Cell Complex Neural Networks

Mustafa Hajij
Department of Mathematics & Computer Science
Santa Clara University
mhajij@scu.edu

Kyle Istvan
kyleistvan@gmail.edu

Ghada Zamzmi
Department of Computer Science & Engineering
University of South Florida
ghadh@mail.usf.edu

Abstract

Cell complexes are topological spaces constructed from simple blocks called cells. They generalize graphs, simplicial complexes, and polyhedral complexes that form important domains for practical applications. They also provide a combinatorial formalism that allows the inclusion of complicated relationships of restrictive structures such as graphs and meshes. In this paper, we propose **Cell Complexes Neural Networks (CXNs)**, a general, combinatorial and unifying construction for performing neural network-type computations on cell complexes. We introduce an inter-cellular message passing scheme on cell complexes that takes the topology of the underlying space into account and generalizes message passing scheme to graphs. Finally, we introduce a unified cell complex encoder-decoder framework that enables learning representation of cells for a given complex inside the Euclidean spaces. In particular, we show how our cell complex autoencoder construction can give, in the special case **cell2vec**, a generalization for node2vec.

1. Introduction

Motivated by the recent success of neural networks in various domains and data types (e.g., [30, 31, 42, 45, 44, 4]), we propose *Cell Complex Neural Networks (CXNs)*, a general unifying scheme that allows neural network-type computations on cell complexes; i.e., we define a neural network structure on cell complexes.

Cell complexes are topological spaces constructed from pieces called *cells* that are homeomorphic to topological balls of varying dimensions. They form a natural generalization of graphs, simplicial complexes, and polyhedral complexes [25]. They also provide a combinatorial formalism that allows the inclusion of complicated relationships not available to more restrictive structures such as graphs and simplicial complexes, while retaining most of intuitive structure of these simpler objects.

Because the simplest nontrivial types of cell complexes are graphs [25], our work can be considered as a generalization of Graph Neural Networks (GNNs) [11]. Earlier work that generalizes regular Convolutional Neural Network (CNN) to graphs was presented in [19] and extended in [38, 18, 28]. Further, a significant effort has been made towards generalizing deep learning to manifolds, most notably geometric deep learning and the work of Bronstein et al. [10, 8]. Other work includes utilizing filters on local patches using geodesic polar coordinates [33] and heat kernels propagation schemes [9], among many others [29, 9, 41, 49, 34]. We refer the reader to recent reviews [53, 48] of GNNs and its variations and to [13] for a recent survey on geometric deep learning.

Topological Data Analysis and Beyond Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

The main contributions of this work are summarized as follows. We propose *CXNs*, a general unifying and simple training scheme on cell complexes that vastly expands the domains upon which we can apply deep learning protocols. Our method encompasses most of the popular types of GNNs, and generalizes those architectures to higher-dimensional domains such as 3D meshes, simplicial complexes, and polygonal complexes. The training on a cell complex is defined in an entirely combinatorial fashion, and hence, naturally extends general message passing schemes currently utilized by GNNs. This combinatorial description allows for intuitive manipulation, conceptualization, and implementation.

We introduce an *inter-cellular message passing scheme* on cell complexes that takes the topology of the underlying space into account. Precisely, we define a message passing scheme that is induced from the boundary and coboundary maps used to compute homology and cohomology in algebraic topology. As a concrete example of this scheme, we define *Convolutional Cell Complex Networks (CCXN)*. Also, we propose a *Cell Complex Autoencoder (CXNA)* to incorporate CXNs in a deep learning model and meaningfully represent cells of the complex in the Euclidean space. We provide examples of representational learning on cell complexes that generalize well-known representational learning on graphs such as graph factorization [2] and node2vec [2]. Computationally, a cell complex net is defined using adjacency matrices, analogous to those used to encode the structure of a graph neural network. This means their implementation can be readily adapted from the existing graph neural networks libraries (e.g., [17]). The rest of this paper is organized as follows. Section 1.1 presents background and important notations about cell complexes. The proposed *CXNs* is presented in Section 2. Cell complex autoencoder *CXNA* is introduced in Section 3. The topology-based message passing scheme on CXNs is discussed in Section 4.4 of the Appendix followed by a presentation of potential applications in Section 4.5.

1.1. Cell Complexes: Background and Notations

A cell complex is a topological space X obtained as a disjoint union of *cells*, each of these cells is homeomorphic to the interior of a k-Euclidean ball for some k. These cells are attached together via attaching maps in a locally reasonable manner[1]. In our setting, the set of k-cells in X is denoted by X^k, and it is called the $k-$ *skeleton* of X. The set of all cells in X whose dimension is less than k is denoted by $X^{<k}$. The set $X^{>k}$ is defined similarly. The dimension of a cell $c \in X$ is denoted by $d(c)$, and the dimension of a cell complex is the largest dimension of one of its cells.

A cell complex is called *regular* if every attaching map is a homeomorphism onto the closure of the image of its corresponding cell. In this paper, all cell complexes will be regular and consist of finitely many cells. Regular cell complexes generalize graphs, simplicial complexes, and polyhedral complexes while retaining many desirable combinatorial and intuitive properties of these simpler structures. The information of attaching maps of a regular cell-complex are stored combinatorially in a sequence of matrices called the boundary matrices ($\partial_k : \mathbb{R}^{|X^k|} \to \mathbb{R}^{|X^{k-1}|}$). These matrices describe, roughly speaking, the number of times k-cells wrap around $(k-1)$-cells in X. The definition of these matrices ∂_k depends on whether the cells of X are oriented or not. Our method is applicable to both oriented and non-oriented cell complexes. However, we only discuss the non-oriented case for the sake of brevity, and leave the oriented case to section 4.3 of the Appendix. Since our cell complexes are regular and non-oriented, the entries of ∂_k are in $\{0,1\}$. Dually, we define $\partial_k^* : \mathbb{R}^{|X^{k-1}|} \to \mathbb{R}^{|X^k|}$ to be the transpose of the matrix ∂_k.

Let X be a cell complex, c^n denotes an n-cell in X, and $facets(c^n)$ denotes the set of all $(n-1)$-cells X that are incident to c^n. Similarly, let $cofacets(c^n)$ denotes the set of all $(n+1)$-cells of X that are incident to c^n. Note that $facets(c^n)$ or $cofacets(c^n)$ might be the empty set. We say that two n-cells a^n and b^n are *adjacent* if there exists an $(n+1)$-cell c^{n+1} such that $a^n, b^n \in facets(c^{n+1})$. Likewise, we say that a^n and b^n are *coadjacent* in X if there exists an $(n-1)$-cell c^{n-1} with $a^n, b^n \in cofacets(c^{n-1})$. The set of all cells adjacent to a cell a in X is denoted by $\mathcal{N}_{adj}(a)$ while the set of all cells coadjacent to a cell a in X is denoted by $\mathcal{N}_{co}(a)$. If a^n, b^n are n-cells in X, then we define the set $\mathcal{CO}[a^n, b^n]$ to be the intersection of $cofacets(a^n) \cap cofacets(b^n)$. Note that this notation is symmetric: $\mathcal{CO}[a^n, b^n] = \mathcal{CO}[b^n, a^n]$ [2]. Similarly, the set $\mathcal{C}[a^n, b^n]$ is defined to be the intersection of $facets(a^n) \cap facets(b^n)$.

[1] The reader is referred to [25] for further technical details of cell complex definition and algebraic topology.

[2] This is not the case when X is not oriented. See Section 4 in the Appendix for more details.

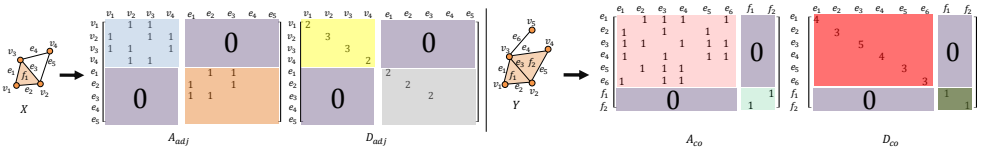

Figure 1: Examples of adjacency and coadjacency matrices for simplicial complexes. Left: the adjacency matrix A_{adj} and adjacency degree matrix D_{adj} of the simplicial complex X. The blue and the orange submatrices in A_{adj} represent A^0_{adj} and A^1_{adj} of X, respectively. The yellow and grey submatrices of D_{adj} represent D^0_{adj} and D^1_{adj}, respectively. Right: the coadjacency matrix A_{co} and coadjacency degree matrix D_{co} of the simplicial complex Y. The pink and the light green matrices in A_{co} represent A^1_{co} and A^2_{co} of Y, respectively. The red and dark green submatrices of D_{co} represent D^1_{co} and D^2_{co}, respectively.

Note that these notions generalize the analogous notions of adjacency and coadjacency matrices defined on graphs. More precisely, let X be a cell complex of dimension n. Let N denotes the total number of cells in the complex X and define $\hat{N} := N - |X^n|$. Let $c_1,\cdots,c_{\hat{N}}$ denotes all the cells in $X^{<n}$. Then, we define the matrix A_{adj} of dimension $\hat{N} \times \hat{N}$ by storing a $|\mathcal{CO}[c_i, c_j]|$ in $A_{adj}(i,j)$ if the cell c_i is adjacent to c_j, otherwise, we store a 0 in $A_{adj}(i,j)$. Note that the matrix $A_{adj}(i,j)$ does not store the adjacency information of n-cells in X since the dimension of the complex X is n. We denote the adjacency matrix between k-cells in X by A^k_{adj}, where $0 \leq k < n$. The *adjacency degree matrix* D_{adj} is defined via $D_{adj}(i,i) = \sum_j A_{adj}(i,j)$, and hence, we define the *adjacency degree matrix between k-cells* D^k_{adj} $(0 \leq k < n)$ similarly. Finally, the coadjacency matrix A_{co}, the coadjacency degree matrix D_{co}, the k-cells coadjacency matrices A^k_{co}, and the k-cells coadjacency degree matrices D^k_{co} for $0 < k \leq n$, are defined similarly. Examples of these matrices are presented in Figure 1.

2. Cell Complex Neural Networks (CXNs)

We define below a general CXNs using a message passing scheme that generalizes the notations of message passing schemes in graphs. Section 4 of the Appendix provides a brief review of message passing schemes on graphs (Section 4.2), and introduce message passing schemes on cell complexes (Section 4.4). We also present here *CCXN* (Section 2.1) as an example of CXNs. The forward propagation computation of a cell complex neural net requires the following data as inputs: (1) A cell complex X of dimension n, possibly oriented and (2) For each m-cell c^m in X, we have an initial vector $h^{(0)}_{c^m} \in \mathbb{R}^{l^0_m}$. The forward propagation algorithm then performs a sequence of message passing executed between cells in X. Precisely, given the desired depth $L > 0$ of the net one wants to define on the complex X, the forward propagation algorithm on X consists of $L \times n$ *inter-cellular message passing scheme* defined for $0 < k \leq L$:

$$h^{(k)}_{c^0} := \alpha^{(k)}_0\left(h^{(k-1)}_{c^0}, E_{a^0 \in \mathcal{N}_{adj}(c^0)}\left(\phi^{(k)}_0(h^{(k-1)}_{c^0}, h^{(k-1)}_{a^0}, F_{e^1 \in \mathcal{CO}[a^0,c^0]}(h^{(k-1))}_{e^1})\right)\right) \in \mathbb{R}^{l^k_0}, \tag{1}$$

$$\vdots$$

$$h^{(k)}_{c^{n-1}} := \alpha^{(k)}_{n-1}\left(h^{(k-1)}_{c^{n-1}}, E_{a^{n-1} \in \mathcal{N}_{adj}(c^{n-1})}\left(\phi^{(k)}_{n-1}(h^{(k-1)}_{c^{n-1}}, h^{(k-1)}_{a^{n-1}}, F_{e^n \in \mathcal{CO}[a^{n-1},c^{n-1}]}(h^{(0)}_{e^n})\right)\right) \in \mathbb{R}^{l^k_{n-1}} \tag{2}$$

where $h^{(k)}_{e^m}, h^{(k)}_{a^m}, h^{(k)}_{c^m} \in \mathbb{R}^{l^k_m}$, E, F are permutation invariant differentiable functions[3], and $\alpha^{(k)}_j, \phi^{(k)}_j$ are trainable differentiable functions where, $0 \leq j \leq n-1$ and $0 < k \leq L$ [4]. Note that for each cell a^n in X, the vectors $h^{(0)}_{a^n}$ are never updated during the training of a CXN. Although the formulation above is simple, it can generalize most types of the popular types of GNNs (e.g., [28, 46]).

[3]The permutation invariance condition on F and E may not be necessary in general (e.g. on triangulated meshes only cyclic invariance is needed). However, a permutation invariant function is needed whenever there is no canonical way to order the cofaces adjacent to a face in the complex.

[4]Examples of the functions $\alpha^{(k)}_j, \phi^{(k)}_j$ in practice are MLP. A concrete example is given in 2.1

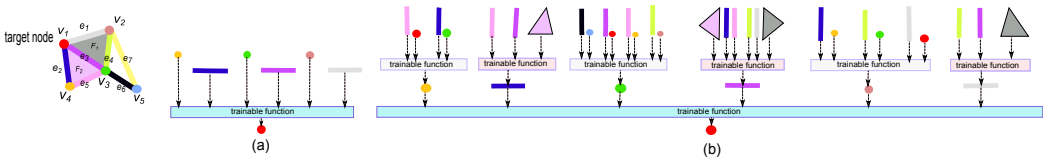

Figure 2: Two layers Cell Complex Neural Network (CXN). The computation is demonstrated with respect to the red target vertex. The information flow goes from lower cells to higher incident cells.

Figure 2 demonstrates the above construction/formulation on a simplicial complex network with depth $L = 2$. We will use X to denote this complex. Note that we abuse the notation in the figure and do not distinguish between a simplex s and its vector $h_s^{(k)}$. For each vertex $\{v_i\}_{i=1}^5$ in X, we assume we are given a vector $h_{v_i}^{(0)}$; we have $h_{e_j}^{(0)}$ for each edge $\{e_j\}_{j=1}^7$, and have $h_{F_i}^{(0)}$ for the faces $\{F_i\}_{i=1}^2$. In the first stage, we start the computation for cells with dimension 0. In this stage, each v_i, $0 \leq i \leq 5$, computes : $h_{v_i}^{(1)} := \alpha_0^{(k)}\left(h_{v_i}^0, E_{v_j \in \mathcal{N}_{adj}(v_i)}\left(\phi_0^{(1)}(h_{v_i}^{(0)}, h_{v_j}^{(0)}, h_{e_{ij}}^{(0)})\right)\right)$, where e_{ij} is the edge that connects v_i to v_j. Figure 2 (a) shows this graph for v_1. Notice that all nodes share the same trainable functions $\alpha_0^{(1)}$ and $\phi_0^{(1)}$. Further, each edge e_i induces a computational graph and computes $h_{e_i}^{(1)}$, $1 \leq i \leq 7$: $h_{e_i}^{(1)} := \alpha_1^{(k)}\left(h_{e_i}^0, E_{e_j \in \mathcal{N}_{adj}(e_i)}\left(\phi_1^{(1)}(h_{e_i}^{(0)}, h_{e_j}^{(0)}, h_{F_{ij}}^{(0)})\right)\right)$, here F_{ij} denotes the unique face that bounds both edges e_i and e_j. Note that all edges share the same trainable functions $\alpha_1^{(1)}$ and $\phi_1^{(1)}$. In stage 2, we compute $h_s^{(2)}$ for all simplices s of dimension 0 and 1. Figure 2 shows this computation for $h_{v_1}^{(2)}$. Note that (1) the computational graphs that feed into it are the ones computed in stage 1 and (2) how the information from this node flows from the surrounding nodes, edges, and faces.

2.1. Convolutional Cell Complex Nets (CCXN)

We present *CCXN*, the simplest type of cell complex neural networks. Specifically, using the adjacency matrices on a cell complex X defined in 1.1 and Figure 1, we extend the definition of convolutional graph neural networks (*CGNN*) [28] to *CCXN*. The input for a CCXN is specified by cell embeddings $H^{(0)} \in \mathbb{R}^{\hat{N} \times d}$ that define the initial cells features on every cell in $X^{<n}$. Here, d is the embedding dimension of the cells. The *convolutional message passing scheme* on X is defined by :

$$H^{(k)} := \text{ReLU}(\hat{A}_{adj} H^{(k-1)} W^{(k-1)}) \tag{3}$$

where $\hat{A}_{adj} = I_{\hat{N}} + D_{adj}^{-1/2} A_{adj} D_{adj}^{-1/2}$, $H^{(k)} \in \mathbb{R}^{\hat{N} \times d}$ are the cell embeddings computed after k steps of applying 3, and $W^{(k-1)} \in \mathbb{R}^{d \times d}$ is a trainable weight matrix at the layer k. We discuss the following few remarks about the definition above. First, observe that we chose the embedding dimension of cells to be d for all $H^{(k)}$. However, this restriction is not necessary in general and we chose it only for notational convenience. Second, we train the CCXN with a single weight W^k for every layer k for all cells. This restriction is also not necessary in general. As indicated in equations (1,2), one may choose to train a different CCXN for every k-cells adjacency matrix A_{adj}^k individually. In this case, we need to train $n - 1$ cell complex networks. Finally, the matrix $\hat{A}_{adj}$ in equation 3 is typically normalized to avoid numerical instabilities when stacking multiple layers [28]. Specifically, with the *renormalization trick*, we make the substitution $I_{\hat{N}} + D_{adj}^{-1/2} A_{adj} D_{adj}^{-1/2} \to \tilde{D}_{adj}^{-1/2} \tilde{A}_{adj} \tilde{D}_{adj}^{-1/2}$ in equation 3 where $\tilde{A}_{adj} = A_{adj} + I_{\hat{N}}$ and $\tilde{D}_{adj}(i,i) = \sum_j \tilde{A}_{adj}(i,j)$. Observe that with this simplified case, this version for CCXN is a generalization of CGNN where the only difference being the generalized notion of adjacency on cell complexes.

3. Cell Complex Autoencoders (CXNA) and Cell Complex Representations

In this section, we present how to incorporate the cell complex structure into a deep learning model. Given a cell complex X, we want to learn a function that embeds the cells of X into some Euclidean space such that the structure information of these cells is preserved. Inspired by the success of graph

autoencoders in representational learning on graphs [24], we propose a general method to learn cell complex representations. While the method provided in [24] is general and encompasses various representational learning strategies (e.g., Graph Factorization [2], node2vec [20], and DeepWalk [35]), it assumes a node to node message passing scheme using edges. Because our setting has to accommodate for different message passing schemes on a cell complex, we present below an autoencoder definition that is consistent with the inter-cellular message passing scheme in equation 1. Other cell autoencoder definitions that are consistent with different message passing schemes can be defined similarly. It is important to note that we are aware of a related work [21] appeared simultaneously with our work, and [6, 39] where a vector representation based on random walks on simplices in a simplicial complex was suggested. Contrary to these works, our approach is a unified framework that describes these special cases to a larger set of vector-based representations and a larger set of complexes.

A cell complex autoencoder consists of three components: (1) an encoder-decoder system, the trainable component of the autoencoder, (2) a similarity measure on the cell complex, which is a user-defined similarity function that represents a notion of similarity between the cells in the complex, and (3) a loss function, which is a user-defined function utilized to optimize the encoder-decoder system according to the similarity measure. Mathematically, let X be a cell complex of dimension n. Then, an *encoder* on X is a function of the form:

$$enc : X^{<n} \to \mathbb{R}^d.$$

This encoder associates to every k-cell c^k ($0 \leq k < n$) a feature vector $\mathbf{z}_c \in \mathbb{R}^d$ that encodes the structure of this cell and its relationship to other cells in X. A *decoder* is a function of the form:

$$dec : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$$

This decoder associates to every pair of cell embeddings a measure of similarity that quantifies some notion of relationship between these cells. The pair (enc,dec) on X is called a *cell complex encoder-decoder system* on X [5]. The functions enc and dec are typically trainable functions that are optimized using user-defined similarity measure and loss function. A similarity measure on a cell complex is a function of the form $sim_X : X^{<n} \times X^{<n} \to \mathbb{R}^+$ such that $sim_X(a^k,c^l)$ reflects a user-defined similarity between the two cells a^k and c^l in $X^{<n}$. We will assume that $sim_X(a^k,c^l) = 0$ whenever $k \neq l$. An example of a similarity measure defined on X is A_{adj} defined in Section 2.1. We want the encoder-decoder system specified above to learn a representation embedding of the cells in $X^{<n}$ such that: $dec(enc(a^k), enc(c^l)) = dec(\mathbf{z}_{a^k}, \mathbf{z}_{c^l}) \approx sim_X(a^k,c^l)$. To this end, let $l : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ be a user-defined loss function and define:

$$\mathcal{L}_k = \sum_{\text{all possible } \mathcal{CO}[a^k,c^k] \subset X^{k+1}} l(dec(enc(\mathbf{z}_{a^k}), enc(\mathbf{z}_{c^k})), sim(a^k,c^k)), \tag{4}$$

and finally define $\mathcal{L} := \sum_{k=0}^{n-1} \mathcal{L}_k$.

Different concrete CXNAs can be provided as shown in Table 1. After training the encoder-decoder model, we can use the encoder to generate the embeddings for k-cell, $0 \leq k < n$.

Table 1: Various definitions of CXNAs.

Method	Decoder	similarity	Loss
Laplacian eigenmaps [5]	$\|\mathbf{z}_a - \mathbf{z}_c\|_2^2$	general	$dec(\mathbf{z}_a,\mathbf{z}_c).sim(a,c)$
Inner product methods [1]	$\mathbf{z}_a^T \mathbf{z}_c$	$A_{adj}(a,c)$	$\|dec(\mathbf{z}_a,\mathbf{z}_c) - sim(a,c)\|_2^2$
Random walk methods [20, 35]	$\frac{e^{\mathbf{z}_a^T \mathbf{z}_c}}{\sum_{b \in X^k} e^{\mathbf{z}_a^T \mathbf{z}_b}}$	$p_X(a\|c)$	$-log(dec(\mathbf{z}_a,\mathbf{z}_c))$

We end this section by noting how the random walk method given in Table 1 effectively defines *cell2vec*, a cell complex representation method that generalizes node2vec [20] and DeepWalk [35] to cell complexes[6].

[5] In our definition of the encoder-decoder system, we chose to embed all cells on X in the same ambient space. This assumption is not needed in general and we are only making this restriction for notational convenience. Alternatively, one may have a sequence of similarity measure to describe the similarity between cells that have the same dimension.

[6] Random walks on graphs can be naturally extended to cell complexes by using the adjacency relations on cell complexes as a mean to define a random walk between cells of the same dimension.

Referencias

[1] A. Ahmed, N. Shervashidze, S. Narayanamurthy, V. Josifovski, and A. J. Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pages 37–48, 2013.

[2] R. Angles and C. Gutierrez. Survey of graph database models. *ACM Computing Surveys (CSUR)*, 40(1):1–39, 2008.

[3] S. Arora. A survey on graph neural networks for knowledge graph completion. *arXiv preprint arXiv:2007.12374*, 2020.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[5] M. Belkin and P. Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. *Advances in neural information processing systems*, 14:585–591, 2001.

[6] J. C. W. Billings, M. Hu, G. Lerda, A. N. Medvedev, F. Mottes, A. Onicas, A. Santoro, and G. Petri. Simplex2vec embeddings for community detection in simplicial complexes. *arXiv preprint arXiv:1906.09068*, 2019.

[7] D. Bommes, B. Lévy, N. Pietroni, E. Puppo, C. Silva, M. Tarini, and D. Zorin. Quad-mesh generation and processing: A survey. In *Computer Graphics Forum*, volume 32, pages 51–76. Wiley Online Library, 2013.

[8] D. Boscaini, J. Masci, S. Melzi, M. M. Bronstein, U. Castellani, and P. Vandergheynst. Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. In *Computer Graphics Forum*, volume 34, pages 13–23. Wiley Online Library, 2015.

[9] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in neural information processing systems*, pages 3189–3197, 2016.

[10] M. M. Bronstein, J. Bruna, Y. LeCun, A. Szlam, and P. Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

[11] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[12] E. Bunch, Q. You, G. Fung, and V. Singh. Simplicial 2-complex convolutional neural nets. *NeurIPS workshop on Topological Data Analysis and Beyond*, 2020.

[13] W. Cao, Z. Yan, Z. He, and Z. He. A comprehensive survey on geometric deep learning. *IEEE Access*, 8:35929–35949, 2020.

[14] Y. Chen, M. Rohrbach, Z. Yan, Y. Shuicheng, J. Feng, and Y. Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019.

[15] T. K. Dey, K. Li, C. Luo, P. Ranjan, I. Safa, and Y. Wang. Persistent heat signature for pose-oblivious matching of incomplete models. In *Computer Graphics Forum*, volume 29, pages 1545–1554. Wiley Online Library, 2010.

[16] S. Ebli, M. Defferrard, and G. Spreemann. Simplicial neural networks. *NeurIPS workshop on Topological Data Analysis and Beyond*, 2020.

[17] M. Fey and J. E. Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[18] C. Gallicchio and A. Micheli. Graph echo state networks. In *The 2010 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2010.

[19] M. Gori, G. Monfardini, and F. Scarselli. A new model for learning in graph domains. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 2, pages 729–734. IEEE, 2005.

[20] A. Grover and J. Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864, 2016.

[21] C. Hacker. k-simplex2vec: a simplicial extension of node2vec. *NeurIPS workshop on Topological Data Analysis and Beyond*, 2020.

[22] M. Hajij, B. Wang, C. Scheidegger, and P. Rosen. Visual detection of structural changes in time-varying graphs using persistent homology. In *2018 IEEE Pacific Visualization Symposium (PacificVis)*, pages 125–134. IEEE, 2018.

[23] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 35–44, 1993.

[24] W. L. Hamilton, R. Ying, and J. Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017.

[25] A. Hatcher. *Algebraic topology*. , 2005.

[26] Z. Jia, S. Lin, M. Gao, M. Zaharia, and A. Aiken. Improving the accuracy, scalability, and performance of graph neural networks with roc. *Proceedings of Machine Learning and Systems (MLSys)*, pages 187–198, 2020.

[27] J. Jung, W. Jin, H.-m. Park, and U. Kang. Accurate relational reasoning in edge-labeled graphs by multi-labeled random walk with restart. *WORLD WIDE WEB-INTERNET AND WEB INFORMATION SYSTEMS*, 2020.

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

[29] I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein. Intrinsic shape context descriptors for deformable shapes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 159–166. IEEE, 2012.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[32] M. Marinov and L. Kobbelt. A robust two-step procedure for quad-dominant remeshing. In *Computer Graphics Forum*, volume 25, pages 537–546. Wiley Online Library, 2006.

[33] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015.

[34] F. Monti, D. Boscaini, J. Masci, E. Rodola, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5115–5124, 2017.

[35] B. Perozzi, R. Al-Rfou, and S. Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710, 2014.

[36] L. Piegl and W. Tiller. *The NURBS book*. Springer Science & Business Media, 1996.

[37] A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Advances in neural information processing systems*, pages 4967–4976, 2017.

[38] F. Scarselli, M. Gori, A. C. Tsoi, M. Hagenbuchner, and G. Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[39] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie. Random walks on simplicial complexes and the normalized hodge 1-laplacian. *SIAM Review*, 62(2):353–391, 2020.

[40] M. Schlichtkrull, T. N. Kipf, P. Bloem, R. Van Den Berg, I. Titov, and M. Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018.

[41] D. I. Shuman, B. Ricaud, and P. Vandergheynst. Vertex-frequency analysis on graphs. *Applied and Computational Harmonic Analysis*, 40(2):260–291, 2016.

[42] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[43] J. Stam. Flows on surfaces of arbitrary topology. *ACM Transactions On Graphics (TOG)*, 22(3):724–731, 2003.

[44] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[45] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[46] P. Veličković, G. Cucurull, A. Casanova, A. Romero, P. Lio, and Y. Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

[47] Q. Wang, Z. Mao, B. Wang, and L. Guo. Knowledge graph embedding: A survey of approaches and applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743, 2017.

[48] 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*, 2020.

[49] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[50] K. Xu, W. Hu, J. Leskovec, and S. Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

[51] Z. Yasseen, A. Nasri, W. Boukaram, P. Volino, and N. Magnenat-Thalmann. Sketch-based garment design with quad meshes. *Computer-Aided Design*, 45(2):562–567, 2013.

[52] S.-X. Zhang, X. Zhu, J.-B. Hou, C. Liu, C. Yang, H. Wang, and X.-C. Yin. Deep relational reasoning graph network for arbitrary shape text detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9699–9708, 2020.

[53] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

4. APPENDIX

4.1. Why Cell Complex Nets?

This work focuses on the construction of the necessary tools that perform neural networks-type computations over domains that have geometric and combinatorial characteristics. Our ultimate goal is to harness the power of deep learning in solving problems that arise when studying these domains or working on problems that are naturally modeled by such domains.

With the above goal in mind, one may wonder why we chose cell complexes for representation over other less general complexes available in algebraic topology? Cell complexes neural nets (CXNs) are generalization of graph neural networks (GNNs), and the gap of generalization between graphs and cell complexes contain many other complexes (e.g., simplicial complexes, polyhedral complexes, and Δ-complexes) that are less general than cell complexes. We discuss below the reasons of why CXNs form a better and more expressive generalization than, say, graph and simplicial complex neural networks [7].

- **Hierarchical relational reasoning representation:** Graphs are natural objects for modeling relations between entities. In this context and towards building intelligent behaviour, GNNs have been extensively explored to build relational reasoning between various objects [37, 52, 14, 40] and in knowledge graphs [3, 47]. However, we believe building intelligent behaviour goes beyond the ability to create relational reasoning between entities. Abstraction and analogy that humans are capable of require building relations among the relations in a hierarchical manner. The importance of this perspective is evident in the unprecedented success of deep learning models where complicated concepts are built from simpler ones in a hierarchical fashion.

  Therefore, utilizing GNNs for relational reasoning is "shallow" because the edges can be only used to model relationships between entities. Even labeled multi-graph [40, 27] are insufficient to model hierarchical relational reasoning because the relationships modeled by the multi edges or multi nodes remain between the raw entities and no deeper relations can be made. CXNs have the ability to model hierarchical relational reasoning. Precisely, the 1-skeleton of a cell complex can represent the shallow relation between objects while the higher cells can be used to build more abstract relations between the relations in a hierarchical manner. Note that other objects, such as simplicial complexes, have uniform structure, in the sense that k-faces have fixed number of edges, which is not natural to model hierarchical complex relational reasoning. Within this context, knowledge graphs can be generalized to *knowledge cell complexes* where entities and relations in knowledge graphs are replaced with hierarchical representation of abstract entities and relations between them.

- **Nature of data and application:** Contrary to simplicial complex neural networks, our definition is a unifying and combinatorial framework that accommodates for almost all data forms that are significant in practice such as polygonal 2d and 3d meshes. In addition to the nature of data, the application at hand determines, in many cases, the type of complexes one needs to work with. For instance, in several CAD [36, 32] and simulation applications [51], quad meshes are preferred over simplicial complexes [7]. Quad meshes are also preferred when solving PDE on surface and are best suited for defining Catmull-Clark subdivision surfaces [43, 23].

- **Trainability of neural networks over geometric domains and resource efficiency:** As is well-known, GNNs are challenging to train for large graphs [26], and simplicial complexes require massive amount of memory and computational power even without doing deep learning computations on them. Hence, it is essential to consider the complexity of the geometric object representation when designing a neural network over these domains. In this context, building geometric objects with cell complexes requires significantly less number of cells than building the same objects with simplicial complexes or other complexes.

[7] We are aware of few related works about unoriented simplicial neural networks [12, 16] that were published at the same time our work got published. We note, however, that our approach is applicable to a more general set of complexes and handles both oriented and non-oriented cases. See Section 4.3 for the oriented case.

4.2. Graph Neural Networks

Given a graph $G = (V,E)$, a graph neural network on G with depth $L > 0$ updates a feature representation for every node in the graph L times. Initially, every node i is given a feature vector $h_i^{(0)}$. On the k stage of the computation, each node i in the graph collects messages from its neighbors j, represented by their feature vectors $h_j^{(k-1)}$, and aggregates them together to form a new feature representation $h_i^{(k)}$ for the node i. A graph neural network requires the following input data:

1. A graph $G = (V,E)$.
2. For each node $i \in V$ we have an initial vector $h_i^{(0)} \in \mathbb{R}^{l_0}$.

Given the above data, the feedforward neural algorithm on G executes L message passing schemes defined recursively for $0 \leq k \leq L$ by:

$$h_i^{(k)} := \alpha^k\left(h_i^{(k-1)}, E_{j\in\mathcal{N}(i)}\left(\phi^k(h_i^{(k-1)},h_j^{(k-1)},e_{i,j})\right)\right) \in \mathbb{R}^{l_k}, \tag{5}$$

where $e_{ij} \in \mathbb{R}^D$ is an edge feature from the node j to the node i, E is a permutation invariant differentiable function, and α^k,ϕ^k are trainable differentiable functions. Note that at each stage k, all messages share the same differentiable functions ϕ^k and α^k.

Consider the graph given in Figure 3. Let's say we want to build a graph neural network on this graph with depth 2. To illustrate the computation, we pick a vertex v_1 in the graph. In the first stage, the surrounding neighbors of v_1, namely $\{v_2,v_3,v_4\}$, pass their messages to v_1. The information obtained from these messages are aggregated and combined together via trainable differentiable functions α^1 and ϕ^1. In the second stage, all neighbors of v_1 collect the messages information from their respective neighbors in a similar fashion as illustrated in Figure 3.

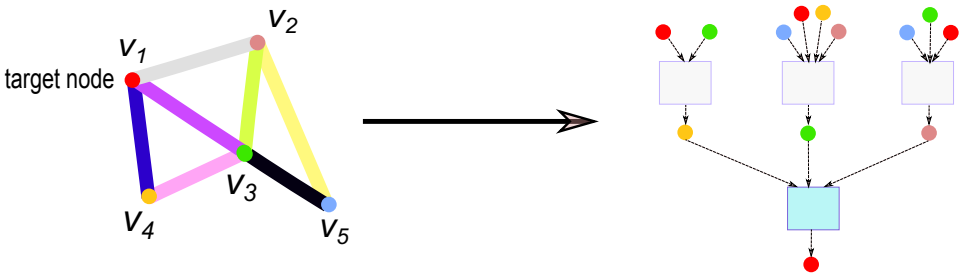

Figure 3: An example of graph neural net with depth 2. The computation are only illustrated on the red node. In this figure, we abuse the notation and do not distinguish between a node i and its associated vector $h_i^{(k)}$. The blue box represents the differentiable functions α^1 and ϕ^1 while the white box represents the functions α^2 and ϕ^2.

4.3. The adjacency relation when complex X is oriented

In this section, we discuss the adjacency and coadjacency relations in a cell complex X when X is oriented. Recall that each cell a in X has two possible orientations. An *oriented cell complex* is a cell complex in which every cell has a chosen orientation. When X is regular, then the entries of ∂_k are in $\{0, \pm 1\}$.

The definitions of $facets(c^n)$ and $cofacets(c^n)$ are more complicated for the oriented case as compared to the non-oriented case. When X is oriented, we store along with each cell in $facets(c^n)$ and $cofacets(c^n)$ the orientation induced by c_n with respect to the maps ∂_n and ∂_n^*, respectively.

In this case, we use $facets^+(c^n) \subset facets(c^n)$, $facet^-(c^n) \subset facets(c^n)$ to the subsets of $facets(c^n)$ to denote the cells that are positively oriented and negatively oriented with respect to c^n, respectively. The set $cofacets^+(c^n)$ and the set $cofacets^-(c^n)$ are defined analogously. Observe that $facets(c^n) = facets^+(c^n) \cup facets^-(c^n)$ and $facets^+(c^n) \cap facets^-(c^n) = \varnothing$. Similarly, $cofacets(c^n) = cofacets^+(c^n) \cup cofacets^-(c^n)$ and $cofacets^+(c^n) \cap cofacets^-(c^n) = \varnothing$.

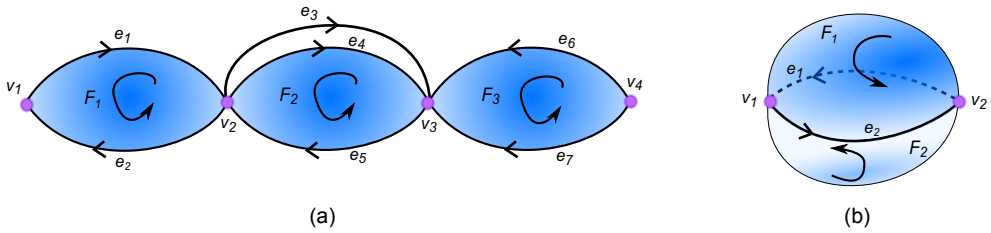

Figure 4: Examples of computing the adjacency and co-adjacency neighbors of cells in cell complexes.

Consider the cell complexes given in Figure 4, we compute few examples of the sets we define above to illustrate the concept. For Figure 4 (a), we have $cofacets(v_1) = \{-e_1, e_2\}$, $cofacets(v_2) = \{e_1, -e_2, -e_3, -e_4, e_5\}$, $cofacets(v_3) = \{e_3, e_4, -e_5, e_6, e_7\}$ and $cofacets(v_4) = \{-e_6, -e_7\}$. Moreover, $facets(F_1) = \{e_1, -e_2\}$. Finally, $cofacets(e_6) = \{F_3\}$ and $cofacets(e_7) = \{-F_3\}$. On the other hand, for Figure 4 (b) $cofacets(e_1) = \{F_1, -F_2\}$ and $cofacets(e_2) = \{F_1, -F_2\}$.

If X is an oriented complex, then a cell b^n is said to be *adjacent* to an n-cell a^n with respect to an $(n+1)$-cell c^{n+1} when $a^n \in facets^+(c^{n+1})$ and $b^n \in facets^-(c^{n+1})$. Similarly, an n-cell b^n is said to be *coadjacent* to a^n with respect to an $(n-1)$-cell c^{n-1} if $a^n \in cofacets^+(c^{n-1})$ and $b^n \in cofacets^-(c^{n-1})$. The set of all adjacent cells of a cell a in X is denoted by $\mathcal{N}_{adj}(a)$. Similarly, the set of all coadjacent cells of a cell a in X is denoted by $\mathcal{N}_{co}(a)$.

In Figure 4 (a), we have $\mathcal{N}_{adj}(v_2) = \{v_1, v_3\}$ $\mathcal{N}_{adj}(v_4) = \varnothing$. For Figure 4 (b), we have $\mathcal{N}_{co}(F_1) = \{F_2\}$. This is because $F_1 \in cofacets^+(e_1)$ and $F_2 \in cofacets^-(e_1)$. On the other hand, $\mathcal{N}_{co}(F_2) = \varnothing$. Note that in this example $\mathcal{N}_{adj}(e_1) = \mathcal{N}_{adj}(e_2) = \varnothing$

If a^n and b^n are n-cells in X, then we define the set $\mathcal{CO}[a^n, b^n]$ to be the intersection $cofacets(a^n) \cap cofacets^+(b^n)$. The set $\mathcal{CO}[a^n, b^n]$ describes the set of all incident $(n+1)$-cells of b^n that have a^n as an adjacent cell. Note that in general $\mathcal{CO}[a^n, b^n] \neq \mathcal{CO}[b^n, a^n]$. Similarly, the set $\mathcal{C}[a^n, b^n]$ is defined to be the intersection of $facets(a^n) \cap facets^+(b^n)$.

In Figure 4 (a), we have $\mathcal{CO}[v_2, v_3] = \{e_3, e_4\}$ whereas $\mathcal{CO}[v_3, v_2] = \{e_4\}$. On the other hand, we have $\mathcal{CO}[e_1, e_2] = \mathcal{CO}[e_2, e_1] = \varnothing$ in Figure 4 (b).

4.4. General Message Passing Scheme

The inter-cellular message passing scheme given in Section 2 updates the vectors on the flows from a given 0-cell and gathers the information from surrounding cells in a radial fashion defined by the adjacency matrices of the cell complex. Although this message passing scheme is natural from the perspective of generalizing GNNs passing schemes, it forms a single method out of many other natural methods that can be defined in the context of CXNs.

4.4.1. Co-adjacency Message Passing Scheme

The message passing scheme given in Section 2 does not update the vectors associated with the final $n-$ cells on the complex. In certain applications, it might be desirable to make the flow of information go from the lower cells to the higher cells. An example of such an application is mesh segmentation where it is desirable to update the information associated with a face on the mesh after gathering local surrounding information. The scheme given in Section 2 can be easily adjusted for this purpose. To this end, we utilize the data on the cells complex as before while re-defining the message passing schemes as follows:

$$h_{c^n}^{(k)} := \alpha_{n-1}^{(k)}\left(h_{c^n}^{(k-1)}, E_{a^n \in \mathcal{N}_{co}(c^n)}\left(\phi_{n-1}^{(k)}(h_{c^n}^{(k-1)}, h_{a^n}^{(k-1)}, F_{e^{n-1} \in \mathcal{C}[a^n, c^n]}(h_{e^{n-1}}^{(k-1)}))\right)\right) \in \mathbb{R}^{l_n^k}, \tag{6}$$

$$\vdots$$

$$h_{c^1}^{(k)} := \alpha_1^{(k)}\left(h_{c^1}^{(k-1)}, E_{a^1 \in \mathcal{N}_{co}(c^1)}\left(\phi_1^{(k)}(h_{c^1}^{(k-1)}, h_{a^1}^{(k-1)}, F_{e^0 \in \mathcal{C}[a^1, c^1]}(h_{e^0}^{(0)}))\right)\right) \in \mathbb{R}^{l_1^k} \tag{7}$$

Note that the initial vector associated with zero cells in X is never updated in this case. An example of these computations is illustrated in Figure 5.

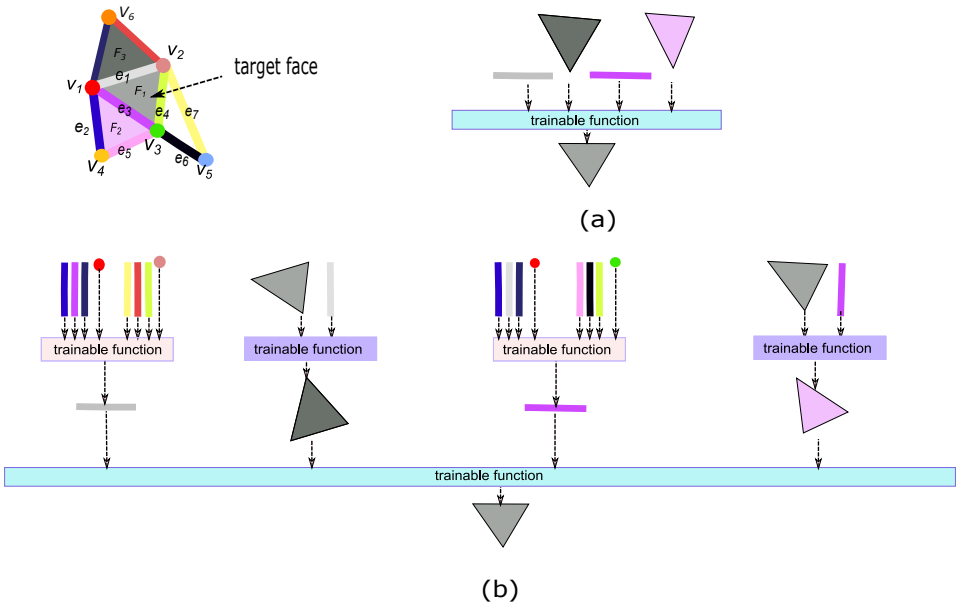

Figure 5: CXN with 2 layers. This example demonstrates a simplicial complex neural network for clarity. The computation is only demonstrated with respect to the light grey face. The information flow goes from higher cells to lower incident cells.

4.4.2. Homology and Cohomology-Based Passing Schemes

This subsection briefly outlines a message passing scheme that is consistent with the boundary and coboundary maps in the context of homology and cohomology of a cell complex.

Let c_m be a cell in a, possible oriented, cell complex X and let $h^{(k)}_{c_m}$ be its embedding. Let $Bd(x)$ be set of cells y of dimension $k-1$ such that $y \in \partial(x)$ and x and y have compatible orientations. Similarly, let $CoBd(x)$ denotes all cells of $y \in X$ with $h \in \partial(y)$ and both x and y have compatible orientations. Denote by $\mathcal{I}(x)$ to the union $Bd(x) \cup CoBd(x)$, we may define the passing scheme as follows:

$$h^{(k)}_{c^m} := \alpha^{(k)}_m \left(h^{(k-1)}_{c^m}, E_{a \in \mathcal{I}(x)} \left(\phi^{(k)}_{m,d(a)} (h^{(k-1)}_{c^m}, h^{(k-1)}_a) \right) \right) \in \mathbb{R}^{l^k_m} \tag{8}$$

Notice that the trainable function $\phi^{(k)}_{m,d(a)}$ needs to accommodate for the fact that the dimension of the vector associated with a, namely $h^{(k-1)}_a$, may vary for different $a \in \mathcal{I}(x)$.

4.5. Potential Applications

The proposed CXNs has several potential applications. For example:

1. **Studying the type of underlying spaces.** The *topological type* of the underlying space is a central question in topology. Specifically, given two spaces A and B, are they equivalent up to a given topological equivalence? In practice, this can be translated to a similarity question between two structures. Indeed, TDA has been extensively utilized towards this purpose [15, 22]. On the other hand, deep learning allows studying the structure of the underlying space by building complex relationship between various elements in this space. Cell complexes form a general class of topological spaces that encompasses graphs, simplicial complexes, and polyhedral complexes. Hence, CXNs provides a potential tool to study the structure similarity between discrete domains such as 3D shapes and discrete manifolds [8].
2. **Learning cell complex representation.** The cell complex autoencoder framework introduced in 3 extends the applications of graphs autoencoder to a much wider set of possibilities.

[8] Within this context, GNNs with their current neighborhood aggregation scheme have been shown to not being able to solve the graph isomophism problem [50].

In particular, cell complexes are natural objects for *language embedding* as they can be used to build complex relationships of arbitrary length. Specifically, we can build a cell complex out of a corpus of text: words are vertices, they share an 1-cells if they are adjacent in the corpus, within a sentence, sentences form the boundaries of 3-cells, paragraphs form the boundaries of 4-cells, chapters form the boundaries of 4-cells, etc. Notice that unlike less general complexes (e.g. simplical complexes), a k-cell in a cell complex may have an arbitrary number of $(k-1)$ incident cells.