ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-6327-7/23/$31.00 ©2023 IEEE | DOI: 10.1109/ICASSP49357.2023.10096866

POOLING STRATEGIES FOR SIMPLICIAL CONVOLUTIONAL NETWORKS

Domenico Mattia Cinque, Claudio Battiloro, Paolo Di Lorenzo

DIET Department, Sapienza University of Rome, Via Eudossiana 18, 00184, Rome, Italy
E-mail: {domenicomattia.cinque, claudio.battiloro, paolo.dilorenzo}@uniroma1.it

ABSTRACT

The goal of this paper is to introduce pooling strategies for simplicial convolutional neural networks. Inspired by graph pooling methods, we introduce a general formulation for a simplicial pooling layer that performs: i) local aggregation of simplicial signals; ii) principled selection of sampling sets; iii) downsampling and simplicial topology adaptation. The general layer is then customized to design four different pooling strategies (i.e., max, top-k, self-attention, and separated top-k) grounded in the theory of topological signal processing. Also, we leverage the proposed layers in a hierarchical architecture that reduce complexity while representing data at different resolutions. Numerical results on real data benchmarks (i.e., flow and graph classification) illustrate the advantage of the proposed methods with respect to the state of the art.

***Index Terms*—** Topological signal processing, topological deep learning, simplicial neural networks, pooling.

1. INTRODUCTION

In the last years, Graph Neural Networks (GNNs) [1–3] have shown remarkable results in learning tasks involving data defined on irregular domains (e.g., graphs), such as social networks, recommender systems, cybersecurity, natural language processing, genomics, and many more [3, 4]. However, GNNs are designed to work with graphs, which consider only pairwise relationships between data. On the contrary, many real-world phenomena involve multi-way relationships as, e.g., in biological or social networks. Some recent works in topological signal processing [5, 6] have shown that multi-way relationships can be described using simplicial complexes, which are specific instances of hyper-graphs with powerful algebraic representation able to model higher-order interactions among nodes.

Related works. Despite its recent birth, many contributions have been made to the field of simplicial deep learning. In [7], the authors introduced a basic simplicial neural network (SNN) architecture that performs convolution exploiting high-order Laplacians without independently exploiting upper and lower neighbourhoods. In [8], message passing neural networks (MPNNs) are adapted to simplicial complexes, with the aggregation and updating functions taking into account data defined on adjacent simplices, enabling message exchange even among signals of different orders. The work in [9] exploits the simplicial filters introduced in [10] to design a flexible and low-complexity simplicial convolutional networks (SCNs) with spectral interpretability. Finally, in [11, 12], simplicial attentional architectures are introduced.

Motivated by the fact that, both in convolutional neural networks (CNNs) and in GNNs, the introduction of pooling layers was proved to be useful for reducing the number of model parameters while improving the learning performance, in this work we aim to endow SCNs with pooling strategies. However, while for CNNs the pooling operation relies on aggregation based on the natural local neighbourhood provided by the regular grid domain, even on simpler graph domains the definition of local patches is not straightforward. Early works tried to overcome this issue by using graph clustering algorithms such as GraClus [13] or spectral methods [14] to produce a node assignment that generalizes the notion of locality present in regular domains. The most recent trends are instead focused on differentiable learnable operators that can learn a node assignment [15], or simply keep some nodes while discarding the others [16, 17]. Other works [18] discuss the class of *global pooling* methods that reduce the graph to a single vector, ignoring topological information. To the best of our knowledge, no previous works tackled the problem of pooling for SCNs.

Contribution. The goal of this work is to introduce pooling strategies for SCNs. Taking inspiration from the select-reduce-connect (SRC) paradigm [19], we introduce a general simplicial pooling layer that comprises three steps: i) a local aggregation step responsible for providing a meaningful summary of the input signals; ii) a selection step responsible for selecting a proper subset of simplices; finally, iii) a reduction step that downsamples the input complex and the aggregated signals of step i) based on the simplices selected in step ii). By tailoring steps ii) and iii), we introduce four different simplicial pooling layers that generalize the well-known graph pooling strategies. Also, we exploit the proposed simplicial pooling layers in a jumping knowledge (JK) hierarchical architecture [20], which aggregates the intermediate embeddings produced by the simplicial pooling layers to produce the final output. Finally, we assess the performance of the proposed methods on real-world graph and trajectory classification tasks, showing favorable comparisons with respect to other techniques in terms of performance and robustness to compression.

2. BACKGROUND

Simplicial complex and signals. Given a finite set of vertices $\mathcal{V}$, a k-simplex $\mathcal{H}_k$ is a subset of $\mathcal{V}$ with cardinality $k+1$. A face of $\mathcal{H}_k$ is a subset with cardinality k and thus a k-simplex has $k+1$ faces. A coface of $\mathcal{H}_k$ is a $(k+1)$-simplex that includes $\mathcal{H}_k$ [5,21]. The lower neighbourhood $\mathcal{N}_\downarrow$ of $\mathcal{H}_k$ is the set of simplices with which it shares a face. Similarly, the upper neighbourhood $\mathcal{N}_\uparrow$ of $\mathcal{H}_k$ is the set of simplices with which it

shares a co-face. A simplicial complex $\mathcal{X}_K$ of order K is a collection of k-simplices $\mathcal{H}_k$, $k = 0, \dots, K$ such that, for any $\mathcal{H}_k \in \mathcal{X}_K$ then $\mathcal{H}_{k-1} \in \mathcal{X}_K$ if $\mathcal{H}_{k-1} \subset \mathcal{H}_k$. We denote the set of k-simplex in $\mathcal{X}_k$ as $\mathcal{D}_k := \{\mathcal{H}_k : \mathcal{H}_k \in \mathcal{X}_K\}$, with $|\mathcal{D}_k| = N_k$ and $\mathcal{D}_k \subset \mathcal{X}_K$.

A k-simplicial signal is defined as a mapping from the set of all k-simplices contained in the complex to the real numbers:

$$\mathbf{z}_k : \mathcal{D}_k \to \mathbb{R}, \qquad k = 0, 1, \dots, K. \tag{1}$$

In this paper, w.l.o.g., we will focus on complexes $\mathcal{X}_2$ of order up to two, thus with a set of vertices $\mathcal{D}_0 = \mathcal{V}$ with $|\mathcal{V}| = V$, a set of edges $\mathcal{D}_1 = \mathcal{E}$ with $|\mathcal{E}| = E$ and a set of triangles $\mathcal{D}_2 = \mathcal{T}$ with $|\mathcal{T}| = T$. Given a simplex $\mathcal{H}_{k-1} \subset \mathcal{H}_k$, we write $\mathcal{H}_{k-1} \sim \mathcal{H}_k$ to indicate that the orientation of $\mathcal{H}_{k-1}$ is coherent with the one of $\mathcal{H}_k$, whereas $\mathcal{H}_{k-1} \not\sim \mathcal{H}_k$ if it is not.

Algebraic representations. The structure of a simplicial complex $\mathcal{X}_K$ is described by the set of its incidence matrices $\mathbf{B}_k$, with $k = 1, \dots, K$. The $\mathbf{B}_k$'s describe which k-simplices are incident to which $(k-1)$-simplices:

$$[\mathbf{B}_k]_{i,j} = \begin{cases} 0 & \mathcal{H}_{k-1,i} \not\subset \mathcal{H}_{k,j}; \\ 1 & \mathcal{H}_{k-1,i} \subset \mathcal{H}_{k,j} \text{ and } \mathcal{H}_{k-1,i} \sim \mathcal{H}_{k,j}; \\ -1 & \mathcal{H}_{k-1,i} \subset \mathcal{H}_{k,j} \text{ and } \mathcal{H}_{k-1,i} \not\sim \mathcal{H}_{k,j}. \end{cases} \tag{2}$$

From the incidence information, we can build the high order combinatorial Laplacian matrices [22] of order $k = 0, \dots, K$:

$$\begin{aligned} \mathbf{L}_0 &= \mathbf{B}_1 \mathbf{B}_1^T \\ \mathbf{L}_k &= \mathbf{B}_k^T \mathbf{B}_k + \mathbf{B}_{k+1} \mathbf{B}_{k+1}^T = \mathbf{L}_{k,d} + \mathbf{L}_{k,u} \\ \mathbf{L}_K &= \mathbf{B}_K^T \mathbf{B}_K. \end{aligned} \tag{3}$$

The term $\mathbf{L}_{k,d}$ in (3), also known as lower Laplacian, encodes the lower adjacency of k-order simplices; the second term $\mathbf{L}_{k,u}$, also known as upper Laplacian, encodes the upper adjacency of k-order simplices. Thus, for example, two edges are lower adjacent if they share a common vertex, whereas they are upper adjacent if they are faces of a common triangle. Note that the vertices of a graph can only be upper adjacent if they are incident to the same edge. This is why $\mathbf{L}_0$ contains only one term, and it corresponds to the usual graph Laplacian.

Hodge Decomposition. High order Laplacians admit a Hodge decomposition [21], such that any k-simplicial signal $\mathbf{z}_k \in \mathbb{R}^{N_k}$ can be decomposed as:

$$\mathbf{z}_k = \mathbf{B}_k^T \mathbf{z}_{k-1} + \mathbf{B}_{k+1} \mathbf{z}_{k+1} + \mathbf{z}_{k,h}, \tag{4}$$

for $k = 0, 1, \dots, K$. The first term $\mathbf{B}_k^T \mathbf{z}_{k-1}$ of (4) is called *irrotational* component, the second term $\mathbf{B}_{k+1} \mathbf{z}_{k+1}$ *solenoidal* component, and the third term $\mathbf{z}_{k,h}$ *harmonic* component. In the sequel, we will focus w.l.o.g. on edge flow signals and complexes of order 2. Therefore, we will drop the subscripts and denote $\mathbf{z}_1$ with $\mathbf{z}$, $\mathbf{L}_1$ with $\mathbf{L}$, $\mathbf{L}_{1,d}$ with $\mathbf{L}_d$, $\mathbf{L}_{1,u}$ with $\mathbf{L}_u$ and $\mathcal{X}_2$ with $\mathcal{X}$. Moreover, we denote the lower and upper neighborhoods of the i-th edge with $\mathcal{N}_{i,\downarrow}$ and $\mathcal{N}_{i,\uparrow}$, respectively.

3. SIMPLICIAL CONVOLUTIONAL NETWORKS WITH POOLING LAYERS

The Hodge decomposition in (4) suggests to separately filter the components of simplicial signals. Indeed, the work in [10] introduced linear shift-invariant (LSI) filters for simplicial signals, which can be seen as a generalization of LSI graph filters that exploit both upper and lower connectivities. A simplicial convolutional neural network is made by the concatenation of several layers composed by a point-wise non-linearity applied to a bank of LSI simplicial filters plus a residual connection [9]. In this paper, we generalize the layer structure of [9] introducing a a family of pooling strategies encoded into the mapping $\mathcal{P}(\cdot)$. In particular, letting $\mathbf{X} \in \mathbb{R}^{E \times G}$ be the matrix collecting G edge signals on its columns, the layer of an SCN endowed with pooling mechanisms (SCNP) can be written as:

$$\mathbf{Y} = \sigma\left[\mathcal{P}\left(\underbrace{\sum_{p=1}^{J_d} \mathbf{L}_d^p \mathbf{X} \mathbf{D}_p}_{\mathbf{Z}_d} + \underbrace{\sum_{p=1}^{J_u} \mathbf{L}_u^p \mathbf{X} \mathbf{U}_p}_{\mathbf{Z}_u} + \underbrace{\mathbf{X}\mathbf{H}}_{\mathbf{Z}_h}\right)\right], \tag{5}$$

where $\mathbf{Y} \in \mathbb{R}^{E' \times F}$, with $E' \leq E$; the filters and residual weights $\{\mathbf{D}_p\}_{p=1}^{J_d}$, $\{\mathbf{U}_p\}_{p=1}^{J_u}$ and $\mathbf{H} \in \mathbb{R}^{G \times F}$ are learnable parameters, while the order J_d and J_u of the filters, the number F of output signals, and the non-linearity $\sigma(\cdot)$ are hyperparameters to be chosen at each layer. Therefore, an SCNP of depth L is built as the stack of L layers defined as in (5); the SCN layer in [9] is recovered from (5) removing the pooling stage.

3.1. Design of simplicial pooling mapping

In this paragraph, we present a general formulation for a simplicial pooling mapping, which will then be tailored to design four different pooling strategies. Let us first denote the input to the pooling mapping in (5) as $\mathbf{Z} = \mathbf{Z}_d + \mathbf{Z}_u + \mathbf{Z}_h \in \mathbb{R}^{E \times F}$, and let the simplicial complex structure be encoded into $\mathcal{X} = (\mathbf{L}_u, \mathbf{L}_d)$. We also denote as $\mathbf{Z}' \in \mathbb{R}^{E' \times F}$ the output of the pooling layer in (5). Then, formally, we define a simplicial pooling layer as the mapping

$$\mathcal{P} : (\mathcal{X}, \mathbf{Z}) \mapsto (\mathcal{X}', \mathbf{Z}'), \tag{6}$$

which takes as input a simplicial complex $\mathcal{X}$ and signals $\mathbf{Z} \in \mathbb{R}^{E \times F}$ defined on it, and returns as output a sub-complex $\mathcal{X}' \subset \mathcal{X}$ and signals $\mathbf{Z}' \in \mathbb{R}^{E' \times F}$ defined on it, with $E' < E$.

Following the pooling paradigm introduced in [19] for GNNs, we propose to model the layer in (6) as the composition of three operations: a local aggregation step, a selection step, and a reduction step. The local aggregation step is responsible for providing summary signals $\widetilde{\mathbf{Z}} \in \mathbb{R}^{E \times F}$ of the input signals $\mathbf{Z} \in \mathbb{R}^{E \times F}$ leveraging the connectivity induced by the complex $\mathcal{X}$. Formally, we define it as the local mapping:

$$\textbf{(Aggregation)} \quad \mathcal{A} : (\mathcal{X}, \mathbf{Z}) \mapsto (\mathcal{X}, \widetilde{\mathbf{Z}}). \tag{7}$$

The mapping in (7) is local in the sense that the aggregated signals $[\widetilde{\mathbf{Z}}]_i$ of the i-th edge are function only of the signals of its (lower and/or upper) neighbours $[\widetilde{\mathbf{Z}}]_j$, $j \in \mathcal{N}_{i,\downarrow}$ and/or $\mathcal{N}_{i,\uparrow}$.

The selection step is responsible for choosing a subset $\mathcal{E}' \subset \mathcal{E}$ of edges that will compose the 1-skeleton of the sub-complex $\mathcal{X}'$. Formally, we define it as a mapping:

$$\textbf{(Selection)} \quad \mathcal{S} : (\mathcal{X}, \widetilde{\mathbf{Z}}) \mapsto \mathcal{E}'. \tag{8}$$

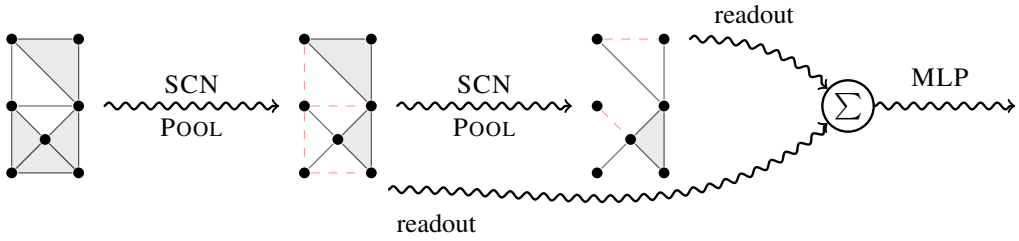

Fig. 1. Example of simplicial pooling and JK hierarchical architecture.

The cardinality of $\mathcal{E}'$ is tuned via the pooling ratio $r \in (0, 1]$ (a hyperparameter to be chosen), such that $|\mathcal{E}'| = E' = \lfloor r \cdot E \rfloor$.

Finally, the reduction step is responsible for properly downsampling the input complex $\mathcal{X}$ and the aggregated signals $\widetilde{\mathbf{Z}}$ to obtain the output sub-complex $\mathcal{X}'$ and the output signals $\mathbf{Z}' \in \mathbb{R}^{E' \times F}$, based on the edge set $\mathcal{E}'$ chosen through the selection set. Formally, we define it as a mapping:

$$\textbf{(Reduction)} \quad \mathcal{R} : (\mathcal{E}', \mathcal{X}, \widetilde{\mathbf{Z}}) \mapsto (\mathcal{X}', \mathbf{Z}'). \tag{9}$$

We assume that the reduction mapping in (9) is given by the concurrent application of two independent operations, i.e., $\mathcal{R} = (\mathcal{R}_S, \mathcal{R}_C)$, which separately downsample signal and simplicial complex structure, respectively, and are defined as:

$$\textit{(Signal reduction)} \quad \mathcal{R}_S : (\mathcal{E}', \widetilde{\mathbf{Z}}) \mapsto \mathbf{Z}', \tag{10}$$

$$\textit{(Complex reduction)} \quad \mathcal{R}_C : (\mathcal{E}', \mathcal{X}) \mapsto \mathcal{X}'. \tag{11}$$

The operations (10)-(11) compute the signals $\mathbf{Z}'$ and the complex structure $\mathcal{X}'$ at the output of the pooling layer, respectively.

In summary, the general pooling mapping $\mathcal{P}$ in (6) is given by the composition of the three operations in (7)-(9), i.e.,

$$\mathcal{P} = \mathcal{R} \circ \mathcal{S} \circ \mathcal{A}. \tag{12}$$

We assume that the aggregation in (7) is kept fixed (e.g., max or mean). Also, the complex reduction in (11) is computed as follows: if an edge e belongs to $\mathcal{E}$ but it is not in $\mathcal{E}'$, the lower connectivity is updated by disconnecting the nodes that are on the boundary of e, while the upper connectivity is updated by removing the triangles that have e on their boundaries.

3.2. Simplicial pooling strategies

In this paragraph, we customize the selection and signal reduction steps in (8) and (10) to design four pooling strategies.

Max pooling: The first method is an extension of the Max Pooling strategy commonly used in CNNs. It selects the subset of edges $\mathcal{E}'$ by ranking the absolute values of the sum of the aggregated signals $\widetilde{\mathbf{Z}}$ of each edge. Formally, we define:

$$\mathcal{S} : \quad \mathbf{y} = \left| \widetilde{\mathbf{Z}} \, \mathbf{1} \right|, \quad \mathcal{E}' = \text{top}_{E'}(\mathbf{y}), \tag{13}$$

$$\mathcal{R}_S : \quad \mathbf{Z}' = [\widetilde{\mathbf{Z}}]_{i \in \mathcal{E}'}, \tag{14}$$

where $\mathbf{1}$ is the vector of all ones, and $\text{top}_{E'}(\cdot)$ selects the indexes of the the E' largest entries of its vector argument.

Top-k pooling: The next layer is a generalization of those proposed in [16, 18] for GNNs. It selects the subset of edges $\mathcal{E}'$ ranking a learnable weighted combination of the aggregated signals $\widetilde{\mathbf{Z}}$ of each edge. Then, it computes the reduced signals as a scaled version of $\widetilde{\mathbf{Z}}$ with coefficients in $[0, 1]$ given by a normalization of the aforementioned weighted combination. Formally, we have:

$$\mathcal{S} : \quad \mathbf{y} = \frac{\widetilde{\mathbf{Z}} \, \mathbf{p}}{\|\mathbf{p}\|_2}, \quad \mathcal{E}' = \text{top}_{E'}(\mathbf{y}) \tag{15}$$

$$\mathcal{R}_S : \quad \mathbf{Z}' = [\widetilde{\mathbf{Z}} \odot \tanh(\mathbf{y}\mathbf{1}^T)]_{i \in \mathcal{E}'}, \tag{16}$$

where $\mathbf{p}$ is a learnable vector, and $\odot$ is the Hadamard product.

Self-Attention Pooling: This method is a generalization of SagPool [17]. The main difference with Top-k is that the ranking is computed over the output of a simplicial convolutional layer as in (5) without pooling and with one output signal, here briefly denoted as SCN. Formally, we have:

$$\mathcal{S} : \quad \mathbf{y} = \text{SCN}(\widetilde{\mathbf{Z}}, \mathbf{L}_d, \mathbf{L}_u), \quad \mathcal{E}' = \text{top}_{E'}(\mathbf{y}), \tag{17}$$

$$\mathcal{R}_S : \quad \mathbf{Z}' = [\widetilde{\mathbf{Z}} \odot \tanh(\mathbf{y}\mathbf{1}^T)]_{i \in \mathcal{E}'}. \tag{18}$$

Separated Top-k pooling: The Hodge Decomposition in (4) and the consequent structure of the SCNP layer in (5) suggest to design pooling layers based on the computation of three different aggregated signals: $\widetilde{\mathbf{Z}}_d$ (obtained from $\mathbf{Z}_d$), $\widetilde{\mathbf{Z}}_u$ (obtained from $\mathbf{Z}_u$), and $\widetilde{\mathbf{Z}}_h$ (obtained from $\mathbf{Z}_h$). Consequently, we will have three corresponding score vectors $\mathbf{y}_d$, $\mathbf{y}_u$, and $\mathbf{y}_h$, respectively. Thus, the "separated" version of the Top$-k$ layer in (15)-(16) is given by:

$$\mathcal{S} : \begin{cases} \mathbf{y}_d = \dfrac{\widetilde{\mathbf{Z}}_d \, \mathbf{p}_d}{\|\mathbf{p}_d\|}, \; \mathbf{y}_u = \dfrac{\widetilde{\mathbf{Z}}_u \, \mathbf{p}_u}{\|\mathbf{p}_u\|}, \; \mathbf{y}_h = \dfrac{\widetilde{\mathbf{Z}}_h \, \mathbf{p}_h}{\|\mathbf{p}_h\|} \\ \mathcal{E}' = \text{top}_{E'}(\mathbf{y}_d + \mathbf{y}_u + \mathbf{y}_h) \end{cases} \tag{19}$$

$$\mathcal{R}_S : \begin{cases} \mathbf{Z}'_d = [\widetilde{\mathbf{Z}}_d \odot \tanh(\mathbf{y}_d \mathbf{1}^T)]_{i \in \mathcal{E}'} \\ \mathbf{Z}'_u = [\widetilde{\mathbf{Z}}_u \odot \tanh(\mathbf{y}_u \mathbf{1}^T)]_{i \in \mathcal{E}'} \\ \mathbf{Z}'_h = [\widetilde{\mathbf{Z}}_h \odot \tanh(\mathbf{y}_h \mathbf{1}^T)]_{i \in \mathcal{E}'} \\ \mathbf{Z}' = \mathbf{Z}'_d + \mathbf{Z}'_u + \mathbf{Z}'_h \end{cases} \tag{20}$$

where $\mathbf{p}_u$, $\mathbf{p}_d$, and $\mathbf{p}_h$ are learnable vectors. Also all the previous methods can be reformulated in this "separated" version, but we leave their presentation and assessment for future works.

	Method	Graph Classification				Edge Flow Classification	
		DD	PROTEINS	MSRC21	NCI109	Ocean Drifters	Synthetic Flow
GCNs	Top-k	72.32 ± 2.65	70.82 ± 2.24	80.35 ± 4.37	68.67 ± 1.14	N/A	N/A
	SelfAtt	72.88 ± 2.47	71.02 ± 2.56	77.68 ± 1.09	$\mathbf{70.41} \pm 0.89$	N/A	N/A
SCNPs	NoPool (SCNs)	78.98 ± 2.64	70.61 ± 1.33	86.32 ± 3.38	68.77 ± 0.51	98.25 ± 1.11	100.0 ± 0.0
	Random	$\mathbf{79.66} \pm 2.16$	72.86 ± 3.11	88.42 ± 2.00	68.14 ± 1.24	76.25 ± 6.72	98.82 ± 0.58
	Max	72.88 ± 3.39	70.61 ± 2.33	89.82 ± 1.47	67.17 ± 1.62	100.0 ± 0.0	100.0 ± 0.0
	Top-k	75.42 ± 4.56	$\mathbf{76.61} \pm 2.04$	$\mathbf{90.18} \pm 1.57$	67.55 ± 2.45	$\mathbf{100.0} \pm 0.0$	100.0 ± 0.0
	SelfAtt	76.27 ± 3.37	71.63 ± 2.64	85.96 ± 2.48	66.78 ± 1.39	86.13 ± 5.89	99.99 ± 0.04
	SepTop-k	70.06 ± 2.05	71.63 ± 4.23	$\mathbf{90.18} \pm 3.64$	67.65 ± 1.32	99.62 ± 0.56	$\mathbf{100.0} \pm 0.0$

Table 1. Accuracy on graph and trajectory classification. The results are reported as mean $\pm$ std over 5 random seeds.

3.3. Hierarchical Architecture

In this paragraph, we introduce a JK hierarchical architecture aimed at exploiting the different data representations obtained after each pooling stage $l \in \{1, ..., L\}$. A pictorial overview of the proposed JK hierarchical architecture is shown in Fig. 1. In particular, applying a readout operation, each intermediate compact representation obtained at the output of layer l collapses the current signals (and complex) into a single embedding vector. For instance, a possible choice is concatenating the mean and the maximum of the current signals. These vectors are then aggregated to compose a global final embedding. For instance, if the same number of output signals is used at each layer, the intermediate representations can be summed to obtain a single global embedding vector. Finally, the global embedding can be passed through a multi-layer perceptron (MLP), if it is needed for the task. In the case of transductive (semisupervised) tasks, both the intermediate and global embeddings might be unnecessary and can be neglected.

4. NUMERICAL EXPERIMENTS

In this section, we assess the performance of the proposed simplicial pooling layers and hierarchical architecture on two learning tasks: trajectory classification [8], and real-world graph [23] classification [1]. We compare the four proposed simplicial pooling layers with a random pooling strategy, and with plain SCNs having no pooling layers. Also, for graph classification, we show the results obtained using GCNs [3] equipped with the "graph counterpart" of the proposed simplicial pooling layers. All the hyper-parameters are tuned to obtain the best performance per each dataset, except for the pooling ratio, which we keep fixed at $r = 0.7$. We compute the intermediate and global embeddings via mean-maximum concatenation and sum, respectively. The models are trained for 150 epochs using the Adam optimizer [24] and early stopping with patience 25. All the experiments are averaged on five random seeds.

We first test the proposed simplicial pooling on two flow classification tasks, namely the synthetic flow and ocean drifter datasets, whose details can be found in [6, 8]. In Table 1 (right side), we compare the accuracy obtained by all the considered methods, illustrating the gain introduced by the proposed simplicial pooling layers for both datasets. Then, to assess the accuracy-complexity tradeoff obtained by the proposed strategies, in Fig. 2 we show the accuracy of the classification task

[1] https://github.com/domenicocinque/spm

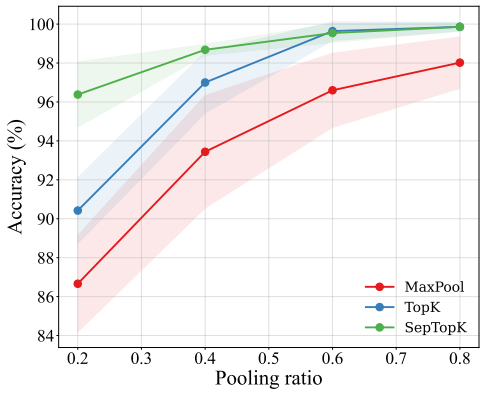

Fig. 2. Accuracy versus pooling ratio.

versus the pooling ratio for the synthetic flow dataset considering three pooling methods. As we can see from Fig. 2, the accuracy mildly decreases with the pooling ratio, especially for the Separated Top-k strategy, illustrating the very good accuracy-complexity trade-off obtained by the proposed methods.

Finally, we study the performance of the proposed simplicial pooling layers on real-world graph classification tasks on the popular TUDataset [23] collection. To obtain simplicial complexes from the graphs, we follow the clique complex lifting procedure proposed in [8], while the input edge signals are computed as the average of the graph signals of the boundary nodes. Then, in Table 1 (left side), we can see how, in most cases, the proposed SCNPs outperform SCNs with random pooling layers or no pooling, and the GCNs counterpart architectures.

5. CONCLUSIONS

In this paper, we have proposed a general formulation of a pooling layer for simplicial convolutional neural networks, designed as the composition of a local aggregation mapping, a selection mapping, and a reduction mapping. The proposed methodology is then tailored to design four different simplicial pooling layers, which generalize known graph pooling strategies for simplicial neural architectures. Numerical results on real and synthetic benchmarks illustrate the favorable performance of the proposed strategies with respect to other methods available in the literature. Future extensions include more complex simplicial architectures [8], or cell complex neural networks [25–27].

6. REFERENCES

[1] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst, "Geometric deep learning: going beyond euclidean data," *IEEE Signal Processing Magazine*, vol. 34, no. 4, pp. 18–42, 2017.

[2] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun, "Spectral networks and locally connected networks on graphs," *arXiv preprint arXiv:1312.6203*, 2013.

[3] Thomas N Kipf and Max Welling, "Semi-supervised classification with graph convolutional networks," *arXiv preprint arXiv:1609.02907*, 2016.

[4] D. Shuman, S. K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *IEEE Signal Processing Magazine*, vol. 30, 10 2012.

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

[6] Michael T Schaub, Austin R Benson, Paul Horn, Gabor Lippner, and Ali Jadbabaie, "Random walks on simplicial complexes and the normalized hodge 1-laplacian," *SIAM Review*, vol. 62, no. 2, pp. 353–391, 2020.

[7] Stefania Ebli, Michaël Defferrard, and Gard Spreemann, "Simplicial neural networks," in *NeurIPS 2020 Workshop on Topological Data Analysis and Beyond*, 2020.

[8] Cristian Bodnar, Fabrizio Frasca, Yu Guang Wang, Nina Otter, Guido Montufar, Pietro Liò, and Michael M Bronstein, "Weisfeiler and lehman go topological: Message passing simplicial networks," in *ICLR 2021 Workshop on Geometrical and Topological Representation Learning*, 2021.

[9] Maosheng Yang, Elvin Isufi, and Geert Leus, "Simplicial convolutional neural networks," in *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2022, pp. 8847–8851.

[10] Maosheng Yang, Elvin Isufi, Michael T. Schaub, and Geert Leus, "Finite impulse response filters for simplicial complexes," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 2005–2009.

[11] L. Giusti, Claudio Battiloro, Paolo Di Lorenzo, Stefania Sardellitti, and Sergio Barbarossa, "Simplicial attention neural networks," *ArXiv*, vol. abs/2203.07485, 2022.

[12] Christopher Wei Jin Goh, Cristian Bodnar, and Pietro Lio, "Simplicial attention networks," in *ICLR 2022 Workshop on Geometrical and Topological Representation Learning*, 2022.

[13] Inderjit S Dhillon, Yuqiang Guan, and Brian Kulis, "Weighted graph cuts without eigenvectors a multilevel approach," *IEEE transactions on pattern analysis and machine intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[14] Yao Ma, Suhang Wang, Charu C. Aggarwal, and Jiliang Tang, "Graph convolutional networks with eigenpooling," *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019.

[15] Zhen Zhang, Jiajun Bu, Martin Ester, Jianfeng Zhang, Chengwei Yao, Zhi Yu, and Can Wang, "Hierarchical graph pooling with structure learning," *arXiv preprint arXiv:1911.05954*, 2019.

[16] Hongyang Gao and Shuiwang Ji, "Graph u-nets," *IEEE transactions on pattern analysis and machine intelligence*, vol. PP, 2019.

[17] Junhyun Lee, Inyeop Lee, and Jaewoo Kang, "Self-attention graph pooling," in *36th International Conference on Machine Learning, ICML 2019*. International Machine Learning Society (IMLS), 2019, pp. 6661–6670.

[18] Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò, "Towards sparse hierarchical graph classifiers," *arXiv preprint arXiv:1811.01287*, 2018.

[19] Daniele Grattarola, Daniele Zambon, Filippo Maria Bianchi, and Cesare Alippi, "Understanding pooling in graph neural networks," *arXiv:2110.05292*, 2021.

[20] Keyulu Xu, Chengtao Li, Yonglong Tian, Tomohiro Sonobe, Ken-ichi Kawarabayashi, and Stefanie Jegelka, "Representation learning on graphs with jumping knowledge networks," in *International conference on machine learning*. PMLR, 2018, pp. 5453–5462.

[21] Lek-Heng Lim, "Hodge laplacians on graphs," *Siam Review*, vol. 62, no. 3, pp. 685–715, 2020.

[22] T.E. Goldberg, "Combinatorial laplacians of simplicial complexes," *Senior Thesis, Bard College*, 2002.

[23] Christopher Morris, Nils M. Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann, "Tu-dataset: A collection of benchmark datasets for learning with graphs," in *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[24] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, 2015.

[25] Cristian Bodnar, Fabrizio Frasca, Nina Otter, Yuguang Wang, Pietro Lio, Guido F Montufar, and Michael Bronstein, "Weisfeiler and lehman go cellular: Cw networks," *Advances in Neural Information Processing Systems*, vol. 34, pp. 2625–2640, 2021.

[26] Mustafa Hajij, Kyle Istvan, and Ghada Zamzmi, "Cell complex neural networks," *arXiv preprint arXiv:2010.00743*, 2020.

[27] Lorenzo Giusti, Claudio Battiloro, Lucia Testa, Paolo Di Lorenzo, Stefania Sardellitti, and Sergio Barbarossa, "Cell attention networks," *arXiv:2209.08179*, 2022.