Identifying Contemporaneous and Lagged Dependence Structures by Promoting Sparsity in Continuous-time Neural Networks

Fan Wu*
Arizona State University
Tempe, Arizona, USA
fanwu8@asu.edu

Woojin Cho
Yonsei University
Seoul, Korea
snowmoon@yonsei.ac.kr

David Korotky
Oregon State University
Corvallis, Oregon, USA
korotkyd@oregonstate.edu

Sanghyun Hong
Oregon State University
Corvallis, Oregon, USA
sanghyun.hong@oregonstate.edu

Donsub Rim
Washington University in St. Louis
St. Louis, Missouri, USA
rim@wustl.edu

Noseong Park
KAIST
Daejeon, Korea
noseong@kaist.ac.kr

Kookjin Lee*†
Arizona State University
Tempe, Arizona, USA
Kookjin.Lee@asu.edu

ABSTRACT

Continuous-time dynamics models, e.g., neural ordinary differential equations, enable accurate modeling of underlying dynamics in time-series data. However, employing neural networks for parameterizing dynamics makes it challenging for humans to identify dependence structures, especially in the presence of *delayed* effects. In consequence, these models are not an attractive option when capturing dependence carries more importance than accurate modeling, e.g., in tsunami forecasting.

In this paper, we present a novel method for identifying dependence structures in continuous-time dynamics models. We take a two-step approach: (1) During training, we promote weight sparsity in the model's first layer during training. (2) We prune the sparse weights after training to identify dependence structures. In evaluation, we test our method in scenarios where the exact dependence-structures of time-series are known. Compared to baselines, our method is more effective in uncovering dependence structures in data even when there are delayed effects. Moreover, we evaluate our method to a real-world tsunami forecasting, where the exact dependence structures are unknown beforehand. Even in this challenging scenario, our method still effective learns physically-consistent dependence structures and achieves high accuracy in forecasting.

*F. Wu and K. Lee acknowledge the support from the U.S. National Science Foundation under grants CNS2210137 and IIS2338909.
†Corresponding author

ACM ISBN 979-8-4007-0436-9/24/10…$15.00
https://doi.org/10.1145/3627673.3679751

CCS CONCEPTS

• **General and reference → General conference proceedings**; • **Information systems → Geographic information systems**; • **Applied computing → Environmental sciences**.

KEYWORDS

Causality Learning, Neural Ordinary Differential Equations, Tsunami Modeling

ACM Reference Format:
Fan Wu, Woojin Cho, David Korotky, Sanghyun Hong, Donsub Rim, Noseong Park, and Kookjin Lee. 2024. Identifying Contemporaneous and Lagged Dependence Structures by Promoting Sparsity in Continuous-time Neural Networks. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management (CIKM '24), October 21–25, 2024, Boise, ID, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3627673.3679751

1 INTRODUCTION

An emerging paradigm for modeling underlying dynamics in time-series data is to use continuous-time dynamics neural networks, such as neural ordinary differential equations (NODEs) [3, 6, 33]. Widely known as a continuous-depth extension of residual networks [11], NODEs have a great fit for data-driven dynamics modeling as they construct dynamics in the form of ODEs. This new paradigm has enabled breakthroughs in many applications, e.g., healthcare [33], computational physics [15–17], or climate modeling [14, 31].

A promising approach to improve modeling performance is to increase the *expressivity* of the neural networks used to parameterize a system, *e.g.*, by employing convolutional neural networks or adding extra dimensions in the state space [6]. While these complex models are shown effective, it becomes harder for humans to interpret the learned dynamics. This poses a greater challenge for tasks that need human-interpretable dependence structures, e.g., predicting high-consequence events like tsunamis.

In this paper, we present a novel method for identifying dependence structures in time-series data from continuous-time neural networks. Unlike existing approaches that perform post-training analyses of models, we focus on learning human-interpretable dependence structure proactively during training. To this end, we employ the score-based structure learning for dynamical systems proposed by [2]. Because the technique can quantify the impact of a past event on the future evolution of data, we extend it to the models, such as neural delay differential equations [40], effective in capturing delayed (or lagged) effects.

Our contributions. We *first* present a novel method for capturing contemporaneous and also lagged dependence structures using continuous-time neural networks. We adapt the score-based structure learning, presented by [2], to neural delay differential equations (NDDEs). Our method consists of two steps: The first step is to train a model while promoting the sparsity. We design a training objective composed of a data-matching loss and a sparsity-promoting penalty term. Once trained, the second step is to prune the columns of input layer weights with the norms smaller than a threshold we define. The columns with high norms will only survive in the input layer, and those corresponds to important input elements for interpreting dependence structures.

Second, we show the effectiveness of our method in learning complex (and also chaotic!) dynamics and their underlying dependence structures. To demonstrate this effectiveness, we conduct a comprehensive evaluation of our method on three systems where their underlying complex dynamics are well-known: the Lorenz-96, Mackey–Glass,and Lotka–Volterra systems. We also compare our method extensively with four baseline methods presented in prior work. The results show that the dependence structures our method identifies from the data matches with the known ground-truth structures more accurately than those captured by the baselines.

Third, we further show our method's effectiveness in identifying *unknown* dependence structures from time-series data. We apply our method to tsunami forecasting at the Strait of Juan de Fuca [25]. This scenario presents a significant challenge as the exact dependence structures in the time-series data are unknown, and the data includes lagged effects. We use NDDEs for our method and train them on the tsunami dataset [23]. The results indicate that our method surpasses the baseline [20]. It more accurately predicts the highest peaks of sea-surface elevations in areas near large populations. Our method is also able to identify the physically-consistent dependence structures between a tsunami event and prior tide observations, aligned with the interpretations of domain experts.

2 PRELIMINARIES

Neural ordinary differential equations (NODEs). parameterize the time-continuous dynamics of hidden states in the data as a system of ODEs using a neural network [3]:

$$\frac{dz}{dt} = f(z, t; \Theta) \tag{1}$$

where $z(t) \in \mathbb{R}^{n_z}$ denotes a time-continuous hidden state, f is a velocity function, parameterized by a neural network whose parameters are denoted as Θ. A typical parameterization of f is a multi-layer perceptron (MLP) $\Theta = \{(W^\ell, \boldsymbol{b}^\ell)\}_{\ell=1}$. Θ^l and $\boldsymbol{b}^l$ are weights and biases of the ℓth layer, respectively. The forward pass of NODEs is equivalent to solving an initial value problem via a black-box ODE solver. Given an initial condition z_0 and f, it computes:

$$z(t_0), z(t_1) = \text{ODESolve}(z_0, f, \{t_0, t_1\}; \Theta). \tag{2}$$

Neural delay differential equations (NDDEs). NODEs have a limitation in the context of dependence structure modeling. NODEs take only the current state of the input variables $z(t)$ (shown in Eqn. 1), which is not suitable for capturing delayed effects. To enable modeling of delayed effects we use the computational formalism NDDEs [40] offers. NDDEs are an extension of NODEs which takes extra input variables $z_{\le\tau}(t)$ as follows:

$$\frac{dz}{dt} = f(z(t), z_{\le\tau}(t), t; \Theta), \tag{3}$$

where $z_{\le\tau}(t) = \{z(t-\gamma) : \gamma \in [0, \tau]\}$ denotes the trajectory of the solution in the past up to time $t - \tau$. To avoid numerical challenges in handling a continuous form of delay, we use a discrete form:

$$\frac{dz}{dt} = f(z(t), z(t-\tau_1), \ldots, z(t-\tau_m), t; \Theta), \tag{4}$$

where m indicates the number of discrete delayed variables. While NDDEs are universal approximators [40], most studies use this continuous-depth neural networks for performing standard classification tasks, e.g., visual recognition [40, 41]. A few studies use NDDEs for time-series modeling, but they are limited to simulated ODE data [13, 28]. It is also unknown whether we can identify dependence structures from these models. To our best knowledge, we are the first who leverage NDDEs for modeling real-world time-series data and discovering *delayed* dependence structures.

Score-based learning for dependence structure discovery. In recent years, there have been advances in structure discovery in continuous-time, such as neural graphical modeling (NGM) [2], the score-based structure learning for dynamical systems. The system of ODEs in Eqn. 1 can be re-written as:

$$\frac{dz_j}{dt} = f_j(z, t; \theta_j), \; j = 1, \ldots, n_z \tag{5}$$

where z_j denotes the jth element of the hidden state such that $z = [z_1, \ldots, z_{n_z}]^\top$. The velocity function f consists of n_z individual neural networks such that $f(z) = [f_1(z), \ldots, f_{n_z}(z)]^\top$ with $f_j \in \mathbb{R}$. Each networks are parameterized by so called model parameters θ_j (i.e., $\Theta = \cup_{j=1}^{n_z} \theta_j$).

Assuming the typical parameterization (i.e., affine transformation followed by nonlinearity), each individual function f_j can be written as follows (we omit the bias terms for clarity):

$$f_j(z, t; \theta_j) = W_j^L \sigma(\cdots \sigma(W_j^2 \sigma(W_j^1 z)) \cdots), \; j = 1, \ldots, n_z, \tag{6}$$

where σ is the nonlinear activation, and W_j^l is the weight of the lth layer of the jth neural network.

Given the dynamics model in Eqn. 5, the (in-)dependence structure within the processes z can be identified via the definition of *local independence* and graphical representations of the processes. Here, we briefly recap the definition of *local independence* (we refer readers to the reference [2] for the formal definition); a process z_i is locally independent of z_j given z_k, if the past of z_k up until time t gives the same information for predicting $z_i(t)$ (z_i at time t) as the past of z_i and z_j until time t. This independence structure is represented as a directed graph $\mathcal{G} = (V, E)$, where V denotes a set of vertices representing distinct processes $\{z_j\}_{j=1}^{n_z}$ and E denotes a set of edges, where a directed edge $z_i \rightarrow z_j \in E$ if and only if z_i is not locally conditionally independent z_j.

With local independence and the graph $\mathcal{G}$, here we briefly restate Lemma 1 in [2] (Proposition 3.6 in [27]); given dynamics models in the form of Eqn. 5, two processes z_i and z_j are locally dependent if and only if z_i appears in the differential equation of z_j (i.e., $||\frac{\partial f_j}{\partial z_i}||_{L_2} \neq 0$, where $|| \cdot ||_{L_2}$ is the functional L_2 norm). Moreover, for any $\boldsymbol{f}'$ such that $||\frac{\partial f_j'}{\partial z_i}||_{L_2} = 0$, there exists an equivalent vector field $\boldsymbol{f}$ such that the ith column of its input layer has the Euclidean norm zero, i.e., $||[W_j^1]_i||_2 = 0$. For the graph model $\mathcal{G}$, we can define the adjacency matrix $\mathcal{A} \in \{0,1\}^{n_z \times n_z}$, where $\mathcal{A}_{ij} \neq 0$ if and only if $||\frac{\partial f_j}{\partial z_i}||_{L_2} \neq 0$, suggesting the dependence structure can be identified through the input layer parameters W_j^1, $j = 1, \ldots, n_z$. Thus, sparsity-promoting regularizes such as the group lasso [10] or the adaptive group Lasso [13] can be applied to the input layer parameters W_j^1, leading to the identification of the dependence structure (e.g., Fig. 1(a) for graphical illustration).

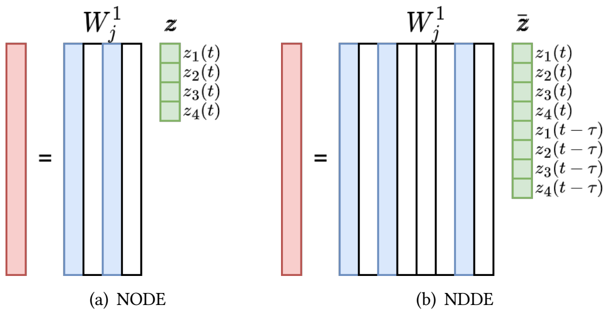


Figure 1: An illustrative operation of the input layer (the white bars indicate zero-norm columns in weights).

3 PROPOSED METHOD

Now we present our method for (delayed) dependence structure discovery. Score-based learning showed its effectiveness in identifying dependence structures of the contemporaneous variables (those in the same time index). But it has not been studied if the same approach will be effective when the system has delayed effects. Our work addresses challenges in trivial adaptation of the score-based learning to such systems and offers a method to characterize delayed structures.

3.1 (Delayed) Dependence Structure

We find that NDDEs' velocity function (Eqn. 4) has the same structure as those in NODEs (Eqn. 5):

$$f(\bar{z}(t); \Theta) = \begin{bmatrix} f_1(z(t), z(t-\tau_1), \ldots, z(t-\tau_m); \theta_1) \\ \vdots \\ f_{n_z}(z(t), z(t-\tau_1), \ldots, z(t-\tau_m); \theta_{n_z}) \end{bmatrix}, \quad (7)$$

where $f_j \in \mathbb{R}$ is the jth element of the velocity function, *i.e.*, $\frac{dz}{dt} = f_j$. The input layer of the jth velocity element, $f_j(z) = W_j^1 \bar{z} + \boldsymbol{b}_j^1$. $\bar{z}(t)$ is a vertical concatenation of all delayed variables:

$$\begin{aligned} \bar{z}(t) &= [z(t)^\mathsf{T}, z(t-\tau_1)^\mathsf{T}, \ldots, z(t-\tau_m)^\mathsf{T}] \\ &= [z_1(t), \ldots, z_{n_z}(t), z_1(t-\tau_1), \ldots, z_1(t-\tau_m), \ldots]. \end{aligned} \quad (8)$$

Inspired by the prior work's approach [2], we propose to identify the dependence structure through the input layer parameters W_j^1; if the ith column of the first layer weight, W_j^1, contains only zero elements (i.e., $\|[W_j^1]_i\|_2 = 0$), the jth element in $\bar{z}(t)$ is independent from the ith element of $\bar{\boldsymbol{z}}$. Through this process, we are able to reveal the structure in terms of the adjacency matrix $A \in \mathbb{R}^{n_z(m+1) \times n_z}$ (a graphical illustration of this process is shown in Fig. 1(b)).

3.2 Sparsity-promoting Objective and Pruning

We now propose to minimize the following sparsity promoting loss (group lasso), and by doing so, we aim to identify the structures from the multivariate time-series over the course of the training:

$$\sum_{i=1}^{n_{\text{train}}} L(z(t_i), \tilde{z}(t_i)) + \alpha \sum_{i=1}^{n_z} \sum_{j=1}^{n_z} ||[W_j^1]_i||_2, \quad (9)$$

where α is the penalty. In addition to the sparsity-promoting loss, we leverage a magnitude-based pruning method to capture the structure more explicitly. As the training proceeds, we prune a column whose ℓ_2-norm becomes smaller than a certain threshold, ρ:

$$[W_j^1]_i = \mathbf{0} \quad \text{if} \quad \|[W_j^1]_i\|_2 \leq \rho, \quad (10)$$

where $\mathbf{0}$ is a zero-valued vector. By applying the sparsity-promoting loss and the pruning method, we expect that the entries of $\bar{z}$, that are non-causal to specific time-series, are pruned and, thus, the lags where dependencies exist can be identified automatically.

3.3 Putting All Together

In the implementation, we employ the standard mini-batching and a variant of stochastic gradient descent (SGD) to train the neural network architectures described in the previous subsections. With the sparsity promoting penalty and the pruning, the entire training process shares the commonality with the training algorithm proposed in [17], shown in Algorithm 1. We implement our method in `Python` using a deep learning framework, `PyTorch` [32]. For the NODEs/NDDEs capability, we use the `TorchDiffeq` library [3]. We will use the notation $\boldsymbol{x}$ to distinguish the time-stepped-series as opposed to the continuous series $\boldsymbol{z}$.

3.4 Discussion

Neural Granger Causality. Neural Granger Causality (NGC) [37] is a method to learn dependence structure based on the Granger causal interactions [8]. This method has a similarity with the proposed method (and also with NGM [2]), which is to estimate the dependence structure via measuring the signal intensity at the input layers and, to achieve this goal, component-wise MLPs or RNNs (whose input layer resembles the structures shown in Figure 1(b)) have been utilized. The major difference lies in time continuity; NGC could handle only the discrete dynamics whereas the proposed

Algorithm 1 Dependence Structure Discovery in NDDEs

```
Initialize $\Theta$
for ($i = 0$; $i < n_{\max}$; $i = i + 1$) do
    Sample $n_{\text{batch}}$ trajectories randomly from $\mathcal{D}_{\text{train}}$
    Sample initial points randomly from the sampled trajectories: $\bar{z}^r(s(r))$, $s(r) \in [0, \ldots, m - \ell_{\text{batch}} - 1]$ for $r = 1, \ldots, n_{\text{batch}}$
    $\tilde{z}(t_1), \ldots, \tilde{z}(t_m) = \text{ODESolve}(\bar{z}^r_{s(r)}, f, \{t_1, \ldots, t_m\}; \Theta)$, for $r = 1, \ldots, n_{\text{batch}}$
    Compute the loss (Eqn. 9)
    Update $\Theta$ via Adam
    Prune $\Theta$ based on the magnitude (Eqn. 10)
end for
```

method (and NGM) learn a continuous-in-time dynamics. As investigated in [2, Appendix A.1], a discrete-in-time causal learning can be inconsistent, i.e., causal strengths may change dramatically as a function of the measurement time interval Δt at which the process is observed. This suggests that "Describing causality within dynamical systems in discrete-time is inherently an ill-posed problem" [2]. The proposed method inherits continuous-in-time dynamics modeling from NGM and, thus, is naturally expected to mitigate the weaknesses shown in discrete-time causality learning.

Utility of pruning. Employing only the sparsity-promoting loss does not produce a sparse representation of dependence structure, but produce dense dependence structure with many weak causal entries (i.e., coefficients with small magnitude). To mitigate this issue, proximal gradient descent (PGD) [30] has been considered in NGM [2]), which essentially replaces a small enough model parameter with zero (i.e., setting a column with a small L_2-norm to zero via thresholding). However, those zeroed model parameters can be kept updated in later gradient update steps (potentially resulting in nonzero coefficient). Instead, in this method, the pruning mechanism is employed; once a model parameter is zeroed out, that model parameter is set as "non-trainable", making training more efficient and the result more interpretable.

4 EVALUATION ON SYSTEMS WITH KNOWN DEPENDENCE STRUCTURE

We first showcase the effectiveness of the proposed method with two canonical chaotic ODE benchmark problems, the Lorenz-96 system (Section 4.1), the Mackey–Glass (MG) equation (Section 4.2), and a two-dimensional ODE benchmark problem, the Lotka–Volterra (LV) equation with delays (Section 4.3). The Lorenz-96 system has been an important testbed for climate modeling. We use this system to showcase the structure learning in the context of NODEs with the assumption that there is no delayed effect (for a sanity check). The main focus is on two following benchmarks exhibiting delayed dependence structures: the MG and LV equations. The MG equation is another chaotic system, describing the healthy and pathological behaviour in certain biological contexts (*e.g.*, blood cells). We use MG to demonstrate the structure learning with *delayed* variables in the context of NDDEs. Lastly, LV describes a dynamics of prey-predator interactions in ecological systems. We consider LV to demonstrate the structure learning in multi-dimensional systems with delayed variables. In the all experiments, we measure performance in two different angles, accuracy in trajectory reconstruction and structure discovery.

Baselines. As the baselines of comparison, we consider four methods: DYNOTEARS [29], PCMCI+ [34], Neural Granger causality (NGC) [37], and Rhino [7]. DYNOTEARS is a scored-based algorithm, following the standard structural vector autoregression (SVAR) model. PCMCI+ is a conditional independence (CI)-based algorithm, which utilizes the Peter–Clark (PC) algorithm and the Momentary Conditional Independence (MCI) test. NGC is based on the Granger causal interactions [8], which are estimated via measuring the signal intensity at the input layers of component-wise MLPs or RNNs. Rhino is a functional causal algorithm that integrates vector auto-regression (VAR), deep learning and variational inference to discover causal relationships. We use implementations of the `causalnex` API[1] for DYNOTEARS, the `tigramite` API[2] for PCMCI+, the Neural-GC API[3] for NGC, and the `causica` API[4] for Rhino.

4.1 Benchmark 1: The Lorenz-96 System

The Lorenz-96 system [21] is given by the equation:

$$\frac{dx_i}{dt} = (x_{i+1} - x_{i-2})x_{i-1} - x_i + F, \tag{11}$$

where $i = 1, \ldots, N$ with $x_{-1} = x_{N-1}$, $x_0 = x_N$, and $x_1 = x_{N+1}$, F denotes the forcing term.

Setup. For modeling, we use the component-wise NODE where each MLP, f_i, has 4 layers with 100 neurons, where z_i models x_i with $n_z = N$ in Eqn. 5. For the nonlinearity, we use the Tanh activation. We put the detailed experimental setup in Supplementary Materials (**SM**).

Results. Fig. 2 reveals the learned dependence structure by showing the magnitude of the column norms of the weight matrices, $\{W_1^j\}$, in the input layer; the magnitude is normalized to have 1 as the maximum value (black) and 0 as the minimum value (white). The horizontal entries (*i.e.*, $\{x_i\}$) with darker colors (close to the black color) can be considered as the ones with the significant contributions to the vertical entries (*i.e.*, $\{\dot{x}_i\}$). Fig. 2 shows that the models have learned that $\dot{x}_i$ is dependent on $x_{i-2}, x_{i-1}, x_i, x_{i+1}$ and not dependent on other entries. Further experiments with varying N can be found in **SM**. As finding structures in NODEs is not the main contribution in this work and the same benchmark problem has been studied in [2], we refer readers to [2] for more experimental results.

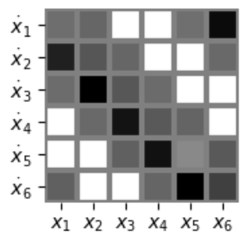


Figure 2: [Lorenz-96] Identified dependence structure for the systems with $N = 6$.

[1]https://github.com/quantumblacklabs/causalnex/blob/develop/causalnex/structure/dynotears.py
[2]https://github.com/jakobrunge/tigramite/blob/master/tigramite/pcmci.py
[3]https://github.com/iancovert/Neural-GC
[4]https://github.com/microsoft/causica/tree/v0.0.0

4.2 Benchmark 2: The Mackey–Glass System

The Mackey–Glass (MG) system [22] is given by the equation:

$$\frac{dx}{dt} = -bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^c}, \tag{12}$$

where a, b and c denote the ODE parameters and τ denotes the lag. The values of $x(t)$ for $t \leq 0$ is defined by the initial function $\phi(t) = .5$.

Setup. For parameterizing NDDEs, we model the right-hand side of the NDDE as an MLP that takes $m = 10$ candidate delayed variables along with the current variable as an input, such that

$$\bar{x}(t) = [x(t), x(t-\tau_1), \ldots, x(t-\tau_{10})] \in \mathbb{R}^{11}, \tag{13}$$

where $\tau_i = i$ (seconds), for $i = 1, \ldots, 10$. We use an MLP consisting of 4 layers with 25 neurons with the Tanh activation function. We refer readers to **SM** for the detailed experimental setup.

Results. Fig. 3 depicts the ground-truth trajectory and the predicted trajectory computed from the learned NDDE model. We repeated the same training for five times with different initializations. Although the mean and the two standard deviation are plotted in Fig. 3, each prediction appear almost identical. Fig. 4 depicts the learned dependence structure. The values are normalized to lie between [0, 1]. We can observe that the most significant contributes are from $x(t-\tau_4)$ and $x(t-\tau_5)$, meaning that the delayed effect appears within a 4~5-seconds window, correctly predicting true positive, but with one false positive error.

Comparisons to baselines. As baselines of comparisons, we consider the recent statistical and machine learning methods for identifying causalities in time-series data with lagged (i.e., delayed) variables: PCMCI+ [34], DYNOTEARS [29], and neural Granger causality [37] - RNN variants (NGC-RNN). We refer readers to **SM** for more descriptions on these methods. For all methods, we repeat three independent runs. Fig. 4 shows a heatmap of numerical values indicating the causal strength, which are normalized in a row-wise fashion (i.e., normalized per each method). The proposed NDDE-based method outperforms all other baselines. The second best performing method is the PCMCI+, which somewhat captures the strongest dependence from the fifth delayed variable $x(t-\tau_5)$ (true positive), but fails to find a correct structure, having many nonzero values in dependent elements (many false positives). The other two baselines find the strongest dependence on the first delayed variable. Moreover, the proposed method is the only method that results in accurate reconstruction of the ground-truth trajectory (Fig. 3).

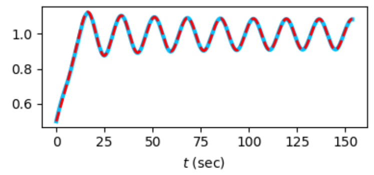


Figure 3: [Mackey–Glass] The ground-truth trajectory (solid blue) and the predicted trajectory computed from the learned model (dashed red).

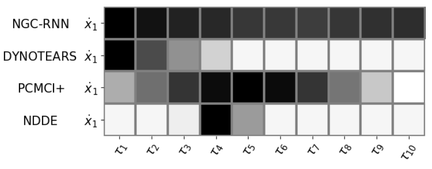


Figure 4: [Mackey–Glass] Identified dependence structure for the systems with 10 delayed variables via four different methods: NGC-RNN, DYNOTEARS, PCMCI+, and NDDE.

Table 1: [MG] Ablation on the sparsity promoting regularizer (wo reg. for indicating the case without the regularizer) and the pruning method (wo prune for indicating the case without the pruning method).

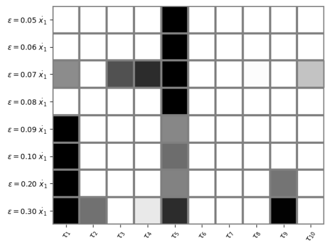
	τ_1	τ_3	τ_5	τ_7	τ_9
wo reg.	1.9E+00	2.0E+00	2.0E+00	1.1E+00	1.3E+00
wo prune	2.4E-04	4.5E-02	1.7E-01	2.2E-04	5.8E-04
ours	0	0	5.2E-02	0	0

Resilience to Gaussian noise and anomalies. We test the resilience of our approach to Gaussian noise and anomalies in three different contexts: (a) when the ground-truth trajectory is corrupted by additive Gaussian noise; (b) when the oracle trajectory contains Gaussian noise and 10 non-trivial anomalies [38], and (c) when we add noise to the velocity function of the ODE describing the Mackey–Glass system (Eqn. 12). (c) is the same approach considered in NGM [2] to inject noise in the measurement. We describe the detailed setup in **SM**.

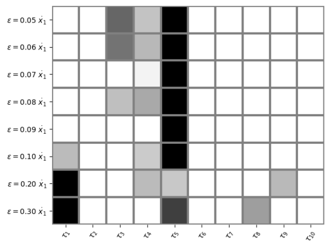
Although increasing noise level tend to negatively affect the performance of the proposed method (i.e., introducing more false positives), the method is successful in finding the true positive ($\tau = 5$) for all scenarios and all noise levels. In Fig. 5(a), our method learns the strongest dependence on the correct ground truth delay with up to around 8% additive noise. In Fig. 5(b), our method does the same up to 10% noise. In Fig. 5(c) the strongest dependence on the correct delay is learned at all noise levels. In the first two cases, adding low levels of noise appears to mitigate the 4~5 second discrepancy mentioned above. In all cases, the addition of noise appears to regularize the result. Particularly, the experimental setup and results in (c), i.e., adding noise to the ODE, shows the resilience of the dependence structures found by our method to all noise levels; the points of the trajectory are smooth while the trajectory itself is irregular due to perturbing evaluation of the system while running the ODE solver.

We also test the same baselines considered above, PCMCI+, DYNOTEARS, and NGC-RNN, on the noise/anomaly-perturbed measurements. All three baselines learn the dependence structures that are roughly the same as shown in Fig. 4; DYNOTEARS fails to capture the true positive dependence while PCMCI+ (NGC-RNN) produces many false positive errors (finding dependence on nearly all variables). We refer readers to **SM** for detailed descriptions on setup and additional results including predicted trajectories from the learned models for all cases.

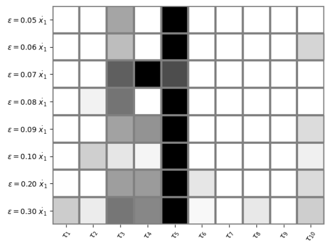
Ablation. Here, we present the results of the ablation study on the usage of the sparsity promoting regularizer (i.e., $\alpha = 0$ in Eq. (9)



(a) Noise added to trajectory.

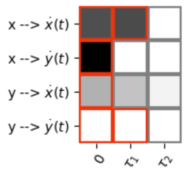


(b) Noise with anomalies.

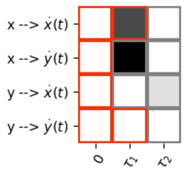


(c) Noise added to ODE

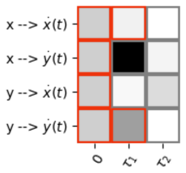
Figure 5: [Mackey–Glass] Identified dependence structures under increasing noise levels.



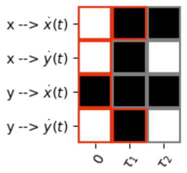
(a) NGC-RNN



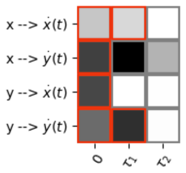
(b) DYNOTEARS



(c) PCMCI+



(d) RHINO



(e) NDDE

Figure 6: [Lotka–Volterra] Identified dependence structure for the systems with 2 delayed variables via five different methods: NGC-RNN, DYNOTEARS, PCMCI+, RHINO and NDDE. Red squares indicate the ground truth dependence structure.

for no regularizer) and pruning (i.e., ρ in Eq. (10) to be close to machine precision for no pruning). Table 1 shows the magnitude of learned dependence from delayed variables t_i (presenting only the odd numbered indices for simplicity, but the trends are same for the even numbered indices). Table 1 essentially shows that without the sparsity promoting regularizer, the dependence structure cannot be effectively identified (i.e., the output variable exhibits strong dependence on all delayed variables). Table 1 also shows that without pruning, although the magnitudes are small, the output variable still exhibits dependence on all delayed variables.

4.3 Benchmark 3: The Lotka–Volterra System

A delayed-version of Lotka–Volterra systems is given by the equation [39]:

$$\begin{aligned}\frac{dx}{dt} &= x(t)(r_1 - a_{11}x(t-\tau) - a_{12}y(t))\\ \frac{dy}{dt} &= y(t)(-r_2 + a_{21}x(t) - a_{22}y(t-\tau))\end{aligned} \tag{14}$$

where $r_1, a_{11}, a_{12}, r_2, a_{21}, a_{22}$ denote the ODE parameters and τ denotes the lag. The initial function is defined as $\phi(t) = [x(t), y(t)] = [0.1, 0.1]$ for $t \leq 0$.

Setup. For parameterizing NDDEs, we model the right-hand side of the NDDE as an MLP that takes $m = 2$ candidate delayed variables along with the current variable as an input, such that

$$\bar{x}(t) = [x(t), y(t), x(t-\tau_1), y(t-\tau_1), x(t-\tau_2), y(t-\tau_2)] \in \mathbb{R}^6 \tag{15}$$

where $\tau_i = 1.1i$, for $i = 1, 2$. We use an MLP consisting of 4 layers with 100 neurons with the Tanh activation function. For details, we refer readers to **SM**.

Results. As baselines of comparisons, in addition to the same baselines considered above, Rhino [7], a variational inference-based structure learning algorithm, is also considered. Again, more descriptions on the method is provided in **SM**. Fig. 6 depicts the structures learned from the baselines and the proposed methods. The values are normalized to lie between [0, 1]. Compared to the 1D MG benchmark problem, the multi-dimensionality increases the problem complexity, which poses more challenges to all baselines and the proposed method. Overall, the proposed NDDE outperforms the baselines, identifying the dependence structure that is most close to the ground-truth shown in Eqn. 14 (i.e., all true positives $x(t), x(t-\tau_1), y(t)$ to $\dot{x}(t)$ and $x(t), y(t), y(t-\tau_1)$ to $\dot{y}(t)$, but two false positives) compared to considered baselines (Figs. 6(a)–6(d)). The next best performing method is NGC resulting in one false positive error and two false negative errors, i.e., missing $y(t), y(t-\tau_1)$ to $\dot{y}(t)$, which can be considered critical.

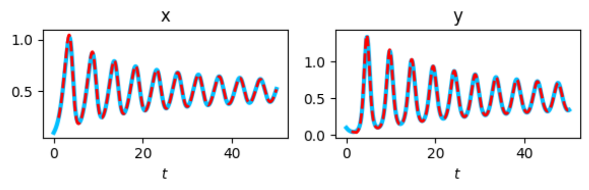


Figure 7: [Lotka–Volterra] The ground-truth trajectory (solid blue) and the predicted trajectory computed from the learned model (dashed red).

Moreover, the proposed method stands as the exclusive approach yielding an accurate reconstruction of the ground-truth trajectories (Fig. 7). Fig. 7 depicts the ground-truth trajectories and the predicted trajectories computed from the learned NDDE model. We observe that the predictions align closely with the ground truth trajectories.

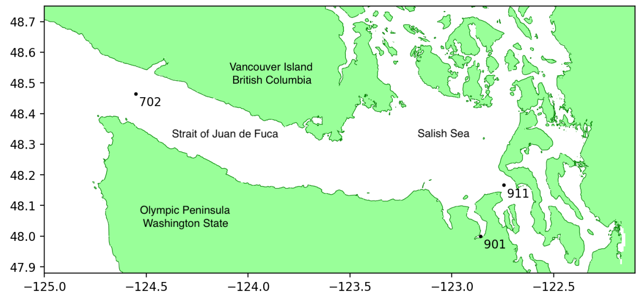


Figure 8: The geography of the tsunami simulation depicting the Strait of Juan de Fuca, and the location of the Gauges 702, 901, and 911.

5 AN APPLICATION TO TSUNAMI FORECASTING

Now we further investigate the forecasting and structure discovery capabilities of the NDDEs equipped with the learning algorithm proposed in Sec. 3. We examine them with a dataset of tsunami realizations used for developing tsunami forecasting models [20]. There exist only a handful of real earthquake events with large magnitudes, so synthetic tsunami data has been generated using numerical tsunami simulations that model the physics. In particular, the tsunami wave propagation modeled by nonlinear partial differential equations [18]. The simulation is initiated by an incoming tsunami wave, which interacts nonlinear with the topography and reflecting waves. This particular setup makes the dataset ideal for interpreting the results.

5.1 Tsunami Dataset

The dataset was created from a set of 1300 synthetic Cascadia Subduction Zone (CSZ) earthquake events ranging in magnitude from Mw 7.8 to 9.3 [25] and made available in [23]. These were generated from methods proposed in [19] using the MudPy software [24]. The resulting seafloor deformation was then used as initial conditions for tsunami wave propagation implemented in [5].

The tsunami data for one earthquake event contains tri-variate time-series data with the variables $(\boldsymbol{x}^{(702)}, \boldsymbol{x}^{(901)}, \boldsymbol{x}^{(911)})$. The time series has duration of 5-hours and is interpolated at 256 uniformly spaced points on the time-grid. This time-series data corresponds to gauge readings for the synthetic tsunami entering the Strait of Juan de Fuca (SJdF). Each variable corresponds to different geological locations denoted by the number designations 702, 901, and 911, as shown in Fig. 8. Gauge 702 is located at the entrance of SJdF, and the other two gauges are located further inside from the entrance: Gauge 901 is located in Discovery Bay, and Gauge 911 is located in the middle of Admiralty Inlet.

Among the 1300 time-series instances of tsunami, the events that with negligible tsunami in the region of our interest were discarded; event with tsunami amplitudes less than 0.1m at Gauge 702 or 0.5m at Gauge 901 were removed from the dataset. This leads to the decrease in the number of time-series instances in the dataset from 1300 to 959. We then split the remaining data into 80/5/15 for the train/validation/test set.

Fig. 9 shows an example of the surface elevation time-series measured at the three gauges. [20] considered tsunamis originating from hypothetical megathrust earthquakes in the CSZ that reach the Puget Sound. Since SJdF is the only path for the tsunamis to reach high-population areas in the Sound (Fig. 8), the authors hypothesized if observing the tsunami near the entrance of the strait (Gauge 702) could be used to forecast its amplitude at Gauge 901 and 911.

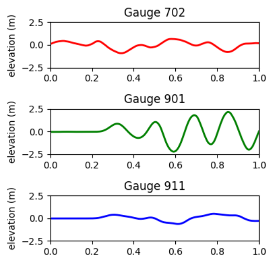


Figure 9: Surface elevation at Gauge 702, 901, and 911.

5.2 Designing Model Architecture

The base of our model is an NDDE, parameterized by using an MLP, that takes the three variables as the original input and the other three variables are their delayed versions, as follows:

$$\bar{x}(t) = [\boldsymbol{x}^{(702)}(t), \boldsymbol{x}^{(901)}(t), \boldsymbol{x}^{(911)}(t), \ldots, \boldsymbol{x}^{(702)}(t-\tau_m), \boldsymbol{x}^{(901)}(t-\tau_m), \boldsymbol{x}^{(911)}(t-\tau_m)] \tag{16}$$

and outputs the time-derivatives of the three variables

$$\dot{\boldsymbol{x}} = [\dot{\boldsymbol{x}}^{(702)}, \dot{\boldsymbol{x}}^{(901)}, \dot{\boldsymbol{x}}^{(911)}]. \tag{17}$$

We consider an MLP with 4 layers with 100 neurons and Tanh nonlinearity for each output element.

Hyperparameter search. As in the empirical study [9] of applying different normalization techniques to NODEs, we examined several combinations of normalization techniques, including layer normalization (LN) [1], weight normalization (WN) [35], and spectral normalization [26]. We found the best working configuration for this particular case study is to set each layer as

$$[\text{Linear} \rightarrow \text{WN} \rightarrow \text{LN} \rightarrow \text{Tanh}].$$

We have tested nine combinations of the weight penalty $\alpha \in \{0.1, 0.01, 0.001\}$ and the pruning threshold $\rho \in \{0.1, 0.01, 0.001\}$. Through this search, we found that $\alpha = \rho = 0.01$ yields the best result. Promoting strong sparsity (larger α and ρ) or weak sparsity (smaller α and ρ) resulted in degradation in forecasting accuracy. We use this setting for our experiments.

Varying m and τ_i. We also vary the value of the number of delayed variables and the time lag $(m, \tau_i) \in \{(6, 12i), (10, 7.2i), (12, 6i), (15, 4.8i), (20, 3.6i)\}$ while fixing the maximum time lag as 72 minutes. Introducing more delayed variables to the model only changes the input layer of the MLP and the specification of the internal layers is fixed. Fig. 10 shows the relative ℓ_2 errors for varying (m, τ_i) and that increasing the number of delayed variables tends to decrease the error up to a certain number, i.e, 12 or 15. However, m larger than 15 resulted in degradation of prediction accuracy.

With the fixed τ and varying m. We test the NDDEs with varying number of delayed variables (*i.e.*, $m \in \{1, 2, 3, 4, 5, 6\}$) where $\tau_i = 12i$ (min), i.e., the time lag is defined as 12 minutes. As depicted in Fig. 11, the increase in the number of delayed variables tends to result in

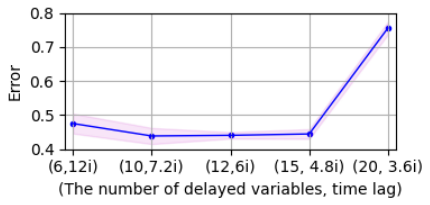


Figure 10: [Tsunami] Averaged relative ℓ_2 errors (blue line) for varying m (the number of delayed variables) and time lag τ_i. The magenta area is covered by the two standard deviations.

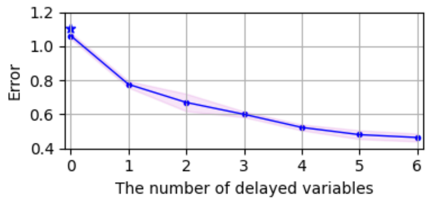


Figure 11: [Tsunami] Averaged relative ℓ_2 errors for varying m (the number of delayed variables) and the fixed time lag $\tau_i = 12i$. Star marker indicates the error of ANODE results.

Table 2: [Tsunami] Performance comparisons

MODEL	REL. ℓ_2 ERROR	MODEL SIZE
RNN	0.968 ± 0.0213	219K
LSTM	0.984 ± 0.0042	875K
GRU	0.909 ± 0.0101	656K
DAE	0.474 ± 0.0265	2,604K
VAE	0.461 ± 0.0111	3,297K
NODE	1.061 ± 0.0085	95K
ANODE	1.104 ± 0.0206	93K
NDDE (ours)	**0.439 ± 0.0118**	107K
LI-NDDE (ours)	**0.404 ± 0.0087**	163K

better performance in terms of prediction accuracy measured in mean-squared errors.

The figure also shows the elevation time series for the same test data considered in the experiments with NODEs. We plot the mean of the predictions as cyan dashed lines and the two standard deviations (as magenta areas). We use the same experimental procedure for forecasting (*i.e.*, solving IVP with the learned NDDEs). As opposed to the predictions made by NODEs, the predictions made by NDDEs match well with the ground-truth trajectories and capture the peak values and locations more accurately.

5.3 Tsunami Forecasting Performance

To evaluate, we train NDDEs with varying number of delayed variables and varying time lags on the tsunami dataset and measure those models' forecasting accuracy as the relative ℓ_2 error: $\|\boldsymbol{x}^{(\text{pred})} - \boldsymbol{x}\|_2 / \|\boldsymbol{x}\|_2$, where $\boldsymbol{x}^{(\text{pred})}$ and $\boldsymbol{x}$ denote the predicted and the ground-truth trajectories, respectively. We repeat the experiments 5 times with different initialization of model parameters.

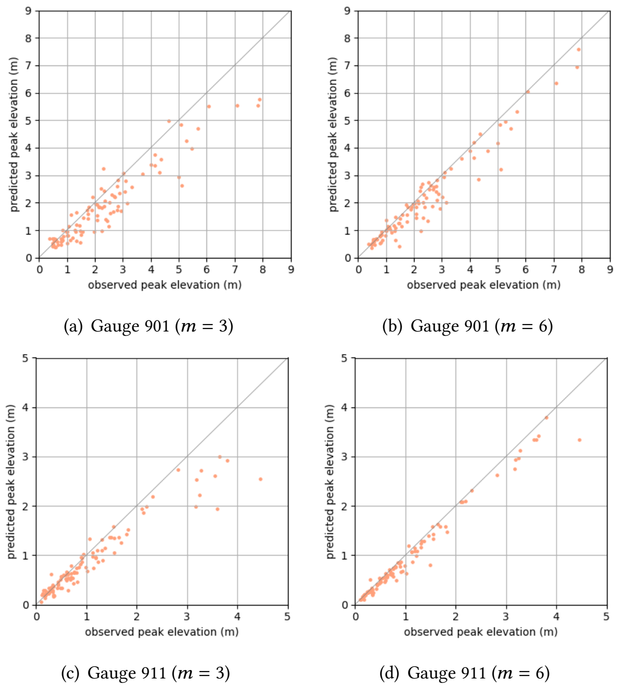


Figure 12: [Tsunami] Observed peak locations (horizontal axis) and predicted peak locations (vertical axis) for elevation time-series measured at Gauge 901 and 911.

Comparison to existing baselines. We also consider classical deep-learning approaches for time-series modeling and the most recent approaches in tsunami forecasting on the same dataset [20]: (1) classical DL approaches: recurrent neural networks (RNNs), long short term memory (LSTM) [12], and gated recurrent unit (GRU) [4], (2) the state-of-the-art Tsunami forecasting approaches [20]: denoising autoencoder (DAE) and variational autoencoder (VAE), (3) a variant of the proposed method: latent-input augmented NDDEs (LI-NDDEs). The LI-NDDEs (1) extract a latent code $\boldsymbol{c}^{(k)}$ from the kth data instance using a small-sized neural network and (2) augment it to the state $\boldsymbol{z}$ (i.e., $\boldsymbol{f}([\boldsymbol{z}, \boldsymbol{c}^{(k)}], t; \Theta)$. We extract a hidden feature by using an autoencoder, whose input is again the first n_{in} steps of the Gauge 702, and augmented the extracted latent code $\boldsymbol{c}$ to the dynamics. The autoencoder used here has the same input/output setting as the previously described baseline autoencoders [20], but is in a small scale (see Table 2 for the model size).

Table 2 shows the relative ℓ_2 errors obtained by using all baselines with the best performing hyperparameters (see **SM** for hyperparameter settings). The results essentially show that the our models outperforms the baseline in terms of prediction accuracy while keeping the model size small. Fig. 12 show the predicted peak values more aligned with the ground-truth trajectories at Gauge 901 and 911. Two sets of the predictions made separately by NDDEs with $m = 3$ and $m = 6$ are depicted. The NDDEs with larger m provide better predictions for peak values.

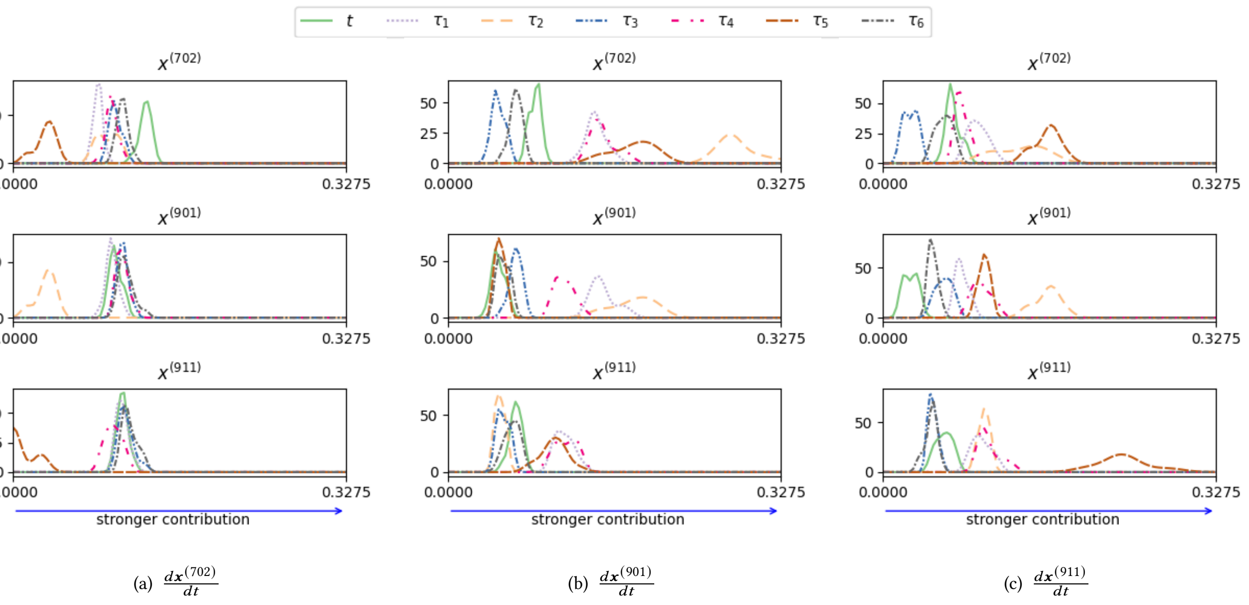


Figure 13: [Tsunami] Dependence structures of NDDE models with $m = 6$: the contributions from $x^{(702)}$ (top), $x^{(901)}$ (middle), and $x^{(911)}$ (bottom) to $\dot{x}^{(702)}$ (left), $\dot{x}^{(901)}$ (middle), and $\dot{x}^{(911)}$ (right). Contemporaneous and lagged dependence structures are indicated by t and $\{\tau_i\}_{i=1}^{6}$.

5.4 Identified Dependence Structures

In Fig. 13, we report the learned dependence structure. The magnitude of the column norms of the weight matrices $\{W_j^1\}$ in the input layer are again considered as the amount of contributions made by the corresponding input variables. We repeat the same experiments 15 times with different initialization, obtain the column norms from each run, and then apply the kernel density estimation based on Scott's factor [36] to the collected column norms.

We first focus on the plots on the diagonal of the 3×3 plot matrix. Gauge 702 has only small contributions from the delayed variables. The gauge observes the incoming wave and there is little reflections coming from the interior of the Sound, so it is expected that there are only small contributions to the result. Gauges 901 and 911 have higher contributions since they do possess dependent behavior coming from the reflections caused by the narrowing strait. This is especially pronounced for 901 which sits in Discovery bay and experiences significant sloshing due to the local topography (see Fig. 9). Next, we observe that the upper diagonal plots reveal larger contributions where as the lower-diagonal plots do not. This indicates that the delayed time-series of Gauge 702 causes those of Gauge 901 and 911 in a significant way, and that delayed time-series at Gauge 901 causes Gauge 911. The significant contributions agree well with the spatio-temporal progression of the physical wave itself.

The most significant delayed variables from Gauge 702 in predicting Gauge 901 are the delays τ_2 and τ_5, which correspond roughly to 24 minutes and 60 minutes of delay. Thus 30-60 minutes of the time-series from 702 is required to predict the time-series at Gauge 911, which agrees with the prediction results from [20]: while 30 minutes of time-series from Gauge 702 was sufficient to predict the wave height at Gauge 901, using 60 minutes improved the result significantly.

We also use the DYNOTEARS, PCMCI+, and Rhino APIs to find dependence structures. The first two methods produce the similar results that the strongest dependence on Gauges 901 and 911 is from Gauge 702 with τ_6, which is ~72 min after the wave reaches the Gauges 901 and 911, while not showing any significant dependence from other τ_i, suggesting that the result do not match with the spatio-temporal progression of the physical wave. Rhino also fails to identify physically consistent dependence structures, e.g., missing τ_5-delayed effect from Gauge 702 to Gauge 901.

6 CONCLUSION

This work presents a computational framework for time-series forecasting and dependence structure discovery. We leverage continuous-time neural networks with delayed variables. We adapt the score-based structure learning algorithm to settings where these models learn systems with delayed dynamics. We also propose a training algorithm for promoting sparse dependence structure in the parameter space. Once trained, we prune the parameters in the input layers and identify dependence structures from the remaining parameters.

We evaluate our method on three ODE benchmark problems: the Lorenz-96, Mackey–Glass, and Lotka–Volterra systems. We show our method's capabilities in accurate dynamics modeling, structure discovery, and resiliency to additive noise and anomalies. Moreover, we demonstrate a practical application of our method to tsunami forecasting. Our method produces highly accurate tsunami forecasting. We also identify the dependence structure of the time series obtained from the three gauges, which are physically-consistent and agree with the domain expert's interpretation.

REFERENCES

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. 2016. Layer normalization. In *NIPS 2016 Deep Learning Symposium*.

[2] Alexis Bellot, Kim Branson, and Mihaela van der Schaar. 2022. Neural graphical modelling in continuous-time: consistency guarantees and algorithms. In *International Conference on Learning Representations*.

[3] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. 2018. Neural ordinary differential equations. In *NeurIPS*. 6572–6583.

[4] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[5] Clawpack Development Team. 2020. Clawpack software. https://doi.org/10.17605/osf.io/kmw6h

[6] Emilien Dupont, Arnaud Doucet, and Yee Whye Teh. 2019. Augmented neural ODEs. *NeurIPS* 32 (2019).

[7] Wenbo Gong, Joel Jennings, Cheng Zhang, and Nick Pawlowski. 2023. Rhino: Deep Causal Temporal Relationship Learning with History-dependent Noise. In *The Eleventh International Conference on Learning Representations*.

[8] Clive WJ Granger. 1969. Investigating causal relations by econometric models and cross-spectral methods. *Econometrica: journal of the Econometric Society* (1969), 424–438.

[9] Julia Gusak, Larisa Markeeva, Talgat Daulbaev, Alexander Katrutsa, Andrzej Cichocki, and Ivan Oseledets. 2020. Towards Understanding Normalization in Neural ODEs. In *ICLR Workshop*.

[10] Trevor Hastie. 2009. *The elements of statistical learning: data mining, inference, and prediction.* Vol. 2. Springer.

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*. 770–778.

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

[13] Samuel I Holt, Zhaozhi Qian, and Mihaela van der Schaar. 2022. Neural Laplace: Learning diverse classes of differential equations in the Laplace domain. In *ICML*.

[14] Jeehyun Hwang, Jeongwhan Choi, Hwangyong Choi, Kookjin Lee, Dongeun Lee, and Noseong Park. 2021. Climate Modeling with Neural Diffusion Equations. In *ICDM*. IEEE, 230–239.

[15] Kookjin Lee and Eric J Parish. 2021. Parameterized neural ordinary differential equations: Applications to computational physics problems. *Proceedings of the Royal Society A* 477, 2253 (2021), 20210162.

[16] Kookjin Lee, Nathaniel Trask, and Panos Stinis. 2021. Machine learning structure preserving brackets for forecasting irreversible processes. *NeurIPS* 34 (2021), 5696–5707.

[17] Kookjin Lee, Nathaniel Trask, and Panos Stinis. 2021. Structure-preserving Sparse Identification of Nonlinear Dynamics for Data-driven Modeling. *arXiv preprint arXiv:2109.05364* (2021).

[18] Randall J. LeVeque, David L. George, and Marsha J. Berger. 2011. Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica* 20 (2011), 211–289. https://doi.org/10.1017/S0962492911000043

[19] Randall J LeVeque, Knut Waagan, Frank I González, Donsub Rim, and Guang Lin. 2016. Generating random earthquake events for probabilistic tsunami hazard assessment. In *Global Tsunami Science: Past and Future, Volume I*. Springer, 3671–3692.

[20] Christopher M Liu, Donsub Rim, Robert Baraldi, and Randall J LeVeque. 2021. Comparison of machine learning approaches for Tsunami forecasting from sparse observations. *Pure and Applied Geophysics* 178, 12 (2021), 5129–5153.

[21] Edward N Lorenz. 1996. Predictability: A problem partly solved. In *Proc. Seminar on predictability*, Vol. 1.

[22] Michael C Mackey and Leon Glass. 1977. Oscillation and chaos in physiological control systems. *Science* 197, 4300 (1977), 287–289.

[23] Diego Melgar. 2016. *Cascadia FakeQuakes waveform data and scenario plots.* https://doi.org/10.5281/zenodo.59943

[24] Diego Melgar. 2020. MudPy. https://doi.org/10.5281/zenodo.3703200

[25] Diego Melgar, Randall J LeVeque, Douglas S Dreger, and Richard M Allen. 2016. Kinematic rupture scenarios and synthetic displacement data: An example application to the Cascadia subduction zone. *Journal of Geophysical Research: Solid Earth* 121, 9 (2016), 6658–6674.

[26] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. 2018. Spectral Normalization for Generative Adversarial Networks. In *ICLR*.

[27] SØREN WENGEL MOGENSEN and NIELS RICHARD HANSEN. 2020. MARKOV EQUIVALENCE OF MARGINALIZED LOCAL INDEPENDENCE GRAPHS. *The Annals of Statistics* 48, 1 (2020), 539–559.

[28] Thibault Monsel, Onofrio Semeraro, Lionel Mathelin, and Guillaume Charpiat. 2023. Neural State-Dependent Delay Differential Equations. (2023).

[29] Roxana Pamfil, Nisara Sriwattanaworachai, Shaan Desai, Philip Pilgerstorfer, Konstantinos Georgatzis, Paul Beaumont, and Bryon Aragam. 2020. Dynotears: Structure learning from time-series data. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 1595–1605.

[30] Neal Parikh, Stephen Boyd, et al. 2014. Proximal algorithms. *Foundations and trends® in Optimization* 1, 3 (2014), 127–239.

[31] Sunghyun Park, Kangyeol Kim, Sookyung Kim, Joonseok Lee, Junsoo Lee, Jiwoo Lee, and Jaegul Choo. 2020. Hurricane Nowcasting with Irregular Time-step using Neural-ODE and Video Prediction. In *ICLR 2020 Workshop*. https://www.climatechange.ai/papers/iclr2020/21

[32] Adam Paszke, et al. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *NeurIPS*. https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[33] Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. 2019. Latent ordinary differential equations for irregularly-sampled time series. *NeurIPS* 32 (2019).

[34] Jakob Runge. 2020. Discovering contemporaneous and lagged causal relations in autocorrelated nonlinear time series datasets. In *Conference on Uncertainty in Artificial Intelligence*. PMLR, 1388–1397.

[35] Tim Salimans and Durk P Kingma. 2016. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. *NeurIPS* 29 (2016).

[36] David W Scott. 2015. *Multivariate Density Estimation: Theory, Practice, and Visualization.* John Wiley & Sons.

[37] Alex Tank, Ian Covert, Nicholas Foti, Ali Shojaie, and Emily B Fox. 2021. Neural Granger causality. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 44, 8 (2021), 4267–4279.

[38] Markus Thill, Wolfgang Konen, and Thomas Bäck. 2020. MarkusThill/MGAB: The Mackey-Glass Anomaly Benchmark. *Version v1. 0.1. Zenodo. doi* 10 (2020).

[39] Xiang-Ping Yan and Wan-Tong Li. 2006. Hopf bifurcation and global periodic solutions in a delayed predator–prey system. *Appl. Math. Comput.* 177, 1 (2006), 427–445.

[40] Qunxi Zhu, Yao Guo, and Wei Lin. 2020. Neural Delay Differential Equations. In *ICLR*.

[41] Qunxi Zhu, Yifei Shen, Dongsheng Li, and Wei Lin. 2022. Neural Piecewise-Constant Delay Differential Equations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9242–9250.