SNN-PDE: Learning Dynamic PDEs from Data with Simplicial Neural Networks

Jae Choi,[1] Yuzhou Chen,[2] Hugo K. Lee,[3] Hyun Kim,[4] Yulia R. Gel[5,6]

[1]Department of Computer Science, University of Texas at Dallas
[2]Department of Computer and Information Sciences, Temple University
[3]Jet Propulsion Laboratory, California Institute of Technology
[4]Air Resources Laboratory, National Oceanic and Atmospheric Administration
[5]Department of Mathematical Sciences, University of Texas at Dallas
[6]National Science Foundation

jaewon.choi@utdallas.edu, yuzhou.chen@temple.edu, huikyo.lee@jpl.nasa.gov, hyun.kim@noaa.gov, ygl@utdallas.edu

Abstract

Dynamics of many complex systems, from weather and climate to spread of infectious diseases, can be described by partial differential equations (PDEs). Such PDEs involve unknown function(s), partial derivatives, and typically multiple independent variables. The traditional numerical methods for solving PDEs assume that the data are observed on a regular grid. However, in many applications data records are irregularly spaced. Furthermore, in problems involving prediction analytics such as forecasting wildfire smoke plumes, the primary focus may be on a set of irregular locations associated with urban development. In recent years, deep learning (DL) methods and, in particular, graph neural networks (GNNs) have emerged as a new promising tool that can complement traditional PDE solvers in scenarios of the irregular spaced data, contributing to the newest research trend of physics informed machine learning (PIML). However, most existing PIML methods tend to be limited in their ability to describe higher dimensional structural properties exhibited by real world phenomena, especially, ones that live on manifolds. To address this fundamental challenge, we bring the elements of the Hodge theory and, in particular, simplicial convolution defined on the Hodge Laplacian to the emerging nexus of DL and PDEs. Contrary to conventional Laplacian and the associated convolution operation, the simplicial convolution allows us to rigorously describe diffusion across higher order structures and to better approximate the complex underlying topology and geometry of the data. The new approach, Simplicial Neural Networks for Partial Differential Equations (SNN PDE) offers a computationally efficient yet effective solution for time dependent PDEs. Our studies of synthetic data and wildfire processes show that SNN PDE improves upon state of the art baselines in handling unstructured grids and irregular time intervals of complex physical systems and offers competitive forecasting capabilities for weather and air quality forecasting.

Introduction

Physics-informed machine learning (PIML) is the recently emerged research trend and (arguably) one of the most actively developing research directions in scientific machine learning. PIML allows to describe various complex phenomena in physical and biological systems, governed by multiscale dynamic processes (Shukla et al. 2022). In broader terms, the idea of PIML is to complement physical laws and the more traditional mathematical methods that describe them, with the ML tools such as convolutional neural networks (CNNs) and graph neural networks (GNNs). One of the primary advantages of such fusion of methods is that PIML allows for modeling various dynamic phenomena using unstructured data at arbitrary space-time locations, with applications ranging from oceanography to biomedical imaging and biosurveillance.

Indeed, the traditional numerical methods to solve partial differential equations (PDEs) assume that the data input arrives on a regular grid. However, in many real-world studies such as air pollution monitoring and infectious disease tracking, the data records are intrinsically irregularly spaced both in space and time. Furthermore, conventional PDE solutions are often limited to course-grained representations due to computational costs. This in turn restricts applicability of such solutions to studies requiring finer-resolution grids such as the ones related to the impact of smoke plumes and wildfires on health outcomes.

The emerging PIML approach to tackle this problem is to employ DL tools such as CNNs and GNNs to approximate differential operators (Iakovlev, Heinonen, and Lähdesmäki 2020; Brandstetter, Worrall, and Welling 2022; Moshe Eliasof and Treister 2021; Brandstetter, Welling, and Worrall 2022; Horie and Mitsume 2022; Rakhoon Hwang and Hwang 2022). However, such techniques are largely based on the conventional convolution operation, focusing only on the pairwise interactions among the system entities, and as a result they tend to overlook the important information delivered by higher-order geometric structures exhibited by real world phenomena, especially, ones that live on manifolds.

We propose to address this fundamental gap, by introducing the elements of the Hodge theory and, in particular, simplicial convolution defined on the Hodge-Laplacian to DL-assisted solutions of PDEs. While the Hodge theory has been used in scientific ML before, for example, for regularization of GNNs (Seo and Liu 2019), it has never been used in conjunction with ML-enabled PDE solvers. Contrary to the conventional convolution based on the Laplacian operator, the simplicial convolution induced by the Hodge-Laplacian enables for systematic and rigorous characterization of diffusion across higher-order structures described via

simplicial complexes and, in turn, leads to a better approximation of the complex underlying topology and geometry of the data. The resulting Simplicial Neural Networks for Partial Differential Equations (SNN-PDE) offers a computationally efficient yet effective solution for time-dependent PDEs for sparse data records measured at arbitrary locations and time points. Our experiments on a wide range of synthetic dynamic systems and the wildfire data provided by the National Oceanic and Atmospheric Administration (NOAA) demonstrate that SNN-PDE delivers significant gains upon state-of-the-art baselines in handling unstructured grids and irregular time intervals of complex dynamical systems.

The key contributions of our approach can be summarized as follows:

- We introduce the concepts of Hodge theory and the associated multi-channel simplicial convolution on Hodge-Laplacian to the DL-assisted solutions of time-dependent PDEs. Such an approach allows for extracting important topological and geometric properties from the unstructured data.
- We propose a dual graph convolutional encoder that accelerates and improves message passing across the graph, enabling more efficient training for problems involving (hidden) long-range dependencies.
- The new Simplicial Neural Networks for Partial Differential Equations (SNN-PDE) model shows competitive results both on synthetic datasets and NOAA wildfire data, outperforming state-of-the-art baselines in solving time-dependent PDEs on unstructured grids and irregular time intervals.
- The proposed Hodge-based approach opens a new path for the DL-enabled solutions of PDEs on manifolds, which yet remains uncharted territory.

Related Work

Graph Neural Networks and Physics-Informed Deep Learning Recently, Graph Neural Network (GNN) has emerged as a primary tool for graph classification (Zhou et al. 2020; Xia et al. 2021). Different methods have been proposed to capture the structural and semantic properties of graphs. For instance, Weisfeiler–Lehman (WL) (Shervashidze et al. 2011) proposes an efficient family of kernels for large graphs with discrete node labels, and Shortest Path Hash Graph Kernel (HGK-SP) (Morris et al. 2016) derives kernels for graphs with continuous attributes from discrete ones. Graph Convolutional Network (GCN) (Kipf and Welling 2017) extends the convolution operation from regular grids to graphs. Message Passing Neural Networks (MPNN) (Gilmer et al. 2017) operate on molecules as graphs by treating atoms as nodes and bonds as edges and its core idea is to iteratively aggregate neighbor representations. Graph Attention Networks (Veličković et al. 2018) introduces and applies the attention mechanism to aggregate node features with the learned weights. EdgeConv (Wang et al. 2019) is proposed based on graph convolution operation and learns graph structural information by extracting edge feature information between a node and its neighbors. Furthermore, GNN frameworks have demonstrated proficiency in modeling intricate physical domains, e.g., fluids, rigid solids, and deformable materials interacting with one another (Sanchez-Gonzalez et al. 2020). NeuralPDE (Dulny, Hotho, and Krause 2022) can inherently learn continuous dynamics with arbitrary time discretizations by using convolutional neural networks. PDE-Net (Long et al. 2018) is a feedforward neural networks that uses the convolutional kernel to approximate a differential operator over the dynamics of complex systems. GNN-PDE (Iakovlev, Heinonen, and Lähdesmäki 2020) introduces continuous-time representation and learning of the dynamics of PDE-driven systems based GNNs. MP-PDE (Brandstetter, Worrall, and Welling 2021) proposes to solve PDEs with an end-to-end neural solver (i.e., based on message passing neural network).

Simplicial Neural Networks Modeling higher-order interactions on graphs is an emerging direction in graph representation learning. While the role of higher-order structures for graph learning has been documented for a number of years (Agarwal, Branson, and Belongie 2006; Johnson and Goldring 2012) and involves such diverse applications as graph signal processing in image recognition (Dong et al. 2019), dynamics of disease transmission and biological network, integration of higher-order graph substructures into DL on graphs has emerged only in 2020. As shown by Benson et al. (2018); Schaub et al. (2020), higher-order network structures can be leveraged to boost graph learning performance. Indeed, several most recent approaches (Ebli, Defferrard, and Spreemann 2020; Roddenberry, Glaze, and Segarra 2021; Bodnar et al. 2021; Chen, Gel, and Poor 2022) propose to leverage simplicial information to perform neural networks on graphs. However, neither of these Simplicial Neural Networks (SNNs) are integrated with a topology-based graph convolution layer allowing us to learn both time-aware persistent topological features and simplicial geometry of graphs. However, to the best of our knowledge, none of these so-called Simplicial Neural Networks (SNNs) is used as an approximate representation of the solution of the PDE. In this paper, SNN-PDE is proposed to address this challenge.

Background on Hodge Theory

We start from providing the necessary mathematical background on manifolds and Hodge theory that our key ideas are based upon.

Definition 1 *Let M be a real smooth manifold and let T_xM be the tangent space at $x \in M$. If at each $x \in M$ M is equipped with a positive definite inner product on T_xM, we say that M is a Riemann manifold.*

That is, loosely speaking, manifold M can be viewed as a topological space such that every point $x \in M$ has a neighborhood homeomorphic to an Euclidean space, while a smooth manifold ensures that the concepts of tangent vectors and differentiable functions are well defined. Now, we turn to differential forms that offer a coordinate free approach to multivariable calculus and can be thought of p-dimensional volumes that allow us to integrate over manifolds.

Definition 2 *A differential form of degree p (i.e., p-form) on M defines a linear functional β such that $\beta : \Lambda^p T_x^* M \to \mathbb{R}$, where Λ is the exterior product and $T_x^* M$ is dual to $T_x M$.*

While studying manifolds, especially in conjunctions with PDEs on manifolds, we need to connect the differentiability and global geometry of the manifold. This task can be approached using the Hodge theory. Before we introduce the Hodge-Laplacian on manifold, we define the Hodge $*$ operator that provides duality between differential forms.

Definition 3 *Let M be a compact Riemann manifold of dimension d. Then the Hodge $*$ operator is an isomorphism between the space of smooth differential p-forms $\Omega_p(M)$ and the space smooth differential $d-p$-forms $\Omega_{d-p}(M)$ on M, that is,*

$$* : \Omega_p(M) \to \Omega_{d-p}(M), \tag{1}$$

and the associated inner L^2 inner product on $\Omega_p(M)$ is defined by

$$(\alpha, beta) = \int_M \alpha \wedge *\beta. \tag{2}$$

Armed with these notations, we can now define the exterior derivative $d : \Omega_p(M) \to \Omega_{p+1}(M)$ and its adjoint operator $d*$ with respect to the inner product (2), i.e. $d* : \Omega_{p+1}(M) \to \Omega_p(M)$ and $(d\alpha, \beta) = (\alpha, d * \beta)$, where $\alpha \in \Omega_p(M)$ and $\beta \in \Omega_{p+1}(M)$. Linear operator $d*$ is also referred to as *codifferential* operator.

Definition 4 *The Hodge-Laplacian operator Δ_H is a linear operator on the space of differential forms $\Omega_p(M)$, i.e. $\Delta_H : \Omega_p(M) \to \Omega_p(M)$, and is defined by*

$$\Delta_H = dd * \alpha + d * d\alpha. \tag{3}$$

While it looks mathematically involved, the core idea of Hodge-Laplacian is that it offers a natural generalization of the Laplacian of a function to Laplacian of differential forms, allowing us to study manifolds using PDEs.

Methodology

Consider a smooth $(d-1)$-dimensional manifold $\Gamma \in \mathbb{R}^d$. Let $p \in \Gamma$ be a point with coordinates $(p_1, \dots, p_{d-1})$. We endow Γ with the metric $\mathfrak{g}$ that is inherited from the embedding Euclidean space. That is, for $p \in \Gamma$, $\mathfrak{g}_p(u, v) = \sum u_i v_i$, where u, v belong to the tangent space $T_p\Gamma$ at p. (Throughout the paper, for simplicity and without loss of generality, we consider a case of $d = 3$.)

Consider a continuously differentiable function $f : \Gamma \to \mathbb{R}$. Then its corresponding gradient operator is given by

$$\nabla_\Gamma f = \sum_i a^i \frac{\partial f}{\partial p_i}, \tag{4}$$

where $a^i = \sum_j g^{ij} \frac{\partial f}{\partial p_j}$ and g^{ij} are the elements of the inverse of the metric tensor $G = \{g_{ij}\}_{i,j=1}^2$ which is associated with the metric $\mathfrak{g}$ (that is, $g_{ij} = \mathfrak{g}\big(\frac{\partial}{\partial p_1} + \frac{\partial}{\partial p_2}\big)$, $1 \le i, j \le 2$).

In turn, its corresponding Hodge-Laplacian on a local basis $\big\{\frac{\partial}{\partial p_i}\big\}$ with metric $\mathfrak{g}$ is given by

$$\Delta_H f = -\sum_{ij} \frac{1}{|G|} \frac{\partial}{\partial p_i} \left(|G| g^{ij} \frac{\partial f}{\partial p_j} \right). \tag{5}$$

That is, on a two-dimensional manifold Γ for a scalar function f, Hodge-Laplacian $\Delta_H f = dd * f + d * df$ reduces to the surface Laplacian times -1 $-d * df$, since $d * f = 0$ for a 0-form f.

Armed with these differential operators $\nabla_\Gamma f$ and $\Delta_H f$, we turn to PDEs on manifolds. In particular, we consider a time-dependent PDE

$$\frac{\partial u}{\partial t}(\boldsymbol{s}, t) = F\big(\boldsymbol{s}, u, \nabla_{\Gamma \boldsymbol{s}} u, \Delta_{H \boldsymbol{s}} u\big), \boldsymbol{s} \in \Gamma, \tag{6}$$

where $\nabla_{\Gamma s} u$ and $\Delta_{H s}$ are the gradient and Hodge-Laplacian evaluated at $\boldsymbol{s}$.

We can now follow the ideas of Liang and Zhao (2013) and Iakovlev, Heinonen, and Lähdesmäki (2020) on the discretization of the differential operators. The key approach here is to view (6) as a system of locally coupled differential equations. In particular, we select $\mathcal{N}$ locations $\{\boldsymbol{s}_1, \dots, \boldsymbol{s}_\mathcal{N}\}$ and consider a discretized version of F, such that the approximation $\hat{F}$ at $\boldsymbol{s}_i$ depends on the system's state u_i at x_i as well as locations $\{\boldsymbol{s}_{i1}, \dots, \boldsymbol{s}_{i\mathcal{N}(i)}\}$ and their states $\{u_{i1}, \dots, u_{i\mathcal{N}(i)}\}$ in a certain vicinity of $\boldsymbol{s}_i$, where $\mathcal{N}(i)$ is the number of points (or nodes) in the neighborhood of $\boldsymbol{s}_i$. For instance, in case of Γ is $\mathbb{R}$, for $i = 1, \dots, \mathcal{N}$ we get

$$\frac{du(\boldsymbol{s}_i, t)}{dt} = F(\boldsymbol{s}_i, \dots, \boldsymbol{s}_{\mathcal{N}(i)}, u_i, \dots, u_{\mathcal{N}(i)}), \tag{7}$$

The resulting system of ordinary differential equations (ODEs) can be solved by the appropriate ODE solver such Runge–Kutta methods.

Note that the local neighborhoods and $\mathcal{N}(i)$ vary among locations. As such, $\hat{F}$ depends both on the location $\boldsymbol{s}_i$ and all $\mathcal{N}(i)$ locations in its vicinity. To address this problem, Liang and Zhao (2013) proposes to use k-nearest neighbors (kNN) and the moving least squares (MLS) and develop a local approximation approach both for manifold Γ and the differential operators ∇_Γ and Δ_H. In turn, Iakovlev, Heinonen, and Lähdesmäki (2020) who consider PDEs on $\mathbb{R}^d$, suggest to represent $\hat{F}$ through $\hat{F}_\theta$ where $\hat{F}_\theta = f_U(f_M(\boldsymbol{s}_i, \dots, \boldsymbol{s}_{\mathcal{N}(i)}, u_i, \dots, u_{\mathcal{N}(i)}))$ (where f_U and f_M are node update and message functions respectively) is the message passing neural networks (MPNNs) (Gilmer et al. 2017).

Given that we are particularly interested in cases when F may live on manifold Γ, we propose to employ simplicial graph neural networks (SGNN) to learn F. The idea is here intuitive as GNNs and SNNs are based on a convolution defined on the Hodge Laplacian and, hence, SGNN is expected to account for the inherently geometric structure of F.

Remark 1 *Note that a more formal treatment of PDEs on manifolds requires a local approximation of Γ. However, this goes beyond the limits of a single paper, and our idea is to test how simplicial convolution defined on the Hodge Laplacian can help in addressing this class of problems, including a more traditional case of PDEs on $\mathbb{R}^d$.*

Next, we present Simplicial Neural Networks for Partial Differential Equations (SNN-PDE), a novel neural network architecture based on the Hodge theory that contains two modules, i.e., dual graph convolutional encoder and multi-channel simplicial convolutional encoder.

Dual Graph Convolutional Encoder

We consider a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{S})$, where $\mathcal{V}$ is the set of nodes ($|\mathcal{V}| = \mathcal{N}$) and $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges ($|\mathcal{E}| = \mathcal{M}$). Furthermore, each node v_i is associated with a d_c-dimensional vector of attributes $\boldsymbol{s}_i$ which is the i-th row of matrix $\boldsymbol{S} \in \mathbb{R}^{\mathcal{N} \times d_c}$ (where d_c is the input feature dimension). For measurement positions, $d_c = 2$. Connectivity of $\mathcal{G}$ can be encoded in a form of an adjacency matrix $\boldsymbol{A} \in \mathbb{R}^{\mathcal{N} \times \mathcal{N}}$ with entries $[\boldsymbol{A}]_{ij} = 1$ if nodes i and j are connected and 0, otherwise. To systematically incorporate both global topological information and node features from the spatial dimension, here we introduce the components of our proposed dual graph convolutional encoder as follows. Figure 1 illustrates the message passing steps in the multi-hop grid-structural graph convolution.

Multi-Hop Grid-Structural Graph Convolution The multi-hop grid-structural graph convolution is designed to capture higher-order structural information from the predefined graph topology by considering the multi-hop neighborhood of each node. In this context, we utilize a random walk-based graph convolution approach to realize this module. Specifically, given a normalized graph Laplacian $\boldsymbol{L}_{\mathcal{G}} = \tilde{\boldsymbol{D}}^{-\frac{1}{2}} \tilde{\boldsymbol{A}} \tilde{\boldsymbol{D}}^{\frac{1}{2}}$ (where $\tilde{\boldsymbol{A}} = \boldsymbol{A} + \boldsymbol{I}$ is the adjacency matrix with self-loop added to each node and $\tilde{\boldsymbol{D}}_{uu} = \sum_v \tilde{\boldsymbol{A}}_{uv}$ is a diagonal matrix), the weighted multi-hop grid-structural graph Laplacian $\tilde{\boldsymbol{L}}_{\mathcal{G}}^{[r]}$ is defined as

$$\tilde{\boldsymbol{L}}_{\mathcal{G}}^{[r]} = \sum_{\tau=1}^{r} \alpha_\tau \times \boldsymbol{L}_{\mathcal{G}}^{\tau}, \tag{8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_\tau, \ldots, \alpha_r)$ is a set of parameters (where $r \geq 2$) and α_τ denotes the weight that controls the importance of each Laplacian power, and $\boldsymbol{L}_{\mathcal{G}}^{\tau}$ denotes the Laplaican $\boldsymbol{L}_{\mathcal{G}}$ multiplied by itself τ times. The proposed weighted multi-hop grid-structural graph Laplacian is to establish a flexible framework to capture the global topological information and partial similarities between neighborhoods with various radii τ. Finally, the multi-hop grid-structural graph convolutional layer can be formulated as

$$\boldsymbol{Z}_{\mathcal{G}}^{(\ell+1)} = \sigma(\tilde{\boldsymbol{L}}_{\mathcal{G}}^{[r]} \boldsymbol{Z}_{\mathcal{G}}^{(\ell)} \boldsymbol{\Theta}_{\mathcal{G}}), \tag{9}$$

where $\sigma(\cdot)$ stands for a nonlinear activation function, $\boldsymbol{Z}_{\mathcal{G}}^{(\ell)}$ and $\boldsymbol{Z}_{\mathcal{G}}^{(\ell+1)}$ are the input and output activations for layer ℓ (where $\boldsymbol{Z}_{\mathcal{G}}^{(0)} = \boldsymbol{S} \in \mathbb{R}^{\mathcal{N} \times d_c}$), and $\boldsymbol{\Theta}_{\mathcal{G}}^{(\ell)} \in \mathbb{R}^{d_\ell^{\mathcal{G}} \times d_{\ell+1}^{\mathcal{G}}}$ is the ℓ-th layer's trainable weights.

Multi-Hop Adaptive Graph Convolution Given the node embedding dictionary $\boldsymbol{E}^{\phi} = (e_1^{\phi}, e_2^{\phi}, \ldots, e_N^{\phi}) \in \mathbb{R}^{N \times d_e}$ (where $x_u^{\phi} \in \mathbb{R}^{d_e}$ and d_e is the dimension of node embedding), we aim to seek a non-negative function $\boldsymbol{\Xi}_{u,v} = \mathscr{G}(w_u^{\phi}, w_v^{\phi})$ which represents the pairwise similarity between any two nodes u and v, and adaptively learn the topology of the graph. Concretely, the multiplication between $\boldsymbol{E}^{\phi}$ and $(\boldsymbol{E}^{\phi})^{\top}$ can (i) give a sum pooling of second-order features from the outer product of all the embedding vector pairs (e_u^{ϕ}, e_v^{ϕ}) and (ii) infer the hidden spatial dependencies of nodes

$$\boldsymbol{\Xi}_{uv} = \mathscr{G}(e_u^{\phi}, e_v^{\phi}) = \frac{\exp\left(\text{ReLU}(e_u^{\phi}(e_v^{\phi})^{\top}\right)}{\sum_{u=1}^{N} \exp\left(\text{ReLU}(e_u^{\phi}(e_v^{\phi})^{\top}\right)},$$

where $\text{ReLU}(\cdot) = \max(0, \cdot)$ is a nonlinear activation function, which is used to eliminate weak connections proactively, and the role of the softmax function is applied to normalized the learned graph $\boldsymbol{\Xi}$. Similar to the multi-hop grid-structural graph convolution, to enhance the representational power of graph convolution, we define the weighted multi-hop adaptive graph Laplacian $\tilde{\boldsymbol{\Xi}}^{[r']}$ as

$$\tilde{\boldsymbol{\Xi}}^{[r']} = \sum_{\tau=1}^{r'} \beta_\tau \times \boldsymbol{\Xi}^{\tau}, \tag{10}$$

where $\boldsymbol{\beta} = (\beta_1, \ldots, \beta_\tau, \ldots, \beta_{r'})$ is a set of parameters (where $r' \geq 2$) which controls contributions of the different adaptive graph Laplacian from one-hop to multiple hops for the graph representation learning. Armed with the multi-hop adaptive graph Laplacian as defined above, we define a graph convolution working on the adaptive graph structure representation as follows

$$\boldsymbol{Z}_{\mathcal{A}}^{(\ell+1)} = \sigma(\tilde{\boldsymbol{\Xi}}_{\mathcal{A}}^{[r]} \boldsymbol{Z}_{\mathcal{A}}^{(\ell)} \boldsymbol{\Theta}_{\mathcal{A}}), \tag{11}$$

where $\boldsymbol{Z}_{\mathcal{A}}^{(\ell)}$ and $\boldsymbol{Z}_{\mathcal{A}}^{(\ell+1)}$ are the input and output activations of the ℓ-th layer (where $\boldsymbol{Z}_{\mathcal{A}}^{(0)} = \boldsymbol{S} \in \mathbb{R}^{\mathcal{N} \times d_c}$), and $\boldsymbol{\Theta}_{\mathcal{A}}^{(\ell)} \in \mathbb{R}^{d_\ell^{\mathcal{A}} \times d_{\ell+1}^{\mathcal{A}}}$ is the ℓ-th layer's trainable weights.

Multi-Channel Simplicial Convolutional Encoder

Spatio-temporal grids often reveal intricate dependencies within their substructures, which are far more complex than what dyadic interactions or pairwise node relations can depict. The Hodge Theory offers a systematic approach to address these higher-order polyadic interactions. Specifically, the discrete Hodge theory extends the concept of the conventional combinatorial graph Laplacian, i.e., describing diffusion from one node to another via edges on graph $\mathcal{G}$ to encapsulate diffusion across higher-order substructures of $\mathcal{G}$. These substructures can be represented as k-simplices of $\mathcal{G}$. Over the past few years, convolutional architectures rooted in the principles of Hodge theory and designed for simplicial complexes have emerged as a novel trend in graph neural networks. However, they have not been applied to dynamic data. Our objective is to weave in the concept of simplicial convolution and introduce Hodge Laplacians at different dimensions to capture the varying dependencies between simplices. We design the higher-order graph convolutional encoder to (i) perform higher-order convolutions, (ii) encode multi-scale higher-order information, and (iii) mix different dimensions in simplicial complexes and provide an

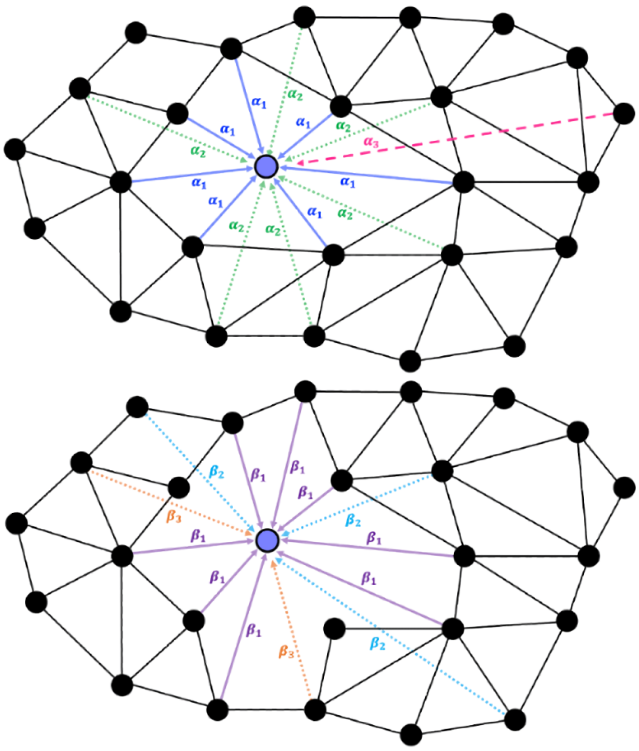

Figure 1: The illustration of message passing steps in the dual graph convolutional encoder over a grid. *Left*: multi-hop grid-structural graph convolution; *Right*: multi-hop adaptive graph convolution.

efficient way to explore each dimension for graph representation learning.

Since the goal is to predict the state of each node (i.e., 0-simplex), we propose a novel multi-channel simplicial convolutional layer that performs simplicial convolution operations over multi-dimensional simplices. Combined with the attention mechanism, different higher-order information can be adequately fused. Specifically, we utilize the simplicial convolution operation to extract the node embedding from k-simplex as follows

$$\boldsymbol{Z}_{k,\mathcal{S}}^{(\ell+1)} = (\boldsymbol{B}_1^\top \cdots \boldsymbol{B}_k^\top)(\sigma(\boldsymbol{L}_k \boldsymbol{Z}_{k,\mathcal{S}}^{(\ell)} \boldsymbol{\Theta}_{k,\mathcal{S}}^{(\ell)})), \tag{12}$$

where $\boldsymbol{L}_k$ denotes the k-Hodge Laplacian, $\boldsymbol{B}_k$ is the incidence matrix representation of the boundary operator ∂_k (for more details, please refer to Appendix A), $\boldsymbol{\Theta}_{k,\mathcal{S}}^{(\ell)}$ stands for the trainable weight, and $\boldsymbol{Z}_{k,\mathcal{S}}^{(\ell)}$ and $\boldsymbol{Z}_{k,\mathcal{S}}^{(\ell+1)}$ are input and output activations of the ℓ-th simplicial convolutional layer for the k-simplex. For the $\boldsymbol{Z}_{k,\mathcal{S}}^{(0)}$, for instance, when $k=1$, $\boldsymbol{Z}_{1,\mathcal{S}}^{(0)}$ represents the edge feature matrix, i.e., $\boldsymbol{z}_{uv,1,\mathcal{S}}^{(0)} = f(\boldsymbol{s}_u, \boldsymbol{s}_v)$; when $k=2$, $\boldsymbol{Z}_{2,\mathcal{S}}^{(0)}$ is the triangle feature matrix, i.e., $\boldsymbol{z}_{uvw,2,\mathcal{S}} = f(\boldsymbol{s}_u, \boldsymbol{s}_v, \boldsymbol{s}_w)$, where the $f(\cdot)$ is an aggregation functions (e.g., summation, maximum, and mean). After conducting the above simplicial convolution operations over $\mathcal{K}$ different simplexes, we have $\mathcal{K}$ embeddings, i.e., $\{\boldsymbol{Z}_{1,\mathcal{S}}^{(\ell+1)}, \ldots, \boldsymbol{Z}_{k,\mathcal{S}}^{(\ell+1)}, \ldots, \boldsymbol{Z}_{\mathcal{K},\mathcal{S}}^{(\ell+1)}\}$ (where $\mathcal{K} \geq 1$). Considering the node state can be correlated with one of them or even their combinations, we use the attention mechanism to focus on the importance of task relevant parts of the learned representations for decision making, i.e., $(\rho_1, \ldots, \rho_k, \ldots, \rho_\mathcal{K}) = Att(\boldsymbol{Z}_{1,\mathcal{S}}, \ldots, \boldsymbol{Z}_{k,\mathcal{S}} \ldots, \boldsymbol{Z}_{\mathcal{K},\mathcal{S}})$ (for the sake of simplicity, we omit the superscript $(\ell+1)$). In practice, we compute the attention coefficient as follows

$$\begin{aligned}\rho_k &= \text{softmax}_i(\Upsilon_{\text{Att}} \tanh(\boldsymbol{\Phi} \boldsymbol{Z}_{k,\mathcal{S}})) \\ &= \frac{\exp(\Upsilon_{\text{Att}} \tanh(\boldsymbol{\Phi} \boldsymbol{Z}_{k,\mathcal{S}}))}{\sum_{k \in \{1,\ldots,\mathcal{K}\}} \exp(\Upsilon_{\text{Att}} \tanh(\boldsymbol{\Phi} \boldsymbol{Z}_{k,\mathcal{S}}))},\end{aligned} \tag{13}$$

where $\Upsilon_{\text{Att}} \in \mathbb{R}^{1 \times d_{\text{out}}}$ is a linear transformation, $\boldsymbol{\Phi}$ is the trainable weight matrix, and the softmax function is used to normalize the attention vector. Then, we obtain the final embedding Z by combining all embeddings

$$\boldsymbol{Z}_\mathcal{S} = \rho_1 \times \boldsymbol{Z}_{1,\mathcal{S}} + \cdots + \rho_k \times \boldsymbol{Z}_{k,\mathcal{S}} + \cdots + \rho_\mathcal{K} \times \boldsymbol{Z}_{\mathcal{K},\mathcal{S}}.$$

The framework of our multi-channel simplicial convolutional encoder is shown in Figure 2. We then combine the embeddings from both dual graph convolutional encoder and multi-channel simplicial convolutional encoder.

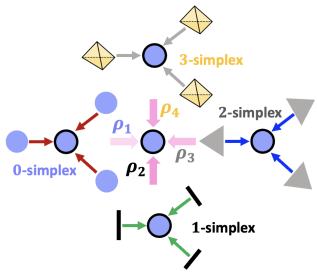

Figure 2: Schematic diagram of message passing steps in the multi-channel simplicial convolutional encoder.

$$\boldsymbol{Z}_{\hat{F}_\theta} = \gamma_\mathcal{G} \times \boldsymbol{Z}_\mathcal{G} + \gamma_\mathcal{A} \times \boldsymbol{Z}_\mathcal{A} + \gamma_\mathcal{S} \times \boldsymbol{Z}_\mathcal{S}, \tag{14}$$

where $\gamma_\mathcal{G}, \gamma_\mathcal{A}$, and $\gamma_\mathcal{S}$ are hyperparameters which control the influence of different modules. Lastly, we use the final node embeddings $\boldsymbol{Z}_{\hat{F}_\theta}$ to evaluate the PDE surrogate and we use our SNN-PDE model to present $\hat{F}_\theta$ (where θ represent parameters of the SNN-PDE) as

$$\begin{aligned}\frac{d\hat{u}(\boldsymbol{s}_i, t)}{dt} &= \hat{F}_\theta(\{\boldsymbol{s}\}_i^{\mathcal{N}(i)}, \{u\}_i^{\mathcal{N}(i)}) \\ &= \text{SNN-PDE}(\{\boldsymbol{s}\}_i^{\mathcal{N}(i)}, \{u\}_i^{\mathcal{N}(i)}) \\ &= \boldsymbol{z}_{i,\hat{F}_\theta},\end{aligned}$$

which can be used to estimate states $\hat{\boldsymbol{u}}(t) = (\hat{u}(\boldsymbol{s}_1, t), \hat{u}(\boldsymbol{s}_2, t), \ldots, \hat{u}(\boldsymbol{s}_\mathcal{N}, t))$ (where $\{\boldsymbol{s}\}_i^{\mathcal{N}(i)}$ denotes $\{\boldsymbol{s}_i, \ldots, \boldsymbol{s}_\mathcal{N}\}$ and $\{u\}_i^{\mathcal{N}(i)}$ represents $\{u_i, \ldots, u_{\mathcal{N}(i)}\}$).

Experimental Study

Datasets We validate our proposed SNN-PDE model on both synthetic and real-world datasets. On (i) synthetic

Model	ConvDiff_N750	ConvDiff_N1500	ConvDiff_N3000	Heat	Burgers
CNN	0.367±0.062	0.273±0.087	0.176±0.049	0.184±0.063	0.571±0.073
ResNet	0.292±0.023	0.259±0.056	0.170±0.025	0.217±0.034	0.467±0.096
EdgeConv	0.240±0.038	0.238±0.032	0.170±0.045	0.289±0.061	0.459±0.050
PDE-Net 2.0	0.319±0.094	0.274±0.062	0.205±0.016	0.029±0.077	0.447±0.112
GNN-PDE	0.119±0.015	0.078±0.012	0.071±0.013	0.378±0.317	0.330±0.010
R-MPGNN	0.150±0.043	0.102±0.057	0.079±0.036	0.117±0.018	0.366±0.070
EA-GNN	0.191±0.014	0.255±0.048	0.209±0.015	0.137±0.021	0.413±0.036
M-GNN	0.162±0.023	0.232±0.003	0.206±0.016	0.052±0.036	0.398±0.026
SNN-PDE (ours)	***0.108±0.010	***0.072±0.004	***0.049±0.006	***0.022±0.006	***0.306±0.050

Table 1: Performance on dynamic physical systems. The best results are given in bold while the best performances achieved by the runner-ups are underlined.

Model	Eastern U.S.	Western U.S.	Stanford Bunny
CNN	80.861±1.631	224.304±0.659	58.802±10.225
ResNet	75.907±2.632	220.696±0.964	33.419±5.101
EdgeConv	80.576±3.631	218.673±2.069	0.217±0.005
PDE-Net 2.0	73.708±2.960	223.152±0.802	0.223±0.010
GNN-PDE	73.574±7.332	226.800±1.522	0.224±0.022
R-MPGNN	73.708±3.288	217.183±1.317	0.214±0.002
EA-GNN	71.656±4.164	214.041±0.980	0.221±0.013
M-GNN	74.620±5.628	217.613±1.014	0.219±0.007
SNN-PDE (ours)	*64.424±2.361	200.657±0.907	*0.213±0.001

Table 2: Performance on the Eastern, Western U.S. wildfire, and Stanford bunny datasets. The best results are given in bold while the best performances achieved by the runner-ups are underlined.

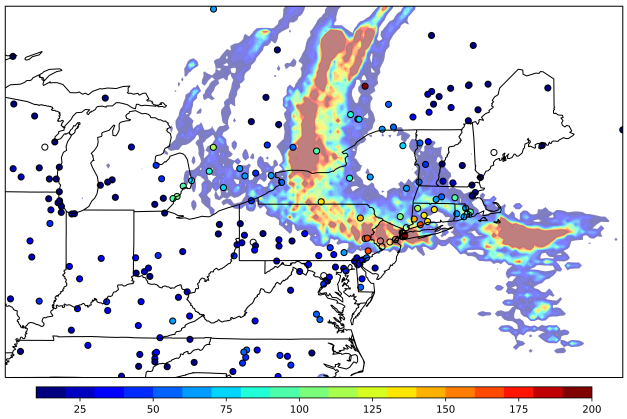

Figure 3: Spatial distribution of observed and simulated surface $PM_{2.5}$ concentrations over the northeastern US during the Quebec fire in June 2023 and daily averaged $PM_{2.5}$ concentrations for 00 UTC on June 7, 2023.

datasets we focus on dynamical systems on $\mathbb{R}$: (1) the convection-diffusion (ConvDiff) equation which is a partial differential equation that can be used to model a variety of physical phenomena and is defined as $\frac{\partial u(s,y,t)}{\partial t} = D\nabla^2 u(s,y,t) - \boldsymbol{v} \cdot \nabla u(s,y,t)$ (where D is a diffusion coefficient, $\boldsymbol{v}$ is the velocity field, and u is the concentration of some quantity of interest); in the ConvDiff equation, we set the number of observation points (i.e., nodes) to be 750, 1500, and 3000 respectively (i.e., ConvDiff_N750, ConvDiff_N1500, and ConvDiff_N3000), and the timestamps is set to be 0.02 seconds with in 11 training timestamps per simulation, (2) the heat equation which is defined as $\frac{\partial u}{\partial t} = D\nabla^2 u$ (where u denotes the temperature field) and describes the behavior of diffusive systems; in heat equation, we set the number of observation points to be 4100, the training set contains 24 simulations and 21 timestamps on the time interval $[0, 0.1]$ seconds with time step 0.005 seconds, and the test set contains 50 simulations and 21 timestamps on the time interval $[0, 0.3]$ seconds; (3) the Burgers' equations which contain a system of two coupled nonlinear PDEs and are defined as $\frac{\partial \boldsymbol{u}(s,y,t)}{\partial t} = D\nabla^2 \boldsymbol{u}(s,y,t) - \boldsymbol{u}(s,y,t) \cdot \nabla \boldsymbol{u}(s,y,t)$ (where $\boldsymbol{u}$ is the velocity vector field); in Burgers' equations, we set the number of observation points to be 5000, the training set contains 24 simulations and 21 timestamps on the time interval $[0, 0.8]$ seconds with time step 0.04 seconds, and the test set contains 50 simulations and 21 timestamps on the time interval $[0, 2.4]$ seconds. For (ii) the real-world dataset, we consider HYSPLIT (Stein et al. 2015), short for Hybrid Single-Particle Lagrangian Integrated Trajectory model, which simulates substance dis-

persion in the atmosphere from local to global scales. It calculates trajectories, transport, dispersion, and deposition of pollutants. It is widely used for back trajectory analysis, pinpointing air mass origins, and predicting pollutant movements. The model combines Lagrangian and Eulerian methods, evolved over decades to handle complex interactions, and offers a good platform for testing physics-informed DL solutions on manifolds. The HYSPLIT also simulates PM2.5 originating from the Quebec wildfire, and The National Oceanic and Atmospheric Administration (NOAA) Smoke Forecasting System integrates wildfire data from various sources. Employing the NOAA Air Resource Laboratory (ARL) HYSPLIT model, it predicts 48-hour smoke transport and concentration of particulate matter less than 2.5 μm in diameter ($PM_{2.5}$), by using the GBBEPX fire emissions inventory (Kim et al. 2020). Figure 3 shows the surface PM2.5 concentrations from the AirNow observations (dots) and the HYSPLIT simulation (contours) at 00UTC on the 7th of June in 2023. Columns from 0 to 500m are integrated to represent the surface concentrations. No background $PM_{2.5}$ concentrations were considered for the HYSPLIT simulations. The smoke plumes originating from the Quebec fire were transported into the Northwestern United States, as evidenced by the elevated levels of PM2.5. Furthermore, we select neighbors for each node by applying Delaunay triangulation to the measurement positions and use the Packing of Parallelograms algorithms for network generation.

Baselines We compare our proposed SNN-PDE with 8 types of state-of-the-art baselines (SOAs), including (i) CNN which consists of multiple convolutional layers, (ii) ResNet (He et al. 2016) which contains 4 residual blocks and a linear CNN layer, (iii) EdgeConv (Wang et al. 2019) which generates edge features that describe the relationships between a node and its neighbors, (iv) PDE-Net 2.0 (Long, Lu, and Dong 2019) which includes a combination of numerical approximation of differential operators by convolutions and a symbolic multi-layer neural network for model recovery, (v) GNN-PDE (Iakovlev, Heinonen, and Lähdesmäki 2020) which utilizes message passing neural networks to evaluate the PDE surrogate, (vi) R-MPGNN (Pilva and Zareei 2022) which applies message passing graph neural networks to parameterize governing equations and learn solver schemes for linear/nonlinear PDEs, (vii) EA-GNN (Gladstone et al. 2023) which introduce virtual edges that yield faster information propagation, and (viii) M-GNN (Gladstone et al. 2023) which is based on multigrid solvers and contains an M-Net block in the network architecture.

Experiment Settings We implement SNN-PDE with Pytorch framework on one NVIDIA RTX A5000 GPU card with up to 48GB memory. In our experiments, we utilize the adaptive-order implicit Adams solver with the relative tolerance rtol $= 1e^{-7}$ and absolute tolerance atol $= 1e^{-5}$. For synthetic and real-world datasets, we set 200 and 10 epochs as training iterations respectively, and we run all models 5 times and report the mean test accuracy and standard deviation. The learning rate is searched in $\{1e^{-7}, 1e^{-6}, 1e^{-5}, 1e^{-4}, 1e^{-3}, 1e^{-2}, 1e^{-1}\}$, the embedding dimension of the node embedding dictionary $\boldsymbol{E}^{\phi}$ is searched in $\{1, 2, 3, 5, 10\}$, and the hidden layer dimension nhid is searched in $\{4, 8, 16, 32, 40, 60\}$. All models are evaluated in terms of the mean relative error where the relative error between the observed states $\boldsymbol{y}(t_i)$ and the estimated $\boldsymbol{u}(t_i)$ at timestamp t_i is defined as $\epsilon_{t_i} = (||\boldsymbol{y}(t_i) - \hat{\boldsymbol{u}}(t_i)||)/||\boldsymbol{y}(t_i)||$. We also perform a one-sided two-sample t-test between the best result and the best performance achieved by the runner-up, where *, **, *** denote p-value < 0.1, 0.05, 0.01 (i.e., denote significant, statistically significant, and highly statistically significant results, respectively. The source code is available at https://github.com/SNNPDE/SNN-PDE.git.

Experiment Results The evaluation results on 8 synthetic and real-world datasets are summarized in Tables 1 and 2. Table 1 shows the performance comparison among 8 baselines on three convection-diffusion equations with different numbers of nodes (i.e., ConvDiff_N750, ConvDiff_N1500, and ConvDiff_N3000), heat equation (i.e., Heat), and Burgers' equations (i.e., Burgers) for nodes' states estimation. We observe that our SNN-PDE always highly and significantly outperforms baseline models on all 5 datasets. In particular, the average relative gain of SNN-PDE over the runner-ups is 20.616% which demonstrates the effectiveness of the proposed SNN-PDE for unstructured grids estimation. In terms of baseline methods, although CNN-based models (i.e., CNN and ResNet) can extract high-level pairwise interaction patterns, they fail to consider topological structure information. Moreover, GNN-based models, e.g., GNN-PDE and EdgeConv only account for the node-level information and tend to exhibit limited abilities to capture higher-order spatial dependencies and information from long-distant nodes. Compared with other GNN-based physics solvers (i.e., EA-GNN and M-GNN), our SNN-PDE improves upon M-GNN (which performs best among these 2 models) 50.000%, 222.222%, 320.408%, 136.364%, and 30.065% on datasets ConvDiff_N750, ConvDiff_N1500, ConvDiff_N3000, Heat, and Burgers. A common limitation of these baseline models is that they do not simultaneously and systematically capture local- and global topological structures and higher-order relationships. Table 2 shows the performance comparison on 2 real-world datasets and Stanford bunny retrieved from Point Clean Net database (Turk and Levoy 1994). Similarly, we can observe that our SNN-PDE always achieves competitive performances on all 3 datasets which reveals that local and global topological information and higher-order dependencies can enhance the model expressiveness. Note that, although the mean relative error of GNN-PDE is lower than that of SNN-PDE, its standard deviation is around 42 times larger than the standard deviation of SNN-PDE. Furthermore, to evaluate the contributions of different components in our SNN-PDE model, we perform ablation studies on ConvDiff_N750, Burgers, and Western U.S. datasets and see Appendix B for a discussion.

Computational Complexity For higher-order simplices, incidence matrices $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ can be calculated efficiently with the computational complexity $\mathcal{O}(\mathcal{N}+\mathcal{M})$ and $\mathcal{O}(\mathcal{M}+\mathcal{Q})$ respectively, where $\mathcal{N}$ is the number of 0-simplices (i.e., nodes), $\mathcal{M}$ is the number of 1-simplices (i.e., edges), and $\mathcal{Q}$ is the number of 2-simplices (i.e., filled triangles).

Conclusion

By bringing the concepts of the Hodge theory and the associated notion of simplicial convolution, we have developed a new competitive and computationally efficient DL-assisted solver of time-dependent PDEs, while accounting for complex topological and geometric properties of the underlying data. In the future, we will advance the proposed methodology to approximate the underlying manifold structure and solve boundary value problem, as well as will explore the utility of SNN-PDE for predictive analytics.

A. Discrete Version of Hodge-Laplacians on Graphs

We start from defining discretization of the differential forms on what is referred to as *simplicial complexes*, as well as *boundary* and *co-boundary* operators, which as would be seen below, are inherently related to their continuous counterparts – exterior derivative and codifferential operators.

Definition 5 *A family $\aleph$ of finite subsets of a set $\mathcal{V}$ is an abstract simplicial complex if for every $\sigma \in \aleph$, $\tau \subseteq \sigma$ implies $\tau \in \aleph$. I.e., Δ is closed under the operation of taking subsets. If $|\sigma| = k+1$, then σ is called a k-*simplex*. Every subset $\tau \subset \sigma$ such that $|\sigma| = k$ is called a face of σ. All simplices in $\aleph$ that have σ as face are called co-faces. Dimension of $\aleph$ is the largest dimension of any of its faces, or ∞ if there is no upper bound on the dimension of the faces.*

For a k-simplex of $k > 0$, we can also define its *orientation* by (arbitrary) selecting some order for its nodes, and two orderings are said to be equivalent if they differ by an even permutation. As a result, for a given k-simplex σ with orientation $[i_0, i_2, \ldots, i_k]$, any face of σ is assigned its own orientation (or "identifyer") $[i_0, i_1, \ldots, i_{j-1}, i_{j+1}, \ldots, i_k]$ (i.e., we omit the j-th element). To study diffusion among higher-order substructures of $\mathcal{G}$, we now form a real-valued vector space C^k which is endowed with basis from the oriented k-simplices and whose elements are called k-*chains*. Diffusion through higher-order graph substructures can be then defined via linear maps among spaces C^k of k-chains on $\mathcal{G}$ (Lim 2020). (The dual of a k-chain, i.e., $\omega : C^k \to \mathbb{R}$, is called a *co-chain*, and co-chains are natural discrete counterpart of differential forms.) For the gentle introduction to differential forms in continuous and discrete cases see Desbrun, Kanso, and Tong (2006).

A linear map $\partial_k : C^k \to C^{k-1}$ is called a *boundary* operator. This is a discrete analogue of codifferential operator $d*$. The adjoint of the boundary map induces the *co-boundary* operator $\partial_k^T : C^k \to C^{k+1}$, which is a discrete analogue of exterior derivative d. We define matrix representations of ∂_k and $\partial_k^\top$ as $\boldsymbol{B}_k$ and $\boldsymbol{B}_k^\top$, respectively.

Definition 6 *An operator over oriented k-simplices $L_k : C^k \to C^k$ is called the discrete k-Hodge Laplacian. That is, L_k is the discrete analogue of Δ_H, and its matrix representation is given by*

$$\boldsymbol{L}_k = \boldsymbol{B}_k^\top \boldsymbol{B}_k + \boldsymbol{B}_{k+1}\boldsymbol{B}_{k+1}^\top, \tag{15}$$

where $\boldsymbol{B}_k^\top \boldsymbol{B}_k$ and $\boldsymbol{B}_{k+1}\boldsymbol{B}_{k+1}^\top$ are often referred to $\boldsymbol{L}_k^{down}$ and $\boldsymbol{L}_k^{up}$, respectively.

That is, the standard graph Laplacian $\boldsymbol{L}_0 = \boldsymbol{B}_1\boldsymbol{B}_1^\top \in \mathbb{R}^{N\times N}$ is a special case of the above k-th combinatorial Hodge Laplacian and $\boldsymbol{L}_1 \in \mathbb{R}^{M\times M}$ is the Hodge 1-Laplacian.

B. Ablation Study

	Architecture	Mean Relative Error
ConvDiff_N750	**SGNN-PDE**	*****0.108±0.010**
	W/o Multi-Channel Simplicial Conv. Encoder.	0.135±0.037
	W/o Multi-Hop Grid-Structural Graph Conv.	0.113±0.025
	W/o Multi-Hop Adaptive Graph Conv.	0.116±0.023
Burgers	**SGNN-PDE**	*****0.306±0.050**
	W/o Multi-Channel Simplicial Conv. Encoder.	0.336±0.032
	W/o Multi-Hop Grid-Structural Graph Conv.	0.320±0.079
	W/o Multi-Hop Adaptive Graph Conv.	0.326±0.338
Western U.S.	**SGNN-PDE**	*****200.657±0.907**
	W/o Multi-Channel Simplicial Conv. Encoder.	227.515±2.00
	W/o Multi-Hop Grid-Structural Graph Conv.	217.020±3.532
	W/o Multi-Hop Adaptive Graph Conv.	226.686±0.940

Table 3: Ablation study on ConvDiff_N750, Burgers, and Western U.S. datasets.

To investigate the importance of the different components in SGNN-PDE, we have conducted an ablation study of our proposed model on ConvDiff_N750, Burgers, and Western U.S. datasets. We have compared our SGNN-PDE with three ablated variants, i.e., (i) SGNN-PDE without Multi-Channel Simplicial Convolutional Encoder (i.e., W/o Multi-Channel Simplicial Conv.), (ii) SGNN-PDE without Multi-Hop Grid-Structural Graph Convolution (i.e., W/o Multi-Hop Grid-Structural Graph Conv.), and (iii) SGNN-PDE without Multi-Hop Adaptive Graph Convolution (i.e., W/o Multi-Hop Adaptive Graph Conv.). Table 3 shows that SGNN-PDE significantly outperforms all ablated variants. As expected, we find that the proposed multi-channel simplicial convolutional encoder significantly improves the results as as it utilizes higher-order relationships and learns important embeddings beyond the node-space. Moreover, both multi-hop grid-structural graph convolution and multi-hop adaptive graph convolution operations play important roles for node representation learning by fusing content features and long-distance (and dynamic) neighboring features of multi-hop information.

Acknowledgements

This project has been supported in part by the NASA AIST grant 21-AIST21_2-0059. In addition, Y.C., H.L., and N.L. have been supported by the NSF Grant DMS-2335846/2335847 and Y.C. has also got support from the NSF grant TIP-2333703. The paper is based upon work supported by (while Y.R.G. was serving at) the NSF. H.L.'s contribution to this work was performed at the JPL, Caltech, under a contract with NASA. The views expressed in the article do not necessarily represent the views of NSF or NASA.

References

Agarwal, S.; Branson, K.; and Belongie, S. 2006. Higher order learning with graphs. In *ICML*.

Benson, A. R.; Abebe, R.; Schaub, M. T.; Jadbabaie, A.; and Kleinberg, J. 2018. Simplicial closure and higher-order link prediction. *PNAS*, 115(48): E11221–E11230.

Bodnar, C.; Frasca, F.; Wang, Y. G.; Otter, N.; Montúfar, G.; Lio, P.; and Bronstein, M. 2021. Weisfeiler and lehman go topological: Message passing simplicial networks. In *ICML*.

Brandstetter, J.; Welling, M.; and Worrall, D. E. 2022. Lie point symmetry data augmentation for neural PDE solvers. In *ICML*.

Brandstetter, J.; Worrall, D.; and Welling, M. 2022. Message passing neural PDE solvers. In *ICLR*.

Brandstetter, J.; Worrall, D. E.; and Welling, M. 2021. Message Passing Neural PDE Solvers. In *ICLR*.

Chen, Y.; Gel, Y. R.; and Poor, H. V. 2022. BScNets: Block Simplicial Complex Neural Networks. In *AAAI*.

Desbrun, M.; Kanso, E.; and Tong, Y. 2006. Discrete differential forms for computational modeling. In *SIGGRAPH*, 39–54.

Dong, X.; Thanou, D.; Rabbat, M.; and Frossard, P. 2019. Learning graphs from data: A signal representation perspective. *IEEE Signal Process. Mag*, 36(3): 44–63.

Dulny, A.; Hotho, A.; and Krause, A. 2022. Neuralpde: Modelling dynamical systems from data. In *German Conference on Artificial Intelligence (Künstliche Intelligenz)*, 75–89. Springer.

Ebli, S.; Defferrard, M.; and Spreemann, G. 2020. Simplicial Neural Networks. In *NeurIPS 2020 Wkshp on TDA and Beyond*.

Gilmer, J.; Schoenholz, S. S.; Riley, P. F.; Vinyals, O.; and Dahl, G. E. 2017. Neural message passing for quantum chemistry. In *ICML*.

Gladstone, R. J.; Rahmani, H.; Suryakumar, V.; Meidani, H.; D'Elia, M.; and Zareei, A. 2023. GNN-based physics solver for time-independent PDEs. *arXiv:2303.15681*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*, 770–778.

Horie, M.; and Mitsume, N. 2022. Physics-Embedded Neural Networks: Graph Neural PDE Solvers with Mixed Boundary Conditions. *NeurIPS*.

Iakovlev, V.; Heinonen, M.; and Lähdesmäki, H. 2020. Learning continuous-time PDEs from sparse data with graph neural networks. In *ICLR*.

Johnson, J. L.; and Goldring, T. 2012. Discrete hodge theory on graphs: a tutorial. *CiSE*, 15(5): 42–55.

Kim, H. C.; Chai, T.; Stein, A.; and Kondragunta, S. 2020. Inverse modeling of fire emissions constrained by smoke plume transport using HYSPLIT dispersion model and geostationary satellite observations. *Atmospheric Chemistry and Physics*, 20(17): 10259–10277.

Kipf, T. N.; and Welling, M. 2017. Semi-supervised classification with graph convolutional networks. In *ICLR*.

Liang, J.; and Zhao, H. 2013. Solving partial differential equations on point clouds. *SISC*, 35(3): A1461–A1486.

Lim, L. H. 2020. Hodge laplacians on graphs. *SIAM Rev.*, 62: 685–715.

Long, Z.; Lu, Y.; and Dong, B. 2019. PDE-Net 2.0: Learning PDEs from data with a numeric-symbolic hybrid deep network. *JCP*, 399: 108925.

Long, Z.; Lu, Y.; Ma, X.; and Dong, B. 2018. Pde-net: Learning pdes from data. In *ICML*.

Morris, C.; Kriege, N. M.; Kersting, K.; and Mutzel, P. 2016. Faster kernels for graphs with continuous attributes via hashing. In *ICDM*, 1095–1100.

Moshe Eliasof, E. H.; and Treister, E. 2021. PDE-GCN: Novel Architectures for Graph Neural Networks Motivated by Partial Differential Equations. In *NeurIPS*.

Pilva, P.; and Zareei, A. 2022. Learning time-dependent PDE solver using Message Passing Graph Neural Networks. *arXiv:2204.07651*.

Rakhoon Hwang, J. Y. S., Jae Yong Lee; and Hwang, H. J. 2022. Solving PDE-constrained Control Problems using Operator Learning. In *AAAI*.

Roddenberry, T. M.; Glaze, N.; and Segarra, S. 2021. Principled simplicial neural networks for trajectory prediction. In *ICML*, 9020–9029.

Sanchez-Gonzalez, A.; Godwin, J.; Pfaff, T.; Ying, R.; Leskovec, J.; and Battaglia, P. 2020. Learning to simulate complex physics with graph networks. In *ICML*.

Schaub, M. T.; Benson, A. R.; Horn, P.; Lippner, G.; and Jadbabaie, A. 2020. Random walks on simplicial complexes and the normalized Hodge 1-Laplacian. *SIAM Rev.*, 62(2): 353–391.

Seo, S.; and Liu, Y. 2019. Differentiable physics-informed graph networks. In *ICLR Wksp on Representation Learning on Graphs and Manifolds*.

Shervashidze, N.; Schweitzer, P.; Van Leeuwen, E. J.; Mehlhorn, K.; and Borgwardt, K. M. 2011. Weisfeiler-lehman graph kernels. *JMLR*, 12(9).

Shukla, K.; Xu, M.; Trask, N.; and Karniadakis, G. E. 2022. Scalable algorithms for physics-informed neural and graph networks. *DCE*, 3: e24.

Stein, A.; Draxler, R. R.; Rolph, G. D.; Stunder, B. J.; Cohen, M. D.; and Ngan, F. 2015. NOAA's HYSPLIT atmospheric transport and dispersion modeling system. *BAMS*, 96(12): 2059–2077.

Turk, G.; and Levoy, M. 1994. Zippered polygon meshes from range images. In *SIGGRAPH*, 311–318.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; and Bengio, Y. 2018. Graph Attention Networks. *ICLR*.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic graph cnn for learning on point clouds. *ACM TOG*, 38(5): 1–12.

Xia, F.; Sun, K.; Yu, S.; Aziz, A.; Wan, L.; Pan, S.; and Liu, H. 2021. Graph learning: A survey. *IEEE TAI*, 2(2): 109–127.

Zhou, J.; Cui, G.; Hu, S.; Zhang, Z.; Yang, C.; Liu, Z.; Wang, L.; Li, C.; and Sun, M. 2020. Graph neural networks: A review of methods and applications. *AI Open*, 1: 57–81.