Causal Graph ODE: Continuous Treatment Effect Modeling in Multi-agent Dynamical Systems

Zijie Huang[*], Jeehyun Hwang[*], Junkai Zhang[*], Jinwoo Baik, Weitong Zhang
Dominik Wodarz, Yizhou Sun, Quanquan Gu, Wei Wang
Department of Computer Science
University of California, Los Angeles
Los Angeles, CA 90095
`zijiehuang@cs.ucla.edu, jeehyunhwang@cs.ucla.edu, zhang@cs.ucla.edu`
`jbaik23@ucla.edu, wt.zhang@ucla.edu, dwodarz@uci.edu`
`yzsun@cs.ucla.edu, qgu@cs.ucla.edu, weiwang@cs.ucla.edu`

Abstract

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

1 Introduction

Many real-world multi-agent systems are dynamic and continuous, where agents (nodes) interact and exhibit complex behaviors over time. This results in time-evolving node trajectories and dynamic interaction edges. An example is the spread of COVID-19 in the U.S., where states act as agents and daily migration patterns across states form interaction edges [13, 26]. Estimating the counterfactual outcomes over time in such systems is crucial for various applications, such as formulating effective policies and designing medical treatment plans [35, 4, 3]. This can achieve more accurate predictions than non-causal methods by considering the influence of biased confounders. For example, the health status of the residents in each state (confounders) can impact their level of adherence to the state's policies(treatments), which can influence future confirmed cases/deaths (outcomes). Furthermore, causal inference for multi-agent dynamical systems enables effective decision-making by addressing causal questions such as "What if we remove a policy at a specific time" or "What if we change the order of different policies". Therefore, it serves as a promising tool for policymakers.

[*]Equal contribution.

However, two challenges exist when applying causal inference to multi-agent dynamical systems. One is that most existing methods [35, 4] assume that nodes are independent, meaning their trajectories are determined solely by their own treatments. Another challenge is that current methods lack the ability to capture the continuous and dynamic effects of multiple treatments on such systems.

To tackle these challenges, we propose the Causal Graph Ordinary Differential Equations (CAG-ODE) model to estimate the continuous counterfactual outcome of a multi-agent dynamical system in the presence of multiple treatments and time-varying confounding and interference. Building upon the recent success of graph ordinary differential equations (ODE) in capturing the continuous interaction among agents [13, 12, 25], our key innovation is to learn time-dependent representations of simultaneous treatments and incorporate them into the ODE function to accurately account for their casual effects on the system. As nodes and edges are jointly evolving, we utilize two coupled treatment-induced ODE functions to account for their dynamics respectively. To mitigate confounding bias, we further design two adversarial learning losses, which enable our model to learn balanced continuous trajectory representations unaffected by treatments or interference. Experiments on both real and simulated datasets demonstrate the effectiveness of our proposed model.

2 Problem Definition

We consider a multi-agent dynamical system as an evolving interaction graph $\mathcal{G}^t = \{\mathcal{V}, \mathcal{E}^t\}$, where nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$ are agents and $\mathcal{E}^t$ are the weighted edges among them. $\mathcal{W}_A = \{\mathbf{W}_A^1, \mathbf{W}_A^2, \ldots, \mathbf{W}_A^T\}$ represents the weighted adjacency matrix sequence, where $\mathbf{W}_A^t \in \mathbb{R}^{n\times n}$ and $w_{i\to j} \in \mathbb{R}$ is the weight of the directed edge pointing from node i to node j.

Following [16], we denote the observational data at timestamp t as $(\mathbf{X}^t, \mathbf{A}^t, \mathbf{Y}^t)$, where $\mathbf{X}^t \in \mathbb{R}^{N\times d_1}$ represents the time-varying covariates (e.g., the health status of residents). $\mathbf{A}^t \in \{0,1\}^{N\times K}$ are time-dependent treatments, where $\mathbf{A}_{ij}^t = 1$ denotes the j^{th} treatment assigned to node i at timestamp t, and K is the number of heterogeneous treatments. $\mathbf{Y}^t \in \mathbb{R}^{N\times d_2}$ is the time-dependent outcome, such as the number of confirmed cases in each state. The historical observations up to time t is represented as $\mathcal{H}^t = \{\overline{\mathbf{X}}^t, \overline{\mathbf{A}}^t, \overline{\mathbf{Y}}^t\}$, where $\overline{\mathbf{X}}^t, \overline{\mathbf{A}}^t, \overline{\mathbf{Y}}^t$ contain all $\mathbf{X}^{t^-}, \mathbf{A}^{t^-}, \mathbf{Y}^{t^-}\left(t^- \leq t\right)$. We predict the unbiased potential outcomes $\mathbb{E}(\mathrm{Y}^{t^+}\left(\mathbf{A}^{t^+} = a\right)|\mathcal{H}^t, \mathcal{W}_{\mathcal{A}})$[2]. Here, a is the dynamic treatment trajectory (e.g. sequences of state policies).

We utilize the GraphODE framework [12, 13, 24]to model the dynamic behavior of a multi-agent system in a continuous manner: $\dot{\boldsymbol{z}}_i^t := \frac{d\boldsymbol{z}_i^t}{dt} = g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right)$. Here $\boldsymbol{z}_i^t \in \mathbb{R}^d$ denotes the latent state variable for agent i at timestamp t and g denotes the ODE function that drives the system to move forward, which is parameterized as a graph neural network (GNN). The latent initial states $\boldsymbol{z}_1^0, \cdots \boldsymbol{z}_N^0$ are computed from a spatial-temporal GNN encoder $\boldsymbol{z}_i^0 = f_{\text{ENC}}(\mathcal{H}^0)$. Given the initial states and the ODE function, the solution $\boldsymbol{z}_i^T$ can be evaluated at any desired time T using any numerical ODE solver [29] on $\boldsymbol{z}_i^T = \boldsymbol{z}_i^0 + \int_{t=0}^{T} g\left(\boldsymbol{z}_1^t, \boldsymbol{z}_2^t \cdots \boldsymbol{z}_N^t\right) \mathrm{d}t$. Finally, a decoder outputs the predicted dynamics based on the latent states. Details can be found in Appendix C.1.

3 The Proposed Model: CAG-ODE

We present Causal Graph ODE (CAG-ODE) to predict continuous counterfactual outcomes for multi-agent dynamical systems with dynamic multi-treatment effects as illustrated in Figure 1. Contrary to GraphODEs [13, 15], CAG-ODE can perform causal reasoning by injecting treatment effects into the ODE functions, which we call *treatment-induced coupled graph ODE*. The multi-treatment effects are captured by a novel treatment fusing module that assigns temporal weights to the treatments using an attention mechanism. CAG-ODE also utilizes two adversarial learning losses to ensure unbiased estimations of counterfactual outcomes.

3.1 Treatment-Induced GraphODE

Treatment Fusing. Treatments can have time-varying effects in multi-agent dynamical systems and they can occur simultaneously, resulting in a combined effect. We propose a novel treatment

[2]The potential outcome can be formalized with *do* operation [30].

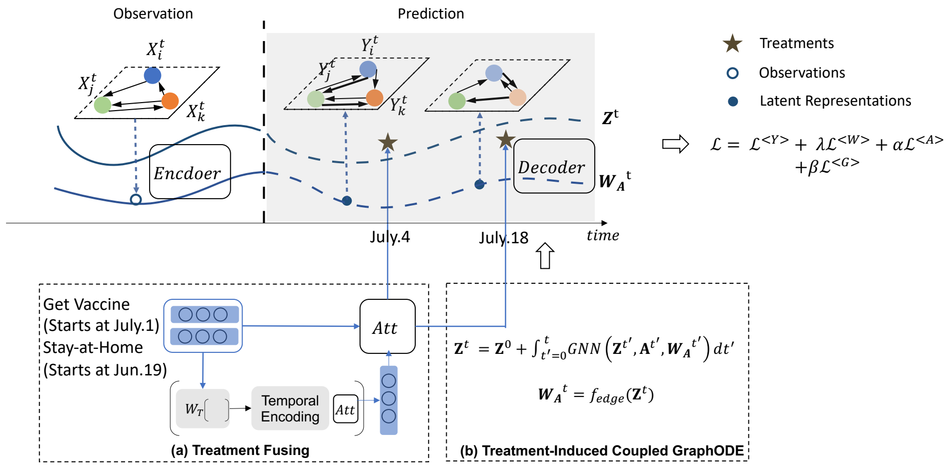


Figure 1: Overall Framework of CAG-ODE.

fusing module that assigns temporal weights to multiple treatments through an attention mechanism as shown below. Let $\boldsymbol{e}_k \in \mathbb{R}^K$ be the one-hot representation of treatment k. We first add it with the temporal encoding TE[13, 36] to account for the time elapsed since the start of the treatment t'. $\mathbf{A}^t_{ik} \in \{0, 1\}$ is an indicator showing whether treatment k is applied to agent i at time t. A contraction matrix $\mathbf{W}_q$ is then used to transform this sparse representation into a more compact form.

$$\widehat{\boldsymbol{o}}^t_{ik} = \mathbf{A}^t_{ik}\boldsymbol{e}_k + \mathrm{TE}\left(t - t'\right)\mathbb{1}[\mathbf{A}^t_{ik} = 1], \quad \boldsymbol{o}^t_{ik} = \mathbf{W}_q\widehat{\boldsymbol{o}}^t_{ik}, \tag{1}$$

$$\mathrm{TE}(\Delta t)_{2i} = \sin\left(\Delta t/M^{2i/d}\right), \mathrm{TE}(\Delta t)_{2i+1} = \cos\left(\Delta t/M^{2i/d}\right), M = 10000. \tag{2}$$

We compute the combined treatment representation as a weighted sum of all in-effect treatments (Eqn 3). We first compute an attention vector m^t_i as the tanh-transformed average of all the treatment representations. Each treatment's weight is derived from the dot product of its representation and m^t_i.

$$\boldsymbol{o}^t_i = \frac{1}{K}\sum_k\left(\boldsymbol{m}^{t\top}_i \boldsymbol{o}^t_{ik}\boldsymbol{o}^t_{ik}\right), \boldsymbol{m}^t_i = \tanh\left(\left(\frac{1}{K}\sum_k \boldsymbol{o}^t_{ik}\right)\mathbf{W}_m\right). \tag{3}$$

Treatment-Induced GraphODE. We use two coupled ODEs to predict the latent trajectories for nodes and edges respectively, accounting for their co-evolution as in [13]. We incorporate the treatment representations into the ODEs to enable counterfactual predictions in the future. Specifically, we use a linear transformation $\mathbf{W}$ to merge the latent states of nodes $\mathbf{Z}^t$ and the treatment representation $\boldsymbol{O}^t$ as $\mathbf{W}[\mathbf{Z}^t||\mathbf{O}^t]$ with $||$ as the concatenation operator. Details of the ODE function can be found at C.2.

Outcome Prediction. Given the treatment representations, the ODE functions, and the latent initial states, we calculate the reconstruction loss of nodes and edges as:

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

where $\widehat{\mathbf{Y}}^t$ and $\widehat{\mathbf{W}}^t_A$ are computed from a decoder based on the latent trajectories (details in Appendix C.2).

3.2 Domain Adversarial Learning

In observational data, treatment assignments are not randomized but are biased based on time-varying confounder values. This can lead to increased variance and bias in counterfactual estimation [35]. In multi-agent dynamical systems, unbalanced interference from neighboring agents further exacerbates this effect and alters the state of each agent. To obtain an unbiased counterfactual prediction, we need to ensure that the distribution of latent representation trajectories is invariant to treatments and interference [16]. To achieve this, we incorporate two adversarial learning losses into the optimization objective function and use gradient reversal layers for the implementation. We use $L^{\langle A\rangle}, L^{\langle G\rangle}$ to

Table 1: Root Mean Square Error (RMSE) for factual outcome evaluation across different prediction lengths(the duration for which predictions are made). Ablation study can be found at Appendix F.2.

Dataset	Covid-19			Tumor Growth		
Prediction Length	7-days	14-days	21-days	14-days	21-days	28-days
CG-ODE	4063	4454	4659	18.37	21.00	24.58
TE-CDE	7999	7470	6832	55.45	55.38	71.23
COVID-POLICY	3973	4268	3944	28.99	42.81	32.91
CAG-ODE	**3710**	**3925**	**3933**	**10.91**	**10.82**	**14.84**

denote the treatment balancing and interference balancing loss respectively. Implementation details can be found at Appendix C.3.

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

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

4 Experiments

4.1 Experiment Setup

We evaluate the performance of our model using two datasets: 1.) The **COVID-19 dataset**, which captures the daily COVID-19 trends of U.S. states from April.12.2020 to Dec.31.2020. The daily population flows among states are represented as dynamic edges. Treatments are state-level COVID-19 policies. We ask the model to predict the daily confirmed cases in each state. 2.) The **Tumor Growth simulation dataset** [8], which describes the tumor growth dynamics in different regions of patients, where they may receive differing treatments. We aim to predict the tumor volumes in each region. Additional details about the datasets can be found in Appendix D.

4.2 Performance Evaluation

We conduct a comparative analysis of our model with three baseline models: one non-causal continuous multi-agent baseline CG-ODE [13], and two causal models: TE-CDE [35] and COVID-Policy [26]. We also compare variants of our model, including models without treatment balancing, interference balancing, both components, or the attention module. The results of factual Outcome predictions are shown in Table 1, and the results of counterfactual outcome predictions is displayed in Appendix F.3. As the COVID-19 is a real-world dataset that does not have counterfactual outcomes, we evaluate only the Tumor Growth dataset. To assess the accuracy of short-term and long-term predictions, the prediction lengths for the COVID-19 and Tumor Growth datasets are set to 7, 14, 21 days and 14, 21, and 28 days, respectively. We include longer-range predictions on the Tumor-Growth dataset in Appendix F.4.

Factual Outcome Predictions. Table 1 shows that our model, CAG-ODE, consistently outperforms the baseline models across all prediction lengths for both datasets. This underscores the effectiveness of our model in capturing the dynamic interactions among objects, especially over longer time periods. Comparing our model with TE-CDE, we observe a performance gap that highlights the benefits of incorporating interference balancing and spatial correlation in the model. Additionally, on the Tumor Growth dataset, our model outperforms the COVID-POLICY model, indicating its broader generalizability across different types of data. Furthermore, our model exhibits proficiency in both short-term and long-term predictions. The analysis of our model variants further emphasizes the importance of each component in the model. Additional experiments with different configurations and multiple runs using the Tumor Growth dataset can be found in Appendix F.5.

5 Conclusion

In this paper, we introduce CAG-ODE for estimating continuous counterfactual outcomes in multi-agent-dynamical systems considering dynamic multi-treatment effects. Our model builds upon existing GraphODEs and we propose a novel treatment fusing module that captures the dynamic effects of multiple treatments occurring simultaneously. Through extensive experiments on two datasets, we demonstrate the superior performance of CAG-ODE across various prediction settings, validating its effectiveness. Furthermore, we leverage our model to analyze policy effects for the COVID-19 transmission, providing valuable insights for policymakers (in Appendix F.1).

References

[1] Anonymous. CF-GODE: Continuous-time causal inference for multi-agent dynamical systems. In *29th SIGKDD Conference on Knowledge Discovery and Data Mining - Research Track*, 2023.

[2] Guangji Bai, Chen Ling, and Liang Zhao. Temporal domain generalization with drift-aware dynamic neural networks. *arXiv preprint arXiv:2205.10664*, 2022.

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

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

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

[6] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems 31*, pages 6571–6583. 2018.

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

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

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

[10] Ehsan Hajiramezanali, Arman Hasanzadeh, Krishna Narayanan, Nick Duffield, Mingyuan Zhou, and Xiaoning Qian. Variational graph recurrent neural networks. In *Advances in Neural Information Processing Systems 32*, pages 10701–10711. 2019.

[11] Ziniu Hu, Yuxiao Dong, Kuansan Wang, and Yizhou Sun. Heterogeneous graph transformer. In *Proceedings of the 2020 World Wide Web Conference*, 2020.

[12] Zijie Huang, Yizhou Sun, and Wei Wang. Learning continuous system dynamics from irregularly-sampled partial observations. In *Advances in Neural Information Processing Systems*, 2020.

[13] Zijie Huang, Yizhou Sun, and Wei Wang. Coupled graph ode for learning interacting system dynamics. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, 2021.

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

[15] Zijie Huang, Wanjia Zhao, Jingdong Gao, Ziniu Hu, Xiao Luo, Yadi Cao, Yuanzhou Chen, Yizhou Sun, and Wei Wang. Tango: Time-reversal latent graphode for multi-agent dynamical systems, 2023.

[16] Song Jiang, Zijie Huang, Xiao Luo, and Yizhou Sun. Cf-gode: Continuous-time causal inference for multi-agent dynamical systems. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 997–1009, 2023.

[17] Song Jiang and Yizhou Sun. Estimating causal effects on networked observational data via representation learning. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pages 852–861, 2022.

[18] Yuhao Kang, Song Gao, Yunlei Liang, Mingxiao Li, and Jake Kruse. Multi-scale dynamic human mobility flow dataset in the u.s. during the covid-19 epidemic. *Scientific Data*, 1-13, 2020.

[19] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

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

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

[22] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR'17*, 2017.

[23] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

[24] Xiao Luo, Haixin Wang, Zijie Huang, Huiyu Jiang, Abhijeet Sadashiv Gangan, Song Jiang, and Yizhou Sun. Care: Modeling interacting dynamics under temporal environmental variation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[25] Xiao Luo, Jingyang Yuan, Zijie Huang, Huiyu Jiang, Yifang Qin, Wei Ju, Ming Zhang, and Yizhou Sun. Hope: High-order graph ode for modeling interacting dynamics. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23, 2023.

[26] Jing Ma, Yushun Dong, Zheng Huang, Daniel Mietchen, and Jundong Li. Assessing the causal impact of covid-19 related policies on outbreak dynamics: A case study in the us. In *Proceedings of the ACM Web Conference 2022*, WWW '22, page 2678–2686, 2022.

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

[28] Valentyn Melnychuk, Dennis Frauen, and Stefan Feuerriegel. Causal transformer for estimating counterfactual outcomes. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 15293–15329. PMLR, 2022.

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

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

[31] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

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

[33] Alvaro Sanchez-Gonzalez, Jonathan Godwin, Tobias Pfaff, Rex Ying, Jure Leskovec, and Peter W. Battaglia. Learning to simulate complex physics with graph networks. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, pages 8459–8468, 2020.

[34] Aravind Sankar, Yanhong Wu, Liang Gou, Wei Zhang, and Hao Yang. Dysat: Deep neural representation learning on dynamic graphs via self-attention networks. In *WSDM'20*, 2020.

[35] Nabeel Seedat, Fergus Imrie, Alexis Bellot, Zhaozhi Qian, and Mihaela van der Schaar. Continuous-time modeling of counterfactual outcomes using neural controlled differential equations. In *International Conference on Machine Learning*, pages 19497–19521. PMLR, 2022.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30*, pages 5998–6008. 2017.

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

[38] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *ICLR'19*, 2019.

[39] Chengxi Zang and Fei Wang. Neural dynamics on complex networks. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 892–902, 2020.

[40] Yunhao Zhang and Junchi Yan. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *International Conference on Learning Representations*, 2023.

A Related Works

A.1 Graph Ordinary Differential Equations for Continuous Multi-agent Dynamical Systems

To model the interactions among agents in a dynamical system, recent studies [12, 13, 39, 31] propose using a Graph Neural Network (GNN) as the ODE function g which is learned. Also known as the GraphODE framework, it follows an encoder-processor-decoder architecture. The encoder computes latent initial states for all agents based on historical observations. The GNN-based ODE function then predicts the latent trajectories starting from the learned initial states. Finally, a decoder extracts the predicted dynamic features. In order to regularize the generated trajectories, GraphODE frameworks often adopt a variational autoencoder (VAE) structure [19], where the encoder computes and samples from approximated posterior distributions for initial states. GraphODEs are promising in making long-range predictions and can handle irregularly-sampled observations effectively [12, 39].

A.2 Causal Inference Over Time

Existing time-dependent causal inference methods mainly differ in how they deal with confounding. They differ from traditional statistical time series analysis [20, 40, 2] which we do not discuss in this paper. Traditionally, many statistical tools that are applied, such as marginal structural models (MSMs) [32], utilize the inverse probability of treatment weighting (IPTW), which are linear in treatment and covariate effect. More recently, researchers use representation learning based balancing approaches to aim to learn representations that are not predictable of the treatments in order to ensure unbiased outcome prediction [5, 28]. This is achieved by training the model with both the prediction loss and the treatment balancing loss. However, one major limitation is that they are all discrete methods, which can offer poor performance on continuous systems such as the spread of COVID-19. Currently, there are a series of works [35, 4, 9, 7] that estimate the continuous counterfactual outcomes through neural ordinary differential equations (ODEs) or neural controlled differential equations (CDEs). Despite their success, they assume that nodes are independent of each other, which is unsuitable for multi-agent dynamical systems. One recent work [1] proposed to parameterize the ODE function with a GNN to generalize to multi-agent settings. However, a major limitation is that this model cannot handle evolving graph structures (edges).

A.3 Graph Neural Networks (GNNs) for Dynamical System Simulations

Graph Neural Networks (GNNs) are a class of neural networks that operate on graph-structured data by passing local messages [22, 37, 38, 13]. They have been extensively employed in various applications such as node classification, link prediction, and recommendation systems. By viewing each agent as a node and interaction among agents as edges, GNNs have shown to be efficient for approximating pair-wise node interactions and achieved accurate predictions for multi-agent dynamical systems [21, 33]. The majority of existing studies propose discrete GNN-based simulators where they take the node features at time t as input to predict the node features at time t+1. To further capture the long-term temporal dependency for predicting future trajectories, some work utilizes recurrent neural networks such as RNN, LSTM or self-attention mechanism to make prediction at time t +1 based on the historical trajectory sequence within a time window [10, 34, 11]. However, they all restrict themselves to learn a one-step state transition function. Therefore, when we successively apply these one-step simulators to previous predictions in order to generate the rollout trajectories, error accumulates and impairs the prediction accuracy, especially for long-range prediction.

B Assumptions

To make potential outcomes identifiable from observational data, we adhere to the following three standard assumptions as in existing works [5, 35, 1]:

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

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

The last assumption defines unconfoundedness (strong ignorability) in multi-agent dynamical systems. We first define the interference effects caused by neighbors' treatments of node i as $\mathbf{G}_i^t = \sum_{j \in \mathcal{N}_i} \frac{1}{|N_i|} \mathbf{A}_j^t \in \mathbb{R}^K$, which is the proportion of treated nodes in node i's neighbors for each treatment type. We refer to $\mathbf{G}_i^t$ as interference summary, which assumes that a node is only influenced by treatments of its immediate neighbors as in previous studies [1, 27, 17].

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

It ensures that it is sufficient to only condition on the historical observations and graph sequences up to t to block all backdoor paths so as to estimate the potential outcome at any time in the future.

With these three assumptions, the potential outcome trajectory can be identified as: $\mathbb{E}\left(\mathbf{Y}^{t^+}(\mathbf{A}^{t^+} = a) \mid \mathcal{H}^t, \mathcal{G}\right) = \mathbb{E}\left(\mathbf{Y}^{t^+} \mid \mathbf{A}^{t^+}, \mathbf{G}^{t^+}, \mathcal{H}^t, \mathcal{G}\right)$. This enables us to estimate the potential outcomes by training a machine learning model using observational data, and use the same model to predict counterfactual outcomes given new treatment trajectories.

C Detailed Description on Model Architecture

C.1 Details of Encoder

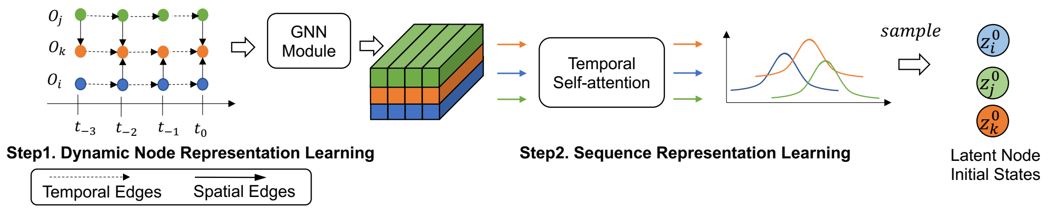


Figure 2: Encoder Computation Flow of Initial States (Adapted from [13]).

The encoder generates the latent initial states for agents and their edges, which are the starting point for future trajectories. It follows the architecture described in [14], which computes the posterior distributions of nodes and edges based on historical observations up to time t_0 and then samples the latent initial states. As agents are mutually influenced over time, the encoder leverages a spatio-temporal GNN to output the latent initial states for all agents using a two-step procedure. It first constructs a spatio-temporal graph, where each node represents an observation at a timestamp. Edges are built between nodes of connected agents within a single timestamp (spatial edges) and between nodes representing observations of the same agent over time (temporal edges). Based on the graph, it employs a GNN to obtain the structural representation for agent i's observation at time t, denoted as h_i^t, through $h_i^t = f_{\text{GNNE}}(\mathcal{H}^t)$. Next, the model computes the sequence representation for each agent, by summarizing its observation node sequence as $\boldsymbol{u}_i = f_{\text{seq}}(h_i^{-T_1}, \cdots, h_i^{-1})$, where T_1 is the observation length. Finally, a neural network calculates the mean and variance of the posterior normal distribution for agent i, from which the latent initial state is sampled:

$$\boldsymbol{z}_i^0 \sim q_\phi(\boldsymbol{z}_i^0 | \mathcal{H}^0) = \mathcal{N}(\boldsymbol{\mu}_{\boldsymbol{z}_i^0}, \sigma^2_{\boldsymbol{z}_i^0}),\ \boldsymbol{\mu}_{\boldsymbol{z}_i^0}, \sigma_{\boldsymbol{z}_i^0} = f_{\text{dist}}(\boldsymbol{u}_i). \tag{4}$$

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

Dynamic Node Representation Learning. To integrate the spatial-temporal structure of multi-agent dynamical systems, we construct a graph as illustrated in Figure 2, where each node represents an observation of an agent at a specific timestamp. Consequently, we need to establish two types of edges: spatial edges at the same timestamp and temporal edges across different timestamps. The spatial edges are formed according to the adjacency matrices, denoted as $w_{i(t) \to j(t)}$. For the temporal edges, we only consider edges from an agent's own previous observations to later observations, denoted as $w_{i(t) \to i(t')}$. Our default setting is $t' = t + 1$ as illustrated in Figure 2 in which we have temporal edges between consecutive observations.

The latent representations of observations are learned from this spatial-temporal graph through an attention mechanism approach. The propagation among L GNN layers is depicted in Equation(5).

$$
\begin{aligned}
\boldsymbol{h}_{i(t')}^{l} &= \boldsymbol{h}_{i(t')}^{l} + \sigma\left(\sum_{j(t)\in\mathcal{N}_{i(t')}} e_{j(t)\to i(t')}^{l} \times \boldsymbol{W}_v \widehat{\boldsymbol{h}}_{j(t)}^{l-1}\right) \\
e_{j(t)\to i(t')}^{l} &= w_{j(t)\to i(t')} \times \alpha_{j(t)\to i(t')}^{l} \\
\alpha_{j(t)\to i(t')}^{l} &= \left(\boldsymbol{W}_k \widehat{\boldsymbol{h}}_{j(t)}^{l-1}\right)^T \left(\boldsymbol{W}_q \boldsymbol{h}_{i(t')}^{l-1}\right) \cdot \frac{1}{\sqrt{d}} \\
\widehat{\boldsymbol{h}}_{j(t)}^{l-1} &= \boldsymbol{h}_{j(t)}^{l-1} + \mathrm{TE}\left(t-t'\right) \\
\mathrm{TE}(\Delta t)_{2i} &= \sin\left(\frac{\Delta t}{10000^{2i/d}}\right), \mathrm{TE}(\Delta t)_{2i+1} = \cos\left(\frac{\Delta t}{10000^{2i/d}}\right)
\end{aligned}
\tag{5}
$$

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

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

$$
\boldsymbol{u}_i = \frac{1}{N}\sum_{t=1}^{T}(\boldsymbol{a}_i^T \widehat{\boldsymbol{h}}_{i(t)} \widehat{\boldsymbol{h}}_{i(t)}), \boldsymbol{a}_i = \tanh\left(\left(\frac{1}{N}\sum_{t=1}^{T}\widehat{\boldsymbol{h}}_{i(t)}\right)\boldsymbol{W}_a\right) \tag{6}
$$

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

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

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

C.2 Details of GraphODE

Specifically, the co-evolution of nodes and edges is depicted in Eqn 7. $\widetilde{\mathbf{W}}_A^t = \mathbf{D}^{-1}\mathbf{W}_A^t$ is the normalized adjacency matrix and $\mathbf{D}$ is the diagonal degree matrix defined as $\mathbf{D}_{ii} = \sum_j \mathbf{W}_{Aij}^t$. $f_e, f_{\text{self}}, f_{\text{edge2value}}$ are all implemented as Multi-Layer Perceptrons (MLPs). To incorporate the treatment effect into the function, we use a linear transformation $\mathbf{W}$ to merge the latent states of nodes $\mathbf{Z}^t$ and the treatment representation $\boldsymbol{O}^t$. In this way, the latent trajectories of agents are affected not only by their own past trajectories and treatments but also by the trajectories and treatments of their interacting agents.

$$
\begin{gathered}
\frac{\mathrm{d}\mathbf{Z}^t}{\mathrm{d}t} = \sigma\left(\widetilde{\mathbf{W}}_A^t \mathbf{W}[\mathbf{Z}^t, \mathbf{O}^t]\right) - \mathbf{Z}^t + \mathbf{Z}^0, \quad \frac{\mathrm{d}\boldsymbol{z}_{i\to j}^t}{\mathrm{d}t} = f_e\left([\boldsymbol{z}_i^t, \boldsymbol{z}_j^t]\right) + f_{\text{self}}\left(\boldsymbol{z}_{i\to j}^t\right), \\
\mathbf{W}_{Aij}^t = f_{\text{edge2value}}\left(\boldsymbol{z}_{i\to j}^t\right), \quad \widetilde{\mathbf{W}}_A^t = \mathbf{D}^{-1}\mathbf{W}_A^t.
\end{gathered}
\tag{7}
$$

Finally, we compute the predicted trajectories for each agent and their interactions based on the decoding likelihoods in Eqn (8), where f_{decN} and f_{decE} are node and edge decoding functions respectively, both implemented as MLPs. They output the means of the normal distributions $p(\boldsymbol{y}_i^t|\boldsymbol{z}_i^t)$ and $p(\boldsymbol{w}_{i\to j}^t|\boldsymbol{z}_i^t)$, which we treat as the predicted values from our model.

$$
\boldsymbol{y}_i^t \sim p(\boldsymbol{y}_i^t|\boldsymbol{z}_i^t) = f_{\text{decN}}(\boldsymbol{z}_i^t), \quad \boldsymbol{w}_{i\to j}^t \sim p(\boldsymbol{w}_{i\to j}^t|\boldsymbol{z}_i^t) = f_{\text{decE}}(\boldsymbol{z}_i^t). \tag{8}
$$

C.3 Details of Domain Adversarial Learning

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

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

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

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

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

C.4 ODE Solver

We use the fourth order Runge-Kutta method from the torchdiffeq python package [6] as the ODE solver, for solving the ODE systems on a time grid that is five times denser than the observed time points. We also utilize the Adjoint method described in [6] to reduce memory use.

C.5 Decoders

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

D Dataset Details

D.1 COVID-19 Dataset

Our experiments used the dataset provided by the Johns Hopkins Coronavirus Resource Center (JHU) [3] from April 12th to December 31st, 2020. That is, we consider 264 time points, with each point representing one day. The dataset contains a comprehensive range of information, but for our experiments, we focus on up to 7 specific features. These features include the daily counts of confirmed cases, deaths, recovered cases, active cases, incident rate (cases per 100,000 people), mortality rate (calculated as the number of recorded deaths multiplied by 100 divided by the number of cases), and testing rate (total test results per 100,000 people). It's worth noting that while the JHU dataset provides cumulative data for confirmed, deaths, recovered, and active cases, our experiments and models specifically consider the daily increases in these features (e.g., the number of new cases reported each day).

To capture dynamic interaction edges, we use a temporal mobility flow network among a selection of 47 states based on COVID-19 USFlows [18].

[3] https://coronavirus.jhu.edu/about/how-to-use-our-data

The treatments are represented as statewide policies that aimed to combat the spread of COVID-19. We identify 58 different statewide policies enacted throughout 2020, from the data given by the Department of Health & Human Services [4]. Each state enacted around 20 of these policies during the time period of April 2020 to December 2020, where the dataset provides the start and end dates of each enacted policy. In our model, treatments are encoded such that for each time point, the value is either 1 or 0 depending on whether the particular policy is enacted (for a given state) at that time or not, respectively.

Overall, the model receives input data for a total of 264 time points, covering each of the 47 states, and includes 7 features. Alongside this, the model is also provided with treatments and a mobility graph. Prior to being used as input for our models, the data is normalized. However, when calculating the test loss for comparison with other baseline models, the output is unnormalized. Our goal is to predict either the number of confirmed cases or the number of deaths for a future period of 7, 14, or 21 days.

D.2 Tumor Growth Dataset

We extend the state-of-the-art pharmacokinetic-pharmacodynamic (PK-PD) model of tumor growth proposed by [8] to simulate a more complex scenario where multiple tumor regions within a single patient interact with each other. The original model characterizes patients suffering from non-small cell lung cancer and models the evolution of their tumor under the combined effects of chemotherapy and radiotherapy. For a detailed description of the original model, we refer the readers to the original paper [8]. In our extended model, we incorporate two new terms: an interference term and a neighborhood covariate term. The volume of the tumor in region i at $t+1$ days after diagnosis is modeled as follows:

$$V_i(t+1) = \left(\underbrace{1 + \rho \log\left(\frac{K}{V_i(t)}\right)}_{\text{Tumor Growth}} - \underbrace{\beta_c C_i(t)}_{\text{Chemotherapy}} - \underbrace{\left(\alpha_r d_i(t) + \beta_r d_i(t)^2\right)}_{\text{Radiotherapy}} + \underbrace{e_{it}}_{\text{Noise}} \right) V_i(t)$$
$$+ \underbrace{\left(\frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \iota_c C_j(t) \right)}_{\text{Chemotherapy Interference}} + \underbrace{\left(\frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \left(\iota_r d_j(t) + \iota_r d_j(t)^2\right) \right)}_{\text{Radiotherapy Interference}} + \underbrace{\left(\frac{1}{N_i} \sum_{j \in \mathcal{N}_i} \kappa V_j(t) \right)}_{\text{Neighborhood Covariates}}$$

where the parameters K, ρ, β_c, α_r, β_r are sampled from the prior distributions described in [8], and $e_{it} \sim N(0, 0.012)$ is a noise term that accounts for randomness in the tumor growth. The prior means for β_c and α_r are adjusted to create three patient subgroups $S(i) \in \{1, 2, 3\}$ as described in [4]. The chemotherapy drug concentration follows an exponential decay with a half life of 1 day. The time-varying confounding is introduced by modeling chemotherapy and radiotherapy assignment as Bernoulli random variables, with probabilities p_c and p_r depending on the tumor diameter. For more details, we refer the reader to the paper [4]. For our newly defined interference and neighborhood covariate terms, we set the hyperparameters ι_c and ι_r to 0.01, and κ to 0.001. These values were carefully chosen to reflect the strength of the interference and neighborhood covariates in the dataset. The number of tumors in each patient N_i is fixed to 15, and for each tumor region, the number of edges connected between the tumor regions is defined randomly from the range of 22 to 45. For additional experiments shown in Appendix F.4, the number of tumor region for each patient is fixed to 5, and the number of edges ranges from 6 to 10. The dataset is input into our model similar to the COVID-19 dataset. Both chemotherapy and radiotherapy are encoded into 0 or 1 value depending on whether it was applied at a specific time point. The input data consists of 60 time points with 4 features, which include tumor volume, patient type, and the two treatments. The data is normalized for model input but unnormalized for test loss calculation. The model's objective is to predict tumor volume for future periods of 7, 14, or 21 days. We also create a longer range dataset with 120 time points, which is described in F.4.

[4]https://catalog.data.gov/dataset/covid-19-state-and-county-policy-orders-9408a

E Training Details

Our model implementation is available at `https://anonymous.4open.science/r/CAG-ODE-repo`. We employ the AdamW optimizer, as proposed in the study by Loshchilov et al. [23], to train our model. The initial learning rate is set at $\eta = 0.005$, and the batch size is set as 8 to accommodate memory constraints.

The Graph Neural Network (GNN) used for the encoder has a singular layer with a hidden dimension of 64. Similarly, the GNN that parameterizes the ODE function is also comprised of a single layer. The dimension of the latent state is set at 20, and the dimension for the embedded treatments is 5.

We assign a weight of 10 for both the treatment balancing term α and the interference balancing term β. Additionally, the weight designated for the edge reconstruction error λ is set at 0.5.

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

We train our model in a sequence-to-sequence setting, where we split the trajectory of each training sample into two parts $[t_1, t_K]$ and $[t_{K+1}, t_T]$. We condition the model on the first part of observations and predict the second part. To fully utilize the data points within each trajectory, we generate training and validation samples by splitting each trajectory into several chunks using a sliding window with three hyperparameters: the observation length and prediction length for each sample, and the interval between two consecutive chunks (samples). We summarize the procedure in Algorithm 1, where K is the number of trajectories and d is the input feature dimension. For both datasets, we set the observation length to be 7 and the interval to be 3. We ask the model to make predictions at varying lengths for evaluation.

Algorithm 1: Data Splitting Procedure.

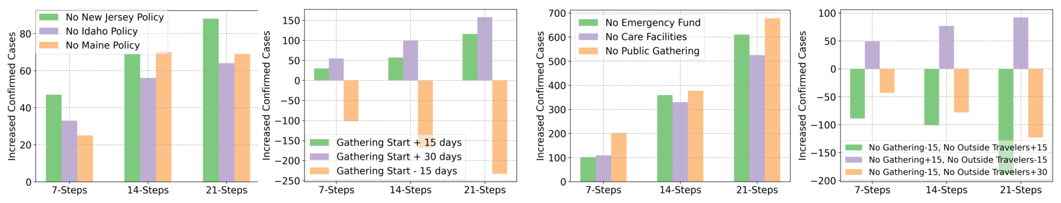
```
Input: Original Training trajectories $X_{\text{input}} \in \mathbb{R}^{K \times N \times T \times d}$;
Observation length $O$; Prediction length $M$; Interval $I$; Trajectory length $T$.
Output: Training samples after splitting $X_{\text{train}}$.
1 sample_length = $O + M$;
2 num_chunk = ($T$ - sample_length )//interval + 1;
3 for i in range (0,K) do
4     for j in range(0,num_chunk,I) do
5         Generate the split training sample as $X_{\text{input}}[i, :, j : j + \text{sample\_length}, :]$
6         Add the training sample to the training set $X_{\text{train}}$.
7     end
8 end
```

F Additional Experiment Results

F.1 Case Study about COVID-19 policies

We conduct a case study to show the impact of different treatments, e.g., COVID-19 related policies, on the COVID-19 dataset as shown in Figure 3. Specifically, we consider four different policy intervention methods and report the resulting average changes in the number of daily confirmed cases across all states in the U.S.:

First, we focus on the removal of policies in three states that have the highest number of announced policies during the time frame of the COVID-19 dataset. By masking out these policies, we observe an increase in the average number of confirmed cases across states in the future. This increase is attributed to both in-state disease spread and population flow to other states. The removal of policies exacerbates the spread of COVID-19 over an extended period, as shown in Figure 3(a), indicating that our model captures the dynamic interference resulting from agents' interactions.



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

Figure 3: Case Study for changing different policies on the COVID-19 dataset

We then explore the effect of changing the starting time of a specific policy for all states. We change the "No Public Gatherings" policy starting time for each state to be 15 days earlier, 15 and 30 days later respectively. As shown in Figure 3(b) when announcing the policy earlier, we observe a decrease in the average number of daily confirmed cases in the future, while announcing the policy later leads to an increase. This intuitive outcome highlights the capability of our model in capturing the causal relationships between policy interventions and COVID-19 spread.

Next, we analyze the impact of the top three most frequent policies across all states by removing them separately. As shown in Figure 3(c), the "Public Gatherings" policy has the largest effect in reducing the spread of COVID-19, even though the most frequent policy is "Emergency Funds". This demonstrates the potential of our model in assisting policymakers to identify the relative importance of each policy over time.

Finally, we study the effects of different orders in policy announcements, specifically focusing on the simultaneous or closely timed announcements of "No Public Gatherings" and "No Traveler from Outside States" policies. We change the announcement dates for the two policies in each state to mimic three scenarios shown in Figure 3 (d). We found that initializing the announcement of "No Public Gatherings" early generally contributes to a reduction in the spread of COVID-19 compared with "No Traveler from Outside States". We further analyze the daily population flow during the given time frame, and found that the majority of population flows are within the same states, indicating that residents of each state pose a high risk of virus transmission compared to people from other states. These insights suggest prioritizing the earlier announcement of the "No Public Gatherings" policy over the "No Traveler from Outside States" policy can better mitigate the spread of COVID-19.

These case study results demonstrate the effectiveness of our model CAG-ODE in analyzing the impact of various treatments or policies on the dynamics of COVID-19 spread. CAG-ODE captures the complex interactions between treatments, disease spread, and population flow, providing valuable insights for policymakers in making informed decisions.

F.2 Ablation Study

To further analyze the performance of our model, we also compare variants of our model. Each variant excludes a specific component to assess its individual impact on performance. The variants include models without treatment balancing, interference balancing, both components or the attention module. As shown in Table 2, all model variants perform worse than CAG-ODE, indicating the rationality of our model design.

F.3 Counterfactual Outcome Predictions

In Table 3, we evaluate the performance when 25%, 50%, and 75% of all observed treatments in each experiment are randomly flipped. The purpose of this experiment is to examine the robustness of the models to counterfactual treatment scenarios, and since CG-ODE does not incorporate causal modeling, it is excluded in this experiment. CAG-ODE outperforms others by a wide margin across all settings. These findings collectively demonstrate the superiority of our proposed model, CAG-ODE, in capturing the dynamics of multi-agent systems and making accurate predictions across different time horizons. We additionally include the visualization of the learned balanced latent representations in Appendix F.6.

Dataset	Covid-19			Tumor Growth		
Prediction Length	7-days	14-days	21-days	14-days	21-days	28-days
CG-ODE	4063	4454	4659	18.37	21.00	24.58
TE-CDE	7999	7470	6832	55.45	55.38	71.23
COVID-POLICY	3973	4268	3944	28.99	42.81	32.91
CAG-ODE	**3710**	**3925**	**3933**	**10.91**	**10.82**	**14.84**
w/o $L^{\langle G\rangle}$	3800	3987	3990	15.57	16.28	16.62
w/o $L^{\langle A\rangle}$	3840	4100	4069	17.90	14.69	20.19
w/o $L^{\langle G\rangle}$,$L^{\langle A\rangle}$	3793	4089	3953	17.28	16.72	24.36
w/o attention	3867	3958	4256	18.91	17.55	34.45

Table 2: Root Mean Square Error (RMSE) for factual outcome evaluation across different prediction lengths(the duration for which predictions are made). For the COVID-19 dataset, we report the mean accuracy with multiple runs.

Prediction Length	14-days			21-days			28-days		
Treatment F.R.	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
TE-CDE	95.61	103.2	100.8	98.65	103.0	97.93	118.3	124.0	121.4
COVID-POLICY	22.99	23.00	22.97	31.85	31.66	32.11	25.46	26.09	26.33
CAG-ODE	**17.23**	**16.98**	**16.96**	**18.64**	**18.84**	**18.85**	**19.91**	**19.88**	**19.87**
w/o $L^{\langle G\rangle}$	20.62	20.53	20.51	19.70	19.60	19.55	21.10	21.41	21.38
w/o $L^{\langle A\rangle}$	22.17	22.35	22.35	20.19	20.10	20.09	20.83	21.14	21.15
w/o $L^{\langle G\rangle}$, $L^{\langle A\rangle}$	19.78	19.75	19.71	19.34	19.29	19.27	21.31	21.40	21.34
w/o attention	19.09	18.37	18.13	22.16	21.78	21.65	27.70	27.44	27.38

Table 3: Root Mean Square Error (RMSE) for counterfactual Outcome evaluation on the Tumor Growth dataset with treatment flipping ratio. Treatment F.R. (**Treatment F**lipping **R**atio) represents the ratio of treatments that are flipped.

F.4 Longer-range Prediction for the Tumor Growth Dataset

In Table 4, we evaluate the performance of our model. An extended version of the simulation dataset with a range of 120 days was used in the experiment. Considered prediction lengths are 35, 49, and 63 days.

Prediction Length	35-days			49-days			63-days		
Treatment F.R.	0.25	0.5	0.75	0.25	0.5	0.75	0.25	0.5	0.75
CAG-ODE	35.00	34.68	34.50	34.12	37.92	37.88	46.71	47.28	48.12

Table 4: Root Mean Square Error (RMSE) for counterfactual outcome evaluation on the longer-range Tumor Growth dataset with treatment flipping ratio: $25\%, 50\%, 75\%$.

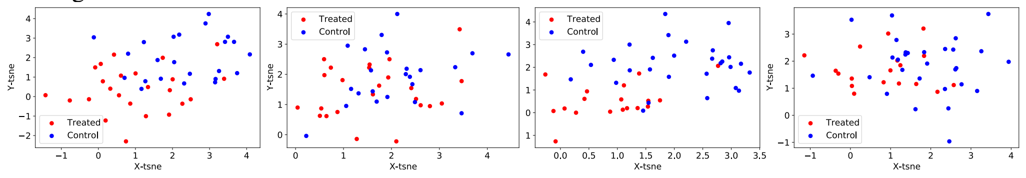
As anticipated, the prediction errors exhibit a moderate increase with longer prediction lengths, as the added duration poses a greater challenge for accurate predictions. Note that the prediction error remains relatively stable when the treatment flipping ratio is increased from 25% to 75%. This observation suggests that the utilization of treatment balancing and interference balancing techniques effectively mitigates the risk of overfitting to confounding factors, ensuring CAG-ODE's robustness.

F.5 Additional Experiment on Tumor Growth Dataset

In order to address the limitations of available datasets and to further validate the robustness of our model, we generated an additional synthetic dataset that models tumor growth by changing the configurations. Compared to the tumor-growth dataset that was used in Table **??**, this new dataset has a lower number of 5 tumor regions per patient and includes a larger sample size of 300 patients.

F.6 Visualization of Learned Balanced Representations

To further understand the effect of treatment balancing loss in CAG-ODE, we visualize the 2-D T-SNE projections of the latent representations of nodes on the COVID-19 dataset, i.e. $\boldsymbol{z}_i^t$ as shown in Figure 4. Specifically, we visualize the latent node representations under two different treatments: "State-of-Emergency" and "No-Public-Gathering". Under each treatment (policy), we use different colors to denote whether a node receives such treatment (treated) or not (control). As shown in Figure 4(a) and (c), the distributions of the learned representations are more distinguishable between the two groups, compared with Figure 4(b) and (d) which have the treatment balancing loss. This indicates that CAG-ODE indeed learns balanced latent representations by employing the treatment balancing loss.



(a) "State-of-Emergency" w/o. Treatment Balancing (b) "State-of-Emergency" with Treatment Balancing (c) "No-Public-Gathering" w/o. Treatment Balancing (d) " No-Public-Gathering" with Treatment Balancing

Figure 4: Treatment Balancing Visualization on the COVID-19 Dataset.

G Limitations

One limitation of CAG-ODE is that when inferring the future trajectories of nodes, we simply assume that all nodes are connected and jointly infer such edge evolution. This would bring huge computational costs when generalized to large-scale dynamical systems. In the future, we will consider more efficient sampling methods to accelerate the edge inference procedure to scale up our model. Another line of future work would be how to model more complex multiple treatment effects, including competing, hierarchical relationships.

H Broader Impacts

Our work significantly enhances the performance of causal inference over multi-agent dynamical systems, which can potentially benefit a wide range of fields including public health, biology, physics, and robotics. Our work also advances the recent study of continuous graphODE for modeling multi-agent system dynamics, providing an efficient tool for further research on AI for science.