Spatial Shortcuts in Graph Neural Controlled Differential Equations

Michael Detzel[*]
Fraunhofer HHI

Gabriel Nobis
Fraunhofer HHI

Jackie Ma
Fraunhofer HHI

Wojciech Samek
Fraunhofer HHI and TU Berlin

Abstract

We incorporate prior graph topology information into a Neural Controlled Differential Equation (NCDE) to predict the future states of a dynamical system defined on a graph. The informed NCDE infers the future dynamics at the vertices of simulated advection data on graph edges with a known causal graph, observed only at vertices during training. We investigate different positions in the model architecture to inform the NCDE with graph information and identify an outer position between hidden state and control as theoretically and empirically favorable. Our such informed NCDE requires fewer parameters to reach a lower Mean Absolute Error (MAE) compared to previous methods that do not incorporate additional graph topology information.

1 Introduction

Effect follows cause. When a problem is represented on a graph, its structure contains information on how causes at one spatial position are linked to effects at another. To learn about physical dynamics from spatial time series data, one can often leverage this structural information. Schölkopf [1] describes differential equations as the gold standard for understanding cause-effect structures and highlight the lack of a time component in statistical machine learning methods. Neural Differential Equations (NDEs) [2] are able to learn a hidden state that evolves continuously in time, and could remedy this lack. With Neural Controlled Differential Equations (NCDE) [3], one can update the hidden state continuously with data incoming at different points in time. If one also wants to account for spatial dependencies, Graph Neural Controlled Differential Equations (GNCDEs) [4] can be used, where a node embedding is learned to capture these spatial dependencies. We incorporate prior known graph topology information into a GNCDE to infer the future dynamics at the vertices. We therefore generate data coming from a graph advection simulation from which we know the underlying graph topology plus the temporal cause and effect relation. We will outline the close connection between the graph information in the data generated and the artificial Neural Network (NN) architecture. Then we train our Informed GNCDEs and let them learn the dynamics to predict their future behavior. We start by describing advection on graphs and the theory before we explain the data generation and our Informed GNCDE. We believe this approach can lead to improvements in domains where graph information is available or where time series data is scarce or partially missing. NCDEs are an effective method in this context due to their ability to handle irregular time series data [5]. With graph information one has the potential to predict with fewer observations, as the graph structure itself does not have to be learned on top of the temporal dynamics. Promising domains for introducing known graph structure are traffic forecasting, river water level forecasting, climate and weather prediction, or disease spread (see A.4).

[*]Corresponding author michael.detzel@hhi.fraunhofer.de

D3S3: Data-driven and Differentiable Simulations, Surrogates, and Solvers @ NeurIPS 2024.

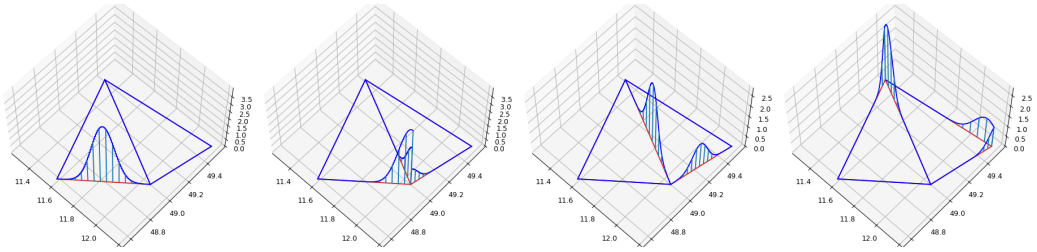

Figure 1: Advection of an initial Gaussian pulse on a graph with 5 edges over time

2 Related Work

Choi et al. [4] consider time series on a graph by using a graph embedding to capture the spatial dependencies, without directly incorporating prior graph knowledge for traffic forecasting. Spatial-Temporal Graph Neural Networks (GNNs) [6, 7] inform NNs with graph information, where Message Passing NNs [8] are combined with Recurrent Neural Networks (RNNs) like Long Short Term Memorys (LSTMs) [9] or Gated Recurrent Units (GRUs) [10]. Poli et al. [11] introduce spatial inductive bias as a node embedding into the vector field while generalizing the notion of Neural Ordinary Differential Equations (NODEs) to graphs. Other attempts of incorporating prior knowledge into NN architecture include Physics Informed Neural Networks (PINNS) [12] which enforce a prior defined Partial Differential Equation (PDE) but do not act on graphs as they avail oneself of a discretized continuous description of the spatial domain. Kosma et al. [13] use NODEs for epidemic spreading on graphs and Verma et al. [14] use NODEs for climate and weather forecasting in discretized continuous space.

3 Learning Advection with Graph Neural Controlled Differential Equations

Advection on Graph Edges. In contrast to Chapman and Mesbahi [15], where the quantities on the vertices are updated discretely in time, we advect quantities that move continuously in time along the edges of a graph and fork or merge at vertices. For the advection simulation, we consider a directed graph $\mathcal{G}$ with a set of vertices $\mathcal{V}$ and edges $\mathcal{E}$. Every edge $e \in \mathcal{E}$ has a time-static continuous domain $\Omega_e = \{x \in \mathbb{R} \mid 0 \leq x < \delta(e)\}$ and a function space $V_e = \{y_e(x,t) | \Omega(e) \times \mathcal{T} \to \mathbb{R}^d\}$ attached to it, where $\delta(e)$ is the length of edge e and $t \in \mathcal{T} = [0, T]$ is a point in time. We first focus on the description of the advection on the interior of the edges without considering transitions between edges: Given an initial state of the dynamic system it evolves according to an advection differential equation

$$\frac{\partial \mathbf{y}}{\partial t} + \nabla \cdot (\mathbf{y} \odot \mathbf{v}) = 0, \quad \mathbf{y}(\,\cdot\,, 0) = \mathbf{y}_0 \tag{1}$$

with $\mathbf{v} = (v_1, \ldots, v_{|\mathcal{E}|})^\intercal$ being the vector of scalar velocity fields $v_e : \Omega_e \to \mathbb{R}$, with $e \in \{1, \ldots, |\mathcal{E}|\}$, that advect the quantity $\mathbf{y} = (y_{e_1}, \ldots, y_{e_{|\mathcal{E}|}})^\intercal$ through space over time. The divergence is intended row-wise and $\odot$ signifies element-wise multiplication. We restrict the entries of $\mathbf{y}$ to positive values, so that the flow direction is always the direction of the directed edge. We assume that the velocity field is constant. In the case of a one-dimensional quantity and velocity field along the edges, this leads to $\frac{\partial y_e}{\partial t} = -v_e \frac{\partial y_e}{\partial x}$, which has the analytical solution $y_e(x,t) = y_e^0(x - vt)$, with initial condition $y_e^0 := y_e(x, 0)$, for all $x \in \Omega(e)$, for all $e \in \mathcal{E}$ and $v_e t \leq x \leq \delta(e)$. With this description of the dynamic, y_e for $e \in \mathcal{E}$ is not required to be differentiable. Now we describe the dynamic on the transition between adjacent edges. For $x < v_e t$ one would look back "beyond the edge". We therefore resort to a matrix formulation to represent the edge transitions of y:

$$\mathbf{y}(\mathbf{x}, t) = \big(\mathbf{A}_\mathcal{E} \mathbf{y}(\mathbf{x}, 0)\big)\Big|_{\mathbf{x} = (\delta(\mathbf{e}) - v_e t \ \mathbf{1}_{|\mathcal{E}|})}, \quad x < v_e t, \tag{2}$$

where $t < \min_{e \in \mathcal{E}} \delta(e)/v_e$ has to hold for every edge to not look further back than the previous edge and where $\mathbf{A}_\mathcal{E}$ is a directed "edge transition matrix", signifying to which edge a quantity transitions when the quantity reaches the end of an edge. $\mathbf{A}_\mathcal{E}$ acts as a function operator on the vector of functions $\mathbf{y}_0 = (y_1^0, \ldots, y_{|\mathcal{E}|}^0)^\intercal$, and $\mathbf{x} = (x_1, \ldots, x_{|\mathcal{E}|})^\intercal$ is a collection vector of spatial coordinates on the

different edges and $\mathbf{1} = (1, \ldots, 1)^\intercal \in \mathbb{R}^{|\mathcal{E}|}$. The entries a_{ij} of $\mathbf{A}_\mathcal{E}$ are defined as

$$a_{ij} = \begin{cases} p_{ij} \in (0,1], & \text{if } e_i \text{ follows on } e_j \\ 0, & \text{else} \end{cases},$$

with p_{ij} being the proportion of the quantity on e_i transported to e_j and $\sum_{i \in \mathcal{E}} p_{ij} = 1$ to enforce conservation. The matrix $\mathbf{A}_\mathcal{E}$ can be obtained from the the vertex adjacency matrix $\mathbf{A}_\mathcal{V}$ (details in A.3). For the numerical simulation the graph edges are spatially discretized into 100 segments s_k from which 50 are uniformly randomly selected without replacement to be initialized with discrete uniformly sampled values $\psi_{e,s_k} \sim \mathcal{U}\{0, 10\}, k \in \{1, 100\}$. We then iteratively advect with a recurrence equation derived from Equation (2) to derive

$$\mathbf{y}(\mathbf{x}, t + \Delta t) = \big(\mathbf{A}_\mathcal{E}(\mathbf{y}(\mathbf{x}, t))\big)\Big|_{\mathbf{x}=(\delta(\mathbf{e})-v\Delta t\ \mathbf{1}_{|\mathcal{E}|})}$$

and use $\Delta t = 5$ minutes to evolve for 48 steps into the future. To obtain data measurements at the nodes, the quantity that passed through the vertex in a given time span is aggregated. We take measurements for Δt and acquire time series on the graph nodes that are used for the NN training. As input data for the GNCDEs, we are facing a time-series of graphs $\{\mathcal{G}_{t_i} \triangleq (\mathcal{V}, \mathcal{E}, \mathbf{Y}_{\mathbf{t_i}})\}_{i=0}^N$, with $\mathbf{Y}_{\mathbf{t_i}}$ being the aggregated quantities at the vertices.

Incorporating topology knowledge into GNCDEs. Let us review the notion of NCDEs [3]: A NCDE is described by

$$\boldsymbol{z}(T) = \boldsymbol{z}(0) + \int_0^T f_{\boldsymbol{\theta}}\Big(\boldsymbol{z}(t)\Big) d\boldsymbol{x}(t),$$

where $T, t \in \mathbb{R}^+$, $\boldsymbol{z}(t) \in \mathbb{R}^{d_z}$, $f_{\boldsymbol{\theta}} : \mathbb{R}^{d_z} \to \mathbb{R}^{d_z} \times \mathbb{R}^{d_x}$ is a vector field modeled by an NN with trainable parameters $\boldsymbol{\theta} \in \mathbb{R}^{d_{\boldsymbol{\theta}}}$ and $\boldsymbol{x}(t) \in \mathbb{R}^{d_x}$ is the control path. This means that the hidden state $\boldsymbol{z}(t))$ is continuously updated over time t from 0 up to time T by the vector field $f_{\boldsymbol{\theta}}$ and the control $\mathbf{x}$. This formulation allows inputting data at arbitrary times t via $\mathbf{x}(t)$. The path of state $\boldsymbol{z}(t)$ encodes the information gathered from time 0 to T and is used to make the final predictions $\{\hat{\mathbf{Y}}_{t_i}\}_{i=1}^M$ with $T < t_i \leq T+\tau$ at time T for the next M time steps up to time $T + \tau$. One can solve and backpropagate through the NCDE given a loss function $\mathcal{L}(\{\hat{\mathbf{Y}}_{t_i}(\mathbf{x})\}_{i=1}^M, \{\mathbf{Y}_{t_i}\}_{i=1}^M)$. Jhin et al. [16] and Kidger [5, p. 66] showed that NCDEs can be coupled to use the encoded hidden path of a first NCDE as a control path input to a second NCDE. Firstly, analogously to Choi et al. [4], we update a d_h-dimensional hidden state $\boldsymbol{H}(t) \in \mathbb{R}^{|\mathcal{V}| \times d_h}$ continuously in time for every node separately to capture the temporal dependencies via

$$\boldsymbol{H}(T) = \boldsymbol{H}(0) + \int_0^T f_{\boldsymbol{\theta}}\Big(\boldsymbol{H}(t)\Big) \frac{d\boldsymbol{X}(t)}{dt} dt. \quad (3)$$

Secondly, we utilize $\boldsymbol{H}$ as control to drive another NCDE to continuously update the hidden state $\boldsymbol{Z}(T) \in \mathbb{R}^{|\mathcal{V}| \times d_z}$, with d_z being the dimension of the hidden state $\boldsymbol{Z}$ that contains now also spatial information via

$$\boldsymbol{Z}(T) = \boldsymbol{Z}(0) + \int_0^T g_{\boldsymbol{\gamma}}\Big(\boldsymbol{Z}(t)\Big) \frac{d\boldsymbol{H}(t)}{dt} dt, \quad (4)$$

where the vector field $g_{\boldsymbol{\gamma}}$ learns the spatial dependencies, making it a GNCDE. We differentiate between the internal and external mappings that the vector field $g_{\boldsymbol{\gamma}}$ describes:

$$g_{\boldsymbol{\gamma}}(\mathbf{Z}(t)) : \mathbb{R}^{d_z} \to \mathbb{R}^{d_z} \times \mathbb{R}^{d_h}, \quad (5)$$

$$g_{\boldsymbol{\gamma}}(\mathbf{Z}(t)) \bullet \cdot : \mathbb{R}^{d_h} \to \mathbb{R}^{d_z}, \quad (6)$$

using $\bullet$ to emphasize the tensor product. Whereas Equation (5) describes the inner workings of the update of the hidden state $\mathbf{Z}$ on itself, the tensor multiplication in Equation (6) describes the direct connection between control (and thereby the data) to the hidden state. Of course both happen simultaneously. The function in Equation (5) describes how a hidden state entry at a node has to encode "where to look" based on the entries of other hidden states and itself, to then "find" the information in the control. In contrast the function in Equation (6) describes the situation in which one "knows where to

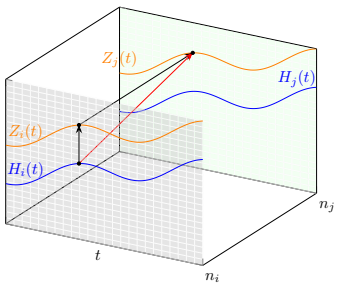

Figure 2: Difference in information transport between hidden state $\mathbf{H}_i$ at node n_i and hidden state $\mathbf{Z}_j$ at node n_j. Connection with $A_\mathcal{V}^{outer}$ leads to a direct connection (red) versus connecting nodes via $A_\mathcal{V}^{inner}$ (black)

	$\mathbf{A}_{\mathcal{V}^{outer}}, \lvert\mathcal{V}\rvert = 4$			$\mathbf{A}_{\mathcal{V}^{outer}}, \lvert\mathcal{V}\rvert = 10$		
$\mathbf{A}_{\mathcal{V}^{inner}}$	Identity MAE	Informed MAE	# params	Identity MAE	Informed MAE	# params
Identity	3.60	2.75	293,184	2.61	1.11	293,244
Informed	3.10	3.52	293,184	-	-	293,244
AGC	2.73	**2.67**	416,744	1.42	**0.66**	416,864

Table 1: MAE on unseen data and number of trainable parameters for different pairs for outer and inner graph mechanism for a **3-layer** architecture for the **4-node graph** and **10-node graph**

look" (at least spatially) to read out the right hidden state update information from the control which is given. In Figure 2 differences in the connections are schematically depicted. In our Informed GNCDE architectures, we use exactly this second notion and introduce a vertex transition matrix $\mathbf{A}_{\mathcal{V}}^{outer}$ into the equation to obtain (A.1 for dimensional details)

$$\boldsymbol{Z}(T) = \boldsymbol{Z}(0) + \int_0^T g_{\boldsymbol{\gamma}}\Big(\boldsymbol{Z}(t)\Big)\mathbf{A}_{\mathcal{V}}^{outer}\frac{d\boldsymbol{H}(t)}{dt}dt \tag{7}$$

with $\mathbf{A}_{\mathcal{V}}^{outer}$ corresponding to the edge transition matrix $\mathbf{A}_{\mathcal{E}}$. Alternatively, one can also introduce topology information into the GNCDE inside g_θ by writing $g_{\boldsymbol{\gamma}} = g_3 \circ \mathbf{A}_{\mathcal{V}}^{inner} g_2 \circ g_1(\boldsymbol{Z}(t))$, where $g_l,\ l \in \{1, 2, 3\}$ are the inner NN layers. Choi et al. [4] use a node embedding called Adaptive Graph Convolution (AGC) similar to the embedding used in Kipf and Welling [17] inside $g_{\boldsymbol{\gamma}}$ to learn dependencies across graph nodes. The AGC is placed inside g_2 and $\mathbf{A}_{\mathcal{V}}^{inner}$ can be considered the identity matrix. One can set $\mathbf{A}_{\mathcal{V}}^{outer}$ to the identity matrix, as well, to obtain their GNCDE, or one can set it to an undirected, directed or directed with prior known proportional weights vertex adjacency matrix. $\mathbf{A}_{\mathcal{V}}^{inner}$ can be chosen analogously.

4 Experiments

We sample 1000 and 10000 time-series for a 4-node graph and a 10-node graph (their adjacencies can be found in A.2), respectively. $\mathbf{A}_{\mathcal{V}_4}$ represents a graph whose underlying topology is depicted in Figure 1, $\mathbf{A}_{\mathcal{V}_{10}}$ represents a simple graph with subsequent edges, that exhibits strong sparsity. The task is to predict the quantities for the next 24 time points given the current and the previous 24 time points. We trained the Informed GNCDE with different variations for the inner and outer graph mechanism and calculate the mean absolute error (MAE) on the test dataset. For the 4-node graph, the informedness on top of the AGC leads to a slight improvement in MAE compared to only using AGC of 2.67 vs. 2.73 (Table 1). Notably, using the outer mechanism alone leads to a MAE of 2.75, with only requiring $293,184$ vs. $416,744$ parameters. For the 10-node graph with longer training time, we obtain lower MAE for just using the outer informedness versus only AGC and significant lower MAE for using informedness on top of AGC with MAE of 0.66 vs. 1.42 (Table 1).

5 Discussion

Position plays a crucial role for the informedness in a NCDE. The position of our informed NCDE approach can be seen analogous to a shortcut connection in a Residual Neural Network (ResNet) [18], only between spatial nodes and not in the virtual time dimension. We suspect a greater impact on prediction performance with larger graphs with sparse connectivity structure. We also observe faster convergence during training for architectures with outer informedness. Informing at the inner and the outer position simultaneously with the same matrix seems to deteriorate performance. Introducing a fully-connected trainable outer matrix in an attempt to learn the adjacency matrix led to exploding gradients. In summary we demonstrated that prior graph information can benefit model performance of a NCDE on a physical time series forecasting task. The position in the architecture where one inserts the information is crucial. One has to investigate further if graph information is detrimental to learning non-causal temporal patterns across distant vertices. We dedicate our future research to exploring larger graphs, integrating graph edge features, and addressing non-constant advection, with the goal of applying our topology informed GNCDE to real-world data that includes exogenous influx at the vertices and probabilistic behavior.

Acknowledgements

This work was supported by the Federal Ministry for Transportation and Digital Infrastructure (BMDV) as grant SOLP (19F2204B); and the Federal Ministry for Economic Affairs and Climate Action (BMWK) as grant DAKI-FWS (01MK21009A).

References

[1] Bernhard Schölkopf. Causality for Machine Learning. pages 765–804. February 2022. doi: 10.1145/3501714.3501755. URL http://arxiv.org/abs/1911.10500.

[2] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/69386f6bb1dfed68692a24c8686939b9-Paper.pdf.

[3] Patrick Kidger, James Morrill, James Foster, and Terry Lyons. Neural controlled differential equations for irregular time series. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6696–6707. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/4a5876b450b45371f6cfe5047ac8cd45-Paper.pdf.

[4] Jeongwhan Choi, Hwangyong Choi, Jeehyun Hwang, and Noseong Park. Graph neural controlled differential equations for traffic forecasting. In *AAAI*, 2022.

[5] P Kidger. *On neural differential equations*. PhD thesis, University of Oxford, 2021.

[6] 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. URL https://ieeexplore.ieee.org/document/9046288/.

[7] Emanuele Rossi, Ben Chamberlain, Fabrizio Frasca, Davide Eynard, Federico Monti, and Michael Bronstein. Temporal graph networks for deep learning on dynamic graphs. In *ICML 2020 Workshop on Graph Representation Learning*, 2020.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/gilmer17a.html.

[9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

[10] Kyunghyun Cho, B van Merrienboer, Caglar Gulcehre, F Bougares, H Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2014)*, 2014.

[11] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph Neural Ordinary Differential Equations. In *AAAI 2020 Workshop on Deep Learning on Graphs: Methodologies and Applications (DLGMA'20)*, June 2021.

[12] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2018.10.045. URL https://www.sciencedirect.com/science/article/pii/S0021999118307125.

[13] Chrysoula Kosma, Giannis Nikolentzos, George Panagopoulos, Jean-Marc Steyaert, and Michalis Vazirgiannis. Neural ordinary differential equations for modeling epidemic spreading. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=yrkJGne0vN. Featured Certification, Expert Certification.

[14] Yogesh Verma, Markus Heinonen, and Vikas Garg. ClimODE: Climate and weather forecasting with physics-informed neural ODEs. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=xuY33XhEGR.

[15] Airlie Chapman and Mehran Mesbahi. Advection on graphs. In *2011 50th IEEE Conference on Decision and Control and European Control Conference*, pages 1461–1466, 2011. doi: 10.1109/CDC.2011.6161471.

[16] Sheo Yon Jhin, Heejoo Shin, Sujie Kim, Seoyoung Hong, Minju Jo, Solhee Park, Noseong Park, Seungbeom Lee, Hwiyoung Maeng, and Seungmin Jeon. Attentive neural controlled differential equations for time-series classification and forecasting. *Knowledge and Information Systems*, 66 (3):1885–1915, March 2024. ISSN 0219-1377, 0219-3116. doi: 10.1007/s10115-023-01977-5. URL https://link.springer.com/10.1007/s10115-023-01977-5.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=SJU4ayYgl.

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

A Appendix

A.1 Dimensions

The dimensions of the domains and codomains of the vector field functions $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\gamma}}$ can be described even more precisely with

$$f_{\boldsymbol{\theta}}(\cdot) : \mathbb{R}^{|\mathcal{V}| \times d_h} \to \mathbb{R}^{|\mathcal{V}| \times d_h} \times \mathbb{R}^{|\mathcal{V}| \times d_x}$$
$$g_{\boldsymbol{\gamma}}(\cdot) : \mathbb{R}^{|\mathcal{V}| \times d_z} \to \mathbb{R}^{|\mathcal{V}| \times d_z} \times \mathbb{R}^{|\mathcal{V}| \times d_h}$$

and $\mathbf{H}(t) \in \mathbb{R}^{|\mathcal{V}| \times d_h}$, $\mathbf{Z}(t) \in \mathbb{R}^{|\mathcal{V}| \times d_z}$, $\mathbf{X}(t) \in \mathbb{R}^{|\mathcal{V}| \times d_x}$, with d_h, d_z and d_x being the dimensionality of the corresponding (hidden) states or control per vertex, leading to a description of the tensor product in Equation (3) with einstein notation while omitting the $\boldsymbol{\theta}$ and $\boldsymbol{\gamma}$ for more clarity

$$\left(\frac{d\mathbf{H}}{dt}\right)_{mh} = \left(f\frac{d\mathbf{X}}{dt}\right)_{mh} = f_{mhnx}\left(\frac{d\mathbf{X}}{dt}\right)_{nx}$$

and for Equation (4)

$$\left(\frac{d\mathbf{Z}}{dt}\right)_{kz} = \left(g\frac{d\mathbf{H}}{dt}\right)_{kz} = g_{kzmh}\left(\frac{d\mathbf{H}}{dt}\right)_{mh}, \tag{8}$$

where n, m, and k are indices for the vertices, h and z are the indices for the dimension of hidden states $\mathbf{H}$ and $\mathbf{Z}$, respectively and x is the index for the dimension of the control path $\mathbf{X}$.

If one now wants to incorporate graph information at the outer position in Equation (7), one would insert $\mathbf{A}_{\mathcal{E}}$ into Equation (8) and obtain

$$\left(\frac{d\mathbf{Z}}{dt}\right)_{kz} = \left(g\frac{d\mathbf{H}}{dt}\right)_{kz} = g_{kzmh}\ \left(\mathbf{A}_{\mathcal{V}}\right)_{mn}\ \left(\frac{d\mathbf{H}}{dt}\right)_{nh}.$$

A.2 Adjacency matrices

The adjaceny matrices for the 4-node graph and the 10-node graph are given by

$$\mathbf{A}_{\mathcal{V}_4} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0.3 & 0.7 \\ 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{A}_{\mathcal{V}_{10}} = \begin{bmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & 1 & 0 \\ 0 & \dots & 0 & 0 & 1 \\ 0 & \dots & 0 & 0 & 0 \end{bmatrix}.$$

A.3 Edge transition matrix

To obtain the edge transition matrix $\mathbf{A}_{\mathcal{E}}$ from a weighted vertex adjacency matrix $\mathbf{A}_{\mathcal{E}}$ with $\sum_{j \in |\mathcal{E}|}(\mathbf{A}_{\mathcal{E}})_{ij} = 1$, one first starts off with the vertex incidence matrix $\mathbf{I}$. Let us define some matrix operations for $\mathbf{I} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$, namely

$$(\mathbf{I}^{+})_{ij} = \begin{cases} (\mathbf{I})_{ij}, & \text{if } (\mathbf{I})_{ij} > 0 \\ 0, & \text{else} \end{cases},$$

$$(\mathbf{I}^{-})_{ij} = \begin{cases} -(\mathbf{I})_{ij}, & \text{if } (\mathbf{I})_{ij} < 0 \\ 0, & \text{else} \end{cases},$$

$$(\mathbf{I}^{c})_{ij} = \begin{cases} 1, & \text{if } (\mathbf{I})_{ij} > 0 \\ (\mathbf{I})_{ij}, & \text{else} \end{cases},$$

thereby $(\cdot)^c$ is making $\mathbf{I}$ conservative. In total the edge transition matrix $\mathbf{I}_{\mathcal{E}}$ is given by

$$\mathbf{A}_{\mathcal{E}} = (\mathbf{I}^{-})^{\intercal}(\mathbf{I}^{c})^{+}.$$

On the one side $(\mathbf{I}^{-})^{\intercal} \in \mathbb{R}^{|\mathcal{E}| \times |\mathcal{V}|}$ considers the incoming magnitude of the transported quantities from edges to vertices and also the amount of quantities, on the other side $(\mathbf{I}^{c})^{+} \in \mathbb{R}^{|\mathcal{V}| \times |\mathcal{E}|}$ distributes the outgoing quantites from the vertices to the edges while solely encoding connectivity and neglecting magnitude.

A.4 Applications

For traffic forecasting, one would consider the traffic flow sensors the nodes of the graph and the road network would be the edges. Analogously in the climate and weather forecasting scenario, weather observation stations measuring quantities like precipitation, temperature, air pressure and wind, would constitute the nodes, and the edges would transport abstract representations that encode the propagation of the physical quantities in space. Here the propagation in three-dimensional space would be encoded as an information link which is represented as a multivariate function on every one-dimensional domain on the edges. In river water level forecasting, one considers the gauging stations as nodes and directed edges that are derived from the geographical river topology connections. For disease spread, a node would be regarded as an administration area for which one can measure the aggregate infections, and the edges would connect these regional areas according to mobility connections, e.g. adjacent areas would be connected, areas that can be reached via train or airplane etc. Here the velocity field on the edges would need to be adapted according to the speed of the means of travel.

A.5 Causality

In total we moved from a space-continuous description of the state on the edges and vertices during the simulation to a discrete space on the vertices for the measurements. In the continuous advection description, we had a clear cause-effect relation due to the fact that we could look back along an edge or back to previous edges with the edge transition matrix. This continuous description of the causal structure is now subsumed to aggregated, discrete values at nodes and edge connections without an attached domain between them. To be able to incorporate this causal structure into the GNCDEs we have to consider a long enough context window to be able to receive information from the previous vertices at earlier times. The task then resembles learning a delay differential equation.

A.6 Additional Results

In Table 2 the MAE on the 4-node graph for different GNCDE configurations with only 2 layers is reported.

Table 2: MAE on unseen data and number of trainable parameters for different selection of pairs for outer and inner graph mechanism for the smaller model with a **2-layer** architecture for the **4-node graph**

	$\mathbf{A}_{\mathcal{V}^{outer}}$			
	Identity		Informed	
$\mathbf{A}_{\mathcal{V}^{inner}}$	MAE	# params	MAE	# params
Identity	3.63	289,024	2.96	289,024
Informed	3.21	289,024	3.60	289,024
AGC	2.83	412,584	**2.78**	412,584

In Table 3 the MAE on the 10-node graph for different GNCDE configurations with only 2 layers is reported.

Table 3: MAE on unseen data and number of trainable parameters for different selection of pairs for outer and inner graph mechanism for the smaller model with a **2-layer** architecture for the **10-node graph**

	$\mathbf{A}_{\mathcal{V}^{outer}}$			
	Identity		Informed	
$\mathbf{A}_{\mathcal{V}^{inner}}$	MAE	# params	MAE	# params
Identity	2.80	289,084	**1.92**	289,084
Informed	2.62	289,084	2.75	289,084
AGC	2.09	412,074	1.94	412,072

A.7 Background on Neural Differential Equations

Neural Ordinary Differential Equations. NODEs (Chen et al. [2]) are time-continuous models that use the structure of a differential equation to model the evolution of state variables $y \in \mathbb{R}^d$:

$$y(0) = y_0, \quad \frac{\mathrm{d}y}{\mathrm{d}t} = f_\theta(t, y(t)).$$

The Lipschitz-continuous function $f_\theta : \mathbb{R} \times \mathbb{R}^d \to \mathbb{R}^d$ on the right hand side, the so called vector field that guides the derivative in the equation is parametrized by an NN with parameters θ, and thereby rendering it "neural". NODEs can be considered the continuous-time limit of residual networks. One can bring an NODE also into an integral formulation:

$$y(0) = y_0, \quad y(t) = y(0) + \int_0^t f_\theta(s, y(s))\,\mathrm{d}s.$$

Neural Controlled Differential Equations. In comparison to the aforementioned NODE formulation in integral form, in a NCDE [3] described by

$$y(0) = y_0, \quad y(t) = y(0) + \int_0^t f_\theta(y(s))\,\mathrm{d}x(s),$$

the continuous path $y : [0, T] \to \mathbb{R}^{d_y}$ must satisfy the equation which is driven by a control path $x(s) : [0, T] \to \mathbb{R}^{d_x}$ that allows continuously introducing information from data at later points in time. $d_x, d_y \in \mathbb{N}$ are the dimensionalities of the two paths x and y, respectively. The vector field $f : \mathbb{R}^{d_y} \to \mathbb{R}^{d_y \times d_x}$ must be Lipschitz-continuous. NCDEs can be considered the continuous version of RNNs.