Simplicial Convolutional Neural Networks

Yang, Maosheng; Isufi, Elvin; Leus, Geert

DOI
10.1109/ICASSP43922.2022.9746017

Publication date
2022

Document Version
Final published version

Published in
Proceedings of the ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)

ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-6654-0540-9/22/$31.00 ©2022 IEEE | DOI: 10.1109/ICASSP43922.2022.9746017

SIMPLICIAL CONVOLUTIONAL NEURAL NETWORKS

Maosheng Yang, Elvin Isufi and Geert Leus

ABSTRACT

Graphs can model networked data by representing them as nodes and their pairwise relationships as edges. Recently, signal processing and neural networks have been extended to process and learn from data on graphs, with achievements in tasks like graph signal reconstruction, graph or node classifications, and link prediction. However, these methods are only suitable for data defined on the nodes of a graph. In this paper, we propose a simplicial convolutional neural network (SCNN) architecture to learn from data defined on simplices, e.g., nodes, edges, triangles, etc. We study the SCNN permutation and orientation equivariance, complexity, and spectral analysis. Finally, we test the SCNN performance for imputing citations on a coauthorship complex.

***Index Terms*—** Simplicial complex, Hodge Laplacian, simplicial filter, simplicial neural network.

1. INTRODUCTION

Graphs are powerful models to represent irregular data by encoding their pairwise relationships. To process such networked data, signal processing concepts have been extended to the graph domain, defining, for instance, the graph Fourier transform and graph filters [1]. Meanwhile, graph neural networks (GNNs) have achieved a good performance in tasks like rating prediction in recommender systems, graph or node classification and link prediction [2, 3, 4, 5, 6].

However, graph signal processing and GNNs are designed for data defined on nodes of a graph. In real-world problems, we might also have data defined on edges, triangles, etc, of a network, such as communication or traffic flow in a data or road network, paper citations of a coauthorship network and so on [7, 8, 9]. To model data defined on such higher-order network structures, we can use a simplicial complex, which is a collection of simplices, i.e., nodes, edges, triangles, etc. Recently, signal processing and neural networks on simplicial complexes have emerged. In [7], simplicial data as well as the simplicial Fourier transform have been defined. The authors of [10, 11] provided an overview of some simplicial signal processing techniques to process flow-typed data. Our previous work [12] analyzed the definition of frequencies for simplicial signals and proposed two types of simplicial filters based on the Hodge Laplacian.

Meanwhile, researchers have also attempted to develop neural networks on simplicial complexes. In [8], a basic simplicial neural network (SNN) was proposed with a convolutional layer composed of a basic simplicial filter [12] and a nonlinearity. Message passing neural networks (MPNNs) have been generalized to simplicial complexes in [13] where the aggregation and updating functions consider in addition to the edge data also data defined on adjacent simplices, i.e., nodes and triangles. The neural network architectures in [14, 15] are instances of [13] by specifying the aggregation functions as simplicial filters and [15] also introduced the notion of admissibility to analyze the architecture. Another attempt in [16] considered recurrent architectures in MPNNs for flow interpolation and graph classification tasks.

Motivated by the principle of the convolution operator, in this paper we propose simplicial convolutional neural networks (SCNNs). Differently from the earlier approach in [8], we build an SCNN with simplicial filters of higher flexibility in exploiting the lower- and upper-neighbors of a simplex. And differently from the MPNN, the proposed SCNN considers multihop information exchange within a layer and enjoys spectral interpretability via the simplicial Fourier transform. Our specific contributions are: i) we propose an inductive SCNN based on the more advanced simplicial filter of [12] and discuss its connections to the related work; ii) we analyze its permutation and orientation equivariances as well as characterize the SCNN in the spectral domain; iii) we test the performance on citation data imputation in a coauthorship complex outperforming the state-of-the-art.

2. SIMPLICIAL SIGNAL PROCESSING

In this section, we recall some important simplicial signal processing concepts, including simplicial complexes and signals, the Hodge decomposition, and simplicial filters.

Simplicial complexes and signals. Given a finite set of vertices $\mathcal{V}$, a k-simplex $\mathcal{S}^k$ is a subset of $\mathcal{V}$ with cardinality $k+1$. A *face* of $\mathcal{S}^k$ is a subset with cardinality k and thus a k-simplex has $k+1$ faces. A *coface* of $\mathcal{S}^k$ is a $(k+1)$-simplex that includes $\mathcal{S}^k$ [7, 17]. A simplicial complex of order K, $\mathcal{X}^K$, is a collection of k-simplices $\mathcal{S}^k$, $k=0,\ldots,K$, with an inclusion property–for any $\mathcal{S}^k \in \mathcal{X}^K$, then $\mathcal{S}^{k-1} \in \mathcal{X}^K$ if $\mathcal{S}^{k-1} \subset \mathcal{S}^k$. We denote the number of k-simplices in $\mathcal{X}^K$

Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, Delft, The Netherlands. This work is supported by the TU Delft AI Labs Programme. e-mail: {m.yang-2, e.isufi, g.j.t.leus}@tudelft.nl.

by N_k. If two simplices share a common face, then they are lower neighbours; if they share a common coface, they are upper neighbours [12]. A graph is a simplicial complex where nodes are 0-simplices, and edges are 1-simplices.

In a simplicial complex, we define a k-simplicial signal $\mathbf{x}^k = [x_1^k, \ldots, x_{N_k}^k]$ as a mapping from the k-simplices to the real space $\mathbb{R}^{N_k}$. For example, $\mathbb{R}^{N_0}$ is the graph signal space in GSP, and $\mathbb{R}^{N_1}$ is the space of edge flows. For an edge flow $\mathbf{x}^1 \in \mathbb{R}^{N_1}$, the sign of its entry denotes the direction of the flow relative to a chosen reference orientation [11, 17].

Hodge Laplacian and decomposition. We represent the relations between $(k-1)$- and k-simplices with the incidence matrix $\mathbf{B}_k$, $k = 1, \ldots, K$. The rows of $\mathbf{B}_k$ are indexed by $(k-1)$-simplices and the columns by k-simplices. E.g., matrix $\mathbf{B}_1$ is the node-to-edge incidence matrix, and $\mathbf{B}_2$ is the edge-to-triangle incidence matrix [11, 12]. We can also use the Hodge Laplacian matrices, $\mathbf{L}_k = \mathbf{B}_k^\top \mathbf{B}_k + \mathbf{B}_{k+1}\mathbf{B}_{k+1}^\top$, where $\mathbf{L}_{k,\mathrm{l}} \triangleq \mathbf{B}_k^\top \mathbf{B}_k$ and $\mathbf{L}_{k,\mathrm{u}} \triangleq \mathbf{B}_{k+1}\mathbf{B}_{k+1}^\top$ are the lower and the upper Laplacians, which encode lower and upper neighbourhoods, respectively. When $k = 0$, the Hodge Laplacian is the graph Laplacian $\mathbf{L}_0 = \mathbf{B}_1\mathbf{B}_1^\top$. Hodge Laplacians admit a *Hodge decomposition*, leading to three orthogonal subspaces which the simplicial signal space can be decomposed into, i.e., $\mathbb{R}^{N_k} = \mathrm{im}(\mathbf{B}_k^\top) \oplus \mathrm{im}(\mathbf{B}_{k+1}) \oplus \ker(\mathbf{L}_k)$, where $\oplus$ is the direct sum of vector spaces and $\mathrm{im}(\cdot)$ and $\ker(\cdot)$ are the *image* and *kernel* of a matrix. For $k = 1$, these subspaces carry the following interpretations [7, 11].

Gradient space. By applying matrix $\mathbf{B}_1$ to an edge flow $\mathbf{x}^1$, we compute its net flow at each node, $\mathbf{B}_1\mathbf{x}^1$. The incidence matrix $\mathbf{B}_1$ is called a *divergence operator*. Its adjoint $\mathbf{B}_1^\top$ differentiates a node signal $\mathbf{x}^0$ along the edges to induce an edge flow $\mathbf{B}_1^\top\mathbf{x}^0$, i.e., it is the *gradient operator*. As a result, any flow within $\mathrm{im}(\mathbf{B}_1^\top)$ can be written as the gradient of a node signal $\mathbf{x}^0$, i.e., $\mathbf{x}^1 = \mathbf{B}_1^\top\mathbf{x}^0$. We call $\mathbf{x}^1 \in \mathrm{im}(\mathbf{B}_1^\top)$ a *gradient flow* and the space $\mathrm{im}(\mathbf{B}_1^\top)$ the *gradient space*.

Curl space. We can induce an edge flow from a triangle signal $\mathbf{x}^2$ as $\mathbf{x}^1 = \mathbf{B}_2\mathbf{x}^2$. The induced flow $\mathbf{x}^1 \in \mathrm{im}(\mathbf{B}_2)$ is called a *curl flow* and the space $\mathrm{im}(\mathbf{B}_2)$ is the *curl space*. The adjoint $\mathbf{B}_2^\top$ is the *curl operator*. We can use it to compute the net edge flow of $\mathbf{x}^1$ circulating along the triangles as $\mathbf{B}_2^\top\mathbf{x}^1$.

Harmonic space. The remaining space $\ker(\mathbf{L}_1)$ is the *harmonic space*. Any edge flow $\mathbf{x}^1 \in \ker(\mathbf{L}_1)$ has zero divergence and curl, i.e., it is *divergence-* and *curl-free*.

Due to the boundary condition $\mathbf{B}_1\mathbf{B}_2 = \mathbf{0}$, any gradient flow $\mathbf{x}^1 \in \mathrm{im}(\mathbf{B}_1^\top)$ is curl-free. The space orthogonal to the gradient space, i.e., $\ker(\mathbf{B}_1) = \mathrm{im}(\mathbf{B}_2) \oplus \ker(\mathbf{L}_1)$, is called the *cycle space*, which consists of both the curl space and harmonic space. Any flow in this space is divergence-free.

Simplicial filters. To process simplicial signals $\mathbf{x}^1$, we use a simplicial convolutional filter of the following form:

$$\mathbf{H} = \epsilon\mathbf{I} + \sum_{l_1=1}^{L_1} \alpha_{l_1}(\mathbf{B}_1^\top\mathbf{B}_1)^{l_1} + \sum_{l_2=1}^{L_2} \beta_{l_2}(\mathbf{B}_2\mathbf{B}_2^\top)^{l_2} \qquad (1)$$

where ϵ, $\boldsymbol{\alpha} = [\alpha_1, \ldots, \alpha_{L_1}]^\top$ and $\boldsymbol{\beta} = [\beta_1, \ldots, \beta_{L_2}]^\top$ are the filter coefficients [12]. For ease of exposition, we only discuss the simplicial filter form (1) for the edge signal space $\mathbb{R}^{N_1}$, but similar discussions apply to general simplicial filters with $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$.

Applying $\mathbf{H}$ to an input edge flow $\mathbf{x}^1$ consists of the *simplicial shifting* operations, $\mathbf{L}_{1,\mathrm{l}}\mathbf{x}^1$ and $\mathbf{L}_{1,\mathrm{u}}\mathbf{x}^1$. The ith entry of $\mathbf{L}_{1,\mathrm{l}}\mathbf{x}^1$ is $[\mathbf{L}_{1,\mathrm{l}}\mathbf{x}^1]_i = \sum_{j\in\{\mathcal{N}_{1,i}\cup i\}}[\mathbf{L}_{1,\mathrm{l}}]_{ij}[\mathbf{x}^1]_j$, which is a local operation within the lower neighborhood of the ith edge, likewise for $\mathbf{L}_{1,\mathrm{u}}\mathbf{x}^1$. Moreover, powers $\mathbf{L}_{1,\mathrm{l}}^k\mathbf{x}^1 = \mathbf{L}_{1,\mathrm{l}}(\mathbf{L}_{1,\mathrm{l}}^{k-1}\mathbf{x}^1)$ can be recursively obtained by applying the local operation k times [12]. This leads to a distributed implementation of simplicial filtering with a complexity of order $\mathcal{O}(N_1D)$ for each shifting with D being the maximal number of neighbours.

As we can observe from (1), in the simplicial domain, different sets of coefficients on $\mathbf{L}_{1,\mathrm{l}}$ and $\mathbf{L}_{1,\mathrm{u}}$ enable an independent and flexible filtering within the lower and upper simplicial neighbourhoods. When $L_1 = L_2$ and $\boldsymbol{\alpha} = \boldsymbol{\beta}$, filter $\mathbf{H}$ [cf. (1)] reduces to the basic form $\mathbf{H} = \sum_{l=0}^{L} h_l\mathbf{L}_1^l$ at the cost of losing expressive power and flexibility [12].

3. SIMPLICIAL CONVOLUTIONAL NEURAL NETWORKS

Upon having a simplicial convolutional filter (1), we can build an SCNN by composing filter banks with elementwise nonlinearities. This SCNN applies to any k-simplicial signal but we again illustrate it for edge signals $\mathbf{x}^1$ for the ease of intuition, and omit the superscript to avoid overcrowded notation.

Consider a P-layer SCNN. In the first layer $p = 1$, we apply F filters $\mathbf{H}_1^f$ [cf. (1)] and an elementwise nonlinearity $\sigma(\cdot)$ to the input $\mathbf{x}_0$ to get a collection of F features $\mathbf{x}_1^f$ as

$$\mathbf{x}_1^f = \sigma[\mathbf{z}_1^f] = \sigma[\mathbf{H}_1^f\mathbf{x}_0], \quad f = 1, \ldots, F \qquad (2)$$

which constitute the output feature matrix $\mathbf{X}_1 = [\mathbf{x}_1^1, \ldots, \mathbf{x}_1^F]$. In subsequent intermediate layers $p = 2, \ldots, P-1$, we have $\mathbf{X}_{p-1} = [\mathbf{x}_{p-1}^1, \ldots, \mathbf{x}_{p-1}^F] \in \mathbb{R}^{N_1\times F}$ as input. Each input signal $\mathbf{x}_{p-1}^g, g = 1, \ldots, F$ is passed through a bank of filters $\mathbf{H}_p^{fg}$ to obtain F intermediate outputs $\mathbf{z}_p^{fg} = \mathbf{H}_p^{fg}\mathbf{x}_{p-1}^g, f = 1, \ldots, F$. To avoid exponential filter growth, the intermediate outputs of the different input signals $\mathbf{x}_{p-1}^g$ are summed, thus, the pth layer generates F features $\mathbf{x}_p^f$ as follows

$$\mathbf{x}_p^f = \sigma\left[\sum_{g=1}^{F}\mathbf{z}_p^{fg}\right] = \sigma\left[\sum_{g=1}^{F}\mathbf{H}_p^{fg}\mathbf{x}_{p-1}^g\right] f = 1, \ldots, F. \quad (3)$$

The processing in (3) is repeated until the last layer $p = P$, where we consider the output has a single feature, and hence each input is processed by a single filter $\mathbf{H}^g$. Thus, the final output of the SCNN is given by

$$\mathbf{x}_P = \sigma\left[\sum_{g=1}^{F}\mathbf{z}_P^g\right] = \sigma\left[\sum_{g=1}^{F}\mathbf{H}_P^g\mathbf{x}_{P-1}^g\right]. \qquad (4)$$

Equations (2), (3) and (4) constitute the SCNN architecture based on the simplicial filter form defined in (1). The lower and upper Hodge Laplacians encode the lower and upper simplicial neighbourhoods, respectively. Simplicial convolutions through filter (1) are performed independently within the lower and upper neighbourhoods and controlled by different sets of coefficients. As we shall show later on, this means that the gradient and curl components of the input features are convolved independently, leading to more expressive power. Next, we discuss the connections of the SCNN with current alternatives and analyze its properties.

Links with related works. In [8], a similar convolutional layer was proposed but based on filter $\mathbf{H} = \sum_{l=0}^{L} h_l \mathbf{L}_1^l$. This SNN architecture is a particular case of the proposed SCNN with less expressive power but also with less parameters. The message passing neural network (MPNN) for simplicial complexes [13] aggregates and updates features from direct simplicial neighbours and simplices of different orders, e.g., nodes and triangles. By considering an order-one simplicial convolution as the message aggregation step, we then obtain the architectures in [14, Eq. 4] and [15, Eq. 7]. When only edge features are available, such approaches are a particular case of the SCNN with order $L_1 = 1$ and $L_2 = 1$. Recurrent architectures are considered for flow interpolation and graph classification in [16]. Compared to these works, the SCNN treats features from the lower and upper neighbours differently and considers features from not only direct neighbours but also for multihop neighbors.

Locality and complexity. The intermediate output $\mathbf{z}_p^{fg}$ at the pth layer collects for each edge information from lower neighbours up to L_1 hops away and upper neighbours L_2 hops away through filter (1).This locality comes from the structure of the Hodge Laplacian, likewise that of GNNs [2].

When only a single feature is available, we have $1 + L_1 + L_2$ parameters in such layers. For layers with multiple input and output features, the number of parameters grows F^2 times. The major complexity comes from the convolutional filtering step, which as seen before it is a weighted linear combination of different shifts of a simplicial signal; a local operation within the simplicial neighbourhoods that can be computed recursively. Hence, an SCNN layer performs the simplicial filtering for each edge with a cost of order $\mathcal{O}((L_1 + L_2)D)$. Again, this complexity grows F^2 times when multiple features are used and P times if P layers are considered.

Equivariance and invariance. Here we show that our SCNN is equivalent with respect to a different simplex labeling and different reference flow orientations, introduced in [13, 15]. Consider the set of simplicial permutation matrices

$$\mathcal{P} = \{\mathbf{P}_k \in \{0,1\}^{N_k \times N_k} : \mathbf{P}_k \mathbf{1} = \mathbf{1}, \mathbf{P}_k^\top \mathbf{1} = \mathbf{1}, k \geq 0\},$$

where products $\mathbf{P}_k \mathbf{x}^k$ permute the k-simplicial signal $\mathbf{x}^k$. Let $\mathbf{P} = (\mathbf{P}_0, \mathbf{P}_1, \dots)$ denote a sequence of permutations. Then, the following holds.

Proposition 1. *The SCNN is a permutation equivariant architecture. For an input edge flow* $\mathbf{x}$*, the output of an edge space SCNN layer with a simplicial filter* $\mathbf{H}$*,* $\mathbf{y} = \sigma[\mathbf{H}\mathbf{x}]$*, becomes* $\mathbf{y}' = \mathbf{P}_1\mathbf{y}$ *after a permutation sequence* $\mathbf{P}$*.*

Proof. An SCNN layer with filter $\mathbf{H}$ gives the output $\mathbf{z} = \mathbf{H}\mathbf{x}$. After a sequence of permutations $\mathbf{P}$, the input edge flow $\mathbf{x}$ becomes $\mathbf{P}_1\mathbf{x}$ and the boundary operators become $\mathbf{P}_0\mathbf{B}_1\mathbf{P}_1^\top$ and $\mathbf{P}_1\mathbf{B}_2\mathbf{P}_2^\top$. Thus, the Hodge Laplacians become $\mathbf{P}_1\mathbf{L}_{1,\mathrm{l}}\mathbf{P}_1^\top$ and $\mathbf{P}_1\mathbf{L}_{1,\mathrm{u}}\mathbf{P}_1^\top$ due to $\mathbf{P}_k^\top\mathbf{P}_k = \mathbf{I}$. Then we can express the permuted intermediate output as

$$\begin{aligned}\mathbf{z}' &= \left(\epsilon\mathbf{I} + \sum_{l_1=1}^{L_1} \alpha_{l_1} (\mathbf{P}_1\mathbf{L}_{1,\mathrm{l}}\mathbf{P}_1^\top)^{l_1} + \sum_{l_2=1}^{L_2} \beta_{l_2} (\mathbf{P}_1\mathbf{L}_{1,\mathrm{u}}\mathbf{P}_1^\top)^{l_2}\right)\mathbf{P}_1\mathbf{x} \\ &= \mathbf{P}_1\left(\epsilon\mathbf{I} + \sum_{l_1=1}^{L_1} \alpha_{l_1}\mathbf{L}_{1,\mathrm{l}}^{l_1} + \sum_{l_2=1}^{L_2} \beta_{l_2}\mathbf{L}_{1,\mathrm{u}}^{l_2}\right)\mathbf{x} = \mathbf{P}_1\mathbf{z}.\end{aligned} \tag{5}$$

Thus, the simplicial filter $\mathbf{H}$ is permutation equivariant. Furthermore, since the nonlinearity $\sigma(\cdot)$ is elementwise, SCNNs are permutation equivariant. □

In addition, in a simplicial complex, we have also set an arbitrary reference orientation for a simplex. A new reference orientation can be modelled by multiplying the rows and columns of the boundary matrices $\mathbf{B}_k$ and $\mathbf{B}_{k+1}$ where that k-simplex appears and the corresponding simplicial signal value by -1. Let then $\mathbf{D}_k$ be diagonal matrices from the set

$$\mathcal{D} = \{\mathbf{D}_k = \mathrm{diag}(\mathbf{d}_k) : \mathbf{d}_k \in \{\pm 1\}^{N_k}, k \geq 1, \mathbf{d}_0 = \mathbf{1}\},$$

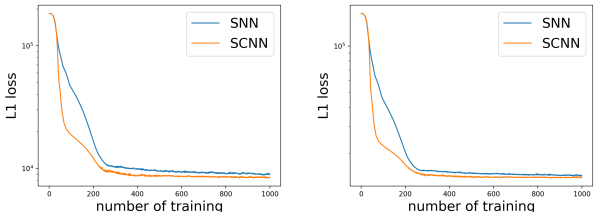
where $\mathbf{D}_k\mathbf{x}^k$ is the updated k-simplicial signal $\mathbf{x}^k$. Let $\mathbf{D} = (\mathbf{D}_0, \mathbf{D}_1, \dots)$ denote a sequence of orientation changes. Then, the following holds.

Proposition 2. *The SCNN is orientation equivariant if the nonlinearity* $\sigma(\cdot)$ *is odd. Without loss of generality, for an input flow* $\mathbf{x}$*, the output of an edge space SCNN layer with a simplicial filter* $\mathbf{H}$*,* $\mathbf{y} = \sigma[\mathbf{H}\mathbf{x}]$ *becomes* $\mathbf{y}' = \mathbf{D}_1\mathbf{y}$ *after a sequence of orientation changes* $\mathbf{D}$*.*

Proof. After an orientation change, the edge flow $\mathbf{x}$ becomes $\mathbf{D}_1\mathbf{x}$ and the boundary operators $\mathbf{B}_1$ and $\mathbf{B}_2$ are updated as $\mathbf{D}_0\mathbf{B}_1\mathbf{D}_1$ and $\mathbf{D}_1\mathbf{B}_2\mathbf{D}_2$. Then, the Hodge Laplacians become $\mathbf{D}_1\mathbf{L}_{1,\mathrm{l}}\mathbf{D}_1$ and $\mathbf{D}_1\mathbf{L}_{1,\mathrm{u}}\mathbf{D}_1$. We can then express the new filter output as $\mathbf{z}' = \mathbf{D}_1\mathbf{H}\mathbf{x} = \mathbf{D}_1\mathbf{z}$, following similar steps as in (5). Thus, simplicial filter $\mathbf{H}$ is orientation equivariant. When the nonlinearity $\sigma(\cdot)$ is an odd function, we have $\mathbf{y}' = \sigma(\mathbf{z}') = \mathbf{D}_1\sigma(\mathbf{z}) = \mathbf{D}_1\mathbf{y}$ that completes the proof. □

Both propositions can be extended to $k > 1$ cases. Permutation and orientation equivariances preserve the output of an SCNN regardless of the choices of the labeling and reference orientation of the edges. In turn, they allow the SCNN to exploit the internal symmetries in the complex.

Spectral analysis. For the spectral analysis, consider first that the eigenvectors of the Hodge Laplacian $\mathbf{L}_1$ span the



(a) 10% missing rate. (b) 20% missing rate.

Fig. 1. Training loss of SNN and SCNN. We see that our SCNN converges much faster in the early training stage and results in a smaller training loss.

three spaces given by the Hodge decomposition: (i) the gradient space $\text{im}(\mathbf{B}_1^\top)$ is spanned by a set of eigenvectors $\mathbf{U}_\text{G}$ of $\mathbf{L}_{1,\text{l}}$ with positive eigenvalues; (ii) the curl space $\text{im}(\mathbf{B}_2)$ is spanned by a set of eigenvectors $\mathbf{U}_\text{C}$ of $\mathbf{L}_{1,\text{u}}$ with positive eigenvalues; and (iii) the harmonic space $\ker(\mathbf{L}_1)$ is spanned by the eigenvectors $\mathbf{U}_\text{H}$ of $\mathbf{L}_1$ with zero eigenvalue. Moreover, we have $\text{im}(\mathbf{L}_1) = \text{im}(\mathbf{U}_\text{G}) \oplus \text{im}(\mathbf{U}_\text{C})$, i.e., gradient and curl spaces make up the image of $\mathbf{L}_1$ [12]. Then, we can eigendecompose $\mathbf{L}_1$ as $\mathbf{L}_1 = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ where $\mathbf{U} = [\mathbf{U}_\text{H}\ \mathbf{U}_\text{G}\ \mathbf{U}_\text{C}]$ provides a simplicial Fourier basis, and $\mathbf{\Lambda} = \text{diag}(\mathbf{\Lambda}_\text{H}, \mathbf{\Lambda}_\text{G}, \mathbf{\Lambda}_\text{C})$ with $\mathbf{\Lambda}_\text{H} = \text{diag}(\mathbf{0}_{N_H})$, $\mathbf{\Lambda}_\text{G} = \text{diag}(\lambda_{\text{G},1}, \ldots, \lambda_{\text{G},N_\text{G}})$, and $\mathbf{\Lambda}_\text{C} = \text{diag}(\lambda_{\text{C},1}, \ldots, \lambda_{\text{C},N_\text{C}})$ collecting the harmonic, gradient, and curl frequencies, respectively; i.e., the simplicial frequencies [12].

For an edge flow $\mathbf{x}$, we can find its simplicial Fourier transform (SFT) as $\tilde{\mathbf{x}} = \mathbf{U}^\top\mathbf{x}$. This further defines three embeddings $\tilde{\mathbf{x}} = [\tilde{\mathbf{x}}_\text{H}^\top\ \tilde{\mathbf{x}}_\text{G}^\top\ \tilde{\mathbf{x}}_\text{C}^\top]$: the harmonic embedding $\tilde{\mathbf{x}}_\text{H} = \mathbf{U}_\text{H}^\top\mathbf{x}$, the gradient embedding $\tilde{\mathbf{x}}_\text{G} = \mathbf{U}_\text{G}^\top\mathbf{x}$, and the curl embedding $\tilde{\mathbf{x}}_\text{C} = \mathbf{U}_\text{C}^\top\mathbf{x}$, which contain the weights of $\mathbf{x}$ at harmonic, gradient, and curl frequencies, respectively.

Via the eigendecomposition of $\mathbf{L}_1$, we can analyze the proposed SCNNs in the spectral domain. First, the frequency response of a simplicial convolutional filter $\mathbf{H}$ is given by

$$\tilde{H}(\lambda_i) = \begin{cases} \epsilon, & \text{for } \lambda_i = 0, \\ \epsilon + \sum_{l_1=1}^{L_1} \alpha_{l_1}\lambda_i^{l_1}, & \text{for } \lambda_i \in \mathcal{Q}_\text{G}, \\ \epsilon + \sum_{l_2=1}^{L_2} \beta_{l_2}\lambda_i^{l_2}, & \text{for } \lambda_i \in \mathcal{Q}_\text{C}, \end{cases} \tag{6}$$

where $\mathcal{Q}_\text{G}$ and $\mathcal{Q}_\text{C}$ respectively collect distinct gradient and curl frequencies. Thus, the ith entry of the SFT of the intermediate output $\mathbf{z}$ of an SCNN layer can be expressed as $\tilde{z}_i = \tilde{H}(\lambda_i)\tilde{x}_i$. This spectral analysis shows that the SCNN layers compute the high-level simplicial components as the pointwise multiplication of the input embedding and the simplicial filter frequency response, ultimately respecting the convolution theorem. Furthermore, we have here different frequency responses for the gradient and curl components, which corresponds to the different weights on lower and upper Laplacians in filter $\mathbf{H}$. However, for a filter with $\boldsymbol{\alpha} = \boldsymbol{\beta}$ as in [8], this would lead to coupling between independent frequencies, and in turn to a limited learning expressiveness.

Table 1. Imputation accuracies for each dimension and missing rate by SNN (first rows) and SCNN (second rows).

Order	0	1	2	3	4	5
N_k	352	1474	3285	5019	5559	4547
10%	0.91 ± 0.003	0.91 ± 0.002	0.90 ± 0.004	0.91 ± 0.004	0.90 ± 0.016	0.90 ± 0.008
10%	0.91 ± 0.004	0.91 ± 0.002	0.91 ± 0.002	0.92 ± 0.001	0.92 ± 0.002	0.92 ± 0.002
20%	0.81 ± 0.006	0.82 ± 0.003	0.82 ± 0.005	0.83 ± 0.004	0.82 ± 0.012	0.83 ± 0.007
20%	0.81 ± 0.007	0.82 ± 0.003	0.83 ± 0.003	0.83 ± 0.002	0.84 ± 0.002	0.84 ± 0.002
30%	0.72 ± 0.006	0.73 ± 0.004	0.73 ± 0.005	0.75 ± 0.002	0.75 ± 0.002	0.75 ± 0.003
30%	0.72 ± 0.005	0.73 ± 0.004	0.74 ± 0.003	0.75 ± 0.002	0.76 ± 0.002	0.77 ± 0.002
40%	0.63 ± 0.007	0.64 ± 0.003	0.65 ± 0.003	0.66 ± 0.004	0.67 ± 0.009	0.67 ± 0.008
40%	0.63 ± 0.006	0.64 ± 0.003	0.65 ± 0.002	0.66 ± 0.002	0.67 ± 0.003	0.69 ± 0.002
50%	0.54 ± 0.007	0.55 ± 0.005	0.56 ± 0.003	0.57 ± 0.003	0.59 ± 0.004	0.60 ± 0.005
50%	0.54 ± 0.006	0.55 ± 0.004	0.56 ± 0.003	0.58 ± 0.003	0.59 ± 0.003	0.61 ± 0.002

4. NUMERICAL RESULTS

We use the SCNN to impute missing citations in a coauthorship complex, in which a paper with $k+1$ authors is represented by a k-simplex, and the k-simplicial signal is the number of citations of the paper. We followed the steps of [8], which lead to the citation dataset in Table 1. We compared the SCNN with the SNN in [8] for the k-simplicial signals with $k = 0, \ldots, 5$. Missing data are generated randomly on the k-simplicial signals at 5 rates, $10\%, 20\%, \ldots, 50\%$. The input of the SCNNs is the k-simplicial signal where missing citations are replaced by the median of known citations. As the SNN in [8], our SCNN has 3 layers with 30 convolutional filters of total length 5 ($L_1 = L_2 = 2$). We used LeakyReLU for $\sigma(\cdot)$ as in [8] although not odd. The reference orientation didn't seem to have much influence. We used the ℓ_1 norm to train the NNs over known citations for 1000 iterations using the Adam optimizer with a learning rate 10^{-3}.

We report the training loss of two instances in Fig. 1 and the mean accuracy $\pm$ the standard deviation in Table 1 over 10 different experiments. A citation value is considered to be correct if the imputed value is within $\pm 5\%$ of the true value. The proposed SCNN approach achieves a smaller training loss and a faster convergence than the SNN (Fig. 1) due to its better expressive power. From Table 1, we observe that both NNs perform similarly for dimensions 0 and 1. This is because for the former, the two NNs are the same, and for the latter, the data dimension is rather small. However, the SCNN gives consistently $1 - 2\%$ better accuracies for $k \geq 2$ with larger data dimensions. With addition hyperparameters tuning, the performance of SCNN could improve further.

5. CONCLUSION

We proposed a simplicial convolutional neural network architecture to learn from data defined on higher-order structures of a network, i.e., simplices in a simplicial complex. We built an SCNN layer through a composition of a simplicial filter and an elementwise nonlinearity. Due to the use of an advanced simplicial filter, our SCNN is able to learn from simplicial neighbours over multiple hops and process simplicial subcomponents (e.g., gradient and curl) independently, compared with the current solutions. The proposed SCNN applies to any k-simplicial signal case. We showed the SCNN is equivariant to permutations of the topology and to orientations of the flows, which allows it to exploit symmetries in a simplicial complex.

6. REFERENCES

[1] A. Ortega, P. Frossard, J. Kovačević, J. M. Moura, and P. Vandergheynst, "Graph signal processing: Overview, challenges, and applications," *Proceedings of the IEEE*, vol. 106, no. 5, pp. 808–828, 2018.

[2] F. Gama, E. Isufi, G. Leus, and A. Ribeiro, "Graphs, convolutions, and neural networks: From graph filters to graph neural networks," *IEEE Signal Processing Magazine*, vol. 37, no. 6, pp. 128–138, 2020.

[3] E. Isufi, M. Pocchiari, and A. Hanjalic, "Accuracy-diversity trade-off in recommender systems via graph convolutions," *Information Processing & Management*, vol. 58, no. 2, p. 102459, 2021.

[4] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[5] E. Isufi, F. Gama, and A. Ribeiro, "Edgenets: Edge varying graph neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[6] Z. Wu, S. Pan, F. Chen, G. Long, C. Zhang, and S. Y. Philip, "A comprehensive survey on graph neural networks," *IEEE transactions on neural networks and learning systems*, vol. 32, no. 1, pp. 4–24, 2020.

[7] S. Barbarossa and S. Sardellitti, "Topological signal processing over simplicial complexes," *IEEE Transactions on Signal Processing*, vol. 68, pp. 2992–3007, 2020.

[8] S. Ebli, M. Defferrard, and G. Spreemann, "Simplicial neural networks," *arXiv preprint arXiv:2010.03633*, 2020.

[9] G. Leus, M. Yang, M. Coutino, and E. Isufi, "Topological volterra filters," in *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2021, pp. 5385–5399.

[10] M. T. Schaub and S. Segarra, "Flow smoothing and denoising: Graph signal processing in the edge-space," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*. IEEE, 2018, pp. 735–739.

[11] M. T. Schaub, Y. Zhu, J.-B. Seby, T. M. Roddenberry, and S. Segarra, "Signal processing on higher-order networks: Livin'on the edge... and beyond," *arXiv preprint arXiv:2101.05510*, 2021.

[12] M. Yang, E. Isufi, M. T. Schaub, and G. Leus, "Finite impulse response filters for simplicial complexes," *arXiv preprint arXiv:2103.12587*, 2021.

[13] C. Bodnar, F. Frasca, Y. G. Wang, N. Otter, G. Montúfar, P. Lio, and M. Bronstein, "Weisfeiler and lehman go topological: Message passing simplicial networks," *arXiv preprint arXiv:2103.03212*, 2021.

[14] E. Bunch, Q. You, G. Fung, and V. Singh, "Simplicial 2-complex convolutional neural nets," *arXiv preprint arXiv:2012.06010*, 2020.

[15] T. M. Roddenberry, N. Glaze, and S. Segarra, "Principled simplicial neural networks for trajectory prediction," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9020–9029.

[16] T. M. Roddenberry and S. Segarra, "Hodgenet: Graph neural networks for edge data," in *2019 53rd Asilomar Conference on Signals, Systems, and Computers*. IEEE, 2019, pp. 220–224.

[17] L.-H. Lim, "Hodge laplacians on graphs," *SIAM Review*, vol. 62, no. 3, pp. 685–715, 2020.