the peer exposure score $p(z_i|x_i, t_i)$ into uniform distribution. Recall that z_i is defined as ratio of treated neighbors of i, and therefore is a continuous variable between 0 and 1. To approximate a continuous uniform distribution over range $[0, 1]$, we propose to uniformly sample a different value $c_i^s \tilde{} [0, 1]$ for every unit i in every training iteration s, that is to say, every i has a varying label in every iteration. In this case, the predicted $\hat{z}_i = d_z(s_i, t_i)$ can be compared with any value from $[0, 1]$ with equal probability for multiple times. Hence, the

randomly generated labels can mimic a continuous uniform distribution. Formally, the $p(z|x,t)$ *regularization loss* J_{uz} at iteration s is:

$$\mathcal{J}_{uz} = \frac{1}{V}\sum_{i=1}^{V}(d_z(s_i,t_i) - c_i^s)^2. \tag{4.25}$$

Note that as shown in Fig. 4.2, the $p(t|x)$ regularization loss $\mathcal{J}_{ut}$ and $p(z|x,t)$ regularization loss $\mathcal{J}_{uz}$ are only used to optimize the encoder, though they propagate gradients to their discriminators. We parameterize the two discriminators d_t, d_z with neural networks.

Estimator. Another objective is to minimize the observed outcomes estimation errors. We simply use neural networks as the estimator, which takes embeddings s_i, treatment t_i and peer exposure z_i as inputs to estimate the potential outcomes. Formally, for the estimator $m : S \times \{0,1\} \times [0,1] \to \mathcal{Y}$, we have the *potential outcome loss.* $\mathcal{J}_m$:

$$\mathcal{J}_m = \frac{1}{V}\sum_{i=1}^{V}(m(s_i,t_i,z_i) - Y^i_{t_i,z_i})^2. \tag{4.26}$$

Optimization. Algorithm. 1 shows the overall optimization procedure. NetEst optimizes the embedding s_i adversarially: (1) it first well trains the discriminators d_t and d_z by minimizing the $p(t|x)$ recover loss $\mathcal{J}_{rt}$ and $p(z|x,t)$ recover loss $\mathcal{J}_{rz}$, (2) then updates the estimator m with $\mathcal{J}_m$ and optimizes the encoder with a multi-task objective $\mathcal{J}_m + \alpha\mathcal{J}_{ut} + \gamma\mathcal{J}_{uz}$, where α and γ are coefficients that control the strengths of $p(t|x)$ and $p(z|x,t)$ regularization.

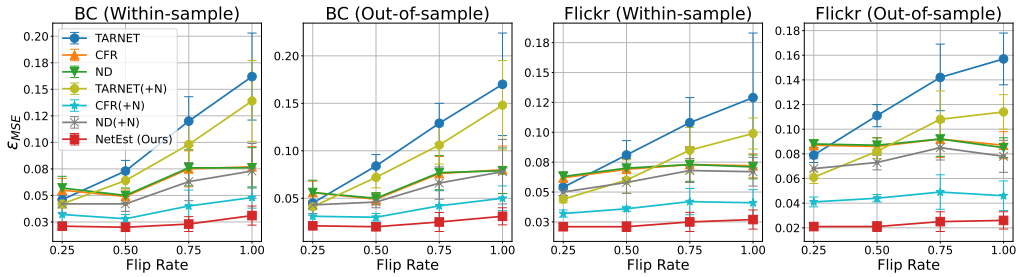

Figure 4.3: Counterfactual estimation errors ϵ_{MSE} v.s. percentages of units whose treatments are flipped (denoted as "flip rate"). From left: BlogCatalog "within-sample", BlogCatalog "out-of-sample", Flickr "within-sample", Flickr "out-of-sample".

4.4 Experiments

In this section, we evaluate the effectiveness of NetEst. We first set up the experiments and then report the results compared to baseline models. We further study why and when NetEst works.

Algorithm 1 The optimization of NetEstimator

Input: Network $G = \langle A, X \rangle$; the observed treatment t_i, peer exposure z_i and outcome $Y^i_{t_i,z_i}$; coefficients α and γ.

Output: Encoder ϕ, $p(t|x)$ Regularizer d_t, $p(z|x,t)$ Regularizer d_z and Estimator m.

Initialize ϕ, d_t, d_z and m;

for w = 1, 2, ..., W **do** ▷ Train model for W epochs

  for o = 1, 2, ..., O **do** ▷ Train d_t for O steps

    Compute $\mathcal{J}_{rt}$;

    Do one step of gradient descent for d_t:

    $\theta_{d_t}^{(o+1)} = \theta_{d_t}^{(o)} - \eta \nabla_{\theta_{d_t}} \mathcal{J}_{rt}$; ▷ η is learning rate

  end for

  for u = 1, 2, ..., U **do** ▷ Train d_z for U steps

    Compute $\mathcal{J}_{rz}$;

    Do one step of gradient descent for d_z:

    $\theta_{d_z}^{(u+1)} = \theta_{d_z}^{(u)} - \eta \nabla_{\theta_{d_z}} \mathcal{J}_{rz}$;

  end for

  for s = 1, 2, ..., S **do** ▷ Train ϕ and m for S steps

    Sample $c_i^s \sim [0, 1]$ for every i;

    Compute $\mathcal{J}_m$, $\mathcal{J}_{ut}$, $\mathcal{J}_{uz}$;

    Do one step of gradient descent for ϕ and m:

    $\theta_{\phi}^{(s+1)} = \theta_{\phi}^{(s)} - \eta \nabla_{\theta_\phi} (\mathcal{J}_m + \alpha \mathcal{J}_{ut} + \gamma \mathcal{J}_{uz})$;

    $\theta_{m}^{(s+1)} = \theta_{m}^{(s)} - \eta \nabla_{\theta_m} \mathcal{J}_m$;

  end for

end for

Return ϕ, d_t,d_z and m.

Table 4.2: Results of causal effects estimation. The PEHE error ϵ_{PEHE} (precision of estimating heterogeneous effects) is reported. The best is boldface while the second best is underlined. "N/A" means the model is not applicable for the peer effects.

Data (Setting)	effects	TARNET	CFR	ND	TARNET(+N)	CFR(+N)	ND(+N)	NetEst_U	NetEst_I	NetEst_P	NetEst
BC (Within-sample)	Individual	$0.1140_{\pm 0.0455}$	$0.1292_{\pm 0.0931}$	$0.1442_{\pm 0.0942}$	$0.1315_{\pm 0.0411}$	$0.1121_{\pm 0.0546}$	$\mathbf{0.0969_{\pm 0.0422}}$	$0.1207_{\pm 0.0345}$	$0.1088_{\pm 0.0452}$	$0.1139_{\pm 0.0437}$	$0.1186_{\pm 0.0542}$
	Peer	N/A	N/A	N/A	$0.4850_{\pm 0.0104}$	$0.3346_{\pm 0.0439}$	$0.4680_{\pm 0.0321}$	$0.1245_{\pm 0.0491}$	$\mathbf{0.0632_{\pm 0.0188}}$	$0.0685_{\pm 0.0176}$	$0.0647_{\pm 0.0188}$
	Total	$0.9952_{\pm 0.0811}$	$0.8708_{\pm 0.0931}$	$0.8558_{\pm 0.0941}$	$0.9027_{\pm 0.0852}$	$0.5566_{\pm 0.1373}$	$0.7472_{\pm 0.1135}$	$0.4101_{\pm 0.0358}$	$0.2268_{\pm 0.0970}$	$0.2483_{\pm 0.0791}$	$\mathbf{0.2214_{\pm 0.1000}}$
BC (Out-of-sample)	Individual	$0.1169_{\pm 0.0457}$	$0.1292_{\pm 0.0931}$	$0.1444_{\pm 0.0944}$	$0.1303_{\pm 0.0406}$	$0.1142_{\pm 0.0540}$	$\mathbf{0.1014_{\pm 0.0443}}$	$0.1199_{\pm 0.0399}$	$0.1040_{\pm 0.0457}$	$0.1114_{\pm 0.0469}$	$0.1159_{\pm 0.0516}$
	Peer	N/A	N/A	N/A	$0.4830_{\pm 0.0110}$	$0.3347_{\pm 0.0440}$	$0.4682_{\pm 0.0323}$	$0.1243_{\pm 0.0498}$	$0.0630_{\pm 0.0198}$	$0.0679_{\pm 0.0182}$	$\mathbf{0.0610_{\pm 0.0181}}$
	Total	$0.9903_{\pm 0.0866}$	$0.8707_{\pm 0.0931}$	$0.8557_{\pm 0.0943}$	$0.8952_{\pm 0.0892}$	$0.5555_{\pm 0.1360}$	$0.0.7438_{\pm 0.1145}$	$0.4051_{\pm 0.0422}$	$0.2205_{\pm 0.1017}$	$0.2436_{\pm 0.0823}$	$\mathbf{0.2166_{\pm 0.1043}}$
Flickr (Within-sample)	Individual	$0.1029_{\pm 0.0231}$	$0.0760_{\pm 0.0445}$	$0.0926_{\pm 0.0470}$	$0.1195_{\pm 0.0434}$	$\mathbf{0.0613_{\pm 0.0306}}$	$0.1483_{\pm 0.0678}$	$0.1855_{\pm 0.0556}$	$0.1529_{\pm 0.0588}$	$0.1632_{\pm 0.0585}$	$0.1513_{\pm 0.0637}$
	Peer	N/A	N/A	N/A	$0.4327_{\pm 0.0177}$	$0.2967_{\pm 0.0370}$	$0.4977_{\pm 0.0066}$	$0.0911_{\pm 0.0188}$	$\mathbf{0.0612_{\pm 0.0298}}$	$0.0759_{\pm 0.0184}$	$0.0734_{\pm 0.0284}$
	Total	$1.0144_{\pm 0.0620}$	$0.9470_{\pm 0.0704}$	$0.9317_{\pm 0.0711}$	$0.8661_{\pm 0.0701}$	$0.5212_{\pm 0.0593}$	$0.8331_{\pm 0.0755}$	$0.3715_{\pm 0.0634}$	$\mathbf{0.2996_{\pm 0.0583}}$	$0.3107_{\pm 0.0669}$	$0.3139_{\pm 0.0545}$
Flickr (Out-of-sample)	Individual	$0.1111_{\pm 0.0215}$	$0.0760_{\pm 0.0445}$	$0.0938_{\pm 0.0467}$	$0.1129_{\pm 0.0376}$	$\mathbf{0.0604_{\pm 0.0297}}$	$0.1346_{\pm 0.0568}$	$0.1769_{\pm 0.0543}$	$0.1464_{\pm 0.0592}$	$0.1546_{\pm 0.0627}$	$0.1392_{\pm 0.0646}$
	Peer	N/A	N/A	N/A	$0.4220_{\pm 0.0204}$	$0.2967_{\pm .0370}$	$0.4876_{\pm 0.0134}$	$0.0827_{\pm 0.0209}$	$\mathbf{0.0544_{\pm 0.0257}}$	$0.0662_{\pm 0.0141}$	$0.0568_{\pm 0.0243}$
	Total	$0.9895_{\pm 0.0648}$	$0.9470_{\pm 0.0704}$	$0.9253_{\pm 0.0621}$	$0.8243_{\pm 0.0654}$	$0.5220_{\pm 0.0586}$	$0.7887_{\pm 0.0822}$	$0.3435_{\pm 0.0647}$	$0.2783_{\pm 0.0572}$	$0.2826_{\pm 0.0586}$	$\mathbf{0.2732_{\pm 0.0571}}$

4.4.1 Experiments Setup

Datasets. For every unit i, only one treatment t_i, peer exposure z_i and outcome $Y^i_{t_i,z_i}$ can be observed(i.e., factual outcome). We can never know the groundtruth counterfactual outcome, and thus it is impossible to evaluate causal effects estimation directly. Therefore, following (Veitch et al., 2019; Ma et al., 2021; Guo et al., 2020c), we use semi-synthetic datasets, i.e., the networks (features, topology) are real but treatments and potential outcomes are simulated. We use two real-world social networks BlogCatalog and Flickr (Guo et al., 2020c; Ma et al., 2021). In both datasets, a unit (node) is a user and an edge indicates their social relationship. Because the raw features of units are high-dimensional and very sparse, following (Guo et al., 2020b; Ma et al., 2021), we use LDA (Blei et al., 2003) to reduce the dimension to 10. "Out-of-sample" estimation requires we have a new network without observed outcomes, therefore, we use METIS (Karypis and Kumar, 1998) to partition the original network into three sub-networks as train/valid/test respectively. We evaluate the "within-sample" estimation on train networks and "out-of-sample" on the test network. Treatments and potential outcomes are simulated according to Fig. 4.1.

Treatments simulation. The treatment t_i is affected by i's features x_i and i's neighbors' features $\{x_j\}_{j\in\mathcal{N}_i}$. Let w_{X_1} be a randomly generated weight vector, then unit i's "propensity to treatment" pt_i is defined as $pt_i = \sigma(w_{X_1} \cdot x_i)$, where $\sigma(\cdot)$ is the sigmoid function. w_{X_1} mimics the causal mechanism of the confounders to treatments. We denote by pt_{N_i} the average of all i's neighbors' propensities, and denote by $tpt_i = \beta_x * pt_i + \beta_n * pt_{N_i}$

the total propensity to treatment of i. Then the treatment t_i is generated following:

$$t_i = \begin{cases} 1 & \text{if } tpt_i > \overline{tpt} \\ 0 & \text{else} \end{cases}, \tag{4.27}$$

where $\overline{tpt}$ is the average of all tpt_i. We set both β_x and β_n as 1. Given t_i and the network topology $\mathcal{A}$, the peer exposure z_i—the ratio of treated neighbors of i—can then be easily calculated.

Potential outcomes simulation. The potential outcome $Y_i|do(t_i, z_i)$ of i is affected by four factors: i's treatment t_i, peer exposure z_i, i's features x_i and i's neighbors' features $\{x_j\}_{j\in\mathcal{N}_i}$. We define "propensity to outcome" $po_i = \sigma(w_{X_2} \cdot x_i)$, where w_{X_2} is randomly generated to represent the causal mechanism of features to potential outcomes. Similarly, We let po_{N_i} be the average of all i's neighbors' propensities. Then the potential outcome is simulated by:

$$Y_i|do(t_i, z_i) = \beta_t \cdot t_i + \beta_z \cdot z_i + \beta_p \cdot po_i + \beta_o \cdot po_{N_i} + \epsilon, \tag{4.28}$$

where ϵ is a noise term. The parameters β_t, β_z, β_p and β_o are strengths to potential outcome of treatment, peer exposure, features, features of neighbors, respectively. We set β_t, β_z, β_p as 1 and β_o as 0.5. following the intuition that a unit's own features should have stronger effects than their neighbors. We use fixed parameters across networks because the causal mechanism is invariant.

Metrics. We consider two metrics: Mean Squared Error ($\epsilon_{MSE} = \frac{1}{V}\sum_{i=1}^{V}(\hat{y}_i - y_i)^2$) for counterfactual estimation where $\hat{y}_i$ and y_i are the estimated and groundtruth potential outcomes, respectively, and ($\epsilon_{PEHE} = \sqrt{\frac{1}{V}\sum_{i=1}^{V}\left(\hat{\tau}(X) - \tau(X)\right)^2}$) for causal effects estimation, where $\hat{\tau}(X)$ is the estimation and $\tau(X)$ is groundtruth. Lower is better for both metrics.

Baselines. NetEst is compared with six baselines and three variants. **CFR** (Shalit et al., 2017): State-of-the-art model for causal effects estimation on independent data,

which is optimized by the estimating observed outcomes estimation, and a so-called Integral Probability Metrics(IPM) that forces treated and control group to be closer. We use the Wasserstein distance implementation of IPM. **TARNet** (Shalit et al., 2017): a variant of CFR without IPM. **NetDeconf** (Guo et al., 2020c): extension of CFR to networked data, which uses GNN for encoding confounders, and Wasserstein distance for representations balancing. **CFR+(N)**, **TARNet+(N)**, **NetDeconf+(N)**: because the above three models do not consider interference, we add the peer exposure (+N) as extra input to them to evaluate their ability under network interference. **NetEst_U, NetEst_I, NetEst_P**: variants of NetEst without any regularizers ($\alpha = \gamma = 0$), only with $p(t|x)$ regularizer ($\alpha = 0.5$, $\gamma = 0$), and only with $p(z|x,t)$ regularizer ($\alpha = 0$, $\gamma = 0.5$), respectively.

Implementation details. We build our model as follow. We use 1 graph convolution layer as encoder[7]. We use 3 fully-connected layers for estimator and the two discriminators. All hidden embedding size is 32. Coefficient α and γ are set as 0.5. For hyperparameters, we use full-batch training and set the learning rate to 0.001 for all modules. All parameters are randomly initialized and updated by the Adam optimizer (Kingma and Ba, 2015a). We run every task for five times (including simulation) and reported the average and 1-standard deviation. The experiment environment is an AWS *g4dn.4xlarge* instance.

4.4.2 Results Comparison

As stated in Sec. 4.2.2, we estimate the counterfactual outcomes and predict three interesting causal effects: individual effects, peer effects and total effects. For counterfactual estimation, a counterfactual treatments assignment T is over the entire network, we therefore simulate the counterfactual outcomes by flipping the treatments of randomly sampled subgroups of units. We try flip rates in $\{0.25, 0.5, 0.75, 1\}$ and report the counterfactual estimation errors in Fig. 4.3. In general, NetEst consistently outperform all baselines in "within-sample" and "out-of-sample" estimations on both datasets, suggesting

[7]Note 1 layer is consistent with the Markov assumption of network effects in Sec. 4.2.3. More layers may be necessary if network effects are beyond the 1-hop neighbors

the *effectiveness* of our model in handling the confounding bias. We notice all models' errors increase with a larger flip rate, but NetEst is still *robust*, showing a much lower error even we flip the treatments of 100% units. NetEst and its variants exhibit a similar superiority against baselines in predicting the three causal effects (Table. 4.2). We observe that NetEst works generally better than other models in estimating the total effects. This empirically demonstrates our conclusion in Sec. 4.3.1 forcing the $p(t|x)$ and $p(z|x,t)$ into uniform distributions is essential for causal effects estimation on networks, which is derived by studying the objective function of causal effects. We note that the two variants NetEst_U and NetEst_I works better under some settings. We speculate that forcing $p(t|x)$ into uniform distribution will also make $p(z|x,t)$ close to be uniformed and vice versa, since both regularizers basically enforce the embeddings to be close with each other (validated later in Fig. 4.5).

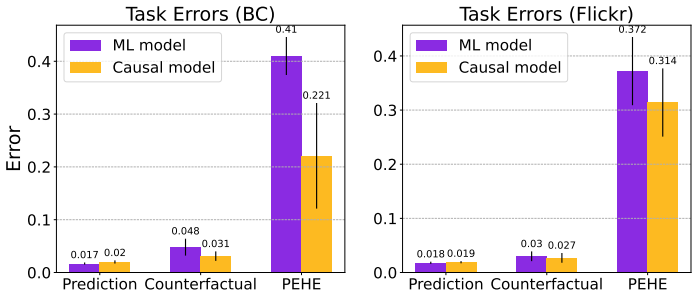

Figure 4.4: Errors of causal model NetEst and graph machine learning model (GCN) on three tasks: prediction, counterfactual estimation and causal effects estimation.

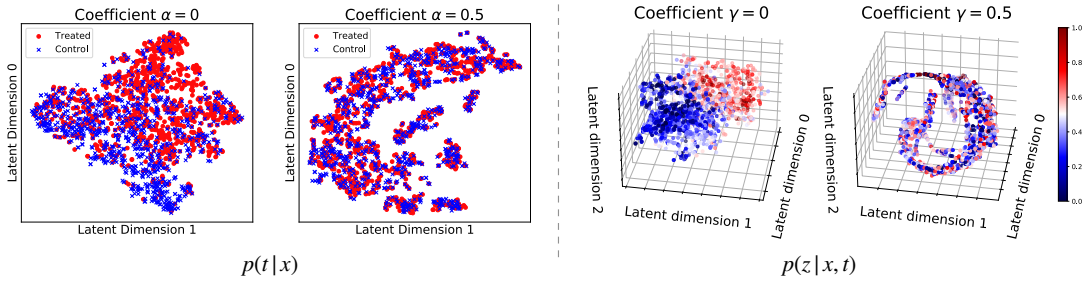

Figure 4.5: T-SNE projections of learned units' embeddings s_i. *Left*: without (α=0) and with (α=0.5) the regularizer for $p(t|x)$. A red point is a unit who was treated while a blue point means a controlled unit. *Right*: without (γ=0) and with (γ=0.5) the regularizer for $p(z|x,t)$. Units are colored by their observed peer exposures z. Red means a higher z, i.e., ratio of treated neighbors in this paper, while blue indicates a lower z. We use 3D projection for $p(z|x,t)$ as z is continuous.

4.4.3 Why Does NetEst Work?

Motivation. We motivate NetEst by modifying the objective functions of graph machine learning models for causal effects estimation. To verify this modification, we compare NetEst with graph machine learning model GCN in Fig. 4.4. Causal model NetEst has worse performance on general prediction task compared to GCN but better on causal estimation tasks. It shows, as intended, forcing $p(t|x)$ and $p(z|x,t)$ into uniform distributions sacrifices the general prediction performance but alleviates the distribution mismatches and therefore favors beneficial for causal problem.

Visualized interpretation. The uniform regularizers are conducted on the embeddings. We further visualize the learned embeddings s_i to understand why adversarial training works. We use t-SNE (Van der Maaten and Hinton, 2008) to project units' embeddings s_i into 2-dimension colored by their binary treatments t_i and 3-dimension colored by their peer exposures z_i in for clearness in Fig. 4.5. With the distribution regularizers, the units points are highly overlapped on both figures. This overlapping means that for a given unit s_i, the discriminators d_t, d_z can not recover the treatment t and peer exposure z, suggesting t and z are uniformly distributed given s_i.

4.4.4 When Does NetEst Work?

The potential outcome scale varies a lot in observational data. We stratify units by their potential outcomes and break down the counterfactual estimation errors in Table. 4.3. We find NetEst works much better on moderate samples than extreme ones. We speculate NetEst can not alleviate the weakness of machine learning models on extreme data just with the proposed distribution regularizers. Understanding and solving this challenge is an interesting future direction.

Table 4.3: Counterfactual estimation errors according to potential outcome percentile. MSE error ϵ_{MSE} is reported.

Potential outcome strata	BC	Flickr
0-10%	0.3008±0.1529	0.2642±0.0784
10%-50%	0.1192±0.0375	0.0966±0.0143
50%-90%	0.0614±0.0235	0.0536±0.0060
90%-100%	0.2423±0.1756	0.4767±0.1368

4.5 Conclusion

This paper studies causal effects estimation on networked data. We theoretically show the objective function of standard graph machine learning has two distribution mismatches against causal effects estimation, motivating our model NetEst that mitigates the distribution gaps via representation learning. Future works could study finding out the optimal treatment strategy, such as vaccine distribution plan, on networks based on estimated causal effects.

CHAPTER 5

CF-GODE: Continuous-Time Causal Inference for Multi-Agent Dynamical Systems

5.1 Introduction

Estimating counterfactual outcomes *over time* is critical to gaining causal understanding for many useful practical applications, such as how to distribute the limited vaccines in the early days to maximize protection over time (Medlock and Galvani, 2009), or how to design proper scheduling of medical treatments to optimize the patient recovery process (Bica et al., 2020). Randomized controlled trials (RCTs) are the gold standard for causal inference, but they can be cost-prohibitive and ethically challenging, particularly when considering the dynamical settings described above. Therefore, estimating counterfactual outcomes from observational data is the key approach to answering causal questions in real-world scenarios. Existing research on observational causal inference over time has begun by utilizing basic linear regression (Robins et al., 2000) and Gaussian processes (Xu et al., 2016) to capture the time-dependencies. Subsequently, advancements have been made by incorporating more advanced deep learning models such as recurrent neural networks (RNNs) (Bica et al., 2020; Fujii et al., 2022) and Transformers (Melnychuk et al., 2022).

Despite the progress, all aforementioned studies have relied on the assumption that units (e.g., people in the vaccine example) are independent of each other, i.e., each unit is solely influenced by its own treatment but not by others. In many realistic scenarios, however, this assumption is not valid. For instance, a person's vaccination not only protects themselves but also those close to them. This type of setting is referred to as a

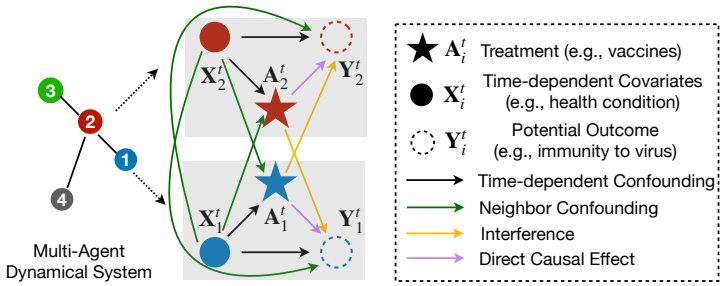

Figure 5.1: Causal graph at time t in a multi-agent dynamical system. The causal variables are represented by shapes, while their relationships are distinguished by colors.

multi-agent dynamical system (Gazi and Fidan, 2007), where units (also known as agents) interact with each other and evolve collectively over time. Many practical problems can be expressed as multi-agent dynamical systems, such as the long-term effects of vaccination where people mutually influence (Halloran and Hudgens, 2012), brain network signals in which the regions of interest (ROI) in a brain are associated (Yu et al., 2022; Cui et al., 2022), and molecular systems movements where the atoms are interconnected (Durrant and McCammon, 2011). Prior approaches for causal inference over time are not applicable to multi-agent dynamical systems since they are not capable of handling the interconnections between units. In this paper, we propose to study this novel problem *counterfactual estimation in multi-agent dynamical systems*, which has received limited attention in the literature.

The dynamic and interrelated nature of multi-agent dynamical systems poses unique and nontrivial challenges to causal inference. We illustrate them along with Fig. 5.1. 1) **Multi-source confounders**. Confounders are variables that have an impact on both treatments and outcomes, leading to spurious correlations between them. Therefore, in observational data, the treatments are not balanced among units with different confounders values, resulting in biased counterfactual outcomes estimation. For example, old people are more likely to receive vaccines, but also face a higher risk of virus infection. If we train a standard supervised model using such imbalanced data, it may wrongly predict that vaccines may increase the infection risk for young people. In multi-agent dynamical systems, the confounders are multi-source, including *time-dependent confounders* and

neighbor confounders. Time-dependent confounders refer to the fact that the confounders typically evolve over time and thus their impact on treatments and outcomes also changes dynamically (Platt et al., 2009; Bica et al., 2020). For instance, people's health conditions change over time, affecting their likelihood of getting vaccinated and future health status. Neighbor confounders mean that a unit's treatment and outcome could also be confounded by the covariates of its near units (neighbors) (Forastiere et al., 2021; Arbour et al., 2016), For example, if family members are in poor health, a unit may be more likely to receive a vaccine. Compared to the independent setting, neighbor confounders are additional confounding factors in multi-agent dynamical systems. 2) **Imbalance of interference**. As discussed in the previous example that vaccines protect not only a unit but also those in close proximity, the outcome of a unit can be influenced by others' treatments in multi-agent dynamical systems. In causal language, this phenomenon is referred to as *interference*. Similar to the treatments, interference is affected by the covariates and thus is not balanced across the units in observational data (Forastiere et al., 2021). For instance, highly educated units are more likely to receive vaccines and typically have more highly educated friends. Therefore, they receive stronger protection through higher vaccination rates among their social networks. Such imbalanced interference causes additional bias in the estimation of counterfactual outcomes. 3) **Continuous dynamics**. In realistic applications, a multi-agent dynamical system is continuous in nature (Porter and Gleeson, 2014). However, most existing causal models are discrete, making them inappropriate for multi-agent dynamical systems. Modeling continuous-time observations (such as covariates and outcomes) and continuously estimating counterfactual outcomes over time remains an open challenge.

In this paper, we address the above challenges and study how to estimate continuous-time counterfactual outcomes, in presence of multi-source confounders and interference, in multi-agent dynamical systems. This is a novel, yet challenging and under-explored problem with valuable real-world applications.

To this end, we model a multi-agent dynamical system as a *graph*, where nodes represent units and edges capture their interactions. Inspired by recent achievements in graph

ordinary differential equations (GraphODE) (Huang et al., 2020), we propose CF-GODE, a novel *causal* model that estimates continuous-time **C**ounter**F**actual outcomes based on **G**raph **O**rdinary **D**ifferential **E**quations in multi-agent dynamical systems. Specifically, we use GraphODE as a backbone to model the continuous trajectory of each unit. However, in this case, traditional GraphODE can only model the pure dynamics of potential outcomes (Huang et al., 2020) and lacks the ability to incorporate additional inputs such as treatments, making it inappropriate for causal inference. To address this issue, in CF-GODE, we propose *Treatment-Induced GraphODE*, a new GraphODE model capable of handling treatments when predicting the future trajectory of potential outcomes. Treatment-Induced GraphODE uses graph neural networks (GNNs) (Kipf and Welling, 2017b) to formulate its differential equations, which can effectively capture the mutual dependencies between units including neighbor confounders and interference. This advantage makes it a natural fit for counterfactual estimation in multi-agent dynamical systems. Then a latent representation is learned for each unit from its observations as the solution to Treatment-Induced GraphODE, which represents the continuous trajectory driven by treatments. The core of ensuring CF-GODE is a causal model is to deal with the aforementioned estimation bias caused by imbalanced treatments and interference in the observational data. We solve this issue via domain adversarial learning (Ganin et al., 2016; Wang et al., 2020), in which we treat the values of treatments (and interference) as domains and ensure the latent representation trajectories are invariant to them. We provide theoretical justification to demonstrate that the domain-adversarial balancing objective functions proposed in CF-GODE can effectively achieve the balancing goal, thereby removing bias in counterfactual estimation and ensuring that CF-GODE is causal.

We summarize our major **contributions** as follows: 1) We study how to estimate counterfactual outcomes in multi-agent dynamical systems, which is a novel yet challenging problem with useful practical implications. 2) We propose CF-GODE, a novel causal model for causal inference multi-agent dynamical systems based on GraphODE and domain-adversarial learning. 3) We provide theoretical analysis to show that CF-GODE is able to handle the imbalanced treatments and interference, ensuring unbiased counterfactual

estimation. 4)We conduct extensive experiments to evaluate CF-GODE's performance on counterfactual outcomes estimation in multi-agent dynamical systems.

5.2 Problem Setup

5.2.1 Problem Formulation

We study how to estimate counterfactual outcomes in the context of multi-agent dynamical systems, where the units engage in mutual interactions and evolve simultaneously over time. Throughout this paper, we use boldface uppercase letters to denote matrices or vectors, boldface uppercase letters with subscripts to signify elements of matrices or vectors, regular lowercase letters to represent values of variables, and calligraphic uppercase letters to indicate sets.

Formally, a multi-agent dynamical system can be represented by a dynamical graph $\mathcal{G}^t = (\mathcal{V}, \mathcal{E}^t)$, where $\mathcal{V} = \{v_1, v_2, ..., v_N\}$ is the set of N units (nodes) and $\mathcal{E}^t$ denotes the edge set at time t. An edge in $\mathcal{E}^t$ describes the intersection between the two units it connects at time t. In this paper, we present an early exploration of causal inference in multi-agent dynamical systems, and for the purpose of simplicity, we assume that the graph structure remains constant over time, i.e., $\mathcal{G}^t = \mathcal{G}$. Each unit is associated with time-varying variables, which are the causal quantities in our case. We introduce them together with the causal framework in the following.

We follow the longitudinal potential outcomes framework (Robins and Hernán, 2009; Rubin, 1978) to formalize the counterfactual outcome estimation as in (Bica et al., 2020; Seedat et al., 2022). The observational data $((\mathbf{X}^t, \mathbf{A}^t, \mathbf{Y}^t) \cup \mathbf{V})$ in a multi-agent dynamical system contains time-dependent covariates $\mathbf{X}^t$ (e.g., health condition), dynamical treatments $\mathbf{A}^t$ (e.g., vaccine allocation), and time-varying outcomes $\mathbf{Y}^t$ (e.g., immunity to infectious disease). It is worth noting that $\mathbf{Y}^t$ is essentially a part of $\mathbf{X}^t$. $\mathbf{V}$ denotes the static covariates of units such as ethnicity. Let the historical records of the multi-agent dynamical system up to time t be represented by $\mathcal{H}^t = \{\bar{\mathbf{X}}^t, \bar{\mathbf{A}}^t, \bar{\mathbf{Y}}^t, \mathbf{V}\}$, where $\bar{\mathbf{X}}^t, \bar{\mathbf{A}}^t, \bar{\mathbf{Y}}^t$

are all the $\mathbf{X}^{t^-}, \mathbf{A}^{t^-}, \mathbf{Y}^{t^-}$ until t ($t^- \leq t$), respectively. In causal inference, we are focused on understanding the potential outcomes $\mathbf{Y}^{t^+}(\mathbf{A}^{t^+} = a)$ that may occur in the future ($t^+ > t$) under a specific treatment a, which explains the impact of the treatment assignment on the dynamics of the system.

Note that a is a treatment trajectory that includes all treatments in the future time. Our goal is to estimate the future potential outcomes sequence driven by treatments in a multi-agent dynamical system, which is formalized as:

$$\mathbb{E}\left(\mathbf{Y}^{t^+}(\mathbf{A}^{t^+} = a) \mid \mathcal{H}^t, \mathcal{G}\right). \tag{5.1}$$

5.2.2 Causal Identification

The potential outcomes represented by $\mathbf{Y}^{t^+}(\mathbf{A}^{t^+} = a)$ are a causal quantity. To make it identifiable from observational data, we must adhere to the following necessary assumptions.

Assumption 1: Positivity (Overlap). The future treatment trajectory is probabilistic regardless of the historical observation, i.e., $0 < P(\mathbf{A}^{t^+} = a \mid \mathcal{H}^t) < 1, \forall \mathcal{H}^t$.

Assumption 2: Consistency. Under the same treatment trajectory a, the potential outcome is equal to the observed outcomes, i.e., $\mathbf{Y}^{\mathbf{t}^+}(\mathbf{A}^{t^+} = a) = Y^{t^+}$.

The above two assumptions are standard for longitudinal counterfactual estimation. To identify the potential outcomes, it is also necessary to assume that there are no unobserved confounders, i.e., the strong ignorability assumption. However, the typical sequential strong ignorability assumption (Bica et al., 2020; Seedat et al., 2022; Melnychuk et al., 2022; Lim et al., 2018) is not appropriate for multi-agent dynamical systems, because the graph structure $\mathcal{G}$ introduces extra graph confounders and interference. A plausible strong ignorability assumption for graphs is first introduced by (Forastiere et al., 2021) and later validated in studies such as (Ma and Tresp, 2021; Ma et al., 2022; Jiang and Sun, 2022). However, the assumption made in these works is limited to static settings. To address this, we extend it to longitudinal settings and adapt it to be applicable to

multi-agent dynamical systems in the following.

We first introduce a summary function, denoted as $g(\cdot)$, that captures the interference effects caused by the treatments of a node's neighboring units in the graph as in (Forastiere et al., 2021). Formally, $\mathbf{G}_i^t = g(\mathbf{A}_{\mathcal{N}_i}^t, \mathbf{A}_{\mathcal{N}_{-i}}^t)$, where $\mathbf{A}_{\mathcal{N}_i}^t$ denotes the treatments of node i's immediate neighbors, and $\mathbf{A}_{\mathcal{N}_{-i}}^t$ is the treatments of all the remaining node that are not directly connected to node i. We refer to $\mathbf{G}_i^t$ as *interference summary*. Here for simplicity, we adopt the assumption put forth in (Forastiere et al., 2021; Arbour et al., 2016; Jiang and Sun, 2022) that a node is only influenced by the treatments of its immediate neighbors, i.e., $g(\mathbf{A}_{\mathcal{N}_i}^t, \mathbf{A}_{\mathcal{N}_{-i}}^t) = g(\mathbf{A}_{\mathcal{N}_i}^t, \mathbf{A}'^t_{\mathcal{N}_{-i}}) = g(\mathbf{A}_{\mathcal{N}_i}^t), \forall \mathbf{A}_{\mathcal{N}_{-i}}^t, \mathbf{A}'^t_{\mathcal{N}_{-i}}$. $g(\cdot)$ can be instantiated using any aggregation functions or models. As in previous studies (Forastiere et al., 2021; Ma and Tresp, 2021; Jiang and Sun, 2022), in this paper, we define $\mathbf{G}_i^t$ as the proportion of treated units in unit i's neighbors, i.e., $\mathbf{G}_i^t := \sum_{j \in \mathcal{N}_i} \frac{\mathbf{A}_i^t}{|N_i|}$. With $\mathbf{G}_i^t$, we present the strong ignorability assumption for multi-agent dynamical systems in the following:

Assumption 3: Strong Ignorability for Multi-Agent Dynamical Systems[8]. Given the historical observations and the graph structure that describes the multi-agent dynamical system, the potential outcome trajectory is independent of the treatments and interference summary, i.e., $\mathbf{Y}^{\mathbf{t}^+}(\mathbf{A}^{t^+} = a) \perp\!\!\!\perp \mathbf{A}^{t^+}, \mathbf{G}^{t^+} \mid \mathcal{H}^t, \mathcal{G}. \forall a, t.$

With these three assumptions, the potential outcome trajectory Eq. (5.1) can be identifiable as:

$$\begin{aligned}
&\mathbb{E}\left(\mathbf{Y}^{t^+}(\mathbf{A}^{t^+} = a) \mid \mathcal{H}^t, \mathcal{G}\right) \\
&= \mathbb{E}\left(\mathbf{Y}^{t^+}(\mathbf{A}^{t^+} = a) \mid \mathbf{A}^{t^+}, \mathbf{G}^{t^+}, \mathcal{H}^t, \mathcal{G}\right) \qquad (5.2) \\
&= \mathbb{E}\left(\mathbf{Y}^{t^+} \mid \mathbf{A}^{t^+}, \mathbf{G}^{t^+}, \mathcal{H}^t, \mathcal{G}\right). \qquad (5.3)
\end{aligned}$$

Eq. (5.2) is true because of assumption 3, while Eq. (5.3) holds under the assumption 2. The above causal identification enables us to estimate the potential outcomes in multi-agent dynamical systems using observational data. More specifically, we can train

[8]Note that similar to the strong ignorability assumptions in static or non-graph sequential settings, assumption 3 can not be verified only from data.

a machine learning model on observational data, which takes treatment trajectory $\mathbf{A}^{t^+}$, interference summary $\mathbf{G}^{t^+}$, historical observation $\mathcal{H}^t$ and graph $\mathcal{G}$ as inputs, and the observed (factual) outcome $\mathbf{Y}^{t^+}$ as targets, to predict the counterfactual outcomes given new treatment trajectories. Our proposed model CF-GODE is grounded in this and will be presented in detail in the subsequent section.

5.3 Proposed Model: CF-GODE

5.3.1 Overview

Our proposed CF-GODE is a causal model that predicts counterfactual outcomes in a multi-agent dynamical system by learning from observational data. We show an overview of our model in Fig. 5.2. Compared to most existing causal models designed for standard sequential settings that consider discrete time intervals and independent units (Bica et al., 2020; Melnychuk et al., 2022), multi-agent dynamical systems are more realistic and present two challenging properties: the dynamics are *continuous* in nature, and units are *influenced* by others. To address these, our proposed CF-GODE takes the advantage of recent breakthroughs in graph ordinary differential equations (GraphODE) (Huang et al., 2020, 2021) and extends it to handle treatments and interference, enabling continuous estimation of counterfactual outcomes in multi-agent dynamical systems. We refer to our ODE model as *Treatment-Induced GraphODE* (Sec. 5.3.2). The time-dependent confounders lead the distribution of covariates to be quite discrepant between units assigned to different treatments, resulting in high variances in counterfactual outcome estimation (Johansson et al., 2016; Shalit et al., 2017; Robins et al., 2000). This effect is further amplified by the imbalanced interference caused by the graph structure in multi-agent dynamical systems (Forastiere et al., 2021; Jiang and Sun, 2022). CF-GODE uses adversarial learning to alleviate this issue and guarantee unbiased estimates of counterfactual outcomes (Sec. 5.3.3).

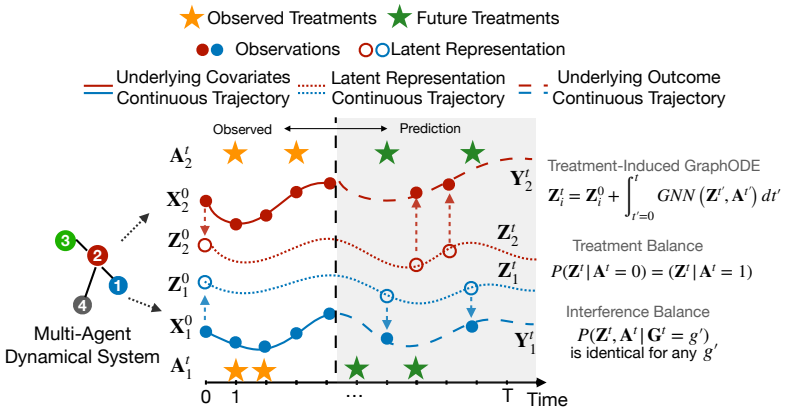

Figure 5.2: Overview of CF-GODE. The initial latent representation $\mathbf{Z^0}$ is first learned from initial observations. Then the continuous latent representation trajectory $\mathbf{Z^t}$ is learned as the solution to treatment-induced GraphODE, which is able to handle treatments as additional inputs. The graph neural network (GNN) based ODE function naturally models the mutual dependencies. The future potential outcomes can be decoded from $\mathbf{Z^t}$ at any given time. To remove confounding bias, $\mathbf{Z^t}$ is balanced with respect to 1) treatments, 2) interference when combined with corresponding treatments.

5.3.2 Treatment-Induced GraphODE

To facilitate continuous-time counterfactual outcome estimation, we propose to learn a continuous latent trajectory $\mathbf{Z}_i^t$ for every node in multi-agent dynamical system that represents their movement. An ideal $\mathbf{Z}_i^t$ should possess two characteristics: 1) the ability to predict observed outcomes, and 2) not to be predictive of the received treatment or interference in observational data[9]. We implement such a $\mathbf{Z}_i^t$ by a novel model called Treatment-Induced GraphODE, which empowers the recent GraphODE (Huang et al., 2020, 2021) to deal with treatment and interference for counterfactual outcomes estimation.

In a multi-agent dynamical system, the future outcomes of node i might be affected by not only its own past movement and current treatment, but also the movements and interference from neighbors (e.g., a unit's health condition and vaccination status have a significant impact on how likely others are to be infected). We model this process and

[9]The second characteristic is discussed in Sec. 5.3.3.

formalize treatment-induced GraphODE as:

$$\mathbf{Z}_i^t = \mathbf{Z}_i^0 + \int_{t'=0}^{t} GNN\left(\mathbf{Z}_1^{t'}, \mathbf{Z}_2^{t'}, ..., \mathbf{Z}_N^{t'}\right) dt'. \tag{5.4}$$

In Eq. (5.4), $\mathbf{Z}^{t'}$ and $\mathbf{A}^{t'}$ denote the latent trajectory representations and treatments of all nodes in the multi-agent dynamical system, respectively. $\phi(\cdot)$ is the ODE function. To comprehensively capture the effects from node i and its connected neighbors, we parameterize $\phi(\cdot)$ using graph neural networks (Kipf and Welling, 2017b) with self-loops. $\mathbf{Z}_i^0$ is the initial state and can be encoded from the initial observations as $\mathbf{Z}_i^0 = f(\mathbf{X}_i^0, \mathbf{V}_i)$, where $f(\cdot)$ is an encoder parameterized by neural networks. With $\mathbf{Z}_i^0$, we can obtain the $\mathbf{Z}_i^t$, which is the solution to treatment-induced GraphODE, by solving an ODE initial-value problem (IVP) in Eq. (5.4), formalized as:

$$\mathbf{Z}_i^0, \mathbf{Z}_i^1 \cdots \mathbf{Z}_i^T = \text{ODESolve}\left(\phi, [\mathbf{Z}_1^0, \mathbf{Z}_2^0 \cdots \mathbf{Z}_N^0], (t_0, t_1 \cdots t_T)\right), \tag{5.5}$$

where T is the number of timestamps for the evaluation of Eq. (5.5). With the solution latent trajectory $\mathbf{Z}_i^t$, we can then use a decoder $d_{\mathbf{Y}}(\cdot)$ to transform it to the predicted outcome $\hat{\mathbf{Y}}_i^t = d_{\mathbf{Y}}(\mathbf{Z}_i^t)$. We also use neural networks to instantiate $d_{\mathbf{Y}}(\cdot)$. We compare the prediction $\hat{\mathbf{Y}}_i^t$ to ground-truths $\mathbf{Y}_i^t$ in all observed timestamps $(t_0, t_1 \cdots t_T)$ using a mean square error as objective, which is formalized as:

$$L^{\langle Y \rangle} = \frac{1}{N}\frac{1}{T}\sum_{i}^{N}\sum_{t}^{T}\left(\hat{\mathbf{Y}}_i^t - \mathbf{Y}_i^t\right)^2. \tag{5.6}$$

5.3.3 Balancing via Adversarial Learning

In the observational data, the treatments applied to each unit $\mathbf{A}_i^t$ are affected by the time-dependent confounders present in the covariates (and thus in its latent representation trajectory $\mathbf{Z}_i^t$). Consequently, the distribution of latent representation trajectory is not balanced among units with different treatment assignments, i.e., $P(\mathbf{A}_i^t|\mathbf{Z}_i^t)$ is not uniform, leading to high variances in the counterfactual outcome estimation (Johansson et al., 2016;

Shalit et al., 2017). In the context of multi-agent dynamical systems, this effect is further exacerbated by the presence of *imbalanced interference* among units. This is because a unit's interference is influenced by its covariates (also the latent representation) and treatments in the observational data, i.e., $P(\mathbf{G}_i^t|\mathbf{Z}_i^t, \mathbf{A}_i^t)$ is not uniform (Forastiere et al., 2021; Jiang and Sun, 2022; Ma and Tresp, 2021). Here we give an intuitive example of the imbalanced interference: consider that a highly educated person is more likely to be surrounded by other highly educated friends, who believe in science and are more likely to be vaccinated, thereby providing stronger protection for this person against infectious diseases, i.e., higher interference.

A sufficient condition to remove the above bias is to ensure that the distribution of latent representation trajectories is invariant to treatments, and when combined with the corresponding treatments, is interference-invariant (Shalit et al., 2017; Bica et al., 2020; Seedat et al., 2022; Forastiere et al., 2021; Jiang and Sun, 2022). This condition is formalized as $P(\mathbf{Z}^t|\mathbf{A}^t = 0) = (\mathbf{Z}^t|\mathbf{A}^t = 1)$ for treatment balancing, and $P(\mathbf{Z}^t, \mathbf{A}^t|\mathbf{G}^t = g')$ is identical for any given value of g' for interference balancing. The treatment $\mathbf{A}^t$ is binary and interference $\mathbf{G}^t$ is continuous as in (Forastiere et al., 2021; Jiang and Sun, 2022). Note that the aforementioned conditions are over the unit groups. This guarantees that the treatment cannot be inferred from the latent representation trajectory, and that the interference is not predictable when the treatment is combined with latent representation. We implement this balancing goal through domain adversarial learning (Ganin et al., 2016), in which the treatment is treated as binary domains and the interference is treated as continuous domains (Wang et al., 2020). Specifically, we use the gradient reversal layer proposed in (Ganin et al., 2016), denoted as $r(\cdot)$, to adversarially optimize the latent representation trajectory at every observed time, making it agnostic towards the treatments and interference.

Treatment Balancing. Formally, the predicted treatment is $\hat{\mathbf{A}}_i^t = d_{\mathbf{A}}\left(r(\mathbf{Z}_i^t)\right)$, where the $d_{\mathbf{A}}$ is a neural network that attempts to recover the treatment from latent representation. The gradient reversal layer $r(\cdot)$ does nothing in the forward pass, but reverses the gradients in the back-propagation. This way, a min-max game is created in

which $d_{\mathbf{A}}$ aims to minimize the treatment prediction loss, while the latent representation learner in treatment-induced GraphODE strives to maximize it, as formalized in the following:

$$L^{\langle A \rangle} = \min_{d^j_{\mathbf{A}}} \max_{f,\phi} \frac{1}{N}\frac{1}{T}\sum_{i}^{N}\sum_{t}^{T}\sum_{j\in\{0,1\}} \mathbb{1}_{(\mathbf{A}_i^t=j)} - \log\left(d^j_{\mathbf{A}}\left(r(\mathbf{Z}_i^t)\right)\right), \tag{5.7}$$

where $d^j_{\mathbf{A}}$ represents the logits of $d_{\mathbf{A}}(\cdot)$ for predicting treatment j. We then provide a theoretical analysis to justify the capability of $L^{\langle A \rangle}$ to attain balanced representations in the following.

Theorem 1. *Let $j \in \{0, 1\}$ be the binary treatment values, and let N and T denote the number of units and observed timestamp lengths, respectively. Let $P_j^t = P(\mathbf{Z}^t \mid \mathbf{A}^t = j)$, be the distribution of latent representation $\mathbf{Z}^t$ for the group of units with treatments j at time t. Let f, ϕ, $d^j_{\mathbf{A}}$ be the initial state encoder, the ODE function of treatment-induced GraphODE, and logits of predicting treatment j. The necessary and sufficient condition for the min-max game in Eq. (5.7) to be optimal is $P_0^t = P_1^t, \forall t \in (t_0, t_1 \cdots t_T)$.*

Theorem 1 suggests that the condition to obtain global optimum of Eq. (5.7) is $P(\mathbf{Z}^t|\mathbf{A}^t = 0) = (\mathbf{Z}^t|\mathbf{A}^t = 1)$. Therefore, by optimizing $L^{\langle A \rangle}$ in Eq. (5.7), we can ensure the latent representation trajectory $\mathbf{Z}^t$ is balanced with respect to treatments. In other words, $\mathbf{Z}^t$ is not predictive of $\mathbf{A}^t$. We prove Theorem 1 in Appendix. 5.6.1.1.

Interference Balancing. The interference $\mathbf{G}_i^t$ is continuous, we thus adapt the continuous domain adversarial learning (Wang et al., 2020) to achieve the interference balancing. Similar to the binary case, we consider the continuous interference as continuous domains, and use the gradient reversal layer $r(\cdot)$ to build a min-max game on interference prediction as follows:

$$L^{\langle G \rangle} = \min_{d_{\mathbf{G}}} \max_{f,\phi} \frac{1}{N}\frac{1}{T}\sum_{i}^{N}\sum_{t}^{T}\left(d_{\mathbf{G}}\left(r([\mathbf{Z}_i^t, \mathbf{A}_i^t])\right) - \mathbf{G}_i^t\right)^2 \tag{5.8}$$

where $d_{\mathbf{G}}$ is the interference predictor which is parameterized by neural networks, and

$[\cdot,\cdot]$ is the concatenation operation. In the following, we also theoretically demonstrate that $L^{\langle G\rangle}$ is able to achieve the interference balancing objective.

Theorem 2. *Let f, ϕ, $d_{\mathbf{G}}$ be the initial state encoder, the ODE function of treatment-induced GraphODE, and the interference predictor. The necessary and sufficient condition for min-max game in Eq. (5.8) to be optimal is $P(\mathbf{Z}^t, \mathbf{A}^t|\mathbf{G}^t = g')$ is identical for any g'.*

Theorem. 2 indicates that if $\mathbb{E}([\mathbf{Z}^t, \mathbf{A}^t] \mid \mathbf{G}^t)$ is identical for any $\mathbf{G}^t = g'$, Eq. (5.8) achieves optimum. Therefore, it is sufficient to balance the combination of representations and treatments with respect to interference $\mathbf{G}^t$ by optimizing the objective function $L^{\langle G\rangle}$. We show the proof of Theorem 2 in Appendix. 5.6.1.2.

5.3.4 Training of CF-GODE

Objective Function. The overall objective function of CF-GODE is formalized in the following:

$$L = L^{\langle Y\rangle} + \alpha_{\mathbf{A}} L^{\langle A\rangle} + \alpha_{\mathbf{G}} L^{\langle G\rangle}, \tag{5.9}$$

where coefficients $\alpha_{\mathbf{A}}$, $\alpha_{\mathbf{G}}$ are the strengths of the treatment balancing and interference balancing, respectively. By adversarially optimizing L, the latent representation trajectory $\mathbf{Z}_i^t$ is able to predict the outcome trajectory $\mathbf{Y}_i^t$ while remaining invariant to the treatments $\mathbf{A}_i^t$ and interference $\mathbf{G}_i^t$ (combined with treatments), which enables the unbiased counterfactual outcome estimation in multi-agent dynamical systems.

Alternative Training as Trade-Off. In practice, we find that directly training CF-GODE with the overall loss function L may not be stable as $L^{\langle A\rangle}$ and $L^{\langle G\rangle}$ could hinder the ability of latent representation trajectory $\mathbf{Z}_i^t$ to predict the outcome. Therefore, we trade-off the training of CF-GODE in an alternative manner between L and $L^{\langle Y\rangle}$, to ensure that $\mathbf{Z}_i^t$ is capable of predicting outcomes. Specifically, we switch the training iterations between L and $L^{\langle Y\rangle}$ with a ratio of K, i.e., $\frac{Iter_L}{Iter_{L^{\langle Y\rangle}}} = K$, where $Iter$ means the number of training iterations and K is a tunable hyperparameter.

5.4 Experiments

5.4.1 Experimental Settings

Dataset. In observational data, we only have factual outcomes but not counterfactual outcomes. Therefore, we use semi-synthetics data to evaluate CF-GODE as in (Ma et al., 2022; Guo et al., 2020c; Veitch et al., 2019). That is, we use two real graphs Flickr and BlogCatalog (Guo et al., 2020c; Chu et al., 2021; Ma et al., 2021) and use a Pharmacokinetic-Pharmacodynamic (PK-PD) model (Goutelle et al., 2008) to simulate the continuous trajectory of treatments and potential outcomes (Bica et al., 2020; Seedat et al., 2022). The data simulation mimics the vaccine example in the real world. We introduce the data simulation process in detail in Appendix. 5.6.2.

Metric. We focus on counterfactual outcomes estimation in this paper, which is a continuous value. Therefore, we use mean square errors (MSE) as our metric to evaluate the performance of our model, which is formalized as $MSE := \frac{1}{N}\frac{1}{T}\sum_i^N \sum_t^T \left(\hat{\mathbf{Y}}_i^t - \mathbf{Y}_i^t\right)^2$.

Baselines. The scope of our model is in continuous-time causal inference, therefore we compare CF-GODE with the following baselines: **CDE** (Kidger et al., 2020): Ordinary differential equations with external inputs to adjust the continuous trajectory. **GraphODE** (Huang et al., 2020) Ordinary differential equations model with graph neural networks (GNNs) based ODE functions. **TE-CDE** (Seedat et al., 2022): the state-of-the-art model for continuous-time counterfactual outcomes estimation based on neural controlled differential equations (NeuralCDE).

Implementation. The parameters of CF-GODE are set as follows: the dimension of latent representations is 64; the ODE solver is the Euler method; the balancing degrees are $\alpha_{\mathbf{A}} = \alpha_{\mathbf{G}} = 0.5$. For training hyperparameters, the learning rate is 0.0001; the default alternative training ratio K is 4. We train the model 5000 epochs and select the best model according to the performance on the validation set. The parameters are optimized by Adam (Kingma and Ba, 2015a). We run all experiments on a Lambda Labs instance with one A100 GPU.

Table 5.1: Counterfactual outcomes estimation errors on two datasets. "BC" is the abbreviation of the BlogCatalog dataset. The errors are broken down in x-step future estimation ($x \in [1, 2, 3, 4, 5]$). MSE errors are reported. The best results are in boldface and the second best results are underlined. CF-GODE-N is the variant of our model without any balancing; CF-GODE-T means balance only w.r.t. treatments; CF-GODE-I denotes balance only w.r.t. interference.

Dataset	Model	1-step	2-step	3-step	4-step	5-step	Overall
Flickr	CDE	0.134±0.015	0.164±0.017	0.198±0.021	0.237±0.023	0.281±0.026	0.203±0.205
	GraphODE	0.237±0.013	0.276±0.010	0.313±0.016	0.347±0.018	0.379±0.021	0.310±0.016
	TE-CDE	0.189±0.025	0.216±0.021	0.246±0.027	0.281±0.041	0.326±0.063	0.252±0.031
	CF-GODE-N	0.089±0.006	0.102±0.007	0.114±0.009	0.126±0.010	0.139±0.012	0.114±0.008
	CF-GODE-T	0.058±0.017	0.066±0.020	0.075±0.023	0.084±0.027	0.098±0.036	0.076±0.025
	CF-GODE-I	0.069±0.007	0.080±0.008	0.091±0.009	0.103±0.011	0.115±0.012	0.092±0.009
	CF-GODE	**0.056±0.009**	**0.060±0.009**	**0.067±0.009**	**0.070±0.010**	**0.077±0.012**	**0.065±0.010**
BC	CDE	0.255±0.120	0.324±0.178	0.407±0.263	0.515±0.383	0.640±0.549	0.427±0.296
	GraphODE	0.195±0.018	0.223±0.023	0.251±0.028	0.280±0.033	0.309±0.040	0.252±0.028
	TE-CDE	0.316±0.086	0.351±0.089	0.399±0.093	0.493±0.137	0.725±0.351	0.457±0.127
	CF-GODE-N	0.167±0.012	0.188±0.015	0.209±0.018	0.228±0.023	0.246±0.028	0.207±0.019
	CF-GODE-T	**0.139±0.015**	**0.154±0.019**	**0.172±0.025**	**0.189±0.027**	**0.202±0.031**	**0.171±0.023**
	CF-GODE-I	0.164±0.016	0.188±0.020	0.210±0.024	0.232±0.029	0.253±0.035	0.209±0.025
	CF-GODE	0.148±0.015	0.166±0.019	0.186±0.023	0.205±0.025	0.229±0.029	0.186±0.021

5.4.2 Can CF-GODE Deliver Accurate Estimations of Counterfactual Outcomes in Multi-Agent Dynamical Systems?

We compare CF-GODE to three lines of models: 1) Continuous-time dynamical prediction models CDE and GraphODE. Note that these baselines are not causal models since they only preserve the dynamical statistical associations. 2) Continuous-time causal inference model TE-CDE. But it is not capable of capturing the mutual dependencies between units in multi-agent dynamical systems. 3) Variants of CF-GODE. We consider three variants: CF-GODE-N means there is no any balancing ($\alpha_{\mathbf{A}} = \alpha_{\mathbf{G}} = 0$); CF-GODE-T denotes balancing only w.r.t. treatments ($\alpha_{\mathbf{A}} = 1$, $\alpha_{\mathbf{G}} = 0$); CF-GODE-I means balancing only w.r.t. interference ($\alpha_{\mathbf{A}} = 0$, $\alpha_{\mathbf{G}} = 1$). For a multi-agent dynamical system with N nodes and length-A treatment trajectories, the total number of possible treatments for all nodes is $O(A \cdot 2^N)$. Therefore, it is intractable to enumerate all treatment combinations. To this end, we randomly flip 50% of all observed treatments in each experiment. We estimate five-step (timestamp) ahead counterfactual outcomes and report estimation errors in Table. 5.1. Generally, CF-GODE and the variants outperform the baselines by substantial

margins. It is noteworthy that, despite being a causal model, TE-CDE performs clearly worse than the family of CF-GODE, because it ignores the mutual influence between units. This underscores our motivation to address this unique challenge in multi-agent dynamical systems. We also note CF-GODE-N is generally the weakest estimator among all variants, confirming the effectiveness of our proposed balancing objectives.

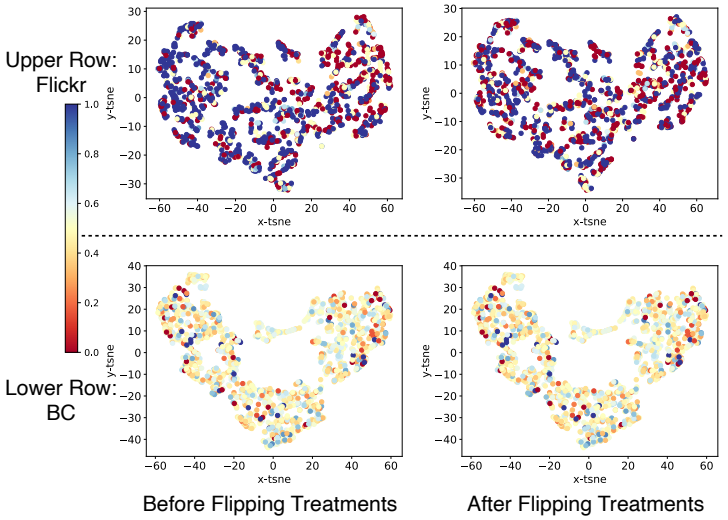

Figure 5.3: T-SNE projections of latent representations "before" (factual) and "after" (counterfactual) flipping the treatments. Each point represents a unit's latent representation. The points are colored by the units' corresponding interference. Upper row: Flickr dataset; Lower Row: BlogCatalog dataset.

Why Does CF-GODE-T Show Superior Performance Than CF-GODE on BlogCatalog Dataset? On BlogCatalog dataset, we observe that balancing solely with respect to treatments (CF-GODE-T) yields the lowest estimation errors, even outperforming balancing both treatments and interference (CF-GODE). To understand this phenomenon, we project all units' latent representations $\mathbf{Z}^t$ into 2-D embeddings using T-SNE (Van der Maaten and Hinton, 2008) and color these 2-D points by their corresponding interference in Fig. 5.3. Specifically, we compare the units' interference before and after flipping the treatments. Compared to Flickr, we notice that in BlogCatalog 1) the latent representations are already comparatively more balanced before flipping the treatments, and 2) the units' interference does not change significantly after flipping the treatments. This suggests that balancing solely with respect to treatments might be

sufficient in the BlogCatalog dataset. Actually, since we use the same data simulation protocol for Flickr and BlogCatalog, this difference in interference distribution is expected to be caused by their distinct graph structures. Specifically, the average and standard derivation of node degrees of the two datasets are Flickr: 2.0 ± 1.7; BlogCatalog: 30.7 ± 25.1. Intuitively, the interference of high degrees nodes is more resistant to flipping a random portion of their neighbors, which is pretty common among nodes in BlogCatalog. We provide further breakdown studies to better understand how node degrees affect counterfactual outcomes estimation in Sec. 5.4.5.

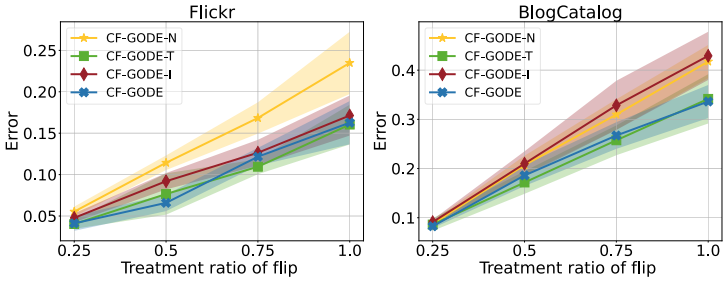

Figure 5.4: Counterfactual outcomes estimation errors w.r.t. the percentage of units in the graph whose treatments are flipped. Left: Flickr dataset; Right: BlogCatalog dataset.

5.4.3 How Does CF-GODE Respond to The Flipping of Counterfactual Treatments?

In the above experiments, the default treatment flipping ratio is set at 50%. It's intriguing to investigate how CF-GODE reacts to different flipping ratios, as this would indicate the degree of difference between factual and counterfactual outcomes in terms of treatments. To this end, we set the flip ratio as $[25\%, 50\%, 75\%, 100\%]$, and present the results of CF-GODE and its variants under these settings in Fig. 5.4. As expected, all models perform worse as the flip ratio increases, since the counterfactual treatments diverge further from the observed factual treatments. However, we observe that with balancing objectives, the error of CF-GODE increases generally slowly, highlighting the need for balancing objectives.

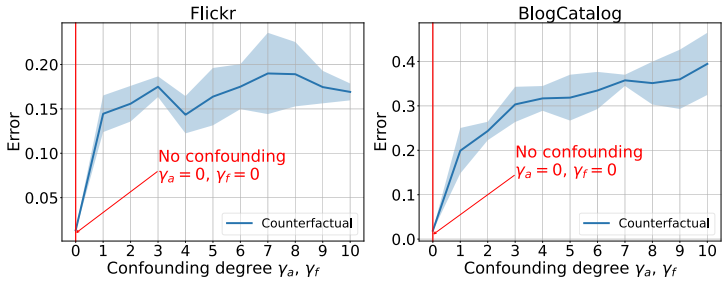

Figure 5.5: Counterfactual outcomes estimation errors w.r.t. 11 different confounding degrees γ_a and γ_f. Note γ_a and γ_f are set as the same values in each experiment. The red line points to the "no confounding bias" setting ($\gamma_a = \gamma_f = 0$).

5.4.4 How Does CF-GODE Respond to Different Confounding Degrees?

In data simulation, we use coefficients γ_a, γ_f to control the degree of the time-dependent and neighbor confounding bias. With larger values of these coefficients, the confounding bias is more severe, leading to increasingly imbalanced data. To study how CF-GODE works under varying confounding degrees, we set $\gamma_a = \gamma_f = \gamma$, where γ is from 1 to 10, and present the counterfactual outcomes estimation errors of CF-GODE under these conditions in Fig.5.5. The errors increase as the confounding bias becomes more severe, but the rate of increase is relatively smooth, particularly on Flickr dataset. This implicates CF-GODE's robustness against high degree confounding bias. Additionally, CF-GODE produces low errors when there is no confounding bias ($\gamma_a = \gamma_f = 0$), which demonstrates the compatibility of CF-GODE with such settings.

5.4.5 How Does Graph Structure Impact Counterfactual Outcomes Estimation?

As discussed in Sec. 5.4.2, the graph structure affects CF-GODE's performance on counterfactual outcomes estimation. To gain deeper insights into this phenomenon, we break down the estimation errors on BlogCatalog dataset according to node degrees in Table. 5.2. Interestingly, we find that CF-GODE's counterfactual estimation errors decrease as the node degrees become higher. Intuitively, this might also be because the interference of high-degree nodes is more stable. However, in this paper, we do not have a

theoretical understanding of the relationships between estimation errors and node degrees.

Table 5.2: The breakdown of counterfactual outcomes estimation errors by units (nodes) degrees in BlogCatalog dataset.

Degree	#Nodes	Percentage%	Error
(0,5]	176	10.2	0.243±0.337
(5,10]	185	10.6	0.235±0.344
(10,20]	389	22.5	0.216±0.296
(20,30]	312	18.0	0.218±0.314
(20,30]	200	11.5	0.221±0.295
(30,40]	165	9.5	0.195±0.284
(40,50]	98	5.7	0.176±0.269
>50	207	12.0	0.187±0.261

5.4.6 Can CF-GODE Be Generalized to New Multi-Agent Dynamical Systems?

Standard counterfactual outcome estimations are typically conducted on units whose factual outcomes have been observed. However, the estimation of the potential outcomes on *new* multi-agent dynamical systems is also of great importance. For instance, to predict the effects of an initial vaccine distribution strategy for a new community. To assess CF-GODE's ability to generalize to new systems, i.e., new graphs, we split the original graph into three subgraphs, denoted as training/validation/testing graphs (details in Appendix. 5.6.2). We train our model on the training graph and evaluate its potential outcome estimation on the testing graph. We report the results in Table. 5.3. We note that the performance of CF-GODE and the variants on new graphs are also generally better than baselines, which is consistent with the estimation of the counterfactual outcomes within the same graph (Sec. 5.4.2). This demonstrates our model's generalizability to new multi-agent dynamical systems.

5.4.7 How Does Alternative Training Affect CF-GODE?

CF-GODE uses an alternative training strategy to trade off the latent representation balancing and potential outcome prediction. We examine how this trade-off is performed under varying alternative ratios K, where $K = \frac{Iter_L}{Iter_{L\langle Y\rangle}}$ represents the alternating training

Table 5.3: Generalization errors of potential outcomes prediction for new multi-agent dynamical systems (new graphs) on two datasets. "BC" is the abbreviation of the BlogCatalog dataset. The errors are broken down in x-step future estimation ($x \in [1, 2, 3, 4, 5]$). MSE errors are reported. The best results are in boldface and the second best results are underlined. CF-GODE-N is the variant of our model without any balancing; CF-GODE-T means balance only w.r.t. treatments; CF-GODE-I denotes balance only w.r.t. interference.

Dataset	Model	1-step	2-step	3-step	4-step	5-step	Overall
Flickr	CDE	0.134±0.017	0.166±0.022	0.201±0.026	0.241±030	0.285±0.034	0.205±0.025
	GraphODE	0.209±0.010	0.243±0.012	0.275±0.015	0.306±0.018	0.335±0.022	0.274±0.014
	TE-CDE	0.193±0.023	0.221±0.021	0.251±0.027	0.285±0.040	0.328±0.061	0.256±0.030
	CF-GODE-N	0.087±0.06	0.099±0.008	0.111±0.009	0.122±0.010	0.134±0.011	0.111±0.008
	CF-GODE-T	**0.057±0.014**	0.064±0.016	0.072±0.018	0.081±0.022	0.092±0.029	0.738±0.020
	CF-GODE-I	0.071±0.007	0.083±0.008	0.096±0.009	0.109±0.010	0.122±0.011	0.096±0.009
	CF-GODE	**0.057±0.008**	**0.062±0.009**	**0.067±0.010**	**0.073±0.011**	**0.081±0.013**	**0.069±0.010**
BC	CDE	0.251±0.113	0.317±0.174	0.399±0.259	0.500±0.380	0.625±0.548	0.418±0.294
	GraphODE	0.183±0.021	0.210±0.026	0.237±0.032	0.265±0.039	0.293±0.046	0.237±0.033
	TE-CDE	0.328±0.092	0.372±0.104	0.440±0.164	0.582±0.378	0.933±0.966	0.457±0.127
	CF-GODE-N	0.168±0.008	0.190±0.009	0.213±0.012	0.235±0.015	0.255±0.021	0.213±0.013
	CF-GODE-T	**0.141±0.011**	**0.157±0.015**	**0.175±0.021**	**0.192±0.022**	**0.203±0.028**	**0.174±0.018**
	CF-GODE-I	0.164±0.014	0.187±0.018	0.209±0.022	0.231±0.028	0.253±0.034	0.209±0.023
	CF-GODE	0.143±0.013	0.162±0.017	0.180±0.022	0.199±0.023	0.214±0.025	0.180±0.019

between the overall loss L and the outcome prediction loss $L^{\langle Y \rangle}$. Fig. 5.6 shows the 2-D T-SNE projections of latent representations and their corresponding counterfactual estimation errors for different K values. We note that compared to solely training on $L^{\langle Y \rangle}$ (i.e., no balancing), training with L is able to force the embeddings more balanced, suggesting the effectiveness of our proposed domain adversarial learning based balancing objectives. In addition, with a smaller K, CF-GODE achieves better estimation errors, while a bigger K leads to more balanced latent representations. These results confirm that our alternative training is able to trade off between latent representation balancing and potential outcome prediction. In practice, choosing an appropriate value of K is expected to be determined through empirical analysis for each dataset.

5.4.8 Case Study: When CF-GODE Is Good, and When It Is Not.

To intuitively understand how CF-GODE works in estimating counterfactual outcomes and to study when CF-GODE would fail, we sample one successful unit and one failure unit from Flickr dataset. We draw their factual outcomes, counterfactual outcomes, and

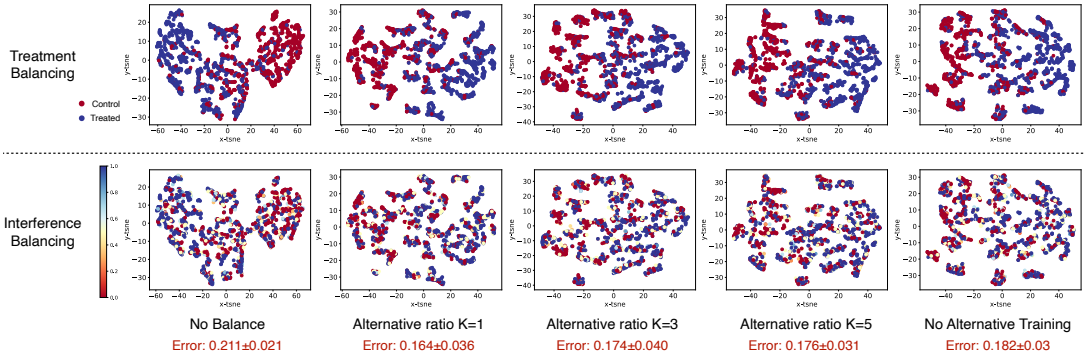

Figure 5.6: The T-SNE visualization of latent representations under different alternative training settings. Each point represents a unit's latent representation. The top line is colored by the observed treatments and the bottom line is colored by observed interference. Five columns from left to right: 1) only trained on $L^{\langle Y \rangle}$ (no balancing); 2) $\frac{Iter_L}{Iter_{L^{\langle Y \rangle}}} = 1$; 3) $\frac{Iter_L}{Iter_{L^{\langle Y \rangle}}} = 3$; 4) $\frac{Iter_L}{Iter_{L^{\langle Y \rangle}}} = 5$; 5) only trained on L (no alternative training). Each setting's corresponding counterfactual estimation error is marked in red.

the estimations made by CF-GODE and CF-GODE-N (without balancing) in Fig. 5.7. In the successful case, the estimate by CF-GODE is able to conform to the counterfactual treatment trajectory, while CF-GODE-N still follows the factual trajectory. This shows the effectiveness of our proposed balancing objectives. However, CF-GODE also makes mistakes. In the failure case, its estimate fails to catch up with the counterfactual outcome trajectory, yielding a non-trivial error. We speculate that this is because the counterfactual outcome of this unit is quite distinct from the factual one in terms of data scale, making counterfactual estimation more difficult.

5.4.9 How Hyperparamters Affect CF-GODE?

The two balancing objectives are core to making CF-GODE causal. Therefore, we finally study the impact of their degrees, represented by $\alpha_{\mathbf{A}}$ and $\alpha_{\mathbf{G}}$ in the loss function, on the model performance. We test $\alpha_{\mathbf{A}}$ and $\alpha_{\mathbf{G}}$ values evenly ranging from $[0, 0, 1.0]$, and present the counterfactual outcomes estimation errors for each combination of $\alpha_{\mathbf{A}}$ and $\alpha_{\mathbf{G}}$ in Fig. 5.8. Our results show that the errors are relatively higher when both $\alpha_{\mathbf{A}}$ and $\alpha_{\mathbf{G}}$ are in low values, i.e., light balancing. On the other hand, with larger values, the estimation errors generally become lower, but with high variance. This indicates the effectiveness of

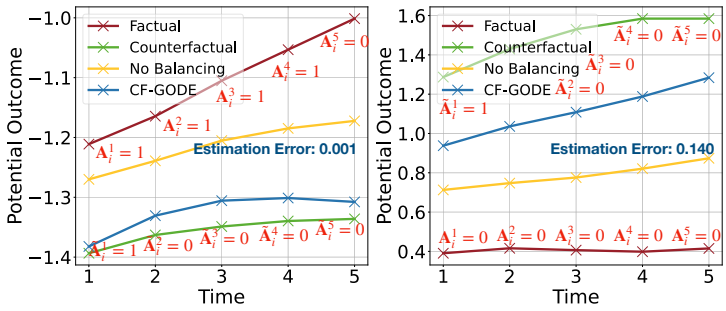

Figure 5.7: The predicted counterfactual outcomes of CF-GODE w/w.o balancing loss functions. The treatments of factual outcomes $\mathbf{A}_i^t$ and treatments of counterfactual outcomes $\tilde{\mathbf{A}}_i^t$ are attached around the corresponding curves at each timestamp. The estimation errors are noted in blue. Results are from Flickr dataset. Left: a successful case; right: a bad case.

the balancing objectives but also highlights the instability of domain adversarial learning based balancing.

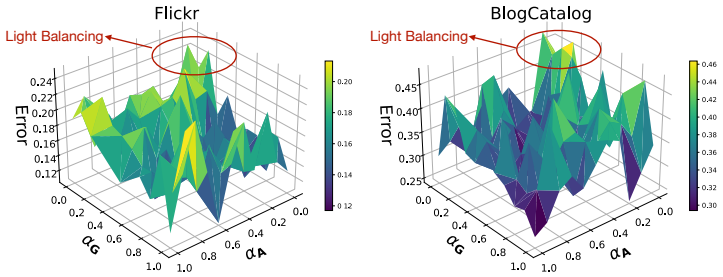

Figure 5.8: The counterfactual estimation errors w.r.t. different combinations of $\alpha_{\mathbf{A}}$ and $\alpha_{\mathbf{G}}$. The "Light Balancing" settings where $\alpha_{\mathbf{A}}$ and $\alpha_{\mathbf{G}}$ are both in small values are circles in red.

5.5 Conclusion

In this paper, we study continuous-time counterfactual outcomes estimation in multi-agent dynamical systems, where units interact with each other. To this end, we propose CF-GODE, a novel causal model based on GraphODE to enable continuous potential outcomes prediction, and domain adversarial learning to remove confounding bias. We provide both theoretical justification and empirical analyses to demonstrate the effectiveness of our model. One limitation of CF-GODE is it needs the assumption of strong ignorability for multi-agent dynamical systems, which is not testable in practice. Recent

studies relax this assumption by inferring latent proxy variables (Wang and Blei, 2019), which could be a potential solution.

5.6 Appendix

5.6.1 Proofs of Theorems

5.6.1.1 Proof of Theorem. 1

Theorem 1. *Let $j \in \{0, 1\}$ be the binary treatment values, and let N and T denote the number of units and observed timestamp lengths, respectively. Let $P_j^t = P(\mathbf{Z}^t \mid \mathbf{A}^t = j)$, be the distribution of latent representation $\mathbf{Z}^t$ for the group of units with treatments j at time t. Let f, ϕ, $d_{\mathbf{A}}^j$ be the initial state encoder, the ODE function of treatment-induced GraphODE, and logits of predicting treatment j. The necessary and sufficient condition for the min-max game in Eq. (5.7) to be optimal is $P_0^t = P_1^t, \forall t \in (t_0, t_1 \cdots t_T)$.*

The proof of Theorem 1 follows (Bica et al., 2020; Melnychuk et al., 2022) and consists of two steps: to find the optimal $d_{\mathbf{A}}^j$ while fixing f and ϕ, and then to prove the optimal f and ϕ while fixing $d_{\mathbf{A}}^j$ can balance the latent representations, i.e., $P_0^t = P_1^t$. The first step is given by Proposition 1.

Proposition 1. *(Proposition 1 in (Melnychuk et al., 2022)) Let $\alpha_j = P(A^t = j)$. When the initial state encoder f and ODE function ϕ are fixed, the optimal $d_{\mathbf{A}}^j$ at time t is:*

$$d_{\mathbf{A}}^j{}^* = \frac{\alpha_j P_j^t}{\sum_{j' \in \{0,1\}} \alpha_{j'} P_{j'}^t}. \tag{5.10}$$

Proof. When fixing f and ϕ, $d_{\mathbf{A}}^j{}^*$ is obtained by:

$$d_{\mathbf{A}}^j{}^* = \underset{d_{\mathbf{A}}^j}{\operatorname{argmin}} \sum_{j \in \{0,1\}} \mathbb{1}_{(\mathbf{A}^t = j)} - \log\left(d_{\mathbf{A}}^j\left(r(\mathbf{Z}^t)\right)\right), \tag{5.11}$$

$$\text{subject to} \sum_{j \in \{0,1\}} d_{\mathbf{A}}^j\left(r(\mathbf{Z}^t)\right) = 1, \tag{5.12}$$

where Eq. (5.11) is adapted from Eq. (5.7). Note Eq. (5.11) can be applied to any i and t in Eq. (5.7), so we disregard the expectation with respect to them. Let $\alpha_j = P(A^t = j)$ and $P_j^t = P(\mathbf{Z}^t \mid \mathbf{A}^t = j)$. Then Eq. (5.11) can be rewritten as:

$$d_{\mathbf{A}}^{j}{}^{*} = \underset{d_{\mathbf{A}}^{j}}{\operatorname{argmin}} \sum_{j\in\{0,1\}} -\mathbb{E}_{\mathbf{Z}^t\sim P_j^t} \alpha_j \log\left(d_{\mathbf{A}}^{j}\left(r(\mathbf{Z}^t)\right)\right) \tag{5.13}$$

$$= \underset{d_{\mathbf{A}}^{j}}{\operatorname{argmin}} \sum_{j\in\{0,1\}} -\int_{\mathbf{Z}'^t} \alpha_j \log\left(d_{\mathbf{A}}^{j}\left(r(\mathbf{Z}'^t)\right)\right) P_j^t d\mathbf{Z}'^t. \tag{5.14}$$

We can also take pointwise optimization for any $\mathbf{Z}'^t$ in Eq. (5.14) Then by combining Equation (5.14) with the constraint in Equation (5.12) and using Lagrange multipliers, we have:

$$d_{\mathbf{A}}^{j}{}^{*} = \underset{d_{\mathbf{A}}^{j}}{\operatorname{argmin}} \sum_{j\in\{0,1\}} -\alpha_j \log\left(d_{\mathbf{A}}^{j}\left(r(\mathbf{Z}'^t)\right)\right) P_j^t + \lambda\left(\sum_{j\in\{0,1\}} d_{\mathbf{A}}^{j}\left(r(\mathbf{Z}^t)\right) - 1\right). \tag{5.15}$$

Let $J = \sum_{j\in\{0,1\}} -\alpha_j \log(d_{\mathbf{A}}^{j}(r(\mathbf{Z}'^t)))P_j^t + \lambda(\sum_{j\in\{0,1\}} d_{\mathbf{A}}^{j}(r(\mathbf{Z}^t)) - 1)$ be the objective in Eq. (5.15) The optimal values can be obtained by taking partial gradients $\frac{\partial J}{\partial d_{\mathbf{A}}^{j}} = 0$ and $\frac{\partial J}{\lambda} = 0$, respectively. By computing them jointly we can obtain $d_{\mathbf{A}}^{j}{}^{*} = \frac{\alpha_j P_j^t}{\sum_{j'\in\{0,1\}} \alpha_{j'} P_{j'}^t}$. □

The second step is to prove Theorem. 1 that the optimal f and ϕ can obtain balanced representations with respect to treatments.

Proof. With Proposition 1, we can fix the optimal $d_{\mathbf{A}}^{j}{}^{*}$ and find the condition where Eq. (5.7) achieves optimum. Putting $d_{\mathbf{A}}^{j}{}^{*}$ into the objective in Eq. (5.7) and applying

similar simplifications as in Eq. (5.14) and Eq. (5.15), we have:

$$f^*, \phi^* = \underset{f,\phi}{\text{argmax}} \sum_{j\in\{0,1\}} -\mathbb{E}_{\mathbf{Z}^t\sim P_j^t} \log\left(\frac{\alpha_j P_j^t}{\sum_{j'\in\{0,1\}} \alpha_{j'} P_{j'}^t}\right) \tag{5.16}$$

$$= \underset{f,\phi}{\text{argmin}} \sum_{j\in\{0,1\}} \mathbb{E}_{\mathbf{Z}^t\sim P_j^t} \log\left(\frac{P_j^t}{\sum_{j'\in\{0,1\}} \alpha_{j'} P_{j'}^t}\right) + \log(\alpha_j) \tag{5.17}$$

$$= \underset{f,\phi}{\text{argmin}} \sum_{j\in\{0,1\}} \text{KL}\left(P_j^t \middle|\middle| \sum_{j'\in\{0,1\}} \alpha_{j'} P_{j'}^t\right) + \log(\alpha_j), \tag{5.18}$$

$$\tag{5.19}$$

where $\text{KL}(\cdot||\cdot)$ is the KL divergence. Note that $\sum_{j\in\{0,1\}} \log(\alpha_j)$ is constant in observation data. $\text{KL}(\cdot||\cdot) \geq 0$ and it reaches 0 when the two operands are equal. Therefore, to have f^*, ϕ^*, for $j \in \{0, 1\}$, we have $P_0^t = P_1^t = \sum_{j'\in\{0,1\}} \alpha_{j'} P_{j'}^t$. Therefore, the optimal f^*, ϕ^* are those who achieve $P_0^t = P_1^t$. We can apply this to all the N units and all the T observed timestamps to obtain the global optimum of Eq. (5.7), which concludes the proof of Theorem 1. □

5.6.1.2 Proof of Theorem 2

Theorem 2. *Let f, ϕ, $d_{\mathbf{G}}$ be the initial state encoder, the ODE function of treatment-induced GraphODE, and the interference predictor. The necessary and sufficient condition for min-max game in Eq. (5.8) to be optimal is $P(\mathbf{Z}^t, \mathbf{A}^t|\mathbf{G}^t = g')$ is identical for any g'.*

The proof of Theorem 2 follows (Wang et al., 2020). Similar to Theorem 1's proof, it first finds the optimum of $d_{\mathbf{G}}$, and then proves that the optimal f and ϕ can balance the representations with respect to interference. We first restate the Lemma 4.1 in (Wang et al., 2020) in Proposition. 2.

Proposition 2. *(Lemma 4.1 in (Wang et al., 2020)) Let $\mathbf{C}^t = [\mathbf{Z}^t, \mathbf{A}^t]$ be the concatenation of $\mathbf{Z}^t$ and $\mathbf{A}^t$. When fixing initial state encoder f and ODE function ϕ, the optimal $d_{\mathbf{G}}$ at time t is:*

$$d_{\mathbf{G}}^* = \mathbb{E}_{\mathbf{G}^t\sim p(\mathbf{G}^t|\mathbf{C}^t)}(\mathbf{G}^t). \tag{5.20}$$

Proof. Eq. (5.20) is adapted from Eq. (5.8). We disregard the expectation with respect to i and t since Eq. (5.20) is applicable to any i and t. If fixing f and ϕ, the optimal $d^*_{\mathbf{G}}$ is given by:

$$d^*_{\mathbf{G}} = \underset{d_{\mathbf{G}}}{\operatorname{argmin}}\ \mathbb{E}_{(\mathbf{Z}^t,\mathbf{A}^t,\mathbf{G}^t)\sim p(\mathbf{Z}^t,\mathbf{A}^t,\mathbf{G}^t)} \left(d_{\mathbf{G}}\left(r([\mathbf{Z}^t,\mathbf{A}^t])\right) - \mathbf{G}^t\right)^2 \tag{5.21}$$

$$= \underset{d_{\mathbf{G}}}{\operatorname{argmin}}\ \mathbb{E}_{(\mathbf{C}^t,\mathbf{G}^t)\sim p(\mathbf{C}^t,\mathbf{G}^t)} \left(d_{\mathbf{G}}\left(r(\mathbf{C}^t)\right) - \mathbf{G}^t\right)^2 \tag{5.22}$$

$$= \underset{d_{\mathbf{G}}}{\operatorname{argmin}}\ \mathbb{E}_{\mathbf{C}^t\sim p(\mathbf{C}^t)}\mathbb{E}_{\mathbf{G}^t\sim p(\mathbf{G}^t|\mathbf{C}^t)} \left(d_{\mathbf{G}}\left(r(\mathbf{C}^t)\right) - \mathbf{G}^t\right)^2. \tag{5.23}$$

As the quadratic expansion in (Wang et al., 2020), the optimal interference predictor is $d^*_{\mathbf{G}} = \mathbb{E}_{\mathbf{G}^t\sim p(\mathbf{G}^t|\mathbf{C}^t)}(\mathbf{G}^t)$. □

Here we introduce and reformulate Theorem 4.1 in (Wang et al., 2020) as Lemma. 1.

Lemma 1. *(Theorem 4.1 in (Wang et al., 2020)) Given $\mathbb{E}_x\mathbb{V}(y \mid x)$ where $\mathbb{V}$ denotes variance, its global optimum can be achieved if and only if for any x, $\mathbb{E}(y \mid x) = \mathbb{E}(y)$.*

With Proposition. 2 and Lemma. 1, we can prove Theorem. 2.

Proof. Fixing the optimal $d^*_{\mathbf{G}}$ in Proposition. 2, the optimal f and ϕ for objective Eq. (5.23) is:

$$f^*, \phi^* = \underset{f,\phi}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{C}^t\sim p(\mathbf{C}^t)}\mathbb{E}_{\mathbf{G}^t\sim p(\mathbf{G}^t|\mathbf{C}^t)} \left(\mathbb{E}_{\mathbf{G}^t\sim p(\mathbf{G}^t|\mathbf{C}^t)}(\mathbf{G}^t) - \mathbf{G}^t\right)^2 \tag{5.24}$$

$$= \underset{f,\phi}{\operatorname{argmax}}\ \mathbb{E}_{\mathbf{C}^t\sim p(\mathbf{C}^t)}\mathbb{V}\left(\mathbf{G}^t \mid \mathbf{C}^t\right). \tag{5.25}$$

Eq. (5.25) has the same form as the equation in Lemma. 1. Then substituting $x = \mathbf{C}^t$ and $y = \mathbf{G}^t$ in Lemma. 1, we have $\mathbb{E}(\mathbf{G}^t \mid \mathbf{C}^t) = \mathbb{E}(\mathbf{G}^t)$. In other words, $\mathbf{G}^t \perp\!\!\!\perp \mathbf{C}^t$. Therefore, we can also use the inverse form that $\mathbb{E}(\mathbf{C}^t) = \mathbb{E}(\mathbf{C}^t \mid \mathbf{G}^t) = \mathbb{E}([\mathbf{Z}^t,\mathbf{A}^t] \mid \mathbf{G}^t)$ for any $\mathbf{G}^t$, which concludes the proof of Theorem. 2. □

5.6.2 Experimental Settings

Datasets. In observational data, we only have outcomes under one treatment trajectory but not the ground-truths of counterfactual outcomes. Therefore, we follow (Ma et al., 2022; Guo et al., 2020c; Veitch et al., 2019) to use semi-synthetic data to evaluate CF-GODE. That is, the graph structure and node features are real, but treatments and potential outcomes are simulated. We use the social networks Flickr and BlogCatalog as in (Guo et al., 2020c; Chu et al., 2021; Ma et al., 2021) as the graph $\mathcal{G}$. We follow these works to first encode the node features into low-dimensional embeddings (10-dimensional in this paper) via LDA (Blei et al., 2003). We then follow (Jiang and Sun, 2022) to use Metis (Karypis and Kumar, 1998) to split the graph into training/validation/testing sets. To simulate treatment and potential outcomes over time, (Geng et al., 2017; Bica et al., 2020; Seedat et al., 2022) use a longitudinal simulation environment, which, however, assumes the units are mutually independent. We extend it into our multi-agent dynamical systems setting by considering the neighbor confounders and interference. During the simulation, we are motivated by the vaccine's use case. Specifically, treatment $\mathbf{A}_i^t$ denotes getting a vaccine or not at time t of unit i. The trajectory of $\mathbf{A}_i^t$ denotes the vaccine records over time, e.g., a unit may have a booster dose after the initial vaccine. In this case, the time-dependent covariates $\mathbf{X}^t$ could be the health condition, static covariates $\mathbf{V}$ could be race or educational background (assuming it does not change during the study) and potential outcome $\mathbf{Y}^t$ could be immunity to the virus. As discussed in Sec. 5.2.1, the potential outcome $\mathbf{Y}^t$ is essentially a part of $\mathbf{X}^t$. This is a common setting in longitudinal causal inference studies (Bica et al., 2020; Seedat et al., 2022; Melnychuk et al., 2022). During the simulation, we follow this protocol: the health condition $\mathbf{X}_i^t$ has a value range $[0.1, 10]$, in which a higher value means a better health condition. Meanwhile, a higher health condition means a lower probability to receive a vaccine (treatment).

Treatment simulation. The treatment $\mathbf{A}_i^t$ is affected by a unit's own time-dependent covariates $\mathbf{X}_i^t$, static covariates $\mathbf{V}_i$ and those of their neighbors. Let $\mathbf{E}_i = w_a \mathbf{V}_i$ denote the effects of static confounders on treatments, where w_a is a generated parameter representing

this mechanism. The treatment is then simulated by Bernoulli generator with probability the $p_i^t(a)$ of unit i at time t:

$$p_i^t(a) = \sigma\Bigg(\underbrace{\gamma_a(\delta_a - \bar{\mathbf{X}}_i^t)}_{\text{time-dependent covariates}} + \underbrace{\gamma_n\Bigg(\delta_n - (\frac{1}{|N_i|}\sum_{j\in\mathcal{N}_i}\bar{\mathbf{X}}_j^t)\Bigg)}_{\text{Neighbor time-dependent covariates}} + \underbrace{\gamma_f \mathbf{E}i}_{\text{Static covariates}} + \underbrace{\gamma_g(\frac{1}{|N_i|}\sum_{j\in\mathcal{N}_i}\mathbf{E}_j)}_{\text{Neighbor static covariates}} \Bigg), \tag{5.26}$$

where $\sigma(\cdot)$ is sigmoid function. γ_a, γ_n, γ_f and γ_g are degrees of time-dependent confounders, neighbor time-dependent confounders, static confounders and neighbor static confounders, respectively. The default values are $[\gamma_a, \gamma_n, \gamma_f, \gamma_g] = [10, 3.3, 10, 3.3]$ ($\frac{\gamma_a}{\gamma_n} = \frac{\gamma_f}{\gamma_g} = 3$), to mimic that a unit's own confounding factors should affect it more than neighbors. Note $\bar{\mathbf{X}}_i^t$ is the average time-dependent covariates until t. This reflects that past time-dependent covariates also affect the treatment. We set $\delta_a = \delta_n = 5$ as adjustments.

Potential outcome simulation. We follow (Geng et al., 2017; Bica et al., 2020; Seedat et al., 2022) to use a Pharmacokinetic-Pharmacodynamic (PK-PD) model (Goutelle et al., 2008) to simulate the continuous trajectory. PK-PD model is a popular bio-mathematical model and a natural fit for our vaccine use case. As mentioned above, $\mathbf{Y}^t$ is essentially a part of $\mathbf{X}^t$. Therefore we directly simulate the trajectory of $\mathbf{X}_i^t$:

$$\frac{d\mathbf{X}_i^t}{dt} = \mathbf{X}_i^t\Bigg(\underbrace{\rho_u \log\left(\frac{K}{\mathbf{X}_i^t}\right)}_{\text{Time-dependt covariates}} + \underbrace{\rho_n \log\left(\frac{K}{\mathbf{X}_i^t}\right)}_{\text{Neighbor time-dependt covariates}} + \underbrace{\rho_f \mathbf{O}_i}_{\text{Static covariates}} + \underbrace{\rho_g \sum_{j\in\mathcal{N}_i}\mathbf{O}_j)}_{\text{Neighbor static covariates}} + \underbrace{\beta_a \mathbf{D}_i^t}_{\text{Treatment}} + \underbrace{\frac{1}{|N_i|}\sum_{j\in\mathcal{N}_i}\beta_n \mathbf{D}_j^t}_{\text{Interference}} + \underbrace{e_i^t}_{\text{Noise}} \Bigg), \tag{5.27}$$

where K controls the effects of time-dependent covariates on future potential outcomes. $\mathbf{O}_i = w_x \mathbf{V}_i$ denote the effects of static confounders on potential outcomes, where w_x represents this mechanism. ρ_u, ρ_n, ρ_f and ρ_g are degrees of time-dependent covariates, neighbor time-dependent covariates, static covariates and neighbor static covariates, respectively. Their default values are $[\rho_u, \rho_n, \rho_f, \rho_g] = [-0.001, -00033, 0.001, 0.00033]$ ($\frac{\rho_u}{\rho_n} = \frac{\rho_f}{\rho_g} = 3$). β_a and β_n control the strengths of treatment and interference. We set them as $[\beta_a, \beta_n] = [0.03, 0.01]$. The values also reflect that a unit's own covariates and treatment should have stronger effects on its future potential outcomes than neighbors. In reality, the effects of vaccines on providing protection decrease over time. To mimic this phenomenon, we use the decay function $\mathbf{D}_i^t = \tilde{\mathbf{D}}_i^t + \mathbf{D}_i^{(t-1)}/2$ to model the effects of treatments over time as in (Bica et al., 2020; Seedat et al., 2022), where $\mathbf{D}_i^t$ denote the protection effect at time t, and $\tilde{\mathbf{D}}_i^t$ means a full protection of vaccines given at t. In other words, the unit receives a vaccine at time t, i.e., $\mathbf{A}_i^t = 1$. We set $\tilde{\mathbf{D}}_i^t = 1$.

CHAPTER 6

Conclusion

This dissertation studies dynamical systems modeling. First, mechanistic factors, including physical laws and general periodic series, are demonstrated useful in improving the accuracy of dynamical prediction. Second, to enable causal reasoning in dynamical systems, this dissertation studies causal effects estimation on static graphs and then extends it to dynamical systems.

Graphs are abstract models for dynamical systems. Building world models that understand dynamics and can predict the next state would be precise and comprehensive for modeling dynamics (LeCun, 2022). Meanwhile, video data encodes rich knowledge about how the world evolves (Bardes et al., 2024). Therefore, future work could explore learning from videos to build world models.

Abrishami, A. and Aliakbary, S. (2019). Predicting citation counts based on deep neural network learning techniques. *Journal of Informetrics*, 13(2):485–499. 4

Alaa, A. M. and van der Schaar, M. (2019). Attentive state-space modeling of disease progression. *Advances in neural information processing systems*, 32. 28

Anderson, R. M. and May, R. M. (1985). Vaccination and herd immunity to infectious diseases. *Nature*, 318(6044):323–329. 51

Arbour, D., Garant, D., and Jensen, D. (2016). Inferring network effects from observational data. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 715–724. 48, 50, 51, 52, 53, 54, 75, 79

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. 38

Bandara, K., Shi, P., Bergmeir, C., Hewamalage, H., Tran, Q., and Seaman, B. (2019). Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *International conference on neural information processing*, pages 462–474. Springer. 26

Bardes, A., Garrido, Q., Ponce, J., Chen, X., Rabbat, M., LeCun, Y., Assran, M., and Ballas, N. (2024). Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*. 102

Barkley, B. G., Hudgens, M. G., Clemens, J. D., Ali, M., and Emch, M. E. (2020). Causal inference from observational studies with clustered interference, with application to a cholera vaccine study. *The Annals of Applied Statistics*, 14(3):1432–1448. 47

Bica, I., Alaa, A. M., Jordon, J., and van der Schaar, M. (2020). 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*. 73, 75, 77, 78, 80, 83, 86, 95, 99, 100, 101

Blei, D. M., Ng, A. Y., and Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022. 67, 99

Caballero Barajas, K. L. and Akella, R. (2015). Dynamically modeling patient's health state from electronic medical records: A time series approach. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 69–78. 26

Cao, Q., Shen, H., Cen, K., Ouyang, W., and Cheng, X. (2017). Deephawkes: Bridging the gap between prediction and understanding of information cascades. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1149–1158. 16, 19

Castillo, C., Donato, D., and Gionis, A. (2007). Estimating number of citations using author reputation. In *International Symposium on String Processing and Information Retrieval*, pages 107–117. Springer. 18

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. 18

Chen, T. and Sun, Y. (2017). Task-guided and path-augmented heterogeneous network embedding for author identification. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 295–304. ACM. 8, 12, 14

Chu, Z., Rathbun, S. L., and Li, S. (2021). Graph infomax adversarial learning for treatment effect estimation with networked observational data. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 176–184. 48, 86, 99

Chung, J., Gulcehre, C., Cho, K., and Bengio, Y. (2014a). Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*. 1

Chung, J., Gülçehre, Ç., Cho, K., and Bengio, Y. (2014b). Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555. 15, 19

Cui, H., Dai, W., Zhu, Y., Kan, X., Gu, A. A. C., Lukemire, J., Zhan, L., He, L., Guo, Y., and Yang, C. (2022). Braingb: a benchmark for brain network analysis with graph neural networks. *IEEE Transactions on Medical Imaging*. 74

Dong, E., Du, H., and Gardner, L. (2020). An interactive web-based dashboard to track covid-19 in real time. *The Lancet infectious diseases*, 20(5):533–534. 44

Dong, Y., Chawla, N. V., and Swami, A. (2017). metapath2vec: Scalable representation learning for heterogeneous networks. In *SIGKDD*. ACM. 12

Dong, Y., Johnson, R. A., and Chawla, N. V. (2015). Will this paper increase your h-index?: Scientific impact prediction. In *Proceedings of the eighth ACM international conference on web search and data mining*, pages 149–158. ACM. 11

Dong, Y., Johnson, R. A., and Chawla, N. V. (2016). Can scientific impact be predicted? *IEEE Transactions on Big Data*, 2(1):18–30. 6

Du, L., Wang, Y., Song, G., Lu, Z., and Wang, J. (2018). Dynamic network embedding: An extended approach for skip-gram based network embedding. In *IJCAI*, pages 2086–2092. 16

Durrant, J. D. and McCammon, J. A. (2011). Molecular dynamics simulations and drug discovery. *BMC biology*, 9(1):1–9. 74

Evans, J. A. and Reimer, J. (2009). Open access and global participation in science. *Science*, 323(5917):1025–1025. 4

Forastiere, L., Airoldi, E. M., and Mealli, F. (2021). Identification and estimation of treatment and interference effects in observational studies on networks. *Journal of the American Statistical Association*, 116(534):901–918. 48, 52, 54, 57, 60, 75, 78, 79, 80, 83

Fujii, K., Takeuchi, K., Kuribayashi, A., Takeishi, N., Kawahara, Y., and Takeda, K. (2022). Estimating counterfactual treatment outcomes over time in multi-vehicle simulation. In *Proceedings of the 30th International Conference on Advances in Geographic Information Systems*, pages 1–4. 73

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *The journal of machine learning research*, 17(1):2096–2030. 76, 83

Gasthaus, J., Benidis, K., Wang, Y., Rangapuram, S. S., Salinas, D., Flunkert, V., and Januschowski, T. (2019). Probabilistic forecasting with spline quantile function rnns. In *The 22nd international conference on artificial intelligence and statistics*, pages 1901–1910. PMLR. 30

Gazi, V. and Fidan, B. (2007). Coordination and control of multi-agent dynamic systems: Models and approaches. In *Swarm Robotics: Second International Workshop, SAB 2006, Rome, Italy, September 30-October 1, 2006, Revised Selected Papers 2*, pages 71–102. Springer. 74

Geng, C., Paganetti, H., and Grassberger, C. (2017). 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. 99, 100

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. *arXiv preprint arXiv:1406.2661*. 62

Goutelle, S., Maurin, M., Rougier, F., Barbaut, X., Bourguignon, L., Ducher, M., and Maire, P. (2008). The hill equation: a review of its capabilities in pharmacological modelling. *Fundamental & clinical pharmacology*, 22(6):633–648. 86, 100

Greenland, S., Pearl, J., and Robins, J. M. (1999). Causal diagrams for epidemiologic research. *Epidemiology*, pages 37–48. 47

Gui, H., Xu, Y., Bhasin, A., and Han, J. (2015). Network a/b testing: From sampling to estimation. In *Proceedings of the 24th International Conference on World Wide Web*, pages 399–409. 47

Gui, H., Zhu, Q., Liu, L., Zhang, A., and Han, J. (2018). Expert finding in heterogeneous bibliographic networks with locally-trained embeddings. *arXiv preprint arXiv:1803.03370*. 12

Guo, R., Li, J., Li, Y., Candan, K. S., Raglin, A., and Liu, H. (2020a). Ignite: A minimax game toward learning individual treatment effects from networked observational data. In *IJCAI*, pages 4534–4540. 48

Guo, R., Li, J., and Liu, H. (2020b). Counterfactual evaluation of treatment assignment functions with networked observational data. In *Proceedings of the 2020 SIAM International Conference on Data Mining*, pages 271–279. SIAM. 51, 67

Guo, R., Li, J., and Liu, H. (2020c). Learning individual causal effects from networked observational data. In *Proceedings of the 13th International Conference on Web Search and Data Mining*, pages 232–240. 48, 67, 69, 86, 99

Halloran, M. E. and Hudgens, M. G. (2012). Causal inference for vaccine effects on infectiousness. *The international journal of biostatistics*. 74

Hamilton, W., Ying, Z., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pages 1024–1034. 6

Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 1025–1035. 48

Hill, J. L. (2011). Bayesian nonparametric modeling for causal inference. *Journal of Computational and Graphical Statistics*, 20(1):217–240. 58

Hirano, K. and Imbens, G. W. (2004). The propensity score with continuous treatments. *Applied Bayesian modeling and causal inference from incomplete-data perspectives*, 226164:73–84. 61

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780. 1

Huang, Z., Sun, Y., and Wang, W. (2020). Learning continuous system dynamics from irregularly-sampled partial observations. *Advances in Neural Information Processing Systems*, 33:16177–16187. 76, 80, 81, 86

Huang, Z., Sun, Y., and Wang, W. (2021). Coupled graph ode for learning interacting system dynamics. In *KDD*, pages 705–715. 80, 81

Hudgens, M. G. and Halloran, M. E. (2008). Toward causal inference with interference. *Journal of the American Statistical Association*, 103(482):832–842. 47

Hyndman, R. J. and Athanasopoulos, G. (2018). *Forecasting: principles and practice*. OTexts. 27, 30, 33, 35, 36, 38

Ismail, A. A., Gunady, M., Corrada Bravo, H., and Feizi, S. (2020). Benchmarking deep learning interpretability in time series predictions. *Advances in neural information processing systems*, 33:6441–6452. 28

Jiang, S., Huang, Z., Luo, X., and Sun, Y. (2023). CF-GODE: continuous-time causal inference for multi-agent dynamical systems. In Singh, A. K., Sun, Y., Akoglu, L., Gunopulos, D., Yan, X., Kumar, R., Ozcan, F., and Ye, J., editors, *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2023, Long Beach, CA, USA, August 6-10, 2023*, pages 997–1009. ACM. 3

Jiang, S., Koch, B., and Sun, Y. (2021). HINTS: citation time series prediction for new publications via dynamic heterogeneous information network embedding. In Leskovec, J., Grobelnik, M., Najork, M., Tang, J., and Zia, L., editors, *WWW '21: The Web Conference 2021, Virtual Event / Ljubljana, Slovenia, April 19-23, 2021*, pages 3158–3167. ACM / IW3C2. 3

Jiang, S. and Sun, Y. (2022). Estimating causal effects on networked observational data via representation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 852–861. 3, 78, 79, 80, 83, 99

Jiang, S., Syed, T., Zhu, X., Levy, J., Aronchik, B., and Sun, Y. (2022). Bridging self-attention and time series decomposition for periodic forecasting. In Hasan, M. A. and Xiong, L., editors, *Proceedings of the 31st ACM International Conference on Information & Knowledge Management, Atlanta, GA, USA, October 17-21, 2022*, pages 3202–3211. ACM. 3

Johansson, F., Shalit, U., and Sontag, D. (2016). Learning representations for counterfactual inference. In *International conference on machine learning*, pages 3020–3029. PMLR. 47, 49, 51, 80, 82

Karypis, G. and Kumar, V. (1998). A fast and high quality multilevel scheme for partitioning irregular graphs. *SIAM Journal on scientific Computing*, 20(1):359–392. 67, 99

Ke, Q., Ferrara, E., Radicchi, F., and Flammini, A. (2015). Defining and identifying sleeping beauties in science. *Proceedings of the National Academy of Sciences*, 112(24):7426–7431. 11

Kidger, P., Morrill, J., Foster, J., and Lyons, T. (2020). Neural controlled differential equations for irregular time series. *Advances in Neural Information Processing Systems*, 33:6696–6707. 86

Kingma, D. P. and Ba, J. (2015a). Adam: A method for stochastic optimization. In Bengio, Y. and LeCun, Y., editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 19, 69, 86

Kingma, D. P. and Ba, J. (2015b). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. 39

Kipf, T. N. and Welling, M. (2017a). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 6, 11, 13, 18, 48, 62

Kipf, T. N. and Welling, M. (2017b). Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. 76, 82

Kitaev, N., Kaiser, L., and Levskaya, A. (2020). Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. 32, 38

LeCun, Y. (2022). A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1). 102

Li, C., Ma, J., Guo, X., and Mei, Q. (2017). Deepcas: An end-to-end predictor of information cascades. In *Proceedings of the 26th international conference on World Wide Web*, pages 577–586. 16, 18, 19

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. (2019). Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *Advances in neural information processing systems*, 32. 26, 31, 32

Lim, B., Alaa, A., and Van Der Schaar, M. (2018). Forecasting treatment responses over time using recurrent marginal structural networks. *advances in neural information processing systems*, 31. 78

Lim, B., Arık, S. Ö., Loeff, N., and Pfister, T. (2021). Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting*, 37(4):1748–1764. 28

Liu, X., Yan, J., Xiao, S., Wang, X., Zha, H., and Chu, S. M. (2017). On predictive patent valuation: Forecasting patent citations and their types. In *AAAI*, pages 1438–1444. 4

Ma, J., Guo, R., Chen, C., Zhang, A., and Li, J. (2021). Deconfounding with networked observational data in a dynamic environment. In *Proceedings of the 14th ACM International Conference on Web Search and Data Mining*, pages 166–174. 48, 67, 86, 99

Ma, J., Wan, M., Yang, L., Li, J., Hecht, B., and Teevan, J. (2022). Learning causal effects on hypergraphs. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1202–1212. 78, 86, 99

Ma, Y. and Tresp, V. (2021). Causal inference under networked interference and intervention policy enhancement. In *International Conference on Artificial Intelligence and Statistics*, pages 3700–3708. PMLR. 78, 79, 83

Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605. 23

Manjunatha, J., Sivaramakrishnan, K., Pandey, R. K., and Murthy, M. N. (2003). Citation prediction using time series approach kdd cup 2003 (task 1). *ACM SIGKDD Explorations Newsletter*, 5(2):152–153. 18

Matrajt, L., Eaton, J., Leung, T., and Brown, E. R. (2020). Vaccine optimization for covid-19: Who to vaccinate first? *Science Advances*, 7(6). 47

McPherson, M., Smith-Lovin, L., and Cook, J. M. (2001). Birds of a feather: Homophily in social networks. *Annual review of sociology*, 27(1):415–444. 47

Medlock, J. and Galvani, A. P. (2009). Optimizing influenza vaccine distribution. *Science*, 325(5948):1705–1708. 73

Melnychuk, V., Frauen, D., and Feuerriegel, S. (2022). 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*, pages 15293–15329. PMLR. 73, 78, 80, 95, 99

Nabi, R., Pfeiffer, J., Bayir, M. A., Charles, D., and Kıcıman, E. (2020). Causal inference in the presence of interference in sponsored search advertising. *Workshop on Causal Discovery and Causality-Inspired Machine Learning*. 47

Ogburn, E. L., Shpitser, I., and Lee, Y. (2020). Causal inference, social networks and chain graphs. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, 183(4):1659–1676. 47

Ogburn, E. L., Sofrygin, O., Diaz, I., and Van der Laan, M. J. (2017). Causal inference for social network data. *arXiv preprint arXiv:1705.08527*. 47

Ogburn, E. L. and VanderWeele, T. J. (2014). Causal diagrams for interference. *Statistical science*, 29(4):559–578. 51

Oreshkin, B. N., Carpov, D., Chapados, N., and Bengio, Y. (2020). N-BEATS: neural basis expansion analysis for interpretable time series forecasting. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. 34, 36

Pearl, J. (2009). *Causality*. Cambridge university press. 47, 48, 50, 52

Platt, R. W., Schisterman, E. F., and Cole, S. R. (2009). Time-modified confounding. *American journal of epidemiology*, 170(6):687–694. 75

Porter, M. A. and Gleeson, J. P. (2014). Dynamical systems on networks: A tutorial. *arXiv preprint arXiv:1403.7663*. 75

Qin, Y., Song, D., Cheng, H., Cheng, W., Jiang, G., and Cottrell, G. W. (2017). A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 2627–2633. AAAI Press. 18, 30

Robert, C., William, C., and Irma, T. (1990). Stl: A seasonal-trend decomposition procedure based on loess. *Journal of official statistics*, 6(1). 1, 27

Robins, J. M. and Hernán, M. A. (2009). Estimation of the causal effects of time-varying exposures. *Longitudinal data analysis*, 553:599. 77

Robins, J. M., Hernan, M. A., and Brumback, B. (2000). Marginal structural models and causal inference in epidemiology. 73, 80

Rosenbaum, P. R. and Rubin, D. B. (1983). The central role of the propensity score in observational studies for causal effects. *Biometrika*, 70(1):41–55. 61

Rubin, D. B. (1978). Bayesian inference for causal effects: The role of randomization. *The Annals of statistics*, pages 34–58. 77

Rubin, D. B. (1980). Randomization analysis of experimental data: The fisher randomization test comment. *Journal of the American statistical association*, 75(371):591–593. 48

Rubin, D. B. (2006). *Matched sampling for causal effects*. Cambridge University Press. 61

Salinas, D., Flunkert, V., Gasthaus, J., and Januschowski, T. (2020). Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191. 26, 30, 38

Schlichtkrull, M., Kipf, T. N., Bloem, P., van den Berg, R., Titov, I., and Welling, M. (2018). Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer. 6, 12

Schuster, M. and Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing*, 45(11):2673–2681. 30

Seedat, N., Imrie, F., Bellot, A., Qian, Z., and van der Schaar, M. (2022). Continuous-time modeling of counterfactual outcomes using neural controlled differential equations. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 19497–19521. PMLR. 77, 78, 83, 86, 99, 100, 101

Shalit, U., Johansson, F. D., and Sontag, D. (2017). Estimating individual treatment effect: generalization bounds and algorithms. In *International Conference on Machine Learning*, pages 3076–3085. PMLR. 47, 48, 49, 51, 55, 60, 68, 69, 80, 83

Shang, J., Liu, J., Jiang, M., Ren, X., Voss, C. R., and Han, J. (2018). Automated phrase mining from massive text corpora. *IEEE Transactions on Knowledge and Data Engineering*, 30(10):1825–1837. 17

Shen, H., Wang, D., Song, C., and Barabási, A.-L. (2014). Modeling and predicting popularity dynamics via reinforced poisson processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28. 4

Sinatra, R., Wang, D., Deville, P., Song, C., and Barabási, A.-L. (2016). Quantifying the evolution of individual scientific impact. *Science*, 354(6312):aaf5239. 4, 14

Sun, Y., Barber, R., Gupta, M., Aggarwal, C. C., and Han, J. (2011a). Co-author relationship prediction in heterogeneous bibliographic networks. In *2011 International Conference on Advances in Social Networks Analysis and Mining*, pages 121–128. IEEE. 8

Sun, Y. and Han, J. (2013). Mining heterogeneous information networks: a structural analysis approach. *Acm Sigkdd Explorations Newsletter*, 14(2). 8

Sun, Y., Han, J., Yan, X., Yu, P. S., and Wu, T. (2011b). Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(11):992–1003. 8

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 14, 19, 33, 35

Tang, J., Zhang, J., Yao, L., Li, J., Zhang, L., and Su, Z. (2008). Arnetminer: Extraction and mining of academic social networks. In *KDD'08*, pages 990–998. 17

Uzzi, B., Mukherjee, S., Stringer, M., and Jones, B. (2013). Atypical combinations and scientific impact. *Science*, 342(6157):468–472. 11

Van der Maaten, L. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, 9(11). 45, 71, 88

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30. 1, 26, 30, 32

Veitch, V., Wang, Y., and Blei, D. (2019). Using embeddings to correct for unobserved confounding in networks. In *Advances in Neural Information Processing Systems*, pages 13792–13802. 48, 51, 67, 86, 99

Velickovic, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., and Bengio, Y. Graph attention networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 48

Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2018). Graph attention networks. In *International Conference on Learning Representations, ICLR*. 13, 25

Wang, D., Song, C., and Barabási, A.-L. (2013). Quantifying long-term scientific impact. *Science*, 342(6154):127–132. 4, 6, 11, 14, 15, 23

Wang, H., He, H., and Katabi, D. (2020). Continuously indexed domain adaptation. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9898–9907. PMLR. 76, 83, 84, 97, 98

Wang, J., Wang, Z., Li, J., and Wu, J. (2018). Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2437–2446. 18

Wang, X., Ji, H., Shi, C., Wang, B., Ye, Y., Cui, P., and Yu, P. S. (2019). Heterogeneous graph attention network. In *The World Wide Web Conference*, pages 2022–2032. 13

Wang, Y. and Blei, D. M. (2019). The blessings of multiple causes. *Journal of the American Statistical Association*, 114(528):1574–1596. 95

Wu, F., Souza, A., Zhang, T., Fifty, C., Yu, T., and Weinberger, K. (2019). Simplifying graph convolutional networks. In *International conference on machine learning*, pages 6861–6871. PMLR. 48

Wu, H., Xu, J., Wang, J., and Long, M. (2021). Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems*, 34:22419–22430. 26, 28, 32

Xiao, S., Yan, J., Li, C., Jin, B., Wang, X., Yang, X., Chu, S. M., and Zhu, H. (2016). On modeling and predicting individual paper citation count over time. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, IJCAI'16. 4

Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. 48

Xu, Y., Xu, Y., and Saria, S. (2016). A bayesian nonparametric approach for estimating individualized treatment-response curves. In *Machine learning for healthcare conference*, pages 282–300. PMLR. 73

Yan, R., Huang, C., Tang, J., Zhang, Y., and Li, X. (2012). To better stand on the shoulder of giants. In *Proceedings of the 12th ACM/IEEE-CS joint conference on Digital Libraries*, pages 51–60. ACM. 6

Yan, R., Tang, J., Liu, X., Shan, D., and Li, X. (2011). Citation count prediction: learning to estimate future citations for literature. In *Proceedings of the 20th ACM international conference on Information and knowledge management*, pages 1247–1252. ACM. 4, 6, 18

Yao, L., Li, S., Li, Y., Huai, M., Gao, J., and Zhang, A. (2018). Representation learning for treatment effect estimation from observational data. *Advances in Neural Information Processing Systems*, 31:2633–2643. 47

Yoon, J., Jordon, J., and van der Schaar, M. (2018). Ganite: Estimation of individualized treatment effects using generative adversarial nets. In *International Conference on Learning Representations*. 47

Yu, T., Yu, G., Li, P.-Y., and Wang, L. (2014). Citation impact prediction for scientific papers using stepwise regression analysis. *Scientometrics*, 101(2):1233–1252. 6

Yu, Y., Kan, X., Cui, H., Xu, R., Zheng, Y., Song, X., Zhu, Y., Zhang, K., Nabi, R., Guo, Y., et al. (2022). Learning task-aware effective brain connectivity for fmri analysis with graph neural networks. *arXiv preprint arXiv:2211.00261*. 74

Yuan, S., Tang, J., Zhang, Y., Wang, Y., and Xiao, T. (2018). Modeling and predicting citation count via recurrent neural network with long short-term memory. *arXiv preprint arXiv:1811.02129*. 4

Yuan, Y., Altenburger, K., and Kooti, F. (2021). Causal network motifs: Identifying heterogeneous spillover effects in a/b tests. In *Proceedings of the Web Conference 2021*, pages 3359–3370. 47

Zhao, L., Song, Y., Zhang, C., Liu, Y., Wang, P., Lin, T., Deng, M., and Li, H. (2019). T-gcn: A temporal graph convolutional network for traffic prediction. *IEEE Transactions on Intelligent Transportation Systems*, 21(9):3848–3858. 18

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. (2021). Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 11106–11115. x, xv, 26, 27, 30, 31, 32, 33, 38, 39

Ziyin, L., Hartwig, T., and Ueda, M. (2020). Neural networks fail to learn periodic functions and how to fix it. *Advances in Neural Information Processing Systems*, 33:1583–1594. x, 26, 27, 33, 42