Attending to Topological Spaces: The Cellular Transformer

Rubén Ballester
Departament de Matemàtiques i Informàtica
Universitat de Barcelona
08007 Barcelona, Spain
ruben.ballester@ub.edu

Pablo Hernández-García
Departamento de Matemáticas
Universidad de Salamanca
pablohg.eka@usal.es

Mathilde Papillon
Department of Electrical Engineering
University of California Santa Barbara
papillon@ucsb.edu

Claudio Battiloro
Department of Biostatistics
Harvard University
cbattiloro@hsph.harvard.edu

Nina Miolane
Department of Electrical Engineering
University of California Santa Barbara
ninamiolane@ucsb.edu

Tolga Birdal
Department of Computer Science
Imperial College London
t.birdal@imperial.ac.uk

Carles Casacuberta
Departament de Matemàtiques i Informàtica
Universitat de Barcelona
08007 Barcelona, Spain
carles.casacuberta@ub.edu

Sergio Escalera
Departament de Matemàtiques i Informàtica
Universitat de Barcelona
08007 Barcelona, Spain
sergio.escalera.guerrero@gmail.com

Mustafa Hajij
Departament of Data Science
University San Francisco
mhajij@usfca.edu

Abstract

Topological Deep Learning seeks to enhance the predictive performance of neural network models by harnessing topological structures in input data. Topological neural networks operate on spaces such as cell complexes and hypergraphs, that can be seen as generalizations of graphs. In this work, we introduce the Cellular Transformer (CT), a novel architecture that generalizes graph-based transformers to cell complexes. First, we propose a new formulation of the usual self- and cross-attention mechanisms, tailored to leverage incidence relations in cell complexes, e.g., edge-face and node-edge relations. Additionally, we propose a set of topological positional encodings specifically designed for cell complexes. By transforming three graph datasets into cell complex datasets, our experiments reveal that CT not only achieves state-of-the-art performance, but it does so without the need for more complex enhancements such as virtual nodes, in-domain structural encodings, or graph rewiring.

1 Introduction

Topological Deep Learning (TDL) [49, 80] is a fast growing field that leverages generalizations of graphs, such as simplicial complexes, cell complexes, and hypergraphs (collectively known

as *topological domains*) to extract comprehensive global information from data [11, 18]. Neural networks that are designed to process and learn from data supported on these topological domains form the core of TDL [80]. By transcending traditional graph-based representations, which are limited to binary relations in graph neural networks (GNNs), TDL exploits novel information contained in higher-order relationships, i.e., interactions involving multiple entities simultaneously. This capability provides new, unique opportunities to address innovative applications across diverse disciplines, including social sciences [110], transportation [62], physics [10], epidemiology [28], as well as scientific visualization and discovery.

In parallel to TDL, the transformer architecture [94] has brought about a paradigm shift in learning on data with various modalities. Using multi-headed attention, transformers can capture long-range dependencies and hierarchical patterns in data like text or video [30, 50]. Particularly, graph transformers [88, 100] form a subset of transformers specifically designed to work with graph-based data. These models suffer from similar expressivity limitations as GNNs compared to Topological Neural Networks (TNNs), as they only represent pairwise relationships within the data.

Contributions. In this work, we propose to bridge the gap between TDL and transformers. We introduce the *Cellular Transformer* (CT) to simultaneously harness the power of the expressive cell complex representation and the attention mechanism. By augmenting the transformer with topological awareness through cellular attention, CT is inherently capable of exploiting complex patterns in data mapped to high order representations, showing competitive or improved performance compared to graph and simplicial transformers and message passing architectures.

Our specific contibutions are summarized as follows.

1. We propose the CT framework, which generalizes the graph-based transformer to process higher-order relations within cell complexes.
2. We introduce cell complex positional encodings and formulate self-attention and cross-attention in topological terms. We demonstrate how to utilize these computational primitives to process data supported on cell complexes in a transformer layer.
3. We benchmark the CT on three classical benchmark datasets, outperforming or achieving results comparable to the state-of-the-art without the need for complex enhancements of the architecture such as virtual nodes, involved in-domain structural or positional encodings or rewiring methods.

2 Related Work

Transformer models have seen significant advancements in various domains, including natural language processing [64, 83, 94], computer vision [1, 30, 50], or graph learning.

Graph transformers. Graph transformers are a special subset of transformers designed to learn from data supported on graphs. This evolving field encompasses three distinct strategies to harnessing the power of transformers in graph contexts. The first approach involves integrating GNNs directly into transformer architectures, either by stacking [88, 100], interweaving [71], or running in parallel [105]. A second method focuses on encoding the graph structure into positional embeddings, which are then added to the input of the transformer model for spatial awareness. These embeddings can be computed in a myriad of ways, such as from Laplacian eigenvectors [31] or SVD vectors of the adjacency matrix [57]. Finally, the third approach hard codes adjacency information into the self-attention. In [31, 77], only adjacent nodes are allowed to attend to each other, as all other attentional weights are zeroed. [75] similarly manipulates self-attention via the kernel matrix instead of the adjacency matrix. We refer the reader to [76] for more information. Our work adopts a combination of the second and third methods in order to adapt the transformer to data supported on cell complexes.

Higher-order transformers. Transformer models which go beyond pairwise relations represent a natural progression from graph-based transformers. The most prominent category of such higher order transformers operates on hypergraphs. In many instances, they have also adopted the self-attention mechanism [56, 66, 98, 106]. Beyond graphs and hypergraphs, transformers operating on topological domains are scarce. To our knowledge, only two higher-order transformers operating on simplicial complexes have been proposed [24, 109]. The first approach, however, does not consider higher order features directly, but rather leverages higher order relations to improve features on nodes. The second approach, although being a fairly general object to define higher-order structures, focuses primarily

on graph learning. It proposes two architectures: one operating on tuples of nodes (i.e., cliques) within the graph, which may not naturally appear in the clique complex of the graph, and another applying a general attention mechanism to all simplices in a lifted graph simultaneously, disregarding the distinct nature of data across dimensions (e.g., properties of atoms in nodes vs. properties of bonds in edges). The latter was only tested on nodes and edges, excluding higher-order elements. A limitation of simplicial approaches is their representative power, as triangles and tetrahedra are scarce in natural data domains.

Non-transformer topological neural networks. Besides transformers, recent years have witnessed a growing interest in higher-order networks [11, 18]. In signal processing and deep learning, various approaches, such as Hodge-theoretic methods, message-passing schemes, and skip connections, have been developed for TNNs. The use of Hodge Laplacians for data analysis has been investigated in [63, 70] and extended to a signal processing context, for example, in [87, 90, 91] for simplicial and cell complexes. Convolutional operators and message-passing algorithms for TNNs have been developed. For hypergraphs, a convolutional operator has been proposed in [2, 36, 61] and has been further investigated in [3, 39, 61]. Message passing on simplicial and cell complexes are proposed in [46, 48]. The expressive power of these networks is studied in [19]. Moreover, message passing on network sheaves can be found in [5, 6, 9, 20, 51, 52]. A model able to infer a latent regular cell complex from data has been introduced in [8]. Recently, message passing-free architectures have been introduced for simplicial complexes [44, 72, 84].

Most attention-based models are designed primarily for graphs, with some recent exceptions that have been introduced in topological domains [3, 40, 42, 65]. Attention on cell complexes was introduced in [41] exploiting higher-order topological information through feature lifting and attention mechanisms over lower and upper neighborhoods, and a generalized attention mechanism on combinatorial complexes was introduced in [49]. However, neither [41] nor [49] consider query-key-value attention and positional/structural encodings. The work in [41] works at the edge level and does not leverage any interplay among cells of different ranks. Furthermore, the work in [49] does not account for dense attention, and, being based on the general notion of combinatorial complex, it is not able to readily encode and leverage specific topological features peculiar to cell complexes.

3 Cell Complexes

Cell complexes encompass various kinds of topological spaces used in network science, including graphs, simplicial complexes, and cubical complexes. A precise definition can be found in [53], and further information is provided in the Appendix.

We constrain ourselves with 2-dimensional regular cell complexes for simplicity, although our constructions and discussion carry over to higher-dimensional cellular complexes similarly. Thus, in this work, as in [87], a *cell complex* is a triplet $\mathcal{X} = (\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2)$ of finite ordered sets, where elements $v \in \mathcal{X}_0$ are called *nodes*, *vertices*, or 0-*cells*, elements $e \in \mathcal{X}_1$ are called *edges* or 1-*cells*, and elements $\sigma \in \mathcal{X}_2$ are called *faces* or 2-*cells*. Additionally, *incidence relations* represent each edge as an ordered pair of vertices $e = [v_1, v_2]$ incident to e, and each face as an ordered sequence of edges $\sigma = [e_1, \dots, e_{m(\sigma)}]$ that form a closed path without self-intersections and constitute the edges incident to σ. We assume that $m(\sigma) \geq 3$ to ensure that the pair $(\mathcal{X}_0, \mathcal{X}_1)$ is a (loopless, simple, directed) graph. The subscript k of each set $\mathcal{X}_k$ is called its *rank*.

The incidence relations endow each edge and each face with an orientation. For an edge $e = [v_1, v_2]$, the oppositely oriented edge is denoted by $-e = [v_2, v_1]$. A collection of edges $e_1, \dots, e_m$ form a closed path if there is a set of distinct vertices $v_1, \dots, v_m$ such that $\pm e_i = [v_i, v_{i+1}]$ for $1 \leq i \leq m$ and $v_{m+1} = v_1$. Incidence relations between cells of consecutive ranks are encoded into *incidence matrices*. Thus, the first incidence matrix $\mathbf{B}_1$ has (i, j) entry equal to -1 if the j-th edge e_j starts at the i-th vertex v_i, 1 if e_j ends at v_i, and 0 otherwise. The entries of the second incidence matrix $\mathbf{B}_2$ are the incidence numbers between faces and edges, where the *incidence number* of a face σ with an edge e is the sign of $\pm e$ if it belongs to a closed path of edges incident to σ, and 0 otherwise. These two matrices satisfy $\mathbf{B}_1\mathbf{B}_2 = 0$, as shown in [43, 53]. The *non-signed incidence matrices* $\mathbf{I}_k$ for $k = 1, 2$, are obtained by replacing incidence numbers in $\mathbf{B}_k$ by their absolute values.

We refer to *neighborhood matrices*, including signed and non-signed incidence matrices, upper and lower adjacency matrices, and Hodge Laplacians, as defined in detail in the Appendix.

3.1 Data on cell complexes: cochain spaces

Cochain spaces are used to process data supported over a cell complex $\mathcal{X} = (\mathcal{X}_0, \mathcal{X}_1, \mathcal{X}_2)$. For $k = 0, 1, 2$, we denote by $\mathcal{C}^k(\mathcal{X}, \mathbb{R}^d)$ the $\mathbb{R}$-vector space of functions $\mathcal{X}_k \to \mathbb{R}^d$, where $d \geq 1$. Here d is called *data dimension* and elements of $\mathcal{C}^k(\mathcal{X}, \mathbb{R}^d)$ are called *k-cochains* or *k-signals* on $\mathcal{X}$. For short, we write $\mathcal{C}^k(\mathcal{X})$ instead of $\mathcal{C}^k(\mathcal{X}, \mathbb{R})$ when $d = 1$; see fig. 1 for an example.

An *annotated cell complex* is a cell complex $\mathcal{X}$ together with a k-cochain $\mathbf{X}_k$ of dimension d_k for each rank k. We view $\mathbf{X}_k$ as a matrix in $\mathcal{M}(|\mathcal{X}_k|, d_k)$, that is, with $|\mathcal{X}_k|$ rows and d_k columns, whose i-th row is the image of the i-th element of $\mathcal{X}_k$. In this work, all datasets consist of annotated cell complexes sharing the same dimensions d_0, d_1 and d_2.

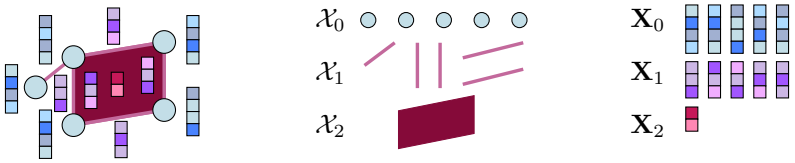


Figure 1: Illustration of an annotated cell complex. Left: An annotated cell complex $\mathcal{X}$ consisting of five vertices, five edges, and one 2-cell. Center: $\mathcal{X}_k$ is the collection of k-cells of $\mathcal{X}$ for $k = 0, 1, 2$. Right: Rows depict values of a cochain $\mathbf{X}_k$ for each k, of dimensions $d_0 = 4$, $d_1 = 3$ and $d_2 = 2$.

4 The Cellular Transformer

In this section, we present a general transformer architecture for cell complexes. First, we discuss different approaches to perform attention on cells and define the cellular transformer layer. Then, we propose different positional encoding methods for the cellular transformer, that identify cells by their relative position in the cell complex or by their *centrality* according to random walks, as in [33].

4.1 Overview

A *cellular transformer* is a neural network which, given an annotated cell complex $\mathcal{X}$, induces a composition of functions $\mathrm{CT} = \mathrm{R} \circ \mathrm{CT}_L \circ \cdots \circ \mathrm{CT}_1 \circ \mathrm{P}$, named layers, where P is a preprocessing layer, as described in section 4.4, R is a readout layer that converts cochains on top of cells into an output prediction value, and CT_l, for $l = 1, \ldots, L$, are cellular transformer layers defined as functions of the form

$$\mathrm{CT}_l \colon \mathcal{C}^0(\mathcal{X}, \mathbb{R}^{d_0^h}) \times \cdots \times \mathcal{C}^n(\mathcal{X}, \mathbb{R}^{d_n^h}) \longrightarrow \mathcal{C}^0(\mathcal{X}, \mathbb{R}^{d_0^h}) \times \cdots \times \mathcal{C}^n(\mathcal{X}, \mathbb{R}^{d_n^h}), \tag{1}$$

where $n = \dim \mathcal{X}$ and h indicates the dimension of hidden layers. In our experiments, we set $d_0^h = \cdots = d_n^h$, whose value depends on the dataset and is specified in the Appendix.

Equation (1) does not provide an explicitly parametrization of the cellular transformer layer as it only describes the function (co)domains. A parametrization of the cellular transformer layers can be given using tensor diagrams together with the cellular attention formulae, described in section 4.2. Transformer and preprocessing layers take as input an annotated cell complex and output the same cell complex with different cochains. We denote input k-cochains on the CT_l layer as $\mathbf{X}_{k,l}$.

4.2 The cellular attention layer and cellular transformer architecture

We propose two cellular attention mechanisms for transformer layers. The first mechanism generalizes self- and cross-attention and depends on the dimensions of the cells. We call it *pairwise cellular attention*. The second mechanism performs the attention over all cells ignoring their dimension. We call it *general cellular attention*. We discuss in section 5.1 the data regimes in which each attention mechanism performs better than the other.

4.2.1 Pairwise cellular attention

Our first mechanism performs pairwise attention between cells of arbitrary ranks according to a tensor diagram (see section 4.3), and then aggregates the outputs received for the same rank. Given source

and target ranks $0 \leq k_s, k_t \leq \dim \mathcal{X}$ and cochains $\mathbf{X}_{k_t}, \mathbf{X}_{k_s}$, the single-head attention from k_s to k_t is a map $\mathcal{C}^{k_s}(\mathcal{X}, \mathbb{R}^{d_s^h}) \times \mathcal{C}^{k_t}(\mathcal{X}, \mathbb{R}^{d_t^h}) \to \mathcal{C}^{k_t}(\mathcal{X}, \mathbb{R}^{d_t^h})$ defined as

$$\mathcal{A}^{\bullet}_{k_s \to k_t}(\mathbf{X}_{k_t}, \mathbf{X}_{k_s}) = \text{softmax}(\mathbf{X}_{k_t}\mathbf{Q}_{k_s \to k_t}(\mathbf{X}_{k_s}\mathbf{K}_{k_s \to k_t})^T \star \phi(\mathbf{N}_{k_s \to k_t}))\mathbf{X}_{k_s}\mathbf{V}_{k_s \to k_t}, \quad (2)$$

where $\mathbf{Q}_{k_s \to k_t} \in \mathcal{M}(d_t^h, p)$, $\mathbf{K}_{k_s \to k_t} \in \mathcal{M}(d_s^h, p)$, and $\mathbf{V}_{k_s \to k_t} \in \mathcal{M}(d_s^h, d_t^h)$ are learnable query, key, and value real matrices with p a fixed hyperparameter shared by all transformer layers. The symbol $\bullet \in \{d, s, c\}$ indicates whether we are performing dense, sparse, or a mixed type of attention, performing dense attention for cells of the same rank and sparse attention otherwise, respectively. The symbol $\star$ is a sum or a Hadamard product for dense or sparse attention, respectively. $\mathbf{N}$ is a neighborhood matrix, and ϕ is a function, possibly with learnable parameters. For our experiments, we set $\phi(\mathbf{N}) = \theta\mathbf{N}$ where θ is a learnable parameter for dense attention and the identity matrix for sparse attention. Attention formulae performs query, key, and value projections without bias for simplicity. A bias term can be added to the projections, as in most transformer implementations. Multi-head attention can also be performed by (1) splitting the cochains $\mathbf{X}_{k_s}$ and $\mathbf{X}_{k_t}$ into multiple cochains $\mathbf{X}^1_{k_s}, \ldots, \mathbf{X}^m_{k_s}$ and $\mathbf{X}^1_{k_t}, \ldots, \mathbf{X}^m_{k_t}$ of smaller dimension; (2) performing single-head attention for each pair of cochains $\mathbf{X}^i_{k_s}, \mathbf{X}^i_{k_t}$; (3) concatenating the outputs of the single-head attention for the different pairs into a full cochain of dimension d_t^h.

For a specific rank k_t, CT layers can produce multiple attention outputs from different rank sources k_s. In the CT layer, we adopt the standard prenorm design [101], where for each rank k_t, the outputs from the various rank sources k_s are summed in the residual connection. The specific algorithm for the CT layer is detailed in the Appendix. In our experiments, we set $\mathbf{N}_{k \to k}$ to be the upper adjacency matrix for $k = 0$ and the lower adjacency matrix if $k > 0$, and $\mathbf{N}_{k_s \to k_t}$ to be the non-signed incidence matrix between dimensions k_s and k_t if $k_s > k_t$, and the transpose matrix otherwise.

4.2.2 General cellular attention

The second mechanism performs attention with all the cells at the same time, disregarding their rank, as proposed in [109]. In the general attention mechanism, cells share the same key and query matrices, but have different value matrices for each rank. The single-head attention formula is

$$\mathcal{A}^{\bullet}_{g}(\mathbf{X}) = \text{softmax}\left(\mathbf{X}Q(\mathbf{X}K)^T \star \phi(\mathbf{N})\right)\left(\text{Concat}\left[(\mathbf{X}_0V^0)^T, \ldots, (\mathbf{X}_{\dim \mathcal{X}}V^{\dim \mathcal{X}})^T\right]\right)^T,$$

using the same notations as in the previous subsection. In this case, the prenorm transformer layer is performed as usual. The algorithm corresponding to the general attention CT layer is detailed in the Appendix. As in section 4.2.1, multi-head attention can be performed by splitting the original cochains into smaller cochains, applying the general single-head attention to pairs of the smaller cochains, and concatenating again into a single, big cochain. In our experiments, we let $\mathbf{N}$ be the following combination of the previous $\mathbf{N}_{k_s \to k_t}$ matrices, where $n = \dim \mathcal{X}$:

$$\mathbf{N} = \begin{bmatrix} \mathbf{N}_{0\to 0} & \mathbf{N}_{1\to 0} & \ldots & 0 \\ \mathbf{N}^T_{1\to 0} & \mathbf{N}_{1\to 1} & \ldots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \ldots & \mathbf{N}_{n\to n} \end{bmatrix}. \quad (3)$$

4.3 Tensor diagrams for cellular transformers

Cellular transformers involve interactions between cochains of different ranks. Tensor diagrams [49] provide a graphical abstraction that illustrates the flow of information of one CT layer. A tensor diagram portrays a CT Layer through the use of a directed graph. Nodes of a tensor diagram represent cochain spaces for different ranks $0 \leq k \leq n$, where n is the maximum allowed rank of cell complexes processed by the CT layer. If the input cell complex $\mathcal{X}$ is of lower dimension than n, the attention on ranks $k > \dim \mathcal{X}$ are ignored. In turn, edges represent either the pairwise attentions performed in the CT layer together with the bias matrices $\mathbf{N}_{k_s \to k_t}$, or simply the matrices used to build the matrix $\mathbf{N}$ from smaller matrices $\mathbf{N}_{k_s \to k_t}$ as in eq. (3), for the general attention. A missing arrow from cochains of rank k_s to cochains of rank k_t implies a zero in the block of $\mathbf{N}$ corresponding to the matrix $\mathbf{N}_{k_s \to k_t}$. An illustration of the tensor diagram used in our experiments is given in fig. 2.

4.4 Positional encodings on cellular complexes

Transformers do not leverage the input structure explicitly by default [94]. Positional encodings help to overcome this problem by injecting positional and structural information about the input

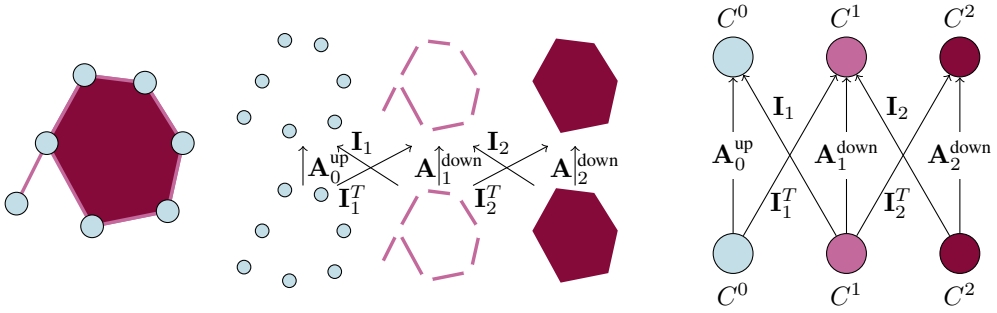


Figure 2: Tensor diagram illustrating the flow of signals between cochains defined on 0-, 1-, and 2-cells. For pairwise attention (section 4.2.1), the neighborhood matrices indicate the bias $\mathbf{N}$ in the attention formula (2). For general attention (section 4.2.2), neighborhood matrices indicates how to build the bias matrix $\mathbf{N}$ by composition of smaller bias matrices $\mathbf{N}_{k_s \to k_t}$ between dimensions.

tokens. For sequences, the first positional encoding used sine and cosine functions depending on the position of the token in the sequence. For graphs, several positional encodings have been studied such as the eigenvectors of the graph Laplacian (LapPE) [32] and Random Walk Positional Encodings (RWPe) [33], where the latter were also adapted for simplicial complex transformers [92, 108].

Let $0 \leq k \leq \dim \mathcal{X}$, where $\mathcal{X}$ is a cell complex. A cellular k-positional encoding of $\mathcal{X}_k$ is a k-cochain $\mathbf{E}_k$ that captures some structural information about $\mathcal{X}_k$ within $\mathcal{X}$ (positional encoding may also be defined on the entire complex $\mathcal{X}$). Given cochains $\mathbf{X}_k$ and positional encodings $\mathbf{E}_k$, the input for the first transformer layer is defined as a function $\mathrm{P}_k \colon \mathcal{C}^k(\mathcal{X}, \mathbb{R}^{d_k}) \times \mathcal{C}^k(\mathcal{X}, \mathbb{R}^{d_{\text{pe}}}) \to \mathcal{C}^k(\mathcal{X}, \mathbb{R}^{d_k^h})$ with $\mathbf{X}_k^1 = \mathrm{P}_k(\mathbf{X}_k, \mathbf{E}_k)$, where P_k combines the signals and the positional encodings. Usual functions are

$$\begin{aligned} \text{SumPE}(\mathbf{X}_k, \mathbf{E}_k) &= \mathbf{X}_k \theta_{\text{in},k} + b_{\text{in},k} + \mathbf{E}_k \theta_{\text{in,pe}} + b_{\text{in,pe}} \\ \text{ConcatPE}(\mathbf{X}_k, \mathbf{E}_k) &= \text{Concat}(\mathbf{X}_k, \mathbf{E}_k) \theta_{\text{in,pe}} + b_{\text{in,pe}}, \end{aligned} \tag{4}$$

where $\theta_\bullet$, $b_\bullet \in$ are learnable parameters. For this paper, we use $\mathrm{P}_k = \text{ConcatPE}$.

Next we discuss three novel positional encodings on cell complexes 4.4.1: Barycentric Subdivision 4.4.2, Random Walk, and Topological Slepians 4.4.3.

4.4.1 BSPe: Barycentric Subdivision Positional Encoding

A popular positional encoding for graph transformers is given by graph Laplacian eigenvectors (LapPE) [32]. LapPE assigns to each vertex v_i a vector $\text{LapPE}(v_i) = (e_i^1, \ldots, e_i^k)$, where $\{e_i^j \mid j = 1, \ldots, k\}$ are eigenvectors of the k smallest eigenvalues of the normalized graph Laplacian for a graph $G = (V, E)$, counting multiplicities, where k is a hyperparameter.

We denote the naive extension from LapPE for cell complexes using the unnormalized Hodge Laplacian matrix, instead of the graph Laplacian one, as **HodgeLapPE**. We use the unnormalized version because normalizing the Hodge Laplacian for dimensions greater than zero is not an easy task [92]. HodgeLapPE are, however, not a good choice for high-order positional encodings *a priori* due to both a lack of normalization and the ambiguous information contained in Hodge Laplacians for nonzero rank. Details on LapPE for graphs and their HodgeLapPE extension are in the Appendix.

To overcome the previous drawbacks, we propose to extend LapPE by taking the original Laplacian positional encodings of the 1-skeleton of the barycentric subdivisions of the cell complexes. The *barycentric subdivision* of a cell complex $\mathcal{X}$, denoted by $\Delta(\mathcal{X})$, is the order complex of its face poset [97], i.e., the abstract simplicial complex whose set of vertices is the set of cells of $\mathcal{X}$ and whose simplices are the totally ordered flags of cells of $\mathcal{X}$. Barycentric subdivisions yield triangulations of cell complexes that preserve their topological properties [25]. The 1-skeleton of the barycentric subdivision of $\mathcal{X}$ is a graph $G = (V, E)$ where V is the set of cells of $\mathcal{X}$ and where two vertices σ_1 and σ_2 are connected if one is a face of the other. The positional encoding of a cell σ is the Laplacian positional encoding of σ seen as a vertex in G.
This positional encoding respects the same theoretical advantages of the LapPE while assigning relative positions to all the cells *at the same time*, and thus relative positions take into account all the cells and not only the cells of a specific dimension. We say that the positional encodings satisfying

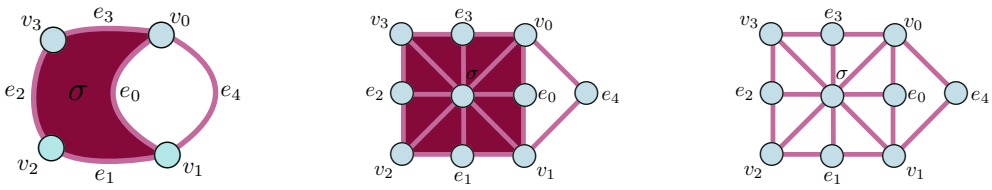


Figure 3: Left: A cell complex $\mathcal{X}$. Center: Barycentric subdivision of $\mathcal{X}$. Right: 1-skeleton of the barycentric subdivision. Each original cell of $\mathcal{X}$ is represented by a node in the 1-skeleton.

this property are called *global*, in contrast to *local* positional encodings, where encodings are assigned independently for each dimension. We denote this positional encoding as **BSPe**.

4.4.2 RWPe: Random Walk Positional Encoding

RWPe take another different way of positioning vertices on a graph based on random walks. Given a vertex $v_i \in V$, the RWPe of v_i is given by the vector $\text{RWPe}(v_i) = \left(\text{RW}_{ii}, \ldots, \text{RW}_{ii}^k\right)$ where $\text{RW} = AD^{-1}$ is the random walk operator of a graph based on edge connectivity. In this case, each vertex is assigned the probabilities of landing again on itself on the random walks from one to k steps. In this case, $\text{RWPe}(v_i)$ is unique and does not need sign or eigenvector selection invariance. For RWPe we have again difficulties defining meaningful random walks on general cell complexes. This problem is first explored in [92] and then in [108] for simplicial complexes, making a special focus on edge random walks. There, transition matrices are defined to perform random walks in simplices of an arbitrary rank and are not affected by the orientation of the simplicial complex.

As a first, more naive approach, we propose to extend RWPe to cell complexes by taking the positional encodings given by the original RWPe for the 1-skeleton of the barycentric subdivision introduced in section 4.4.1. We denote these *global* positional encodings as **RWBSPe**. We also propose a more sophisticated, local approach, denoted **RWPe**, extending the random walks from [92] to cell complexes. The full development of the random walk matrix can be found in the Appendix. From the random walk matrix, positional encodings are taken as in RWPe for each rank of the cell complex.

4.4.3 TopoSlepiansPE: Topological Slepians Positional Encoding

The objective of positional encoding is to assign a relative position to tokens informed by the underlying domain, which for us are cells within a cell complex. Topological Slepians, as introduced in [7], represent a novel category of signals specifically defined over cell complexes. These signals are characterized by their maximal concentration within the topological domain (the cells) and their perfect localization in the corresponding dual domain (the frequencies [89]). This means that the non-zero coordinates of the different topological Slepians are concentrated only on some of the cells of the complex, with a specific spectral content. Topological Slepians have been shown to be particularly efficient for signal representation and sparse coding tasks [7], making them a valuable tool to devise positional encoding.

In this work, we compute Slepians as in Section 4 of [7]. Given an ordered multiset of rank k Slepians $\mathcal{S}_k = \{s_k^i \mid i = 1, \ldots, d_{\text{pe}}\}$, we let the positional encoding of a k-cell σ be $\mathbf{E}(\sigma)_i = s_{k,\sigma}^i$, the coordinate corresponding to σ of the s_k^i Slepian (in practice, we can obtain $|\mathcal{S}_k| = d_{\text{pe}}$ by adding zero vectors). We denote this positional encoding as **TopoSlepiansPE**. Although it comes with guaranteed localization properties, it is a local positional encoding, and suffers from permutation variance and eigenvector sign variance, that the transformers need to learn.

5 Experiments

Due to the lack of cell complex datasets, as argued in [79], we test our cellular transformers in three different graph datasets: `ZINC` [60], `ogbg-molhiv` [55], and the `graph classification benchmark` (`GCB`) [15] in its hard version, that we lift to cell complexes. We lift each graph G to a cell complex by filling its cycles with 2-cells using the `TopoX` library [47]. Cycle filling is not the optimal way of adding cells to our molecules datasets, and may detriment the performance of

Table 1: Models and scores compared for the datasets GCB, ZINC, and ogbg-molhiv. Average and standard deviation for accuracy (↑) are shown for the GCB test dataset; MAE (↓) is reported for the ZINC test dataset; AUC-ROC (↑) is reported for the ogbg-molhiv test dataset. GCB models are described in the Appendix. Results for GCB were extracted from [13], and other results were extracted from [109]. The first seven model rows represent message passing architectures. For the GCB dataset, the other six models are classic machine learning algorithms. For the ZINC and OGB datasets, the second set of rows correspond to graph transformer models, and the third set of rows belong to the simplicial transformer models of [109]. The abbreviation v.n. means virtual node.

GCB		ZINC		ogbg-molhiv	
Model	Accuracy (↑)	Model	MAE (↓)	Model	AUC-ROC (↑)
Graclus [29]	0.690 ± 0.015	GCN [67]	0.367	GCN+v.n.	0.7599
NDP [17]	0.726 ± 0.009	GAT [95]	0.384	GIN+v.n. [102]	0.7707
DiffPool [104]	0.699 ± 0.019	GatedGCN [22]	0.282	DGN [12]	0.7970
Top-K [38]	0.427 ± 0.152	PNA [26]	0.188	PNA	0.7905
SAGPool [69]	0.377 ± 0.145			GSN [21]	0.8039
MinCutPool [16]	0.738 ± 0.019	CIN [19]	0.079	CIN	**0.8094**
ESC + RBF-SVM [73]	0.625 ± 0.046	GIN-AK+ [107]	0.080	GIN-AK+	0.7961
ESC + L1-SVM [73]	0.722 ± 0.010	Graphormer [103]	0.122		
ESC + L2-SVM [73]	0.693 ± 0.016	SAN [68]	0.139	SAN	0.7785
Hist Kernel [74]	0.720 ± 0.000	EGT [58]	0.108		
Jaccard Kernel [74]	0.630 ± 0.000	GPS [85]	**0.070**	GPS	0.7880
Edit Kernel [74]	0.600 ± 0.000	$\mathcal{AS}^{\mathsf{SN}}_{0:1}$	0.080		
Stratedit Kernel [74]	0.600 ± 0.000	$\mathcal{AS}^{\mathsf{SN+VS}}_{0:1}$ [109]	0.073	$\mathcal{AS}^{\mathsf{SN+VS}}_{0:1}$	0.7981
$\mathcal{C}$ (ours)	$\mathbf{0.752 \pm 0.010}$	$\mathcal{C}$ (ours)	0.080	$\mathcal{C}$ (ours)	0.7946

Table 2: Table containing results for the best attention mechanism and positional encoding combinations on the datasets GCB, ZINC, and ogbg-molhiv. Columns indicate combinations of attention mechanisms and positional encodings. Abbreviations for the positional encodings are B for BSPe, H for HodgeLapPE, RB for RWBSPe, T for TopoSlepiansPE, and R for RWPe. Best result, second best result, and third best result for each dataset are highlighted in boldface green, blue, and orange, respectively. For GCB, the third best result is obtained by the zero positional encoding.

Dataset	$\mathcal{A}^s_g$	B	H	RB	T	R
GCB (Accuracy ↑)		**0.7516**	0.7432	0.7442	0.7432	0.7442
ZINC (MAE ↓)		0.0840	0.0824	0.1296	0.1303	0.1051
ogbg-molhiv (AUC-ROC ↑)		0.7681	0.7032	0.7082	0.7192	0.7343
Dataset	$\mathcal{A}^c_{k_s \to k_t}$	B	H	RB	T	R
GCB (Accuracy ↑)		0.7347	0.7337	0.7442	0.7421	0.7379
ZINC (MAE ↓)		0.0833	0.0831	**0.0802**	0.1202	0.0833
ogbg-molhiv (AUC-ROC ↑)		0.7784	0.7658	0.7321	0.7565	0.7338
Dataset	$\mathcal{A}^s_{k_s \to k_t}$	B	H	RB	T	R
GCB (Accuracy ↑)		0.7400	0.7421	0.7389	0.7505	0.7505
ZINC (MAE ↓)		0.0934	0.0852	0.1210	0.1452	0.0973
ogbg-molhiv (AUC-ROC ↑)		**0.7946**	0.7586	0.7058	0.7111	0.7288

the transformers. Yet, we will show that it still allows our proposed transformer to achieve results comparable to the state-of-the-art, and thus represents a first step towards encouraging the community to develop cell complex datasets.

For each dataset, we use its official train, validation and test splits, and we try all the possible combinations of the attention mechanisms presented in section 4.2 with the positional encodings presented in section 4.4 plus a positional encoding called zero that assigns a zero vector of fixed length to each cell, simulating absence of positional encodings. For GCB, we run the experiments with

five different random seeds and report average accuracy and standard deviation. Due to computational constraints, we only run the experiments once for the other two datasets ZINC and ogbg-molhiv, reporting only the obtained score. For the state-of-the-art methods, we report the average and standard deviation achieved for these datasets in [15, 109]. Details on the architectures are in the Appendix.

From state-of-the-art methods in [109], we only report the graph and simplicial architectures, and skip the transformers based on clique-lifting, as these cliques do not appear naturally on the graph as high-order cells, and our purpose is developing general cell complex transformers for cell domains. In their experiments, though, the models, while applicable for arbitrary ranks, were tested using only vertex and edge data, excluding higher-order features. The tensor diagram for the self- and cross-attention and general architectures in each layer can be found in fig. 2. Results with our best performing combinations of attention mechanism and positional encodings are in table 1. A summary of the best attention and positional combinations for CT is reported in table 2. Complete experimental results, training details, and hardware resources used for the experiments are in the Appendix.

5.1 Discussion

Overall, we outperform all previous state-of-the-art methods tested in the GCB dataset and we obtain comparable results to some of the most effective architectures in the other two.

For ZINC, we surpass most of the message passing architectures with the exception of CIN [19] and GIN-AK [107], for which the differences in the performance between them and our models are relatively small compared to the biggest gap in performance for the dataset. For ogbg-molhiv, message passing architectures consistently obtain comparable or better results than graph transformers, except for the GCN [67] and GIN [102] architectures. In the case of graph transformers, the best performing architecture for ZINC is GPS [85], which we surpass in ogbg-molhiv. Following GPS, the second best transformer in ZINC is the simplicial transformer of [109] equipped with virtual simplices, a similar technique to virtual nodes in graph architectures [59, 93]. Without equipping virtual simplices, though, the MAE values for our best model in ZINC and for the simplicial transformers are equal. Both architectures outperform all other non-GPS traditional graph transformers in this dataset, suggesting that high-order interactions are relevant even in the case of graph datasets. For ogbg-molhiv, our best model outperforms graph transformer architectures, and obtains slightly worse performance than the simplicial transformer for vertices and edges of [109] equipped with virtual simplices. Results without virtual simplices were not reported for ogbg-molhiv.

The results corroborate that leveraging high-order information about the dataset in transformer architectures is capable of outperforming or obtaining comparable SOTA results without the need for advanced techniques such as graph rewiring, virtual nodes, or learnable bias matrices in the attention mechanism.

General vs pairwise attention. We observe that pairwise attention mechanisms obtained the best results for the molecule datasets (ZINC and ogbg-molhiv) and the general attention mechanism obtained the best result for the GCB dataset. We observe that in the GCB dataset, vertices, edges, and 2-cells contain *homogeneous* information about clustering properties of the input graphs, where edges and 2-cells use concatenations of vertex features associated to the cell. On the other hand, ZINC and ogb-molhiv contain atomic information for the nodes and 2-cells and bond information for the edges, making the cochains *heterogeneous* at different ranks. The results suggest that

1. general attention, using common query and key projections for all cells at the same time, is more suitable for problems where features in all ranks are homogeneous, i.e., of the same nature;

2. pairwise attention, that uses specific query and key projections for pairwise ranks, is more suitable for problems where features are heterogeneous, because it allows to each rank to attend to specific properties of each rank in an isolated way.

Global vs local positional encodings. We classified our positional encodings into two groups depending on whether they were obtained for all ranks simultaneously (global p.e.) or for each rank isolatedly (local p.e.). We observe that global positional encodings obtained the best results in the three datasets. However, for GCB and ZINC, the second and third best results used local positional encodings. For GCB, the second place was shared between TopoSlepiansPE and RWPe, while the third place was achieved with the zero positional encoding. For the ZINC dataset, the second and third

places were obtained by HodgeLapPE. Interestingly, neither TopoSlepiansPE nor RWPe obtained good results for the molecule datasets overall.

6 Conclusion

In this work, we introduced the Cellular Transformer (CT), a novel transformer architecture for cell complexes that leverages high-order relationships inherent in topological spaces, together with new positional encodings for it. Our experimental results demonstrated that CT achieves state-of-the-art performance or comparable results without the need for additional enhancements such as virtual nodes or learnable bias matrices. This work motivates further exploration in topological deep learning, particularly in the development of cell complex datasets and the improvement of transformer layers and positional encodings, such as it has happened for graph transformers. Future work will focus on extending the CT framework to more efficient and effective attention mechanisms, exploring more sophisticated positional encodings, and applying the framework to a broader range of real-world datasets and techniques where transformers are being used, such as generative or multi-modal models.

6.1 Limitations

Cellular transformers suffer from the same computational limitations as traditional and graph transformers [94], with attention having quadratic complexity on the number of cells. This means that cellular transformers are currently limited to cell complexes with a low number of cells. However, as in other transformers, computational limitations may be overcome with linearized attention mechanisms [23]. We leave the study of efficient cellular transformers as future work. Another drawback that comes with general topological deep learning is that, for harnessing the power of cell complexes in graph datasets, the selection of a good lifting procedure converting graphs into complexes is fundamental. Although there is no general answer as to which lifting algorithm to select in each case, we expect to see proposals in the second edition of the Topological Deep Learning Challenge [14].

References

[1] A. Arnab, M. Dehghani, G. Heigold, C. Sun, M. Lučić, and C. Schmid. 'Vivit: A video vision transformer'. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 6836–6846.

[2] D. Arya and M. Worring. 'Exploiting relational information in social networks using geometric deep learning on hypergraphs'. In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. 2018, pp. 117–125.

[3] S. Bai, F. Zhang, and P. H. Torr. 'Hypergraph convolution and hypergraph attention'. In: *Pattern Recognition* 110 (2021), p. 107637.

[4] S. Barbarossa and S. Sardellitti. 'Topological Signal Processing Over Simplicial Complexes'. In: *IEEE Transactions on Signal Processing* 68 (2020), pp. 2992–3007. ISSN: 1941-0476. DOI: 10.1109/tsp.2020.2981920. URL: http://dx.doi.org/10.1109/TSP.2020.2981920.

[5] F. Barbero, C. Bodnar, H. S. de Ocáriz Borde, M. Bronstein, P. Veličković, and P. Liò. *Sheaf Neural Networks with Connection Laplacians*. 2022. arXiv: 2206.08702 [cs.LG].

[6] C. Battiloro, Z. Wang, H. Riess, P. D. Lorenzo, and A. Ribeiro. 'Tangent Bundle Filters and Neural Networks: From Manifolds to Cellular Sheaves and Back'. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5.

[7] C. Battiloro, P. Di Lorenzo, and S. Barbarossa. 'Topological Slepians: Maximally Localized Representations of Signals Over Simplicial Complexes'. In: *ICASSP 2023 - 2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2023, pp. 1–5. DOI: 10.1109/ICASSP49357.2023.10095803.

[8] C. Battiloro, I. Spinelli, L. Telyatnikov, M. M. Bronstein, S. Scardapane, and P. D. Lorenzo. 'From Latent Graph to Latent Topology Inference: Differentiable Cell Complex Module'. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=0JsRZEGZ7L.

[9] C. Battiloro, Z. Wang, H. Riess, P. Di Lorenzo, and A. Ribeiro. 'Tangent bundle convolutional learning: from manifolds to cellular sheaves and back'. In: *IEEE Transactions on Signal Processing* (2024).

[10] F. Battiston et al. 'The physics of higher-order interactions in complex systems'. In: *Nature Physics* 17.10 (2021), pp. 1093–1098.

[11] F. Battiston, G. Cencetti, I. Iacopini, V. Latora, M. Lucas, A. Patania, J.-G. Young, and G. Petri. 'Networks beyond pairwise interactions: structure and dynamics'. In: *Physics Reports* 874 (2020), pp. 1–92.

[12] D. Beaini, S. Passaro, V. L'etourneau, W. L. Hamilton, G. Corso, and P. Lio'. 'Directional Graph Networks'. In: *International Conference on Machine Learning*. 2020.

[13] *Benchmark dataset for graph classification GitHub repository*. `https://github.com/FilippoMB/Benchmark_dataset_for_graph_classification`. Accessed: 2024-04-14.

[14] G. Bernárdez, L. Telyatnikov, M. Montagna, M. Papillon, M. Ferriol-Galmés, F. Baccini, N. Miolane, M. Hajij, T. Papamarkou, G. Alzamzmi, T. Doster, T. Emerson, H. Kvinge, and B. Rieck. *ICML Topological Deep Learning Challenge 2024: Beyond the Graph Domain*. `https://pyt-team.github.io/packs/challenge.html`. Accessed: 2024-05-15. 2024.

[15] F. M. Bianchi, C. Gallicchio, and A. Micheli. 'Pyramidal Reservoir Graph Neural Network'. In: vol. 470. Elsevier, 2022, pp. 389–404.

[16] F. M. Bianchi, D. Grattarola, and C. Alippi. 'Spectral clustering with graph neural networks for graph pooling'. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 874–883.

[17] F. M. Bianchi, D. Grattarola, L. Livi, and C. Alippi. 'Hierarchical Representation Learning in Graph Neural Networks With Node Decimation Pooling'. In: *IEEE Transactions on Neural Networks and Learning Systems* 33.5 (2022), pp. 2195–2207. DOI: `10.1109/TNNLS.2020.3044146`.

[18] C. Bick, E. Gross, H. A. Harrington, and M. T. Schaub. 'What are higher-order networks?' In: *arXiv:2104.11329* (2021).

[19] C. Bodnar, F. Frasca, N. Otter, Y. G. Wang, P. Lio', G. Montúfar, and M. M. Bronstein. 'Weisfeiler and Lehman Go Cellular: CW Networks'. In: *Neural Information Processing Systems*. 2021.

[20] C. Bodnar, F. D. Giovanni, B. P. Chamberlain, P. Lio, and M. M. Bronstein. 'Neural Sheaf Diffusion: A Topological Perspective on Heterophily and Oversmoothing in GNNs'. In: *Advances in Neural Information Processing Systems*. 2022.

[21] G. Bouritsas, F. Frasca, S. Zafeiriou, and M. M. Bronstein. 'Improving Graph Neural Network Expressivity via Subgraph Isomorphism Counting'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 45.1 (2023), pp. 657–668. DOI: `10.1109/TPAMI.2022.3154319`.

[22] X. Bresson and T. Laurent. 'Residual Gated Graph ConvNets'. In: *ArXiv* abs/1711.07553 (2017).

[23] K. Choromanski, V. Likhosherstov, D. Dohan, X. Song, A. Gane, T. Sarlos, P. Hawkins, J. Davis, A. Mohiuddin, L. Kaiser, D. Belanger, L. Colwell, and A. Weller. *Rethinking Attention with Performers*. 2022. arXiv: `2009.14794 [cs.LG]`.

[24] J. Clift, D. Doryn, D. Murfet, and J. Wallbridge. 'Logic and the 2-Simplicial Transformer'. In: *International Conference on Learning Representations*. 2020. URL: `https://openreview.net/forum?id=rkecJ6VFvr`.

[25] G. E. COOKE and R. L. PINNEY. *Homology of Cell Complexes*. Princeton University Press, 1967. ISBN: 9780691623139. URL: `http://www.jstor.org/stable/j.ctt183pvgt` (visited on 16/05/2024).

[26] G. Corso, L. Cavalleri, D. Beaini, P. Lio', and P. Velickovic. 'Principal Neighbourhood Aggregation for Graph Nets'. In: *ArXiv* abs/2004.05718 (2020).

[27] M. Defferrard, X. Bresson, and P. Vandergheynst. 'Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering'. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016. URL: `https://proceedings.neurips.cc/paper_files/paper/2016/file/04df4d434d481c5bb723be1b6df1ee65-Paper.pdf`.

[28] S. Deng, S. Wang, H. Rangwala, L. Wang, and Y. Ning. 'Cola-gnn: Cross-location attention based graph neural networks for long-term ili prediction'. In: *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 2020, pp. 245–254.

[29] I. S. Dhillon, Y. Guan, and B. Kulis. 'Weighted Graph Cuts without Eigenvectors A Multilevel Approach'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.11 (2007), pp. 1944–1957. DOI: 10.1109/TPAMI.2007.1115.

[30] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. 'An image is worth 16x16 words: Transformers for image recognition at scale'. In: *arXiv preprint arXiv:2010.11929* (2020).

[31] V. P. Dwivedi and X. Bresson. 'A Generalization of Transformer Networks to Graphs'. In: *AAAI Workshop on Deep Learning on Graphs: Methods and Applications* (2021).

[32] V. P. Dwivedi, C. K. Joshi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. 'Benchmarking Graph Neural Networks'. In: *Journal of Machine Learning Research* 24.43 (2023), pp. 1–48. URL: http://jmlr.org/papers/v24/22-0567.html.

[33] V. P. Dwivedi, A. T. Luu, T. Laurent, Y. Bengio, and X. Bresson. 'Graph Neural Networks with Learnable Structural and Positional Representations'. In: *International Conference on Learning Representations*. 2022. URL: https://openreview.net/forum?id=wTTjnvGphYj.

[34] E. Estrada and G. J. Ross. 'Centralities in simplicial complexes. Applications to protein interaction networks'. In: *Journal of Theoretical Biology* 438 (2018), pp. 46–60. ISSN: 0022-5193. DOI: https://doi.org/10.1016/j.jtbi.2017.11.003. URL: https://www.sciencedirect.com/science/article/pii/S0022519317305040.

[35] W. Falcon and The PyTorch Lightning team. *PyTorch Lightning*. Version 1.4. Mar. 2019. DOI: 10.5281/zenodo.3828935. URL: https://github.com/Lightning-AI/lightning.

[36] Y. Feng, H. You, Z. Zhang, R. Ji, and Y. Gao. 'Hypergraph Neural Networks'. In: *Proc. AAAI* 33.01 (2019), pp. 3558–3565.

[37] M. Fey and J. E. Lenssen. 'Fast Graph Representation Learning with PyTorch Geometric'. In: *ICLR Workshop on Representation Learning on Graphs and Manifolds*. 2019.

[38] H. Gao and S. Ji. 'Graph U-Nets'. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 2083–2092. URL: https://proceedings.mlr.press/v97/gao19a.html.

[39] Y. Gao, Z. Zhang, H. Lin, X. Zhao, S. Du, and C. Zou. 'Hypergraph learning: Methods and practices'. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[40] L. Giusti, C. Battiloro, P. Di Lorenzo, S. Sardellitti, and S. Barbarossa. 'Simplicial Attention Networks'. In: *arXiv preprint arXiv:2203.07485* (2022).

[41] L. Giusti, C. Battiloro, L. Testa, P. Di Lorenzo, S. Sardellitti, and S. Barbarossa. 'Cell attention networks'. In: *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2023, pp. 1–8.

[42] C. W. J. Goh, C. Bodnar, and P. Lio. 'Simplicial attention networks'. In: *arXiv preprint arXiv:2204.09455* (2022).

[43] L. J. Grady and J. R. Polimeni. *Discrete calculus: Applied analysis on graphs for computational science*. Vol. 3. Springer, 2010.

[44] S. Gurugubelli and S. P. Chepuri. 'SaNN: Simple Yet Powerful Simplicial-aware Neural Networks'. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=eUgS9Ig8JG.

[45] A. Hagberg, P. J. Swart, and D. A. Schult. 'Exploring network structure, dynamics, and function using NetworkX'. In: (Jan. 2008). URL: https://www.osti.gov/biblio/960616.

[46] M. Hajij, K. Istvan, and G. Zamzmi. 'Cell Complex Neural Networks'. In: *NeurIPS Workshop TDA and Beyond* (2020).

[47] M. Hajij, M. Papillon, F. Frantzen, J. Agerberg, I. AlJabea, R. Ballester, C. Battiloro, G. Bernárdez, T. Birdal, A. Brent, et al. 'TopoX: a suite of Python packages for machine learning on topological domains'. In: *arXiv preprint arXiv:2402.02441* (2024).

[48] M. Hajij, G. Zamzmi, T. Papamarkou, V. Maroulas, and X. Cai. 'Simplicial Complex Representation Learning'. In: *Machine Learning on Graphs (MLoG) Workshop at ACM International WSD Conference* (2022).

[49] M. Hajij, G. Zamzmi, T. Papamarkou, N. Miolane, A. Guzmán-Sáenz, K. N. Ramamurthy, T. Birdal, T. K. Dey, S. Mukherjee, S. N. Samaga, et al. 'Topological Deep Learning: Going Beyond Graph Data'. In: ().

[50] K. Han, Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, Y. Xu, et al. 'A survey on vision transformer'. In: *IEEE transactions on pattern analysis and machine intelligence* 45.1 (2022), pp. 87–110.

[51] J. Hansen and R. Ghrist. 'Toward a spectral theory of cellular sheaves'. In: *Journal of Applied and Computational Topology* (2019).

[52] J. Hansen and T. Gebhart. *Sheaf Neural Networks*. 2020. arXiv: 2012.06333 [cs.LG].

[53] A. Hatcher. *Algebraic Topology*. Cambridge University Press, 2005.

[54] D. Hernández Serrano, J. Hernández-Serrano, and D. Sánchez Gómez. 'Simplicial degree in complex networks. Applications of topological data analysis to network science'. In: *Chaos Solitons Fractals* 137 (2020), pp. 109839, 21. ISSN: 0960-0779. DOI: 10.1016/j.chaos.2020.109839. URL: https://doi.org/10.1016/j.chaos.2020.109839.

[55] W. Hu, M. Fey, M. Zitnik, Y. Dong, H. Ren, B. Liu, M. Catasta, and J. Leskovec. 'Open Graph Benchmark: Datasets for Machine Learning on Graphs'. In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin. Vol. 33. Curran Associates, Inc., 2020, pp. 22118–22133. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/fb60d411a5c5b72b2e7d3527cfc84fd0-Paper.pdf.

[56] Z. Hu, J. Wang, S. Chen, and X. Du. 'A Semi-supervised Framework with Efficient Feature Extraction and Network Alignment for User Identity Linkage'. In: *Database Systems for Advanced Applications: 26th International Conference, DASFAA 2021, Taipei, Taiwan, April 11–14, 2021, Proceedings, Part II 26*. Springer. 2021, pp. 675–691.

[57] M. S. Hussain, M. J. Zaki, and D. Subramanian. 'Edge-augmented graph transformers: Global self-attention is enough for graphs'. In: *arXiv preprint arXiv:2108.03348* (2021).

[58] M. S. Hussain, M. J. Zaki, and D. Subramanian. 'Global Self-Attention as a Replacement for Graph Convolution'. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (2021). URL: https://api.semanticscholar.org/CorpusID:249375304.

[59] E. Hwang, V. Thost, S. S. Dasgupta, and T. Ma. 'An Analysis of Virtual Nodes in Graph Neural Networks for Link Prediction (Extended Abstract)'. In: *The First Learning on Graphs Conference*. 2022. URL: https://openreview.net/forum?id=dI6KBKNRp7.

[60] J. J. Irwin, T. Sterling, M. M. Mysinger, E. S. Bolstad, and R. G. Coleman. 'ZINC: A Free Tool to Discover Chemistry for Biology'. In: *Journal of Chemical Information and Modeling* 52.7 (2012). PMID: 22587354, pp. 1757–1768. DOI: 10.1021/ci3001277. eprint: https://doi.org/10.1021/ci3001277. URL: https://doi.org/10.1021/ci3001277.

[61] J. Jiang, Y. Wei, Y. Feng, J. Cao, and Y. Gao. 'Dynamic Hypergraph Neural Networks.' In: *IJCAI*. 2019, pp. 2635–2641.

[62] W. Jiang and J. Luo. 'Graph neural network for traffic forecasting: A survey'. In: *arXiv preprint arXiv:2101.11174* (2021).

[63] X. Jiang, L.-H. Lim, Y. Yao, and Y. Ye. 'Statistical ranking and combinatorial Hodge theory'. In: *Mathematical Programming* 127.1 (2011), pp. 203–244.

[64] J. D. M.-W. C. Kenton and L. K. Toutanova. 'Bert: Pre-training of deep bidirectional transformers for language understanding'. In: *Proceedings of naacL-HLT*. Vol. 1. 2019, p. 2.

[65] E.-S. Kim, W. Y. Kang, K.-W. On, Y.-J. Heo, and B.-T. Zhang. 'Hypergraph attention networks for multimodal learning'. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 14581–14590.

[66] J. Kim, S. Oh, and S. Hong. 'Transformers generalize deepsets and can be extended to graphs & hypergraphs'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 28016–28028.

[67] T. N. Kipf and M. Welling. 'Semi-Supervised Classification with Graph Convolutional Networks'. In: (2017). URL: https://openreview.net/forum?id=SJU4ayYgl.

[68] D. Kreuzer, D. Beaini, W. L. Hamilton, V. L'etourneau, and P. Tossou. 'Rethinking Graph Transformers with Spectral Attention'. In: *ArXiv* abs/2106.03893 (2021).

[69] J. Lee, I. Lee, and J. Kang. 'Self-Attention Graph Pooling'. In: *Proceedings of the 36th International Conference on Machine Learning*. Ed. by K. Chaudhuri and R. Salakhutdinov. Vol. 97. Proceedings of Machine Learning Research. PMLR, Sept. 2019, pp. 3734–3743. URL: https://proceedings.mlr.press/v97/lee19c.html.

[70] L.-H. Lim. 'Hodge Laplacians on graphs'. In: *Siam Review* 62.3 (2020), pp. 685–715.

[71] K. Lin, L. Wang, and Z. Liu. 'Mesh graphormer'. In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2021, pp. 12939–12948.

[72] K. Maggs, C. Hacker, and B. Rieck. 'Simplicial Representation Learning with Neural k-Forms'. In: *The Twelfth International Conference on Learning Representations*. 2024. URL: https://openreview.net/forum?id=Djw0XhjHZb.

[73] A. Martino, A. Giuliani, and A. Rizzi. '(Hyper)Graph Embedding and Classification via Simplicial Complexes'. In: *Algorithms* 12.11 (2019). ISSN: 1999-4893. DOI: 10.3390/a12110223. URL: https://www.mdpi.com/1999-4893/12/11/223.

[74] A. Martino and A. Rizzi. '(Hyper)graph Kernels over Simplicial Complexes'. In: *Entropy* 22.10 (2020). ISSN: 1099-4300. DOI: 10.3390/e22101155. URL: https://www.mdpi.com/1099-4300/22/10/1155.

[75] G. Mialon, D. Chen, M. Selosse, and J. Mairal. *GraphiT: Encoding Graph Structure in Transformers*. 2021. arXiv: 2106.05667 [cs.LG].

[76] E. Min, R. Chen, Y. Bian, T. Xu, K. Zhao, W. Huang, P. Zhao, J. Huang, S. Ananiadou, and Y. Rong. 'Transformer for graphs: An overview from architecture perspective'. In: *arXiv preprint arXiv:2202.08455* (2022).

[77] E. Min, Y. Rong, T. Xu, Y. Bian, P. Zhao, J. Huang, D. Luo, K. Lin, and S. Ananiadou. 'Masked Transformer for Neighhourhood-aware Click-Through Rate Prediction'. In: *CoRR* abs/2201.13311 (2022). URL: https://arxiv.org/abs/2201.13311.

[78] N. Miolane, M. Hajij, E. Paine, T. Papamarkou, J. Hoppe, J. Meissner, F. Frantzen, M. Papillon, and USFCA-MSDS. *pyt-team/TopoNetX: TopoNetX 0.0.2*. Version 0.0.2. May 2023. DOI: 10.5281/zenodo.7958504. URL: https://doi.org/10.5281/zenodo.7958504.

[79] T. Papamarkou, T. Birdal, M. Bronstein, G. Carlsson, J. Curry, Y. Gao, M. Hajij, R. Kwitt, P. Liò, P. D. Lorenzo, V. Maroulas, N. Miolane, F. Nasrin, K. N. Ramamurthy, B. Rieck, S. Scardapane, M. T. Schaub, P. Veličković, B. Wang, Y. Wang, G.-W. Wei, and G. Zamzmi. *Position Paper: Challenges and Opportunities in Topological Deep Learning*. 2024. arXiv: 2402.08871 [cs.LG].

[80] M. Papillon, S. Sanborn, M. Hajij, and N. Miolane. 'Architectures of topological deep learning: A survey on topological neural networks'. In: *arXiv preprint arXiv:2304.10031* (2023).

[81] J. Park, Y. Hwang, M. Kim, M. K. Chung, G. Wu, and W. H. Kim. 'Convolving Directed Graph Edges via Hodge Laplacian for Brain Network Analysis'. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2023*. Ed. by H. Greenspan, A. Madabhushi, P. Mousavi, S. Salcudean, J. Duncan, T. Syeda-Mahmood, and R. Taylor. Cham: Springer Nature Switzerland, 2023, pp. 789–799. ISBN: 978-3-031-43904-9.

[82] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. 'PyTorch: an imperative style, high-performance deep learning library'. In: *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2019.

[83] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. 'Improving language understanding by generative pre-training'. In: (2018).

[84] K. N. Ramamurthy, A. Guzmán-Sáenz, and M. Hajij. 'Topo-mlp: A simplicial network without message passing'. In: *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE. 2023, pp. 1–5.

[85] L. Rampasek, M. Galkin, V. P. Dwivedi, A. T. Luu, G. Wolf, and D. Beaini. 'Recipe for a General, Powerful, Scalable Graph Transformer'. In: *ArXiv* abs/2205.12454 (2022).

[86] O. Rioul and M. Vetterli. 'Wavelets and signal processing'. In: *IEEE Signal Proc. Mag.* 8.4 (1991), pp. 14–38. DOI: 10.1109/79.91217.

[87] T. M. Roddenberry, M. T. Schaub, and M. Hajij. 'Signal processing on cell complexes'. In: *Proc. IEEE ICASSP* (2022).

[88] Y. Rong, Y. Bian, T. Xu, W. Xie, Y. Wei, W. Huang, and J. Huang. 'Self-supervised graph transformer on large-scale molecular data'. In: *Advances in Neural Information Processing Systems* 33 (2020), pp. 12559–12571.

[89] S. Sardellitti and S. Barbarossa. 'Topological Signal Representation and Processing over Cell Complexes'. In: *arXiv preprint arXiv:2201.08993* (2022).

[90] S. Sardellitti, S. Barbarossa, and L. Testa. 'Topological Signal Processing over Cell Complexes'. In: *Proceeding IEEE Asilomar Conference. Signals, Systems and Computers* (2021).

[91] 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'. In: *Signal Processing* 187 (2021), p. 108149.

[92] M. T. Schaub, A. R. Benson, P. Horn, G. Lippner, and A. Jadbabaie. 'Random Walks on Simplicial Complexes and the Normalized Hodge 1-Laplacian'. In: *SIAM Review* 62.2 (2020), pp. 353–391. DOI: 10.1137/18M1201019. eprint: https://doi.org/10.1137/18M1201019. URL: https://doi.org/10.1137/18M1201019.

[93] H. Shirzad, A. Velingker, B. Venkatachalam, D. J. Sutherland, and A. K. Sinop. 'Exphormer: Sparse transformers for graphs'. In: *International Conference on Machine Learning*. PMLR. 2023, pp. 31613–31632.

[94] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. 'Attention is all you need'. In: *Advances in neural information processing systems* 30 (2017).

[95] P. Velickovic, G. Cucurull, A. Casanova, A. Romero, P. Lio', and Y. Bengio. 'Graph Attention Networks'. In: *ArXiv* abs/1710.10903 (2017).

[96] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors. 'SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python'. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[97] M. L. Wachs. 'Poset topology: tools and applications'. In: *arXiv preprint math/0602226* (2006). arXiv: math/0602226 [math.CO].

[98] J. Wang, K. Ding, L. Hong, H. Liu, and J. Caverlee. 'Next-item recommendation with sequential hypergraphs'. In: *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 2020, pp. 1101–1110.

[99] M. Wang, D. Zheng, Z. Ye, Q. Gan, M. Li, X. Song, J. Zhou, C. Ma, L. Yu, Y. Gai, T. Xiao, T. He, G. Karypis, J. Li, and Z. Zhang. 'Deep Graph Library: A Graph-Centric, Highly-Performant Package for Graph Neural Networks'. In: *arXiv preprint arXiv:1909.01315* (2019).

[100] Z. Wu, P. Jain, M. Wright, A. Mirhoseini, J. E. Gonzalez, and I. Stoica. 'Representing long-range context for graph neural networks with global attention'. In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 13266–13279.

[101] R. Xiong, Y. Yang, D. He, K. Zheng, S. Zheng, C. Xing, H. Zhang, Y. Lan, L. Wang, and T. Liu. 'On Layer Normalization in the Transformer Architecture'. In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by H. D. III and A. Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, 13–18 Jul 2020, pp. 10524–10533. URL: https://proceedings.mlr.press/v119/xiong20b.html.

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

[103] C. Ying, T. Cai, S. Luo, S. Zheng, G. Ke, D. He, Y. Shen, and T.-Y. Liu. 'Do Transformers Really Perform Bad for Graph Representation?' In: *Neural Information Processing Systems*. 2021.

[104] Z. Ying, J. You, C. Morris, X. Ren, W. Hamilton, and J. Leskovec. 'Hierarchical Graph Representation Learning with Differentiable Pooling'. In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/e77dbaf6759253c7c6d0efc5690369c7-Paper.pdf.

[105] J. Zhang, H. Zhang, C. Xia, and L. Sun. 'Graph-bert: Only attention is needed for learning graph representations'. In: *arXiv preprint arXiv:2001.05140* (2020).

[106] R. Zhang, Y. Zou, and J. Ma. 'Hyper-SAGNN: a self-attention based graph neural network for hypergraphs'. In: *International Conference on Learning Representations (ICLR)*. 2020.

[107] L. Zhao, W. Jin, L. Akoglu, and N. Shah. 'From Stars to Subgraphs: Uplifting Any GNN with Local Structure Awareness'. In: *ArXiv* abs/2110.03753 (2021).

[108] C. Zhou, X. Wang, and M. Zhang. 'Facilitating Graph Neural Networks with Random Walk on Simplicial Complexes'. In: *Thirty-seventh Conference on Neural Information Processing Systems*. 2023. URL: https://openreview.net/forum?id=H57w5EOj6O.

[109] C. Zhou, R. Yu, and Y. Wang. *On the Theoretical Expressive Power and the Design Space of Higher-Order Graph Transformers*. 2024. arXiv: 2404.03380 [cs.LG].

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

A Architecture details and experiments

Pairwise attention transformer layer. Following the usual prenorm design [101], the output of the cellular transformer layer for a specific rank k_t is denoted $\mathbf{X}_{k_t,l+1}$ and computed in six steps, as follows:

$$\begin{aligned}
&\mathbf{X}^1_{k_t,l} = \text{LayerNorm}_{k_t}(\mathbf{X}_{k_t,l}),\\
&\mathbf{X}^1_{k_s,l} = \text{LayerNorm}_{k_s}(\mathbf{X}_{k_s,l}) \text{ for each } k_s \text{ in the tensor diagram},\\
&\mathbf{X}^2_{k_s\to k_t,l} = \mathcal{A}^\bullet_{k_s\to k_t}(\mathbf{X}^1_{k_t,l}, \mathbf{X}^1_{k_s,l}) \text{ for each } k_s \text{ in the tensor diagram},\\
&\mathbf{X}^3_{k_s\to k_t,l} = \text{Dropout}(\mathbf{X}^2_{k_s\to k_t,l}) \text{ for each } k_s \text{ in the tensor diagram},\\
&\mathbf{X}^4_{k_t,l} = \mathbf{X}_{k_t,l} + \sum_{k_s} \mathbf{X}^3_{k_s\to k_t,l}\\
&\mathbf{X}^5_{k_t,l} = \text{LayerNorm}(\mathbf{X}^4_{k_t,l}),\\
&\mathbf{X}^6_{k_t,l} = \text{Dropout}(\text{FFN}_2(\text{Dropout}(\text{ReLU}(\text{FFN}_1(\mathbf{X}^5_{k_t,l}))))),\\
&\mathbf{X}_{k_t,l+1} = \mathbf{X}^4_{k_t,l} + \mathbf{X}^6_{k_t,l},
\end{aligned}$$

The LayerNorm is unique for each dimension d and each layer l.

General attention transformer layer. Similarly to the pairwise attention transformer layer, the general attention transformer layer introduced in section 4.2.2 performs the following steps:

$$\begin{aligned}
&\mathbf{X}^1_l = \text{LayerNorm}(\mathbf{X}_l),\\
&\mathbf{X}^2_l = \mathcal{A}^\bullet_g(\mathbf{X}^1_l),\\
&\mathbf{X}^3_l = \text{Dropout}(\mathbf{X}^2_l),\\
&\mathbf{X}^4_l = \mathbf{X}_l + \mathbf{X}^3_l,\\
&\mathbf{X}^5_l = \text{LayerNorm}(\mathbf{X}^4_l),\\
&\mathbf{X}^6_l = \text{Dropout}(\text{FFN}_2(\text{Dropout}(\text{ReLU}(\text{FFN}_1(\mathbf{X}^5_l))))),\\
&\mathbf{X}_{l+1} = \mathbf{X}^4_l + \mathbf{X}^6_l.
\end{aligned}$$

Training details. All experiments use a `Cosine Annealing` scheduler with linear warmup, an `AdamW` optimizer with $\epsilon = 1^{-8}$, $(\mu_1, \mu_2) = (0.9, 0.999)$ and variable peak learning rate, and a gradient clipping norm of 5. All our transformer architectures, after the transformer layers, use a fully-connected readout whose dropout and number of hidden layers is fixed for each set of experiments, followed by a global add pool layer over all the vertex signals to perform prediction or regression. The fully-connected block begins with a number of neurons equivalent to the hidden dimension of the transformers and concludes with a number of neurons corresponding to the network's output number. Throughout the block, each hidden layer has half as many neurons as its predecessor.

Architecture details of the cellular transformer. Table 3 presents the hyperparameters of the cellular transformer for the three datasets `GCB`, `ogbg-molhiv`, and `ZINC`.

Table 3: Cellular transformer parameters for the experiments of the three datasets. The attention type and the positional encodings vary depending on the experiment configuration.

	GCB	ogbg-molhiv	ZINC
#Layers	12	12	12
Hidden dimension (d^h)	80	768	96
FFN inner-layer dimension	80	768	96
# Attention heads (m)	8	32	8
Hidden dimension of each head	10	24	12
Attention dropout	0.1	0.1	0.1
Embedding dropout	0.0	0.0	0.0
Readout MLP dropout	0.1	0.0	0.0
Max epochs	350	200	10000
Peak learning rate	3e−4	2e−4	2e−4
Batch size	256	1024	256
Warmup epochs	35	20	1000
Weight decay	0.01	1e−5	0.01
# hidden layers readout MLP	1	3	2

Signals on cells. All graphs in the three datasets contain at least discrete signals for the vertices. For the GCB dataset, we associate to each edge a signal corresponding to concatenating the signals of its endpoints. For the ogbg-molhiv and ZINC datasets, the edges contain signals, so we do not change them. As a first step in the transformer architecture, we learn an embedding for the discrete features. For the edge features in GCB, each vertex feature is embedded individually. For the three datasets, signals on the 2-cells are given by sum of the embedded signals of their vertices.

GCB **architectures**. The first six models in table 1 are all graph neural networks with different graph pooling layers and common architecture composition given by MP(32)-Pool-MP(32)-Pool-MP(32)-GlobalPool-Dense(Softmax), where MP(32) is a Chebyshev convolutional layer [27] with 32 hidden units, Pool is a pooling message passing layer, GlobalPool is a global pool layer used as readout, and Dense(Softmax) is a dense layer with softmax activation. Skip connections were used. The other state-of-the-art models consist of models proposed in [73, 74].

B Mathematical details and examples

Cell complexes. A definition of cell complexes in the context of algebraic topology can be found in [53]. In brief, a cell complex is a topological space $\mathcal{X}$ that can be decomposed as a union of disjoint subspaces called *cells*, where each cell σ is homeomorphic to $\mathbb{R}^k$ for some integer $k \geq 0$, called the *rank* of σ. Additionally, for every cell σ, the difference $\overline{\sigma} \setminus \sigma$ is a union of finitely many cells of lower rank, where $\overline{\sigma}$ denotes the closure of σ. The *dimension* of a finite cell complex is the maximum of the ranks of its cells. The set of cells of rank k in a cell complex $\mathcal{X}$ is denoted by $\mathcal{X}_k$. The n-*skeleton* of $\mathcal{X}$ is the cell complex spanned by $\mathcal{X}_0, \ldots, \mathcal{X}_n$, for $0 \leq n \leq \dim \mathcal{X}$.

A *characteristic map* for a cell σ of rank k is a map from the Euclidean unit closed ball of dimension k into $\overline{\sigma}$ whose restriction to the open ball is a homeomorphism. A cell complex is called *regular* if each cell σ admits a characteristic map which is itself a homeomorphism from the closed ball to $\overline{\sigma}$. For example, a decomposition of a circle as the union of a 0-cell and a 1-cell is not regular. Geometric realizations of abstract simplicial complexes are regular cell complexes. Cell complexes generalize simplicial complexes as their cells are not constrained to be simplices.

Boundary and coboundary operators. For each rank k, the incidence matrix $\mathbf{B}_k$ of a cell complex $\mathcal{X}$, as defined in section 3, is the matrix of the *boundary operator* $\mathcal{C}_k(\mathcal{X}) \to \mathcal{C}_{k-1}(\mathcal{X})$, where $\mathcal{C}_k(\mathcal{X})$ is the $\mathbb{R}$-vector space spanned by the set $\mathcal{X}_k$ of k-cells of $\mathcal{X}$. The transpose $\mathbf{B}_k^T$ is the matrix of the *coboundary operator* $\mathcal{C}^{k-1}(\mathcal{X}) \to \mathcal{C}^k(\mathcal{X})$ on the dual vector spaces. Thus the matrix $\mathbf{B}_k^T$ encodes reverse incidence relations from $(k-1)$-cells to k-cells.

Neighborhood matrices. We jointly call *neighborhood matrices* the (signed or non-signed) incidence matrices, Laplacians, and adjacency matrices. Laplacians are extensively used in graph learning

[32, 81] and signal processing [4, 90]. For a cell complex $\mathcal{X}$ of arbitrary dimension, the k-th *Hodge Laplacian* is defined in terms of incidence matrices as $\mathbf{L}_k = \mathbf{B}_k^T\mathbf{B}_k + \mathbf{B}_{k+1}\mathbf{B}_{k+1}^T$, where $\mathbf{B}_k = 0$ if $k = 0$ or $k > \dim \mathcal{X}$. The *upper* and *lower* Laplacians are, respectively, $\mathbf{L}_k^{\text{up}} = \mathbf{B}_{k+1}\mathbf{B}_{k+1}^T$ and $\mathbf{L}_k^{\text{down}} = \mathbf{B}_k^T\mathbf{B}_k$. Therefore, for a 2-dimensional cell complex,

$$\mathbf{L}_0 = \mathbf{B}_1\mathbf{B}_1^T, \qquad \mathbf{L}_1 = \mathbf{B}_1^T\mathbf{B}_1 + \mathbf{B}_2\mathbf{B}_2^T, \qquad \mathbf{L}_2 = \mathbf{B}_2^T\mathbf{B}_2.$$

The lower Laplacian $\mathbf{B}_k^T\mathbf{B}_k$ can be interpreted as the matrix of the composite of the boundary operator $\mathcal{C}_k(\mathcal{X}) \to \mathcal{C}_{k-1}(\mathcal{X})$ with the adjoint of the coboundary operator $\mathcal{C}^{k-1}(\mathcal{X}) \to \mathcal{C}^k(\mathcal{X})$ through the isomorphism $\mathcal{C}_k(\mathcal{X}) \cong \mathbb{R}^{|\mathcal{X}_k|}$ determined by the given order of the set $\mathcal{X}_k$ of k-cells of $\mathcal{X}$. The upper Laplacian is the composite of the adjoint of the k-coboundary with the $(k+1)$-boundary.

Two distinct k-cells are upper adjacent if there exists at least one $(k+1)$-cell incident to both. We denote upper adjacency by $\sigma_i \sim_U \sigma_j$. Similarly, two distinct k-cells are lower adjacent if they share at least one common incident $(k-1)$-cell. We denote lower adjacency by $\sigma_i \sim_L \sigma_j$ [34]. The upper and lower adjacency relations are stored using *adjacency matrices* $\mathbf{A}_k^{\text{up}}$ and $\mathbf{A}_k^{\text{down}}$. These are square matrices of size $|\mathcal{X}_k| = \dim \mathcal{C}^k(\mathcal{X})$, which are related to non-signed incidence matrices as follows: $\mathbf{A}_k^{\text{up}}$ is obtained from $\mathbf{I}_{k+1}\mathbf{I}_{k+1}^T$ by replacing its diagonal entries with zeros, and $\mathbf{A}_k^{\text{down}}$ is obtained from $\mathbf{I}_k^T\mathbf{I}_k$ analogously.

Details on LapPE for graphs and their HodgeLapPE extension. A popular positional encoding for graph transformers is graph Laplacian eigenvectors (LapPE) [32]. Let $0 \leq \tilde{\lambda}_1 \leq \cdots \leq \tilde{\lambda}_{\tilde{p}} \leq 2$ be the distinct eigenvalues of the normalized graph Laplacian $\tilde{\mathbf{L}}_0$ for a graph $G = (V, E)$ with $V = \{v_1, \ldots, v_n\}$. As $\tilde{\mathbf{L}}_0$ is a real symmetric matrix,

$$\tilde{\lambda}_i = \min_{\substack{\|x\|=1 \\ x \in \langle \tilde{V}_{i-1} \rangle^\perp}} x^T\tilde{\mathbf{L}}_0 x = \min_{\substack{\|x\|=1 \\ x \in \langle \tilde{V}_{i-1} \rangle^\perp}} \|\mathbf{B}_1^T(D^+)^{1/2}x\|^2 = \min_{\substack{\|x\|=1 \\ x \in \langle \tilde{V}_{i-1} \rangle^\perp}} \sum_{(v_i,v_j)\in E} \left(\frac{x_i}{\sqrt{d(v_i)}} - \frac{x_j}{\sqrt{d(v_j)}} \right)^2, \tag{5}$$

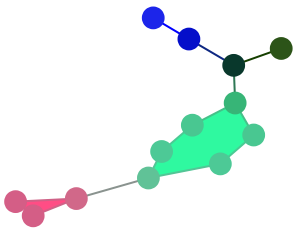


Figure 4: BSPe positional encoding of length three for a cell complex with two 2-cells. To generate a colour from the positional encoding, we normalize each coordinate of the positional encodings to the $[0, 1]$ range, generating normalized RGB colours. Note that close cells are assigned similar colours.

where $d(v)$ is the degree of v and $\tilde{V}_{i-1}$ is a set of linearly independent eigenvectors of $\tilde{\lambda}_1, \ldots, \tilde{\lambda}_{i-1}$. The minimum attained for any unit-norm eigenvector of $\tilde{\lambda}_i$ is unique up to a sign for non-degenerate eigenvalues. Eigenvectors corresponding to small eigenvalues give a *gradient* on the graph. The *close* vertices (with respect to the adjacency) have close eigenvector coordinates, thus, represent a *relative* position encoding of the vertices of the graph. Thus, LapPE assigns to each vertex v_i a vector $\text{LapPE}(v_i) = (e_i^1, \ldots, e_i^k)$, where $\{e_i^j \mid j = 1, \ldots, k\}$ are eigenvectors of the k smallest eigenvalues counting multiplicities, where k is a hyperparameter. To avoid the sign ambiguity for normalized eigenvectors of non-degenerate eigenvalues, positional encodings are multiplied by random signs at each iteration during training, with the objective of making the neural network invariant to this ambiguity.

HodgeLapPE is the extension of LapPE to cells of arbitrary rank using the unnormalized Hodge Laplacian. However, without normalizing the Hodge Laplacian, we can obtain eigenvalues that are not small enough to make the previous Rayleigh quotient (5) meaningful even in the case of simple simplicial complexes. With the unnormalized version we also do not obtain straightforward formulae that require close cells to have close coordinates in the eigenvector. The following examples illustrate the previous drawbacks.

Example B.1 (Large eigenvalues extending LapPE to simplicial complexes using the unnormalized Hodge Laplacian)**.** Take a triangle graph G with vertices v_1, v_2, v_3. The eigenvalues of the unnormalized Hodge Laplacian $\mathbf{L}_1$ given the orientation induced by the ordering of the vertices are 0 and 3. For $\lambda = 3$, an eigenvector of unit length is $(1/\sqrt{2})\,(1, 0, 1)$, that assigns the same coordinates to two of the edges and a different coordinate to the other one, although the three edges are clearly neighborhoods and must have similar representations, making eigenvectors useless in this case.

Example B.2 (Rayleigh quotient of the Hodge Laplacian does not produce a gradient of arbitrary dimensional cells)**.** If we want to produce positional encodings for k-cells using the k-th Hodge Laplacian, we obtain

$$\begin{aligned}\lambda_i = \min_{\substack{\|x\|=1 \\ x\in\langle S_{i-1}\rangle^\perp}} x^T\mathbf{L_k}x = \min_{\substack{\|x\|=1 \\ x\in\langle S_{i-1}\rangle^\perp}} \|\mathbf{B}_k x\|^2 + \|\mathbf{B}_{k+1}^T x\|^2 \\ = \min_{\substack{\|x\|=1 \\ x\in\langle S_{i-1}\rangle^\perp}} (1-\mathbb{1}(k=0)) \sum_{\gamma\in S_{i-1}} \left(\sum_{\substack{\sigma_j\in S_i \\ \gamma<\sigma_j}} (s(\gamma,\sigma_j)x_j)\right)^2 \\ + (1-\mathbb{1}(d=\dim(\mathcal{X}))) \sum_{\gamma\in S_{i+1}} \left(\sum_{\substack{\sigma_j\in S_i \\ \sigma_j<\gamma}} (s(\sigma_j,\gamma)x_j)\right)^2,\end{aligned} \tag{6}$$

where $\gamma < \sigma$ means that γ is a proper face of σ and $s(\gamma,\sigma)$ is the value of γ in the boundary of σ. In this case, taking the previous triangle graph G with three vertices, the eigenvalue $\lambda = 0$ has a unit eigenvector $(1/\sqrt{3})\,(-1,-1,1)$, that assigns to two of the edges the same coordinates and to the third edge the opposite coordinate, though being the three of them neighbors since they are all adjacent. We refer to [4] for more information about Equation (6) in the context of Topological Signal Processing.

C Positional encoding details

In this section, we extend the details about the positional encodings built in section 4.4.

C.1 Random walks on cell complexes

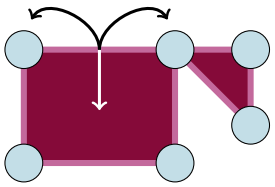
Let $\mathcal{X}$ be a regular cell complex. We describe a random walk on the set of k-cells of $\mathcal{X}$. To this end, we first recall that the number of upper and lower adjacent k-cells of a given cell $\sigma \in \mathcal{X}_k$ are named respectively the $(0,k+1)$-upper and $(0,k-1)$-lower degree of σ [54],

$$\deg_U^{0,k+1}(\sigma) = \#\{\sigma' \in \mathcal{X}_k \colon \sigma \sim_U \sigma'\}; \qquad \deg_L^{0,k-1}(\sigma) = \#\{\sigma' \in \mathcal{X}_k \colon \sigma \sim_L \sigma'\}.$$

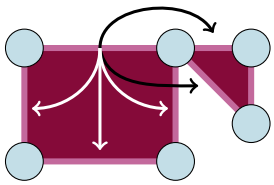
On the one hand, for each $k \geq 0$, we define a random upper k-walk based on upper adjacencies of the k-cells of $\mathcal{X}$. At each step, we move from a k-cell σ_i to any upper adjacent k-cell σ_j with probability proportional to the number of $(k+1)$-cells in common. To describe this process, we consider a weighted undirected graph G_k^{up}, whose vertices are the k-cells of $\mathcal{X}$ and the weight of each edge (σ_i,σ_j) is the number of $(k+1)$-cells whose closure contains both cells (if a k-cell is not upper adjacent to any k-cell, we draw a loop on the corresponding vertex with weight equal to 1). Thus, the upper random k-walk is described by the left stochastic matrix $\mathbf{RW}_k^{\text{up}} = \mathbf{wA}_k^{\text{up}}(\mathbf{D}_k^{\text{up}})^{-1}$, where $\mathbf{wA}_k^{\text{up}}$ and $\mathbf{D}_k^{\text{up}}$ denote the weighted adjacency and diagonal weighted degree matrices of the graph G_k^{up}.

On the other hand, for each $k > 0$, we define a random lower k-walk through lower adjacencies of the k-cells of $\mathcal{X}$. In this case, we move from a k-cell σ_i to any lower adjacent k-cell σ_j with probability proportional to the number of $(k-1)$-faces in common. As in the previous case, the random lower walk can be described as a random walk on a weighted graph G_k^{down}, whose vertices are the k-cells of $\mathcal{X}$ and the weight of an edge (σ_i,σ_j) is set as the number of $(k-1)$-cells that both cells have in common (as before, if a k-cell is not lower adjacent to any other k-cell, then we draw a loop on it with weight equal to 1). The lower random k-walk is described by the left stochastic matrix $\mathbf{RW}_k^{\text{down}} = \mathbf{wA}_k^{\text{down}}(\mathbf{D}_k^{\text{down}})^{-1}$, where $\mathbf{wA}_k^{\text{down}}$ and $\mathbf{D}_k^{\text{down}}$ denote the corresponding weighted adjacency and diagonal weighted degree matrices of the graph G_k^{down}. The matrices $\mathbf{wA}_k^{\text{up}}$ and $\mathbf{wA}_k^{\text{down}}$ correspond respectively to the upper and lower adjacency matrices $\mathbf{A}_k^{\text{up}}$ and $\mathbf{A}_k^{\text{down}}$ with the diagonal entries in null rows replaced with 1.

We can combine both processes to obtain a random walk in which information flows through upper and lower adjacencies, in line with [92]. The idea is as follows: if we are in a k-cell σ with upper and lower adjacent k-cells, we take a step with equal probability via either upper or lower connections. If σ has upper adjacent k-cells but not lower ones, we move following the random upper k-walk process, and vice versa. Lastly, if σ has neither upper nor lower connections, then we do not move.



(a) RWBSPe random walk possible transitions from the upper-left edge.



(b) RWPe random walk possible transitions from the upper-left edge.

Figure 5: Differences between RWBSPe and RWPe random walks. RWBSPe random walks can jump from a cell to all its incident and coincident cells, while RWPe random walks can jump from a cell to all its upper and lower adjacent cells.

The left stochastic matrix that describes the random k-walk is defined for $\sigma_i,\ \sigma_j \in \mathcal{X}_k$ by

$$(\mathbf{RW}_k)_{\sigma_i\sigma_j} = \begin{cases} \frac{1}{2}(\mathbf{RW}_k^{\text{up}})_{\sigma_i\sigma_j} + \frac{1}{2}(\mathbf{RW}_k^{\text{down}})_{\sigma_i\sigma_j} & \text{if } \deg_U^{0,k+1}(\sigma_j) \neq 0 \text{ and } \deg_L^{0,k-1}(\sigma_j) \neq 0 \\ (\mathbf{RW}_k^{\text{up}})_{\sigma_i\sigma_j} & \text{if } \deg_U^{0,k+1}(\sigma_j) \neq 0 \text{ and } \deg_L^{0,k-1}(\sigma_j) = 0 \\ (\mathbf{RW}_k^{\text{down}})_{\sigma_i\sigma_j} & \text{if } \deg_U^{0,k+1}(\sigma_j) = 0 \text{ and } \deg_L^{0,k-1}(\sigma_j) \neq 0 \\ \mathbb{1}(i=j) & \text{if } \deg_U^{0,k+1}(\sigma_j) = \deg_L^{(0,k-1)}(\sigma_j) = 0. \end{cases}$$

An example of the differences between transitions from an edge in the random walks described in this section and the barycentric subdivision random walks of RWBSPe are described in fig. 5.

C.2 Topological Slepians

Let $\mathcal{X}$ be an oriented regular two-dimensional cell complex, with Hodge Laplacians $\mathbf{L}_1$ and $\mathbf{L}_2$. Hodge Laplacians admit a Hodge decomposition [70], such that the k-cochain space can be decomposed as

$$\mathcal{C}^k(\mathcal{X},\mathbb{R}) = \text{im}\big(\mathbf{B}_k^T\big) \oplus \text{im}\big(\mathbf{B}_{k+1}\big) \oplus \ker\big(\mathbf{L}_k\big), \tag{7}$$

where $\bigoplus$ denotes direct sum of vector spaces, and $\ker(-)$ and $\text{im}(-)$ are the kernel and image spaces of a matrix, respectively. The k-cochains can be represented by means of eigenvector bases of the corresponding Hodge Laplacian. Using the decomposition $\mathbf{L}_k = \mathbf{U}_k\mathbf{\Lambda}_k\mathbf{U}_k^T$, the k-th Cellular Fourier Transform (k-CWFT) is the projection of a k-cochain onto the eigenvectors of $\mathbf{L}_k$ [89]:

$$\widehat{\mathbf{X}}_k = \mathbf{U}_k^T\,\mathbf{X}_k. \tag{8}$$

We refer to the eigenvalue set $\mathcal{B}_k$ of the k-CWFT as the frequency domain. An immediate consequence of the Hodge decomposition in (7) is that the eigenvectors belonging to $\text{im}(\mathbf{L}_k^d)$ are orthogonal to those belonging to $\text{im}(\mathbf{L}_k^u)$, for all $k = 1,\dots,K-1$. Therefore, the eigenvectors of $\mathbf{L}_k$ are given by the union of the eigenvectors of $\mathbf{L}_k^u$, the eigenvectors of $\mathbf{L}_k^d$, and the kernel of $\mathbf{L}_k$. We now introduce two localization operators acting onto a k-cell concentration set (thus, onto the topological domain), say $\mathcal{S}_k \subset \mathcal{X}^k$, and onto a spectral concentration set (thus, onto the frequency domain), say $\mathcal{F}_k \subset \mathcal{B}_k$, respectively. In particular, we define a cell-limiting operator onto the k-cell set $\mathcal{S}_k$ as

$$\mathbf{C}_{\mathcal{S}_k} = \text{diag}(\mathbf{1}_{\mathcal{S}}) \in \mathbb{R}^{|\mathcal{X}_k|\times|\mathcal{X}_k|}, \tag{9}$$

where $\mathbf{1}_{\mathcal{S}_k} \in \mathbb{R}^{|\mathcal{X}_k|}$ is a vector having ones in the index positions specified in $\mathcal{S}_k$, and zero otherwise; and $\text{diag}(\mathbf{z})$ denotes a diagonal matrix having $\mathbf{z}$ on the diagonal. A k-cochain $\mathbf{X}_k$ is perfectly localized onto the set $\mathcal{S}_k$ if $\mathbf{C}_{\mathcal{S}_k}\mathbf{X}_k = \mathbf{X}_k$. Similarly, the frequency limiting operator is defined as

$$\mathbf{B}_{\mathcal{F}_k} = \mathbf{U}\,\text{diag}(\mathbf{1}_{\mathcal{F}_k})\,\mathbf{U}^T \in \mathbb{R}^{|\mathcal{X}_k|\times|\mathcal{X}_k|}, \tag{10}$$

that can be interpreted as a band-pass filter over the frequency set $\mathcal{F}_k$. A k-cochain is perfectly localized over the bandwidth $\mathcal{F}_k$ if $\mathbf{B}_{\mathcal{F}_k}\mathbf{X}_k = \mathbf{X}_k$. The matrices in (9) and (10) are proper projection operators.

At this point, *k-topological Slepians* are defined as orthonormal vectors that are maximally concentrated over the k-cell set $\mathcal{S}_k$, and perfectly localized onto the bandwidth $\mathcal{F}_k$:

$$\begin{aligned} s_k^i = &\arg\max_{s_k^i} \|\mathbf{C}_{\mathcal{S}_k} s_k^i\|_2^2 \\ &\text{subject to } \|s_k^i\| = 1, \quad \mathbf{B}_{\mathcal{F}_k} s_k^i = s_k^i, \\ &\qquad \langle s_k^i, s_k^j\rangle = 0, \quad j = 1,\dots,i-1, \text{if } i > 1, \end{aligned} \tag{11}$$

for $i = 1, \ldots, |\mathcal{X}_k|$. As shown in [7], the solution of problem (11) is given by the eigenvectors of the matrix operator $\mathbf{B}_{\mathcal{F}}\mathbf{C}_{\mathcal{F}}\mathbf{B}_{\mathcal{F}_k}$, i.e.,

$$\mathbf{B}_{\mathcal{F}_k}\mathbf{C}_{\mathcal{S}_k}\mathbf{B}_{\mathcal{F}_k}s_k^i = \lambda_k^i s_k^i. \tag{12}$$

It is then clear that the maximum number of k-topological Slepians per each pair of concentration sets $\{\mathcal{S}_k, \mathcal{F}_k\}$ is given by rank$\{\mathbf{B}_{\mathcal{F}_k}\mathbf{C}_{\mathcal{S}_k}\mathbf{B}_{\mathcal{F}_k}\}$. For this reason and to have a more exhaustive representation of structural and topological properties of the complex [7], we choose a sequence of M concentration sets $\{\mathcal{S}_{k,i}, \mathcal{F}_{k,i}\}_{i=1}^M$ and concatenate the corresponding Slepians. From (7), the frequency domain can be partitioned into two separate sets: (i) the set of eigenvalues of $\mathbf{L}_k^{\text{down}}$, say $\mathcal{F}_k^d$, and (ii) the set of eigenvalues of $\mathbf{L}_k^{\text{up}}$, say $\mathcal{F}_k^u$. For the same reason, we can define two distinct sequences of (not necessarily disjoint) k-cell concentration sets: (i) K_k^d sets based on lower adjacency (encoded by $\mathbf{L}_k^{\text{down}}$), that we refer to as *lower sets*; (ii) K_k^u sets based on upper adjacency (encoded by $\mathbf{L}_k^{\text{up}}$), that we refer to as *upper sets*. It is then natural to associate the frequency concentration set $\mathcal{F}_k^u$ to each upper k-cell concentration set, and the frequency concentration set $\mathcal{F}_k^d$ to each lower k-cell concentration set. Finally, suppose that the kernel of the Laplacian $\mathbf{L}_k$ is not empty. In that case, topological Slepians can be combined with the harmonic eigenvectors of $\mathbf{L}_k$, i.e. the eigenvectors associated with the zero eigenvalues, resulting in a mixed positional encoding strategy. The number of topological Slepians can then be controlled either by tuning K_k^d and K_k^u, or by taking just the top Slepians per each pair of concentration sets. In this work, we choose the upper and lower k-cell concentration sets as the adjacency and coadjacency of each k-cell including the k-cell itself, respectively, obtaining $|\mathcal{X}_k|$ pairs of concentration sets for each rank k. In this way, we are also sure that the obtained set of topological Slepians comes with theoretical guarantees, i.e., it is an (A, B)-frame [7, 86] In general, the choice of the localization sets is an interesting directions, that could be easily combined with prior knowledge about the complex and the data at hand.

D Numerical results

The set of results of our experiments for all the combinations of attention mechanisms and positional encodings is reported in table 4.

E Implementation and hardware resources

Implementation was performed mainly using the `TopoNetX` [78] library for cell complex representation and manipulation, `PyTorch` [82] for the deep learning pipelines, `PyTorch Geometric` [37] for feature pooling and dataset loading, `Deep Graph Library` [99] and `Scipy` [96] for sparse tensor algebraic operations and sparse tensor representation and manipulation, `NetworkX` [45] for graph manipulation, and `PyTorch Lightning` [35] as a top layer for experimentation in `PyTorch`. The most critical pieces of software implemented in this project have been the `DataLoader` and the `collate` function to batch cell complexes. The `DataLoader` is implemented in the class `TopologicalTransformerDataLoader`. The collate function is implemented in the function `collate`, both inside the file `src/datasets/cell_dataloader.py`. The `collate` function creates a cell complex batch by performing the disjoint union of cell complexes. As an input, the `collate` function receives an object with the signals for each cell, the neighborhood matrices used in the transformer architecture as bias $\mathbf{N}$ in a sparse format, and other data needed by the experiments such as the label of the dataset and the positional encodings. The neighborhood matrices are batched into a new sparse block matrix, taking into account that different cell complexes may have different dimensions and thus not all the cell complexes have the same neighborhood matrices. Currently, the `collate` function supports adjacency and boundary matrices, although the function can be extended easily. Signals, positional encodings, and labels are simply concatenated. To keep track of which signals and positional encodings correspond to each of the individual cell complex, we also return, for each dimension, a vector of size equal to total number of cells of that dimension in the disjoint union which indicates to which cell complex belong each signal or positional encoding.

The experiments were executed on a server with an AMD EPYC 7452 (128) @ 2.350GHz CPU, 503GiB of RAM memory, x4 PNY Nvidia RTX 6000 Ada Generation 48GB GPUs, and Ubuntu 22.04.4 LTS with the 6.5.0-28-generic Linux kernel. Each experiment was executed on a separated GPU device, using 12 workers per experiment.

Table 4: Results of experiments with cellular transformers in the three datasets GCB, ogbg-molhiv, and ZINC. Reported scores are test accuracy (↑), test AUC-ROC (↑), and test MAE (↓), respectively. Best result, second best result, and third best result are highlighted in boldface green, blue, and orange, respectively.

Datasets →		GCB	ogbg-molhiv	ZINC
Attention type	Positional encodings	Accuracy (↑)	AUC-ROC (↑)	MAE (↓)
$\mathcal{A}^s_g$	BSPe	**0.7516 ± 0.0102**	0.7681	0.0840
$\mathcal{A}^s_{k_s \to k_t}$	BSPe	0.7400 ± 0.0123	**0.7946**	0.0934
$\mathcal{A}^d_{k_s \to k_t}$	RWBSPe	0.6295 ± 0.0263	0.7136	0.3451
$\mathcal{A}^s_g$	zeros	0.7453 ± 0.0086	0.7362	0.1239
$\mathcal{A}^c_{k_s \to k_t}$	HodgeLapPE	0.7337 ± 0.0103	0.7658	0.0831
$\mathcal{A}^d_{k_s \to k_t}$	HodgeLapPE	0.6179 ± 0.0286	0.7670	0.4009
$\mathcal{A}^s_{k_s \to k_t}$	zeros	0.7453 ± 0.0123	0.7008	0.1416
$\mathcal{A}^s_{k_s \to k_t}$	HodgeLapPE	0.7421 ± 0.0191	0.7586	0.0852
$\mathcal{A}^d_g$	BSPe	0.5989 ± 0.0219	0.7177	0.3983
$\mathcal{A}^s_g$	RWBSPe	0.7442 ± 0.0140	0.7082	0.1296
$\mathcal{A}^c_{k_s \to k_t}$	zeros	0.7484 ± 0.0061	0.7435	0.1090
$\mathcal{A}^d_{k_s \to k_t}$	BSPe	0.6095 ± 0.0234	0.7328	0.4096
$\mathcal{A}^d_{k_s \to k_t}$	zeros	0.6400 ± 0.0196	0.5952	0.4348
$\mathcal{A}^c_{k_s \to k_t}$	TopoSlepiansPE	0.7421 ± 0.0094	0.7565	0.1202
$\mathcal{A}^c_{k_s \to k_t}$	RWPe	0.7379 ± 0.0102	0.7338	0.0833
$\mathcal{A}^d_g$	RWBSPe	0.6200 ± 0.0423	0.6949	0.3586
$\mathcal{A}^d_g$	TopoSlepiansPE	0.6179 ± 0.0276	0.6954	0.5175
$\mathcal{A}^c_{k_s \to k_t}$	BSPe	0.7347 ± 0.0136	0.7784	0.0833
$\mathcal{A}^c_{k_s \to k_t}$	RWBSPe	0.7442 ± 0.0108	0.7321	**0.0802**
$\mathcal{A}^s_g$	HodgeLapPE	0.7432 ± 0.0112	0.7032	0.0824
$\mathcal{A}^s_g$	RWPe	0.7442 ± 0.0136	0.7343	0.1051
$\mathcal{A}^d_g$	zeros	0.6105 ± 0.0221	0.6679	0.4526
$\mathcal{A}^d_{k_s \to k_t}$	RWPe	0.6463 ± 0.0406	0.6944	0.3843
$\mathcal{A}^s_{k_s \to k_t}$	RWBSPe	0.7389 ± 0.0144	0.7058	0.1210
$\mathcal{A}^d_{k_s \to k_t}$	TopoSlepiansPE	0.6463 ± 0.0214	0.6690	0.4527
$\mathcal{A}^d_g$	HodgeLapPE	0.5811 ± 0.0250	0.7243	0.3981
$\mathcal{A}^s_{k_s \to k_t}$	TopoSlepiansPE	0.7505 ± 0.0054	0.7111	0.1452
$\mathcal{A}^s_g$	TopoSlepiansPE	0.7432 ± 0.0107	0.7192	0.1303
$\mathcal{A}^s_{k_s \to k_t}$	RWPe	0.7505 ± 0.0054	0.7288	0.0973
$\mathcal{A}^d_g$	RWPe	0.6368 ± 0.0282	0.6836	0.3770

F Licenses

The GCB dataset is distributed under a MIT license. ogbg-molhiv is distributed under a MIT license. The ZINC dataset is free to use and download and its license can be found at https://wiki.docking.org/index.php?title=UCSF_ZINC_License.