GOPHER: Categorical probabilistic forecasting with graph structure via local continuous-time dynamics

Ke Alexander Wang[*]
Stanford University
alxwang@cs.stanford.edu

Danielle Maddix
Amazon Research
dmmaddix@amazon.com

Yuyang Wang
Amazon Research
yuyawang@amazon.com

Abstract

We consider the problem of probabilistic forecasting over categories with graph structure, where the dynamics at a vertex depends on its local connectivity structure. We present GOPHER, a method that combines the inductive bias of graph neural networks with neural ODEs to capture the intrinsic local continuous-time dynamics of our probabilistic forecasts. We study the benefits of these two inductive biases by comparing against baseline models that help disentangle the benefits of each. We find that capturing the graph structure is crucial for accurate in-domain probabilistic predictions and more sample efficient models. Surprisingly, our experiments demonstrate that the continuous time evolution inductive bias brings little to no benefit despite reflecting the true probability dynamics.

1 Introduction

In categorical probabilistic forecasting, we seek to predict a discrete probability distribution $\mathbf{p}(t)$ at some instantaneous time t, based on observed time-stamped data [10]. Consider the example of forecasting the most likely locations of the next earthquake over a finite set of locations at t, given the history of earthquake times and locations. We can view locations as vertices V on a graph $G = (V, E)$ with edges E that represent adjacency. Specifically, the probability of an earthquake at node $v \in V$ in the near future is mostly influenced by the probability of earthquakes at nodes within its neighborhood. This type of graphical structure also appears in other problems, including traffic forecasting [28], information diffusion in social networks [1], epidemic diffusion [24, 13], urban conflict patterns [16], and is an example of a marked temporal point process [6].

In this paper, we consider categorical probabilistic forecasts where there is a graphical structure to inform us of the local dynamics governing $\mathbf{p}(t)$ over time. We formalize the intuition that each component of the probability vector $\mathbf{p}(t) \in \mathbb{R}^{|V|}$ obeys local dynamics using the differential equation

$$\frac{\mathrm{d}p_v}{\mathrm{d}t} = g(p_v, \{p_u \mid u \in N(v)\}, t), \tag{1}$$

which we use to inform our model's inductive bias. Here, g governs the local dynamics, $\mathcal{N}(v) \subseteq V$ denotes the set of neighboring nodes of v, and p_v denotes the probability at node v. To capture the equivariant local dynamics of our forecast $\mathbf{p}(t)$, we propose GOPHER, a model that learns a neural ODE [4] with graph neural network (GNN) [26] dynamics.

Our method GOPHER introduces two inductive biases to aid with probabilistic forecasting over graph-structured categories by 1) utilizing graph structure explicitly and 2) introducing temporal evolution through a neural ODE. To disentangle the benefits of these two biases, we introduce two baseline models, ablating each bias. We find that utilizing the known graph structure results is key, and results in 10x improvements in accuracy and sample efficiency. On the other hand, explicitly modelling the temporal dynamics surprisingly results in little benefits.

[*]Work done as an intern at Amazon Research

2 Gopher: Forecasting with temporal dynamics and graph structure

Let $G = (V, E)$ be a graph, and let $t_i \in \mathbb{R}_0^+$ denote the timestamp of an event at node $v_i \in V$. Given G and an irregularly sampled dataset $\mathcal{D} = \{(t_i, v_i)\}_{i=1}^N$, we want to learn the probability vector $\mathbf{p}(t) \in \mathbb{R}^{|V|}$ of each $v \in V$ at any time t. We wish to model the dynamics of $\mathbf{p}(t)$ such that the change in the probability at node v depends only on the neighborhood $\mathcal{N}(v)$ around v, as described in Equation 1. However, directly parameterizing g from Equation 1 with a neural ODE can violate conservation of probability: $\mathbf{1}^\top \mathbf{p}(t) = 1$.

Instead of explicitly enforcing the sum constraint into our neural ODE, we model the dynamics in a continuous-time embedding space from which we derive the dynamics $\mathrm{d}\mathbf{p}/\mathrm{d}t$. Specifically, let $\mathbf{Z}_0 \in \mathbb{R}^{|V| \times D}$, where D denotes the embedding space dimension. We use $\mathbf{z}_{0,i}$ to denote row i of $\mathbf{Z}_0$ at initial time t_0, corresponding to the embedding of node v_i. We then model the dynamics of the continuous-time embeddings $\mathbf{Z}(t) \in \mathbb{R}^{|V| \times D}$ via

$$\frac{\mathrm{d}\mathbf{Z}}{\mathrm{d}t} = g(\mathbf{Z}, G, t) \quad \text{subject to} \quad \mathbf{Z}(t_0) = \mathbf{Z}_0, \tag{2}$$

where g is the learned graph neural network (GNN) dynamics. To map $\mathbf{Z}(t)$ to a probability space while preserving equivariance, we learn a shared projection $\pi : \mathbb{R}^D \to \mathbb{R}$ such that

$$\mathbf{p}(t) = \mathrm{Softmax}\big(\pi(\mathbf{z}(t)_1), \ldots, \pi(\mathbf{z}(t)_{|V|})\big). \tag{3}$$

Provided that g and π are differentiable, which can be done using smooth activation functions, our model then implicitly models the local temporal dynamic of our problem in Equation 1. Finally, we train our model Gopher by maximizing the log likelihood $\sum_{i=1}^N \log p_{v_i}(t_i)$ with respect to the parameters of π, g, and the initial condition $\mathbf{Z}_0$.

Incorporating node attributes. In some cases G may have node attributes $\{a_v\}_{v \in V}$ for each node $v \in V$ that affect the interaction dynamics, such as the geographical coordinates of each node in a spatial graph or the demographics of a user in a social network. Node attributes can be easily incorporated by letting the initial node embeddings be a learned function of the attributes, $\mathbf{Z}_0[v] = \psi_v(a_v)$, and optimizing with respect to the parameters of $\{\psi_v\}$.

Related works. Our paper lies at the intersection of probabilistic forecasting, neural ODEs, and graph neural networks (GNNS), and can be seen as the discrete analogue of continuous normalizing flows [4, 11, 5] on manifolds [17, 18]. Probabilistic forecasting seeks to predict a full distribution at each time step [14, 10], with contemporary methods often relying on deep probabilistic models [22, 25, 21, 20]. A direct application of categorical probabilistic forecasts is marked temporal point processes, which learn the rate of an event type v at time t, summarized by the conditional intensity function $\lambda(t, v) = \lambda(t) \cdot p_v(t)$ [6]. The inductive bias of a learnable ODE with GNN dynamics has also been explored in the context of other problems, including graph generation [7], node classification [19, 2], multi-particle trajectory prediction [19], learning partial differential equations [15], and knowledge graph forecasting [12].

3 Results: I Can't Believe Temporal Dynamics Don't Matter!

Synthetic datasets. We apply our method to model the mark component of a marked temporal point process (TPP) occurring on the nodes of a graph such that $p_v(t)$ is the probability of an event occurring on vertex v at time t. We create a synthetic dataset where events occur over time on a directed graph G, with node probabilities that obey graph advection as an example of local dynamics [3]. Graph advection conserves the total probability by ensuring $\mathbf{1}^\top \mathrm{d}\mathbf{p}/\mathrm{d}t = 0$. We represent the graph G by the weighted adjacency matrix A, where $A_{uv} > 0$ for $(u, v) \in E$. We sample sequences of events over time $[0, T]$ from a homogeneous Poisson process with constant temporal intensity $\lambda(t) = \lambda = 2.5$ and temporal node probability $\mathbf{p}(t) \in \mathbb{R}^{|V|}$ governed by the graph advection equation [3]

$$\frac{\mathrm{d}\mathbf{p}}{\mathrm{d}t} = -L_{\text{out}}(A)^\top \mathbf{p} \iff \frac{\mathrm{d}p_v}{\mathrm{d}t} = \sum_{v:\ (v,u) \in E} A_{vu} p_v - \sum_{v:\ (u,v) \in E} A_{uv} p_u. \tag{4}$$

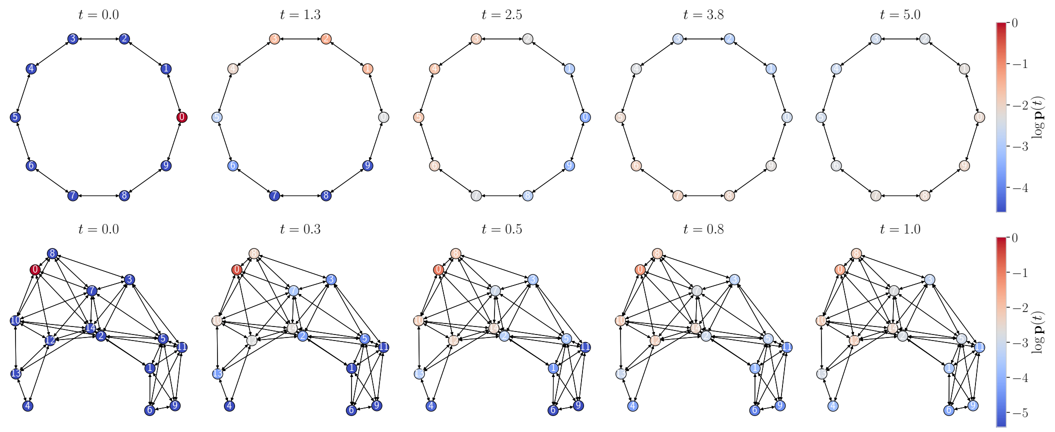
Figure 1: (Top) Advection on a cyclic graph set chosen so that the probability mass is transported more strongly in the counter-clockwise direction. (Bottom) Advection on a random geometric graph with randomly chosen edge weights. (Both) Colors are shown in log-scale to make dynamics more visually-apparent. Light-gray coloring corresponds to $\log(1/|V|)$ which is the steady-state probability mass for each node.

Here, $L_{\text{out}}(A) := D_{\text{out}}(A) - A$ denotes the out-degree graph Laplacian, and $D_{\text{out}}(G)$ denotes the diagonal out-degree matrix with $D_{\text{out}}(G)_{ii} = \sum_j A_{ij}$.

We create two graphs structures for our synthetic datasets, a ring graph and a random geometric graph, and visualize their advection on the graph over time in Figure 1; see Appendix D for more details on their construction. We also visualize the advection dynamics of each component of $\mathbf{p}(t)$ for the ring graph in Figure 9 of Appendix D. We use $T = 5$ seconds for the ring graph dataset and $T = 1$ second for the geometric graph dataset. Since the timestamps are sampled from a Poisson process and *not* equidistantly spaced over $[0, T]$, the continuous time aspect of the problem is clearly evident in the dataset.

Evaluating each inductive bias of GOPHER. We evaluate the accuracy and sample-efficiency improvements from incorporating graph structure and modelling temporal dynamics in GOPHER. To disentangle the effects of these two inductive biases, we compare our model to two baseline models. The first is a two-layer MLP that acts on node-embeddings concatenated with time, which has none of the above inductive biases. The second is a single-layer GNN that also acts on node-embeddings concatenated with time. The GNN incorporates the explicit graph structure, but does not incorporate dynamical systems structure. We refer to the models as NAIVEMLP and NAIVEGNN respectively. In our experiments, GOPHER learns g using a Graph Isomorphism Network (GIN) layer [27] parameterized by a two-layer MLP; we use another two-layer MLP for the projection π. NAIVEGNN uses the same GIN architecture and projection except that it does not learn a differential equation. Finally, NAIVEMLP replaces the GIN layer with a two layer MLP. See Appendix D for further details on our experiment hyperparameters.

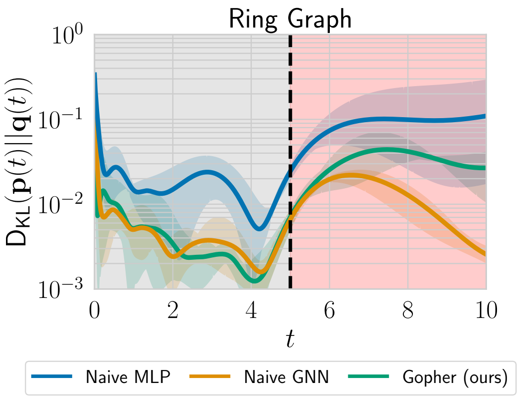

Figure 2: KL divergence between the true probabilities $\mathbf{p}(t)$ and the predicted probabilities $\mathbf{q}(t)$ trained on 1024 sampled sequences. Gray is the training set time interval $[0, T]$ and red is the extrapolation region $[T, 2T]$ beyond the training set.

Figure 2 shows the KL divergence betweeen the ground truth $\mathbf{p}(t)$ and the learned predictions over time for the ring graph. We summarize the KL divergence over $[0, T]$ in Figure 3 by the geometric mean since the error varies over multiple overs of magnitudes over time [9]. In both figures, we show the 95% confidence intervals over 3 seeds. For both datasets, there is 10x difference in accuracy

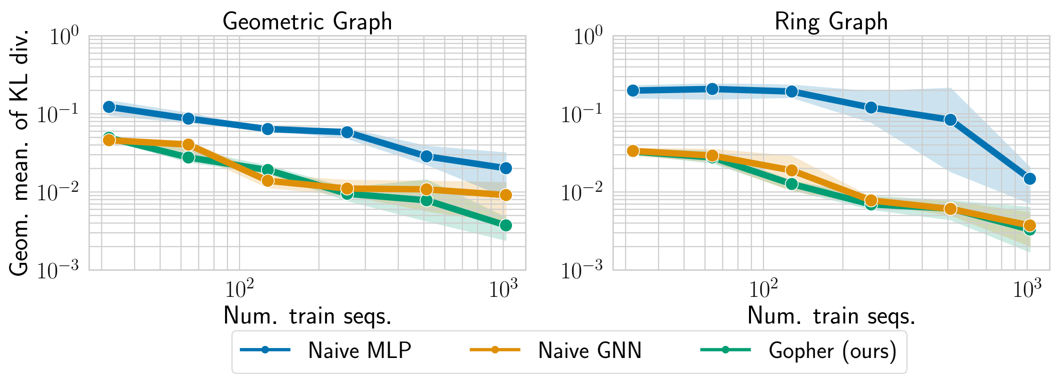

Figure 3: Sample complexity of each model on the geometric graph dataset (left) and cyclic graph dataset (right). The error is measured in terms of the geometric mean of the KL divergence over the evaluation time period $[0, T]$. Prediction of uniform probability corresponds to a geometric mean KL divergence of 0.72 for the geometric graph and 0.15 for the cyclic graph.

between the graph structured models and NAIVEMLP, indicating that utilizing the graph structure is greatly beneficial. Though NAIVEGNN does not explicitly model the local temporal dynamics of the datasets, it performs nearly identically to our model GOPHER in fitting $\mathbf{p}(t)$ over the training interval $[0, T]$. In principle, GOPHER has the best chance of extrapolating to the $[T, 2T]$ time period not seen during training since GOPHER explicitly models the local dynamics. However, GOPHER's poor extrapolation ability suggests that its learned dynamics do not actually reflect the true dynamics. Indeed, in Figure 7 of Appendix C we show that although GOPHER can fit the training data well, it is brittle to edge deletions, further indicating GOPHER does not learn the true dynamics.

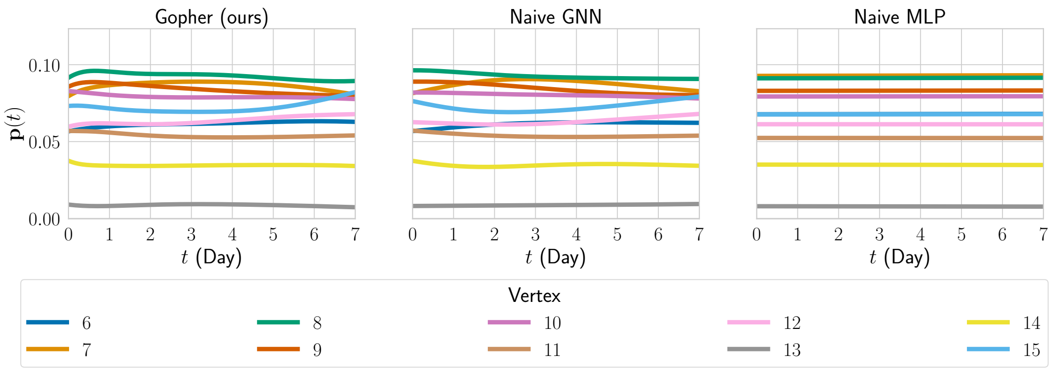

Figure 4: Learned $\mathbf{p}(t)$ over a subset of the counties fit using the COVID-19 dataset preprocessed by Chen et al. [5]. Only the graph-based models are able to capture the variations over time. See Appendix A for the empirical distribution of $\mathbf{p}(t)$.

Real-world dataset. We use data released publicly by the New York Times [23] on daily COVID-19 cases in New Jersey state to construct a real-world categorical probabilistic forecasting dataset, following the preprocessing script of Chen et al. [5]. We aggregate the cases by county and form a graph with 21 nodes where each node is a county and each edge is a county border. Using the train/test split from Chen et al. [5], we obtain per event log likelihoods with 1-standard-deviations of -2.766 ± 0.003 for NAIVEMLP, -2.768 ± 0.003 for NAIVEGNN, and -2.767 ± 0.000 for GOPHER over 3 seeds. However, these likelihoods are not representative of the model differences since we find a large distribution shift between the train and test distribution shown in Figure 5 of Appendix A. This distribution shift causes the models to perform equally poorly on the test set. In actuality, NAIVEMLP completely fails to capture variations in $\mathbf{p}(t)$ over time, as shown in Figure 4.

4 Discussion

Although the inductive biases of GOPHER, directly reflect properties of categorical forecasting with local continuous-time dynamics, our experiments find that, surprisingly, explicitly modelling the temporal dynamics does not improve performance. Most of the performance gains of GOPHER come from incorporating a graph structure, which can be done with a simple baseline model like NAIVEGNN. The failure of GOPHER can be attributed to the fact that the learned dynamics in the embedding space do not accurately reflect the ground truth dynamics in probability space.

References

[1] Eytan Bakshy, Itamar Rosenn, Cameron Marlow, and Lada Adamic. The role of social networks in information diffusion. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 519–528, New York, NY, USA, April 2012. Association for Computing Machinery. ISBN 978-1-4503-1229-5. doi: 10.1145/2187836.2187907.

[2] Ben Chamberlain, James Rowbottom, Maria I. Gorinova, Michael Bronstein, Stefan Webb, and Emanuele Rossi. GRAND: Graph Neural Diffusion. In *International Conference on Machine Learning*, pages 1407–1418. PMLR, July 2021.

[3] Airlie Chapman and Mehran Mesbahi. Advection on graphs. *IEEE Conference on Decision and Control and European Control Confereence (CDC-ECC)*, 50:1461–1466, December 2011.

[4] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[5] Ricky T. Q. Chen, Brandon Amos, and Maximilian Nickel. Neural Spatio-Temporal Point Processes. In *International Conference on Learning Representations*, September 2020.

[6] D. J. Daley and D. Vere-Jones. *An Introduction to the Theory of Point Processes: Volume I: Elementary Theory and Methods*. Probability and Its Applications, An Introduction to the Theory of Point Processes. Springer-Verlag, New York, second edition, 2003. ISBN 978-0-387-95541-4. doi: 10.1007/b97277.

[7] Zhiwei Deng, Megha Nawhal, Lili Meng, and Greg Mori. Continuous Graph Flow. *arXiv:1908.02436 [cs, stat]*, September 2019.

[8] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. Augmented Neural ODEs. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[9] Marc Finzi, Ke Alexander Wang, and Andrew G. Wilson. Simplifying Hamiltonian and Lagrangian Neural Networks via Explicit Constraints. *Advances in Neural Information Processing Systems*, 33:13880–13889, 2020.

[10] Tilmann Gneiting and Matthias Katzfuss. Probabilistic Forecasting. *Annual Review of Statistics and Its Application*, 1(1):125–151, 2014. doi: 10.1146/annurev-statistics-062713-085831.

[11] Will Grathwohl, Ricky T. Q. Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-Form Continuous Dynamics for Scalable Reversible Generative Models. In *International Conference on Learning Representations*, September 2018.

[12] Zhen Han, Zifeng Ding, Yunpu Ma, Yujia Gu, and Volker Tresp. Temporal Knowledge Graph Forecasting with Neural ODE. *arXiv:2101.05151 [cs]*, August 2021.

[13] Wenzhang Huang, Maoan Han, and Kaiyu Liu. Dynamics of an SIS reaction-diffusion epidemic model for disease transmission. *Mathematical Biosciences & Engineering*, 7(1):51, 2010. doi: 10.3934/mbe.2010.7.51.

[14] Robin John Hyndman and George Athanasopoulos. *Forecasting: Principles and Practice*. OTexts, Australia, 2nd edition, 2018.

[15] Valerii Iakovlev, Markus Heinonen, and Harri Lähdesmäki. Learning continuous-time PDEs from sparse data with graph neural networks. In *International Conference on Learning Representations*, September 2020.

[16] Scott Linderman and Ryan Adams. Discovering Latent Network Structure in Point Process Data. In *International Conference on Machine Learning*, pages 1413–1421. PMLR, June 2014.

[17] Aaron Lou, Derek Lim, Isay Katsman, Leo Huang, Qingxuan Jiang, Ser Nam Lim, and Christopher M De Sa. Neural manifold ordinary differential equations. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 17548–17558. Curran Associates, Inc., 2020.

[18] Emile Mathieu and Maximilian Nickel. Riemannian Continuous Normalizing Flows. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 2503–2515. Curran Associates, Inc., 2020.

[19] Michael Poli, Stefano Massaroli, Clayton M. Rabideau, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Continuous-Depth Neural Models for Dynamic Graph Prediction. *arXiv:2106.11581 [cs, stat]*, June 2021.

[20] Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep State Space Models for Time Series Forecasting. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[21] Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs M. Bergmann, and Roland Vollgraf. Multivariate Probabilistic Time Series Forecasting via Conditioned Normalizing Flows. In *International Conference on Learning Representations*, September 2020.

[22] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 36(3):1181–1191, July 2020. ISSN 0169-2070. doi: 10.1016/j.ijforecast.2019.07.001.

[23] The New York Times. Coronavirus (Covid-19) Data in the United States, 2021. URL https://github.com/nytimes/covid-19-data.

[24] Rui Wang, Danielle Maddix, Christos Faloutsos, Yuyang Wang, and Rose Yu. Bridging physics-based and data-driven modeling for learning dynamical systems. In Ali Jadbabaie, John Lygeros, George J. Pappas, Pablo A. Parrilo, Benjamin Recht, Claire J. Tomlin, and Melanie N. Zeilinger, editors, *Proceedings of the 3rd Conference on Learning for Dynamics and Control*, volume 144 of *Proceedings of Machine Learning Research*, pages 385–398. PMLR, 07 – 08 June 2021. URL https://proceedings.mlr.press/v144/wang21a.html.

[25] Yuyang Wang, Alex Smola, Danielle Maddix, Jan Gasthaus, Dean Foster, and Tim Januschowski. Deep Factors for Forecasting. In *International Conference on Machine Learning*, pages 6607–6617. PMLR, May 2019.

[26] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. A Comprehensive Survey on Graph Neural Networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, January 2021. ISSN 2162-237X, 2162-2388. doi: 10.1109/TNNLS.2020.2978386.

[27] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How Powerful are Graph Neural Networks? In *International Conference on Learning Representations*, September 2018.

[28] Bing Yu, Haoteng Yin, and Zhanxing Zhu. Spatio-temporal graph convolutional networks: A deep learning framework for traffic forecasting. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, pages 3634–3640, Stockholm, Sweden, July 2018. AAAI Press. ISBN 978-0-9992411-2-7.

A Distribution shift in our real-world dataset

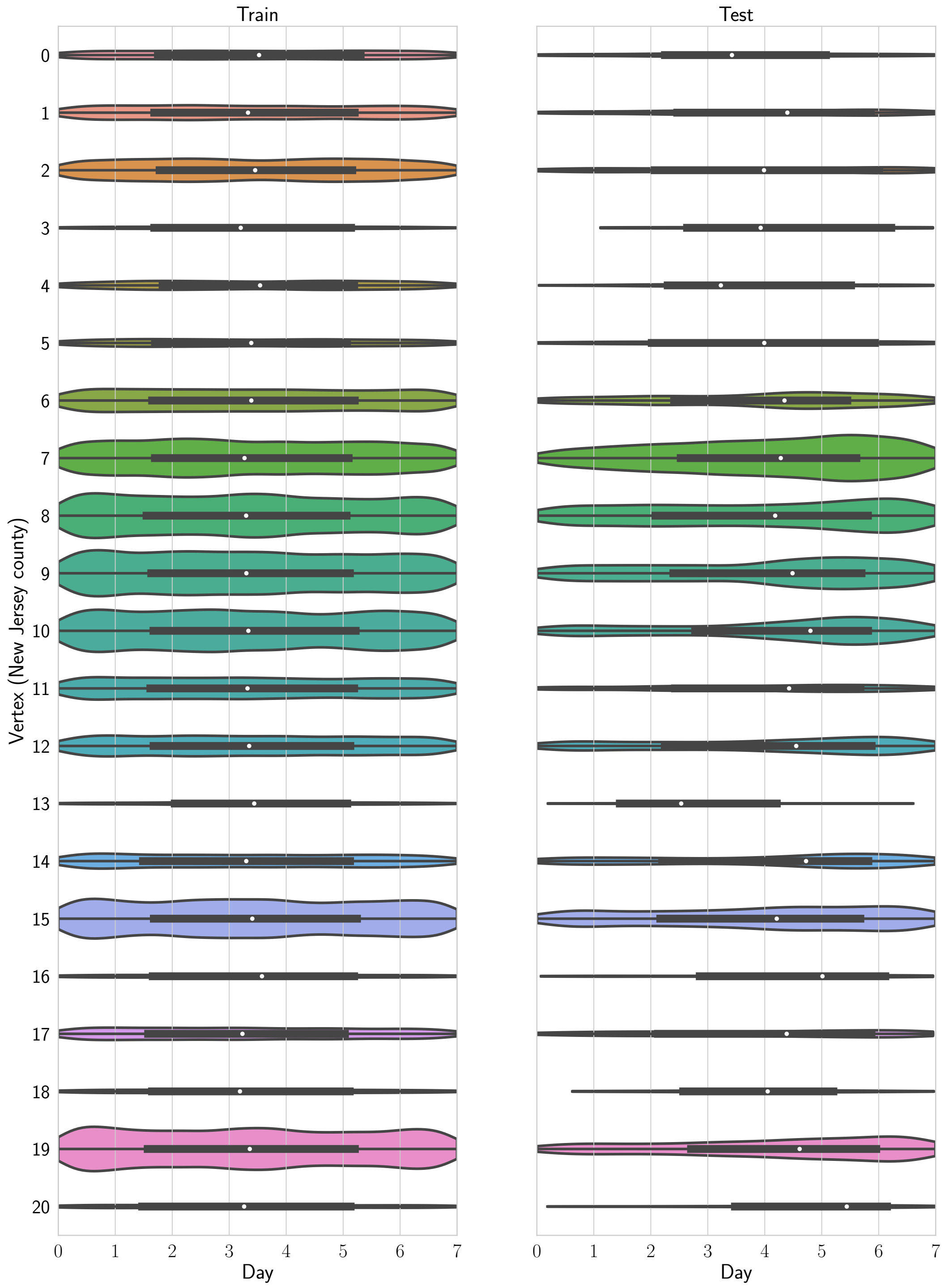

Figure 5: Distribution shift between the train and test distribution of the New Jersey COVID-19 cases created by Chen et al. [5] when binned by the 21 counties. For each of the two distributions, the height of each violin plots is normalized by the total count of observations in that split, i.e. size of training set or size of test set. For most vertices, there are fewer COVID cases later in the 7 day interval in the test set than in the training set.

B Learned forecasts on COVID-19 dataset

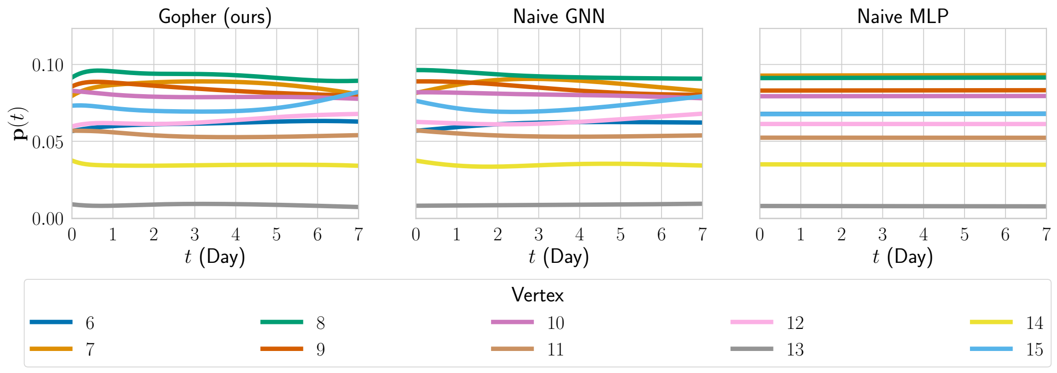

Figure 6: A copy of Figure 4 for easier comparison to the empirical distribution in Figure 5

C Model robustness

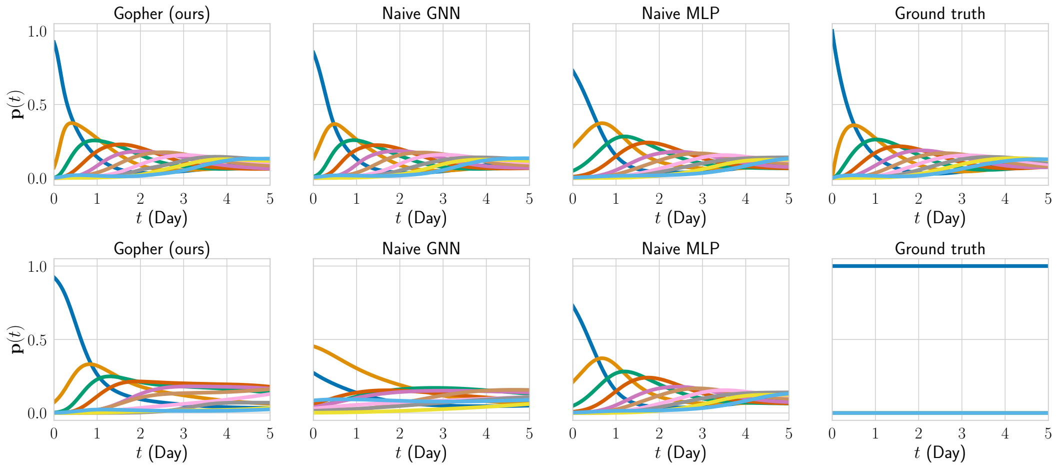

Figure 7: (Top) The learned and ground truth $\mathbf{p}(t)$ for the ring graph dataset. (Bottom) The predicted $\mathbf{p}(t)$ after we remove all edges from the ring graph. Notice that the ground truth of a completely disconnected graph is to have $\mathbf{p}(t) = \mathbf{p}(0)$ for all t. However, all of the models fail completely on this new disconnected graph, suggesting that they do not learn the true dependence of $\mathbf{p}(t)$ on the graph structure.

D Implementation details

We use 2 layer MLPs with 64 hidden units per layer whenever we use a MLP. We use Swish activations to ensure smoothness of our dynamics. We also use an augmented neural ODE [8], using 16 dimensions as augmented dimensions out of the 64 hidden dimensions.

D.1 Model architectures.

GOPHER. We parameterize the dynamics g from Equation 2 using one graph isomorophism network layer parameterized by a MLP. We also use a MLP to model the projection π.

NAIVEGNN. We use the same architecture as GOPHER, namely a GIN layer followed by a projection π. However, instead of using the model to parameterize ODE dynamics, we directly input the node embeddings concatenated with time t through the GNN.

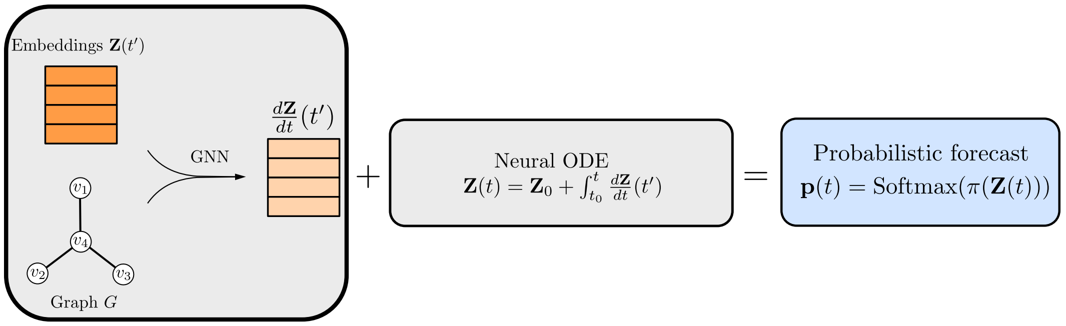
Embeddings $\mathbf{Z}(t')$
GNN
$\frac{d\mathbf{Z}}{dt}(t')$
v_1
v_4
v_2
v_3
Graph G
+
Neural ODE
$\mathbf{Z}(t) = \mathbf{Z}_0 + \int_{t_0}^{t} \frac{d\mathbf{Z}}{dt}(t')$
=
Probabilistic forecast
$\mathbf{p}(t) = \text{Softmax}(\pi(\mathbf{Z}(t)))$

Figure 8: Architecture of GOPHER.

NAIVEMLP. We replace the GIN layer of NAIVEGNN with a MLP, keeping all else the same.

D.2 Training procedure and dataset details.

To maximize hardware parallelism, we parallelize our neural ODE computation across sequence timesteps and across sequences using the time-reparameterization trick outlined in Chen et al. [5]. For the synthetic datasets, we use the AdamW optimizer with 0.01 learning rate and batch size 64 for 30 epochs. For the COVID-19 dataset, we use a 3×10^{-4} learning rate and batch size 4 for 15 epochs. Here, each batch consists of multiple sequences drawn from the training period $[0, T]$. We use $T = 5$ for the ring graph, $T = 1$ for the geometric graph, and $T = 8$ for the New Jersey counties graph. We generate the ring graph dataset by using hand-set coefficients for the edge weights A_{uv} to allow for counter-clockwise transport. We generate the geometric graph dataset by generating a random geometric graph via the `networkx` python package and drawing a random sample of $\{A_{uv}\}$.

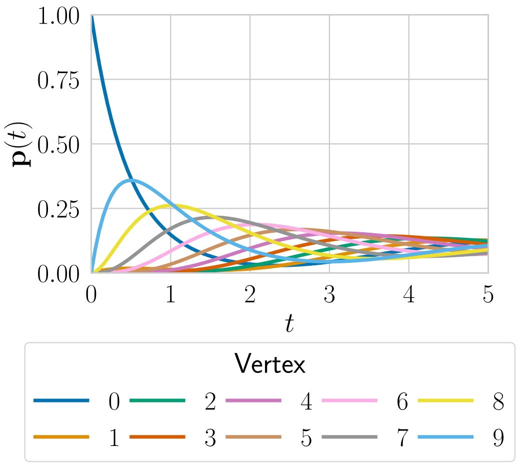

Figure 9: The dynamics of each component of $\mathbf{p}(t)$ on the cyclic graph. Each component corresponds to the probability of a vertex on the graph.