

# HodgeFormer: Transformers for Learnable Operators on Triangular Meshes through Data-Driven Hodge Matrices

Akis Nousias

K3Y Labs

anousias@k3y.bg

Stavros Nousias

Technical University of Munich

stavros.nousias@tum.de

## Abstract

*Currently, prominent Transformer architectures applied on graphs and meshes for shape analysis tasks employ traditional attention layers that heavily utilize spectral features requiring costly eigenvalue decomposition-based methods. To encode the mesh structure, these methods derive positional embeddings, that heavily rely on eigenvalue decomposition based operations, e.g. on the Laplacian matrix, or on heat-kernel signatures, which are then concatenated to the input features. This paper proposes a novel approach inspired by the explicit construction of the Hodge Laplacian operator in Discrete Exterior Calculus as a product of discrete Hodge operators and exterior derivatives, i.e. ( $L := \star_0^{-1} d_0^T \star_1 d_0$ ). We adjust the Transformer architecture in a novel deep learning layer that utilizes the multi-head attention mechanism to approximate Hodge matrices  $\star_0$ ,  $\star_1$  and  $\star_2$  and learn families of discrete operators  $L$  that act on mesh vertices, edges and faces. Our approach results in a computationally-efficient architecture that achieves comparable performance in mesh segmentation and classification tasks, through a direct learning framework, while eliminating the need for costly eigenvalue decomposition operations or complex preprocessing operations.*

## 1. Introduction

Data-driven algorithms are at the core of machine learning innovations in fields such as natural language processing, computer vision, and audio processing. Over the past decade, existing architectures and handcrafted features have been gradually replaced by machine learning models that are able to capture fine-grained data-driven structures and patterns, better leverage the underlying hardware, and can efficiently handle large-scale data.

In recent years, Transformer-based architectures have been at the forefront of breakthroughs in all of the aforementioned fields, due to their scalability in terms of data and hardware, their generality as an architecture, and their

versatility in accommodating different data modalities [14].

Traditional Transformers-based approaches for 3D data typically follow a graph-based application paradigm. Vanilla Transformers are used, usually with node features as input, while structural information is incorporated through positional embeddings, usually via eigen decomposition of a Laplacian-based matrix [19, 23, 38].

In this work, we introduce a novel Transformer architecture inspired by the explicit construction of the Laplacian operator in Discrete Exterior Calculus (DEC) adjusted to triangular 3-dimensional meshes. This kind of connection is not novel by itself and was the main idea of HodgeNet by Smirnov and Solomon [34], where input features were used for learning Hodge Star operators as diagonal matrices. These matrices were then used for constructing Laplacian operators, which were then applied to input features in a spectral learning kind of approach. Specifically, we draw connections between discrete Hodge Star operators and Transformer-style attention mechanisms. While most common constructions of discrete Hodge Star operators are limited to diagonal matrices, these represent only some specific Hodge operator realizations. In fact, the Galerkin method, widely used in Finite Element Methods (FEM) literature [35], prescribes systematic ways to discretize various differential operators, including the Hodge operator. The key idea is to project onto carefully designed basis functions (e.g., piecewise linear or higher order elements) and account for the overlap of these basis functions. These overlaps are used to design the mass matrix, which acts as a discrete Hodge operator. Unlike the diagonal Hodge operator, which considers only self-contributions, the Galerkin Hodge operator [2, 22, 27] incorporates contributions from neighboring elements, resulting in a sparse, non-diagonal matrix.

There are several key insights that allow drawing direct connections with the Transformer-style attention. Specifically, the projection with matrices  $W_Q$ ,  $W_K$ ,  $W_V$  onto a learned space corresponds to a basis projection as in the Galerkin method, and the self-attention mechanism  $Q \cdot K^T$  encodes pairwise interactions and accounts for overlaps of the basis functions. If the self-attention mechanism is lo-

calized, then it aligns with Galerkin-style discretizations [3]. The proposed architecture includes a novel Transformer-inspired layer that enables information propagation directly on the manifold through attention-based learnable Hodge Star matrices and incidence matrices, which serve as discrete exterior derivatives. By incorporating the structure considerations into the architecture, we do not rely at all on eigen-decomposition based methods, spectral features, or preprocessing steps, while achieving competitive performance compared to state-of-the-art approaches on meshes.

The rest of the paper is organized as follows: Sec. 2 presents the related work. Sec. 3 provides preliminaries on aspects relevant to the proposed approach. Sec. 4 presents the methodology, and 5 presents the experimental results. Finally, Sec. 6 discusses the outcomes of this work and concludes this paper.

## 2. Related Work

**Mesh convolutions.** Recent advances in geometric deep learning have adapted convolution operations for sampled 3D manifolds by exploiting relationships between mesh elements, *i.e.* vertices, edges and faces, to construct convolutional filters. MeshCNN, introduced by Hanocka et al. [11], proposed edge-based mesh convolution, mesh pooling and mesh unpooling operations. However, the method was limited by the fixed topology requirement and potential loss of geometric information during pooling. DualConvMesh-Net [32] attempted to bridge local and global features through geodesic and Euclidean convolutions, but faces computational challenges with large meshes. Another category of methods utilizes subdivision schemes to enable convolution operations on meshes. SubdivNet [7] and Subdivision-based Mesh Convolutional Networks [12] enforce regular structure through remeshing or subdivision techniques. The latter defines convolution operations directly on 3D triangle meshes via subdivision structure, treating mesh faces as fundamental convolution units analogous to 2D pixels. However, these subdivision-driven approaches suffer from several limitations, potentially introducing geometric artifacts, failing to preserve fine local features, requiring specific connectivity patterns, such as loop subdivision sequences, and incurring high computational costs from remeshing operations.

**Spectral methods** are based on the eigen-decomposition of discrete operators, typically the Laplace-Beltrami operator, utilizing the translation, rotation, and scale invariance of operations in the spectral domain. HodgeNet [34] introduces a learnable class of sparse operators on meshes, built from standard constructions in discrete exterior calculus. It employs standard geometry processing operations and Hodge star operators ( $\star_0$  and  $\star_1$ ) to encode per-vertex and per-edge features. Spectral approaches by [8] leverage the graph Laplacian for convolution, with subsequent works like CayleyNets [17] improving spectral filtering design. How-

ever, these methods struggle with computational complexity due to the high computational cost of eigen-decomposition. To avoid this limitation, Surface Networks [15] leverage Graph Neural Networks (GNNs) to learn polynomial expansions of both intrinsic (Laplacian) and extrinsic (Dirac) operators to capture mesh geometry, enabling them to process irregular mesh structures directly. Shape descriptors built on the Laplace-Beltrami operator have been particularly successful in capturing geometric properties. The Heat Kernel Signature (HKS) [36] captures multiscale geometric information through the heat diffusion process, providing an isometry-invariant point descriptor. The Wave Kernel Signature (WKS) [1] offers an alternative based on quantum mechanical principles, achieving better feature localization. Building on these descriptors, several neural architectures have emerged. DiffusionNet [33] leverages diffusion-based features to achieve robust segmentation and classification on partial and non-isometric shapes. Geodesic CNNs [25] introduce geometry-adapted convolutional and pooling layers that operate in the spectral domain, utilizing HKS and WKS features as input channels for learning intrinsic shape representations. LaplacianNet [29] introduces a deep learning framework for 3D meshes encoding mesh connectivity using Laplacian spectral analysis. The network uses Mesh Pooling Blocks (MPBs) to split the mesh surface into local pooling patches and aggregate both local and global features. A mesh hierarchy is built from fine to coarse using Laplacian spectral clustering, making the method robust to different triangulations and isometric transformations. The method is dependent on eigen-decomposition and spectral clustering.

**Graph Learning-based E(n)-Equivariant Graph Neural Networks (EGNN)** [31] efficiently extend message passing to geometric graphs by maintaining scalar invariant features and equivariant coordinates for each node, using relative distances to preserve rotations, translations, and reflections. Equivariant Mesh Neural Networks (EMNN)[37] build on this by adding surface-aware messages that leverage mesh face information. EMNN further incorporates multiple vector channels and hierarchical pooling to capture long-range dependencies while maintaining E(3)-equivariance.

**Transformer adaptations** display potential in capturing global relationships in 3D meshes. The Laplacian Mesh Transformer [18] applies Transformer layers to Laplacian eigenvectors and geometric features. MeshMAE [21] introduces self-supervised learning through masked autoencoders for mesh understanding. MeshFormer [24] addresses scalability by combining mesh simplification with hierarchical graph-based attention mechanisms. Mesh Transformer (MeT) [38] proposes a two-stream architecture that processes triangle and cluster features in parallel, using Laplacian eigenvectors for positional encoding. Despite their effectiveness, these Transformer-based methods face significant computational challenges. The eigen-decomposition required

for Laplacian features and the quadratic complexity of attention mechanisms limit their applicability to large-scale meshes, while the need for complex architectural components increases implementation and training difficulty.

### 3. Preliminaries

#### 3.1. Triangular meshes & Discrete Operators

In this work, we focus on triangular meshes  $\mathcal{M}(\mathcal{V}, \mathcal{E}, \mathcal{F})$  with vertices  $n_v = |\mathcal{V}|$ , edges  $n_e = |\mathcal{E}|$ , and faces  $n_f = |\mathcal{F}|$ . In the context of Discrete Exterior Calculus (DEC) [9], a triangular mesh is seen as a 2-dimensional oriented simplicial complex. It is composed of oriented  $k$ -simplices for  $k = \{0, 1, 2\}$  corresponding respectively to vertices, edges, and faces. On each set of  $k$ -simplices, we consider discrete differential  $k$ -forms  $\Omega^k$ , that is, functions that assign values to the elements of each set. In addition, we employ two important constructs:

- i The *discrete exterior derivative*  $d_k$ , along with its adjoint  $d_k^T$ , that is, the linear map  $d_k : \Omega^k \rightarrow \Omega^{k+1}$  mapping discrete  $k$ -forms to discrete  $k+1$  forms.
- ii The *discrete Hodge Star*, which is the discrete analog of the Hodge Star operator  $\star_k : \Omega^k \rightarrow \tilde{\Omega}^{n-k}$  that maps  $k$ -forms  $\Omega^k$  to their dual  $(n-k)$  forms  $\tilde{\Omega}^{n-k}$ .

The discrete exterior derivative and the discrete Hodge Star operator encode different aspects of the mesh. The exterior derivative captures the orientation and connectivity between mesh elements, translating how vertices, edges, and faces relate, while the Hodge Star operator encodes metric information such as angles, lengths, areas, and volumes.

A key feature of the discrete Hodge Star, arising from the nature of the discretization, is that it does not need to act on forms defined on the same mesh. Instead, it naturally relates quantities on two distinct and possibly unrelated meshes: the *primal mesh* (the original simplicial complex) and the *dual* (or secondary) mesh. This decoupling offers a wide range of choices for the dual mesh, and indeed, the literature presents many dual mesh discretizations, such as the circumcentric dual, the barycentric dual, and others that lead to different Hodge Star instantiations. For a triangular mesh, the corresponding sets of primal and dual elements share the same dimensionality, although this correspondence does not hold for other simplicial complexes.

All the above constructs can be naturally represented as matrices, as shown in Tab. 1. The matrix representations of the discrete exterior derivative operators  $d_0 \in \{-1, 0, 1\}^{n_e \times n_v}$  and  $d_1 \in \{-1, 0, 1\}^{n_f \times n_e}$  are simply the *signed incidence matrices* of the input meshes. The matrix representations of the Hodge Star operators, as mentioned above, depend on the discretization of the dual mesh.

By composing discrete operators, we can construct complex objects such as the curl operator  $(\star_2 \cdot d_1)$ , the divergence operator  $(\star_0^{-1} d_0^T \star_1)$ , or the Laplacian operator on vertices

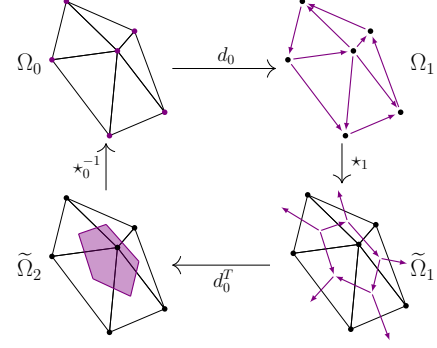


Figure 1. The Hodge Laplacian applied on a 0-form, i.e. values on vertices, as a sequence of discrete exterior derivative and Hodge Star operators.

Object	Matrix Representation
$\Omega^0$ (0-form)	$x_v \in \mathbb{R}^{n_v \times d_v}$
$\Omega^1$ (1-form)	$x_e \in \mathbb{R}^{n_e \times d_e}$
$\Omega^2$ (2-form)	$x_f \in \mathbb{R}^{n_f \times d_f}$
$d_0$ (vertices-to-edges)	$d_0 \in \{-1, 0, 1\}^{n_e \times n_v}$
$d_1$ (edges-to-faces)	$d_1 \in \{-1, 0, 1\}^{n_f \times n_e}$
$\star_0$ (Hodge Star on 0-forms)	$\star_0 \in \mathbb{R}^{n_v \times n_v}$
$\star_1$ (Hodge Star on 1-forms)	$\star_1 \in \mathbb{R}^{n_e \times n_e}$
$\star_2$ (Hodge Star on 2-forms)	$\star_2 \in \mathbb{R}^{n_f \times n_f}$

Table 1. Mapping of different objects to matrices

$L_v := \star_0^{-1} d_0^T \star_1 d_0$ , as shown in Figure 1, or the more general Hodge Laplacian on  $k$ -forms (also known as  $k$ -form Laplacian, or de Rham Laplacian):

$$L_k := d_{k-1} \cdot \star_{k-1}^{-1} \cdot d_{k-1}^T \cdot \star_k + \star_k^{-1} \cdot d_k^T \cdot \star_{k+1} \cdot d_k \quad (1)$$

#### 3.2. Transformers

A Transformer is a function  $T : \mathbb{R}^{n \times d_m} \rightarrow \mathbb{R}^{n \times d_m}$  where  $d_m$  is the embedding (model) dimension, defined by the composition of  $L$  Transformer layers  $T_1(\cdot), \dots, T_L(\cdot)$ :

$$T_l(x) = f_l(A_l(x) + x). \quad (2)$$

Here,  $f_l(\cdot)$  transforms each row of shape  $\mathbb{R}^{1 \times d_m}$  independently of the others and is usually implemented with a small two-layer feedforward network.  $A_l(\cdot)$  is the self-attention function and is the only component acting across sequence positions. The self-attention function  $A_l(\cdot)$  computes, for each position, a weighted sum of all positions in the sequence. Formally, for input  $x \in \mathbb{R}^{n \times d_m}$ :

$$Q = xW_Q, \quad K = xW_K, \quad V = xW_V \quad (3)$$

Typically  $W_Q, W_K, W_V \in \mathbb{R}^{d_m \times d_h}$  where  $d_h = d_m/h$ , with  $h$  the number of attention heads, and  $Q \in \mathbb{R}^{n \times d_h}$ ,

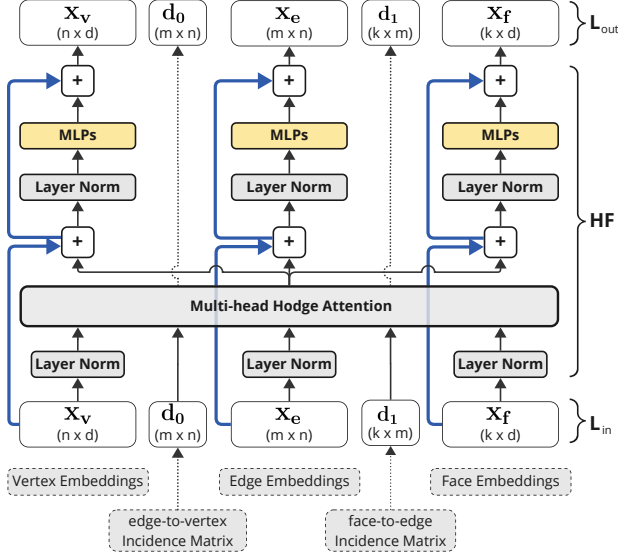


Figure 2. Overview of a HodgeFormer layer operating on vertex, edge and face features via the Multi-Head Hodge Attention mechanism. The layer can be configured so that different mesh elements or combinations of them will be updated.

$K \in \mathbb{R}^{n \times d_h}$ ,  $V \in \mathbb{R}^{n \times d_h}$ . The attention output is:

$$\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^\top}{\sqrt{d_h}} \right) V \quad (4)$$

and the output shape is  $\mathbb{R}^{n \times d_h}$ . For multi-head attention, this process is repeated in parallel for each head; the outputs are concatenated and projected back to  $d_m$ . The feedforward network is applied independently to each position (row) and is typically a two-layer MLP.

## 4. A Transformer Model with Hodge Attention

### 4.1. HodgeFormer Layer

We propose a novel deep learning architecture shown in Fig. 2, named ‘‘HodgeFormer’’, that leverages the multi-head attention mechanism for learning data-driven Hodge Laplacian operators on  $k$ -forms based on the parametrization of the Hodge Star  $\star_k$ . Let  $x_k \in \mathbb{R}^{n_k \times d_k}$  a  $k$ -form on mesh elements with  $k \in \{v, e, f\}$ , respectively, of size  $n_k$  and let  $W_{Q_k} \in \mathbb{R}^{d_k \times d_{h_k}}$ ,  $W_{K_k} \in \mathbb{R}^{d_k \times d_{h_k}}$ , and  $W_{V_k} \in \mathbb{R}^{d_k \times d_{h_k}}$  the learnable projection matrices transforming  $x_k$  to corresponding representations  $Q_k$ ,  $K_k$ , and  $V_k$  as in Eq. (4). For simplicity we set  $d_k = d$  for  $k \in \{v, e, f\}$ , and  $d_{h_k} = d_h = d/h$  for  $h$  attention heads. Then we can parametrize the Hodge Star operator  $\star_k$  as:

$$\star_k(x_k) = \sigma \left( \frac{Q_k K_k^\top}{\sqrt{d_h}} \right), \quad \text{for } k \in \{v, e, f\} \quad (5)$$

where  $\sigma(\cdot)$  is the row-wise softmax non-linearity. For the inverse Hodge Star operators, i.e.,  $\star_k^{-1}$ , we employ dedicated linear maps  $Q_k, K_k$ .

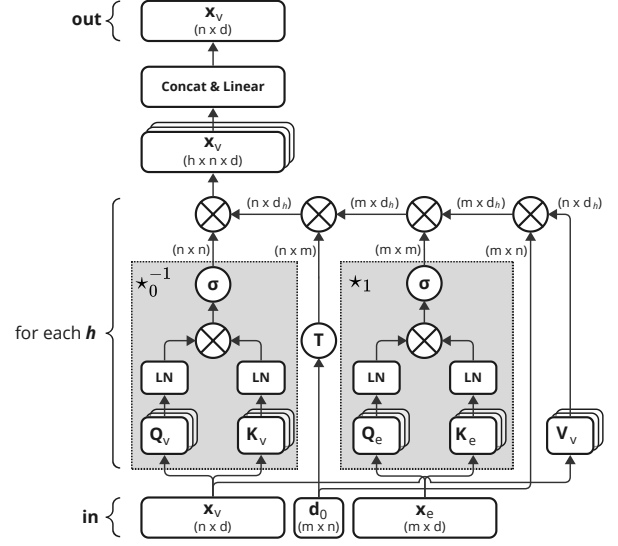


Figure 3. Multi-head Hodge Attention applied on latent vertex features  $x_v$ . The multi-head attention mechanism learns data-driven Hodge Star matrices  $\star_0^{-1}$  and  $\star_1$ . Layer Norm (LN) is applied on  $Q, K$  matrices to increase training stability.

We define the HodgeFormer Layer  $H_l$  as the function that operates and updates  $k$ -forms on vertices, edges and faces:

$$H_l(x_v, x_e, x_f) := G_l(A_{H_l}(x_v, x_e, x_f) + (x_v, x_e, x_f)) \quad (6)$$

where  $A_{H_l}(\cdot, \cdot, \cdot)$  is the Multi-head Hodge Attention operating on each  $k$ -form:

$$x_v, x_e, x_f := A_{H_l}(x_v, x_e, x_f) \quad (7)$$

$$x_v = A_v(x_v, x_e) = L_v(x_v, x_e) \cdot V_v(x_v) \quad (8)$$

$$x_e = A_e(x_v, x_e, x_f) = L_e(x_v, x_e, x_f) \cdot V_e(x_e) \quad (9)$$

$$x_f = A_f(x_e, x_f) = L_f(x_e, x_f) \cdot V_f(x_f) \quad (10)$$

and  $G_l(\cdot)$  a function that transforms the values of each  $k$ -form independently and is implemented with separate two-layer feedforward networks with residual connections.

From Eq. (1), the Hodge Laplacians  $L_v, L_e, L_f$  are defined for vertices (0-form), edges (1-form) and faces (2-form) as :

$$\begin{aligned} L_v &:= \star_0^{-1}(x_v) \cdot d_0^T \cdot \star_1(x_e) \cdot d_0 \\ L_e &:= d_0 \cdot \star_0^{-1}(x_v) \cdot d_0^T \cdot \star_1(x_e) + \\ &\quad \star_1^{-1}(x_e) \cdot d_1^T \cdot \star_2(x_f) \cdot d_1 \\ L_f &:= d_1 \cdot \star_1^{-1}(x_e) \cdot d_1^T \cdot \star_2(x_f) \end{aligned} \quad (11)$$

The HodgeFormer layer  $H_l$  extends the standard transformer layer to independently operate and update  $k$ -form features using multi-head attention via mesh-based Hodge Laplacians and Stars, closely matching the residual and feedforward update principles in canonical transformer networks.



	Acts on	Learns	Requires	Updates
$L_v$	$v$	$\star_0^{-1}, \star_1$	$x_v, x_e$	$x_v$
$L_e$	$e$	$\star_0^{-1}, \star_1^{-1}, \star_1 \star_2$	$x_v, x_e, x_f$	$x_e$
$L_f$	$f$	$\star_1^{-1}, \star_2$	$x_e, x_f$	$x_f$

Table 2. Overview of learnable operators  $L_v, L_e, L_f$ .

## 4.2. End-to-end Architecture

This section presents the proposed end-to-end architecture. As illustrated in Fig. 4, the architecture consists of (i) Embedding Layers, (ii) HodgeFormer and Vanilla Transformer layers, and (iii) a task-specific head.

The input consists of features extracted from the mesh, along with the sparse oriented incidence matrices  $d_0$  and  $d_1$ . Dedicated embedding layers, discussed in Sec. 4.5, operate separately on vertex, edge, and face features and map them to target latent dimensions  $d_v, d_e$ , and  $d_f$  (Sec. 4.5). Then, the latent embeddings for each mesh element are transformed by a mixture of sequentially placed HodgeFormer and Transformer layers and updated accordingly. The final embeddings are passed to the task head, which maps them to the task-specific output dimensions. Putting together HodgeFormer layers with vanilla Transformer layers can be thought of as combining local operators with global mixers. Different mixing strategies of HodgeFormer and vanilla Transformer layers can be applied.

The proposed end-to-end architecture is configurable to combine operations on different mesh elements, i.e., vertices, edges, and faces, where each learned operator acts upon and updates specific k-forms, as depicted in Tab. 2. In the case that specific mesh elements are not selected to be updated, the corresponding Transformer layers are omitted. The size of the architecture in terms of trainable parameters depends on the number and configuration of used layers and the combination of mesh elements selected to operate upon.

## 4.3. Sparse Attention

For the multi-head Hodge attention mechanism, we employ sparse attention by defining sparsity patterns based on local neighborhoods of mesh elements. Instead of computing the full attention matrix, this choice reduces complexity of the attention operator and aligns with the notion of the Hodge Star as a local operator.

Let  $x \in \mathbb{R}^{n \times d}$  a k-form on some mesh element, where  $n$  the number of k-form elements and  $d$  the feature dimension, and  $Q, K, V$  the linearly mapped queries, keys and values respectively. For each k-element  $i \in 1, \dots, n$ , we define a sparsity pattern  $S_i$ , which is the set of key positions that the query at position  $i$  can attend to. Then, the update of each k-form element feature vector  $x_i$  through the sparse attention mechanism can be formalized as

$$x_i = \sum_{j \in S_i} A_{ij} V_j \quad (12)$$

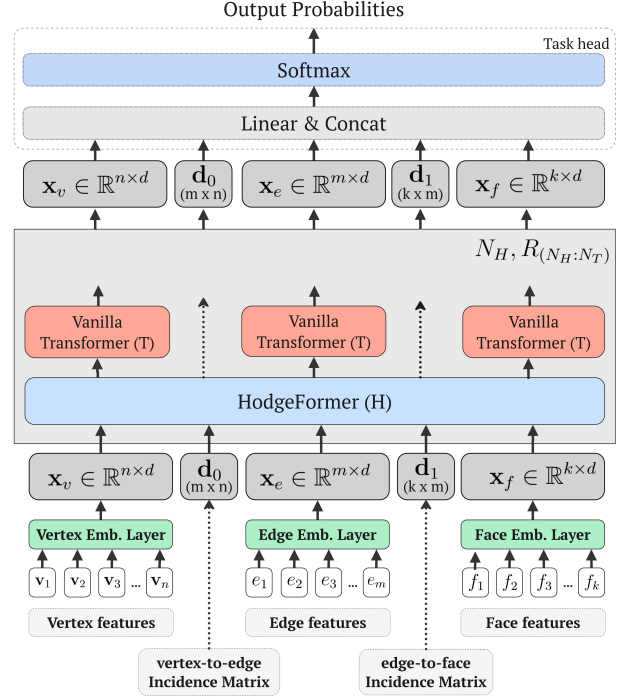


Figure 4. Overview of a deep learning architecture with combined HodgeFormer and vanilla Transformer layers operating on an input triangular mesh.

where,  $A_{ij}$  is the attention weight between elements  $i$  and  $j$ , calculated as  $A_{ij} = \text{softmax}(Q_i \cdot K_j^T) / \sqrt{d}$ .

To construct the sparsity pattern  $S_i$  for each k-form element  $i$ , we extract local neighborhoods by performing breadth-first search (BFS) on the adjacency structure of the corresponding mesh element. In addition to local connections, we introduce random connections to further enhance the connectivity of the attention graph. Following the approach of [30], we select  $\sqrt{n}$  neighbors for each mesh element, where  $n$  is the total number of elements in the mesh.

In practice, the operators  $L_v, L_e$ , and  $L_f$  from Eq. (11) (corresponding to vertex, edge, and face operators, respectively) are not explicitly materialized. Instead, each matrix component of these operators is applied sequentially to the input features, from right to left. This approach, combined with the sparsity pattern described above, results in an overall computational complexity of  $O(n^{1.5}d)$ .

## 4.4. Input Features and Positional Embeddings

HodgeFormer input tokens are directly formed from features on vertices, edges, and faces to enable the application of DEC-based formulations. In addition, positional information is directly incorporated in the architecture as xyz coordinates through the input features for each mesh element.

Specifically, we consider three types of input features: (i) coordinates, (ii) normals and (iii) areas: The **vertex features**

$x_{v_{in}}$  consist of the point 3D coordinates, vertex normals,  $n_{v_i}$  calculated as the weighted average of incident face normals  $n_{f_i}$  and the vertex associated cell area, calculated as the weighted average of incident face areas. The **edge features**  $x_{e_{in}}$  consist of the point 3D coordinates of edge vertices, and the vertices opposite to the respective edge, edge normals calculated as the average of incident vertex normals, and the lengths of the edges that belong to the incident faces. The **face features**  $x_{f_{in}}$  consist of the point 3D coordinates of the face vertices ordered according to the face’s orientation, the face normal and the face area. These design choices aim to incorporate information associated both to the primal and the dual mesh. The effect of each type of input features is showcased in the ablation study in Tab. 5.

## 4.5. Embedding Layer

The input features mentioned in Sec. 4.4,  $x_{v_{in}}$ ,  $x_{e_{in}}$  and  $x_{f_{in}}$  are transformed by an embedding layer into the corresponding matrices  $x_v$ ,  $x_e$  and  $x_f$ . The embedding layer (Eq. (13)), operates separately on each mesh element and involves two steps: (i) Feature aggregation over one-hop neighborhood defined by the mesh element’s adjacency structure and (ii) embedding to the target dimension via a feed-forward MLP.

$$x_k = MLP(x_{k_{in}} + A_k \cdot x_{k_{in}}), \quad k \in \{v, e, f\} \quad (13)$$

By performing a one-hop feature aggregation, we allow our network to take into account neighbors’ information before embedding the input features to the target dimension.

## 5. Experimental Evaluation

### 5.1. Evaluation Setup

The HodgeFormer architecture is implemented in PyTorch [28] with standard backpropagation. The BFS operations for local neighborhood extraction are based on sparse matrix operations using the *graphblas* framework [5], [6]. All experiments and measurements are performed on a single Nvidia RTX 4090 GPU with 24GB VRAM, paired with Intel i9-14900KF processors and 64GB of system RAM.

In all experiments, we use a common latent dimension  $d = 256$  across vertices, edges, and faces as well as in the attention mechanism for the  $W_Q$ ,  $W_K$ ,  $W_V$  mappings. The MLP hidden layers have a dimensionality of  $d_h = 512$ . For the multi-head attention mechanisms, we use  $h = 4$  heads throughout the architecture, with each head operating on  $d_k = d/h = 64$ -dimensional space.

HodgeFormer layers use sparse attention, where each input element attends over  $\sqrt{n}$  elements, consisting of local neighbors and random connections in an 4 : 1 ratio. Plain Transformer Layers follow a standard implementation except for the incorporation of a linear attention mechanism as in [13]. All MLP layers use *ReLU* non-linearities where required. We adopt the pre-LN architecture variant with

residual connections, placing Layer Normalization (Layer-Norm) before the input to each layer block. Following [30], we remove scale and bias terms from LayerNorm. Dropout is employed as a regularization mechanism both in the HodgeFormer and the Transformer layers. Finally, in all experiments we employ the neighbor-aware embedding layer as described in Sec. 4.5.

### 5.2. Experiments and Results

The HodgeFormer architecture achieves competitive results with state-of-the-art models in common tasks across multiple datasets without utilizing spectral features, eigenvalue decomposition operations, or complementary structures.

The architecture is validated with experiments on the tasks of mesh classification and mesh segmentation using benchmark datasets. For all experiments, input meshes are zero-centered and scaled to the unit sphere, during both training and testing. During training, we apply random rotations and small perturbations to vertex positions along mesh edges for data augmentations. All reported results are obtained using architectures of  $N = 6$  HodgeFormer layers mixed with Transformer layers, in either a 2 : 1 or 6 : 2 ratio. Optimization is performed via standard backpropagation using the Adam optimizer, with learning rates decayed using a cosine annealing strategy. The following hyperparameters were tuned to each problem: learning rate in the range  $[10^{-3}, 10^{-4}]$ , batch size in  $[4, 16]$  and number of training epochs in  $[200, 300]$ .

The task of mesh classification is evaluated on the SHREC-11[20] and Cube Engraving [11] datasets. SHREC-11 is a dataset of 600 meshes representing 30 categories of 20 shapes each. Following prior work, we train on randomly selected splits of 10 samples per class and evaluate on the remaining data. Cube Engraving is a synthetic dataset of 2D shapes engraved on a randomly chosen face of a cube, comprising 4,381 shapes from 22 distinct categories. In both experiments, we optimize against a cross-entropy loss with a label smoothing factor of 0.2. HodgeFormer achieves competitive results compared to other methods (Tab. 3). For SHREC-11, the best-performing model uses 6 HodgeFormer layers followed by 2 Transformer layers operating on all mesh elements. In contrast, for Cube Engraving, the top model also uses 6 HodgeFormer layers followed by 2 Transformer layers, but operates only on vertex features.

For the mesh segmentation task, our goal is to predict labels for every face of a mesh. We evaluate HodgeFormer architecture on four datasets: on the *Human-part-segmentation* dataset in its simplified version [26] with hard ground truth labels at faces, and the *Vases*, *Chairs*, and *Aliens* categories from the *Shape COSEG* dataset. In all experiments, we optimize against a class-weighted cross-entropy loss with an additional label smoothing factor of 0.2. For each dataset, we report the highest performing variant of our architecture.

Method	Type	Acts On	Eigen Decomp.	SHREC11 (split-10)	Cube Engrav.	Human Simplified	COSEG Vases	COSEG Chairs	COSEG Aliens
HodgeNet [34]	mlp	v	Yes	94.7%	n/a	85.0%	90.3%	95.7%	96.0%
DiffusionNet [33]	mlp	v	Yes	99.5%	n/a	90.8%	n/a	n/a	n/a
LaplacianNet [29]	mlp	v	Yes	n/a	n/a	n/a	92.2%	94.2%	93.9%
Laplacian2Mesh [10]	cnn	v	Yes	100.0%	91.5%	88.6%	94.6%	96.6%	95.0%
MeT [38]	trns	f	Yes	n/a	n/a	n/a	<b>99.8%</b>	<b>98.9%</b>	<b>99.3%</b>
MeshCNN [11]	cnn	e	<b>No</b>	91.0%	92.2%	85.4%	92.4%	93.0%	96.3%
PD-MeshNet [26]	cnn	ef	<b>No</b>	99.1%	94.4%	85.6%	95.4%	97.2%	98.2%
MeshWalker [16]	rnn	v	<b>No</b>	97.1%	98.6%	n/a	<b>99.6%</b>	98.7%	<b>99.1%</b>
SubDivNet [12]	cnn	f	<b>No</b>	99.5%	<b>98.9%</b>	<b>91.7%</b>	96.7%	96.7%	97.3%
EMNN (MC+H) [37]	gnn	ef	<b>No</b>	<b>100%</b> <sup>†</sup>	n/a	88.7% <sup>†</sup>	n/a	n/a	n/a
EGNN (MC+H) [37]	gnn	ef	<b>No</b>	99.6% <sup>†</sup>	n/a	87.2% <sup>†</sup>	n/a	n/a	n/a
<b>HodgeFormer (ours)</b>	trns	vef	<b>No</b>	98.7%	95.3%	90.3%	94.3%	<b>98.8%</b>	98.3%

Table 3. Performance of different methods on mesh classification and mesh segmentation tasks. Mesh classification is evaluated on the 30-class SHREC11[20] dataset evaluated on splits of 10 samples per class (split-10) and the Cube Engraving dataset [11]. Mesh segmentation is evaluated on the Human dataset in its simplified version of Milano et al. [26] with hard ground truth labels at faces, as well on the Shape COSEG dataset for the categories of Vases, Chairs and Aliens. HodgeFormer achieves results comparable to the state-of-the-art without spectral features, eigenvalue decomposition operations or complex complementary structures. We report their base architecture, the mesh elements they operate on (*v*: vertices, *e*: edges, *f*: faces) and whether they depend on eigen-decomposition methods. The abbreviation "trns" denotes the transformer architecture. Notes: The EMNN (MC+H) and EGNN (MC+H) results shown in the table are as reported by [37]. <sup>†</sup> clarifies that while EGNN and EMNN [37] theoretically do not need pre-calculated Heat Kernel Signatures (HKS), the authors report precomputation of HKS values for both the SHREC and Human Segmentation benchmark datasets

For the *Human* dataset, our highest performing model was a 6-layered HodgeFormer model mixed with 2 Transformer layers in a 2 : 1 ratio, as presented in Fig. S6a, whereas for *Vases*, *Chairs* and *Aliens* datasets, our highest performing model was a 6-layered HodgeFormer model followed by two Transformer layers, as presented in Fig. S6b. As Tab. 3 demonstrates, our approach performs comparably to state of the art without spectral features, eigenvalue decomposition operations, or complex complementary structures.

### 5.3. Model Variations Study

To evaluate the importance of different architecture components and parameter choices, we varied our base model in different ways and measured its performance on a segmentation task. Specifically, in Tab. 4, we compare the main architecture layers *HodgeFormer* and a vanilla *Transformer* layer along with different embedding layers, i.e., the *feed-forward* and the *neighbor* embedding layers. In Tab. 5, we test combinations of different input features based on their type by distinguishing between *coordinates*, *normals*, and *areas*. In Table 6, we evaluate the effect of the number of neighbors taken into account by the sparse attention component on the performance of the HodgeFormer layers. For all experiments, we consider an architecture of 4 layers acting only on mesh vertices with embedding dimension 256 and hidden dimension 512, and a learning rate of  $5e-4$ , and we report test accuracy results, averaged over 5 experiment trials with random training splits. We also consider  $\sqrt{n}$  neighbors

for each set of  $n$  mesh elements with the exception of the experiment of Tab. 6.

In Tab. 4, the HodgeFormer layers clearly outperform vanilla Transformer layers. At the same time, embedding input features while considering their neighbors offers an edge in initial training steps and results in an overall better performance. In Tab. 6, we observe the effect of the number of neighbors for vertices and edges on the HodgeFormer performance. Let  $s_k$  for  $k \in v, e, f$  the number of neighbors each element of a  $k$ -form attends to. For  $s_v = 1, s_e = 1$ , the performance drops significantly as the attention mechanism takes into account only self-contributions for vertices and edges. For  $s_v = 32, s_e = 48$ , which approximates  $\sqrt{n}$  for meshes in the COSEG vases dataset, the performance is high. Yet, for larger numbers of neighbors, we get diminishing returns in model performance. In Tab. 5, we observe that the HodgeFormer layer is capable of achieving satisfactory results using simple coordinate features, although exhibiting lower performance and slower convergence. However, using normals as the sole feature set proves insufficient, resulting in models with poor performance and limited generalization capability.

### 5.4. Efficiency

For the HodgeFormer architecture, preprocessing involves data loading, augmentation, and BFS operations for local neighborhood extraction (enabling sparse attention), all performed on-the-fly during data loading without separate pre-



Figure 5. Mesh segmentation results on the Human Body test set.

computation, supporting continuous data streaming across multiple cores. Tab. 7 compares the runtime performance of HodgeFormer with other recent methods on the human segmentation task, under typical training and testing configurations. HodgeFormer has an attractive runtime profile compared to other methods with fast training, high GPU utilization and no precomputation steps, with a significant part of its execution time spent in I/O and preprocessing, mostly in the BFS-based neighborhood extraction.

Transformer	Embedding	Vases
Vanilla Transformer	FNN	84.82%
Vanilla Transformer	Neighbors + FNN	87.28%
HodgeFormer	FNN	91.67%
HodgeFormer	Neighbors + FNN	92.02%

Table 4. Ablation study on the effect of the Neighbor Embedding layer (Sec. 4.5) evaluated on the COSEG Vases dataset. FNN refers to simple Feed-Forward Neural Network.

Input Features	Vases
coords	86.57%
normals	56.41%
coords-normals	91.66%
coords-areas	88.22%
all (coords-normals-areas)	92.02%

Table 5. Comparing input feature effect on HodgeFormer architecture performance, evaluated on the COSEG Vases dataset.

No. Neighbors (# V, # E)	Vases
1v, 1e	88.37%
8v, 12e	90.72%
16v, 24e	90.84%
32v, 48e	92.02%
64v, 96e	92.06%
128v, 196e	92.13%

Table 6. Ablation study on number of neighbors for the HodgeFormer Sparse Attention, evaluated on the COSEG Vases dataset.

## 6. Discussion, limitations and conclusion

In this work, we draw connections between discrete Hodge operators and Transformer-style attention mechanisms and

	Exec Time (s)	GPU Time (s)	GPU Batch (ms)	Peak Mem (GB)
Training (batch size = 12)				
MeshCNN	26.84	25.59	806.29	4.41
Laplacian2Mesh	641.34	605.17	423.57	4.80
<b>HodgeFormer</b>	17.58	8.36	263.57	14.21
Testing (batch size = 1)				
MeshCNN [11]	1.36	1.09	60.40	0.19
Laplacian2Mesh[18]	2.96	0.13	7.98	0.95
<b>HodgeFormer</b>	1.97	0.40	22.41	0.41

Table 7. Runtime performance comparison of HodgeFormer and other mesh-based methods on train and test sets of Human segmentation dataset, under typical training and testing configurations. We report the following metrics: (a) Execution time per epoch, (b) GPU time per epoch, (c) GPU time per batch, and (d) Peak GPU memory usage. All reported times are averaged over multiple epochs or batches, as appropriate.

propose a general architecture for deep learning on triangular meshes named HodgeFormer. Our architecture omits the construction of commonly used techniques such as expensive spectral features, eigenvalue decomposition operations, and complex complementary structures. Despite these omissions, HodgeFormer retains its expressiveness while fully leveraging the hardware-friendly properties of the Transformer architecture. In addition, by employing sparse attention mechanisms, the HodgeFormer architecture results in an overall complexity of  $O(n^{1.5}d)$ . HodgeFormer was tested on classification and segmentation tasks, demonstrating performance comparable to state-of-the-art methods. Strategies such as patching in MeshMAE [21] or mesh simplification preprocessing in MeshWalker [16] are part of future research on large-scale meshes and large-scale unsupervised training. Compared to a standard Transformer, our architecture has higher memory requirements due to the use of multiple attention mechanisms per layer. This requirement is partially mitigated by the employment of sparse attention. Although the proposed architecture has low complexity and is fully parallelizable, its implementation relies heavily on gather operations. Investigation of preprocessing strategies for input meshes with reordering algorithms such as the Cuthill-McKee [4] may allow the use of contiguous operations and enable better hardware utilization.



## References

- [1] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1626–1633, 2011. 2
- [2] Bernhard Auchmann and Stefan Kurz. A geometrically defined discrete hodge operator on simplicial cells. *IEEE Transactions on Magnetism*, 42(4):643–646, 2006. 1
- [3] Shuhao Cao. Choose a transformer: Fourier or galerkin. *Advances in neural information processing systems*, 34:24924–24940, 2021. 2
- [4] Elizabeth Cuthill and James McKee. Reducing the bandwidth of sparse symmetric matrices. In *Proceedings of the 1969 24th national conference*, pages 157–172, 1969. 8
- [5] Timothy A. Davis. Algorithm 1000: Suitesparse:graphblas: Graph algorithms in the language of sparse linear algebra. *ACM Trans. Math. Softw.*, 45(4), 2019. 6
- [6] Timothy A. Davis. Algorithm 10xx: Suitesparse:graphblas: Graph algorithms in the language of sparse linear algebra. *ACM Trans. Math. Softw.*, 2022. 6
- [7] Fernando De Goes, Mathieu Desbrun, Mark Meyer, and Tony DeRose. Subdivision exterior calculus for geometry processing. *ACM Transactions on Graphics*, 35(4):1–11, 2016. 2
- [8] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 2
- [9] Mathieu Desbrun, Anil N. Hirani, Melvin Leok, and Jerrold E. Marsden. Discrete Exterior Calculus. *arXiv preprint math/0508341*, 2005. 3
- [10] Qiujie Dong, Zixiong Wang, Junjie Gao, Shuangmin Chen, Zhenyu Shu, and Shiqing Xin. Laplacian2mesh: Laplacian-based mesh understanding. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):492–502, 2023. 7
- [11] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019. 2, 6, 7, 8
- [12] Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022. 2, 7
- [13] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020. 6
- [14] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Zamir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. *ACM computing surveys (CSUR)*, 54(10s):1–41, 2022. 1
- [15] Ilya Kostrikov, Zhongshi Jiang, Daniele Panozzo, Denis Zorin, and Joan Bruna. Surface Networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2540–2548, 2018. 2
- [16] Alon Lahav and Ayellet Tal. Meshwalker: Deep mesh understanding by random walks. *arXiv preprint arXiv:2006.05353*, 2020. 7, 8
- [17] Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters. *IEEE Transactions on Signal Processing*, 67(1):97–109, 2019. 2
- [18] Xiao-Juan Li, Jie Yang, and Fang-Lue Zhang. Laplacian mesh transformer: Dual attention and topology aware network for 3d mesh classification and segmentation. In *European Conference on Computer Vision*, pages 541–560. Springer, 2022. 2, 8
- [19] Yuan Li, Xiangyang He, Yankai Jiang, Huan Liu, Yubo Tao, and Lin Hai. Meshformer: High-resolution mesh segmentation with graph transformer. In *Computer Graphics Forum*, pages 37–49. Wiley Online Library, 2022. 1
- [20] Z. Lian, A. Godil, B. Bustos, M. Daoudi, J. Hermans, S. Kawamura, Y. Kurita, G. Lavoué, H. V. Nguyen, R. Ohbuchi, Y. Ohkita, Y. Ohishi, F. Porikli, M. Reuter, I. Sipiran, D. Smeets, P. Suetens, H. Tabia, and D. Vandermeulen. Shrec’11 track: Shape retrieval on non-rigid 3D watertight meshes. In *Eurographics Workshop on 3D Object Retrieval, EG 3DOR*, pages 79–88, 2011. 6, 7
- [21] Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. Meshmae: Masked autoencoders for 3d mesh data analysis. In *European Conference on Computer Vision*, pages 37–54. Springer, 2022. 2, 8
- [22] Lek-Heng Lim. Hodge laplacians on graphs. *Siam Review*, 62(3):685–715, 2020. 1
- [23] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12939–12948, 2021. 1
- [24] Minghua Liu, Chong Zeng, Xinyue Wei, Ruoxi Shi, Linghao Chen, Chao Xu, Mengqi Zhang, Zhaoning Wang, Xiaoshuai Zhang, Isabella Liu, et al. Meshformer: High-quality mesh generation with 3d-guided reconstruction model. *arXiv preprint arXiv:2408.10198*, 2024. 2
- [25] Jonathan Masci, Davide Boscaini, Michael Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 2
- [26] Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. *Advances in Neural Information Processing Systems*, 2020-December:952–963, 2020. 6, 7
- [27] Mamdouh S Mohamed, Anil N Hirani, and Ravi Samtaney. Comparison of discrete hodge star operators for surfaces. *Computer-Aided Design*, 78:118–125, 2016. 1
- [28] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019. 6
- [29] Yi-Ling Qiao, Lin Gao, Jie Yang, Paul L. Rosin, Yu-Kun Lai, and Xilin Chen. Learning on 3d meshes with laplacian encoding and pooling. *IEEE Transactions on Visualization and Computer Graphics*, 28(2):1317–1327, 2022. 2, 7

- [30] Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68, 2021. [5](#), [6](#)
- [31] Victor Garcia Satorras, Emiel Hooeboom, and Max Welling. E (n) equivariant graph neural networks. In *International conference on machine learning*, pages 9323–9332. PMLR, 2021. [2](#)
- [32] Jonas Schult, Francis Engelmann, Theodora Kontogianni, and Bastian Leibe. DualConvMesh-Net: Joint Geodesic and Euclidean Convolutions on 3D Meshes. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 8609–8619, 2020. [2](#)
- [33] Nicholas Sharp, Souhaib Attaiki, Keenan Crane, and Maks Ovsjanikov. DiffusionNet: Discretization Agnostic Learning on Surfaces. *ACM Transactions on Graphics*, 41(3):1–16, 2022. [2](#), [7](#)
- [34] Dmitriy Smirnov and Justin Solomon. Hodgenet: learning spectral geometry on triangle meshes. *ACM Transactions on Graphics (TOG)*, 40(4):1–11, 2021. [1](#), [2](#), [7](#)
- [35] Endre Süli, Christoph Schwab, and Paul Houston. hp-dgfem for partial differential equations with nonnegative characteristic form. In *Discontinuous Galerkin Methods: Theory, Computation and Applications*, pages 221–230. Springer, 2000. [1](#)
- [36] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Computer graphics forum*, pages 1383–1392. Wiley Online Library, 2009. [2](#)
- [37] Thuan Anh Trang, Nhat Khang Ngo, Daniel T Levy, Thieu Ngoc Vo, Siamak Ravanbakhsh, and Truong Son Hy. E(3)-equivariant mesh neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 748–756. PMLR, 2024. [2](#), [7](#)
- [38] Giuseppe Vecchio, Luca Prezzavento, Carmelo Pino, Francesco Rundo, Simone Palazzo, and Concetto Spampinato. Met: A graph transformer for semantic segmentation of 3d meshes. *arXiv preprint arXiv:2307.01115*, 2023. [1](#), [2](#), [7](#)