Computer Vision and Image Understanding 235 (2023) 103773

Contents lists available at ScienceDirect

Computer Vision and Image Understanding

journal homepage: www.elsevier.com/locate/cviu

MeT: A graph transformer for semantic segmentation of 3D meshes

Giuseppe Vecchio [a,*], Luca Prezzavento [a], Carmelo Pino [b], Francesco Rundo [b], Simone Palazzo [a], Concetto Spampinato [a]

[a] Department of Computer Engineering, University of Catania, Viale Andrea Doria 6, Catania 95127, Italy
[b] ADG, R&D Power and Discretes, STMicroelectronics, Str. Primosole, 50, 95121 Catania, Italy

ARTICLE INFO

Communicated by Nikos Paragios

MSC:
41A05
41A10
65D05
65D17

Keywords:
3D mesh segmentation
Graph neural networks
Transformers
3D meshes
Segmentation

ABSTRACT

Polygonal meshes have become the standard for discretely approximating 3D shapes, thanks to their efficiency and high flexibility in capturing non-uniform shapes. This non-uniformity, however, leads to irregularity in the mesh structure, making tasks like segmentation of 3D meshes particularly challenging. Semantic segmentation of 3D mesh has been typically addressed through CNN-based approaches, leading to good accuracy. Recently, transformers have gained enough momentum both in NLP and computer vision fields, achieving performance at least on par with CNN models, supporting the long-sought architecture universalism. Following this trend, we propose a transformer-based method for semantic segmentation of 3D mesh motivated by a better modeling of the graph structure of meshes, by means of global attention mechanisms. In order to address the limitations of standard transformer architectures in modeling relative positions of non-sequential data, as in the case of 3D meshes, as well as in capturing the local context, we perform positional encoding by means the Laplacian eigenvectors of the adjacency matrix, replacing the traditional sinusoidal positional encodings, and by introducing clustering-based features into the self-attention and cross-attention operators. Experimental results, carried out on three sets of the Shape COSEG Dataset (Wang et al., 2012), on the human segmentation dataset proposed in Maron et al. (2017) and on the ShapeNet benchmark (Chang et al., 2015), show how the proposed approach yields state-of-the-art performance on semantic segmentation of 3D meshes.

1. Introduction

Three-dimensional (3D) shapes are at the core of computer graphics and play an important role in many daily-life applications such as vision, robotics, medicine, augmented reality, and virtual reality. In recent years, many approaches have been proposed to encode real-world shapes, including 3D meshes (Botsch et al., 2010) and point clouds (Nguyen and Le, 2013). Meshes have become widely adopted to represent complex real-world objects, which are commonly composed of continuous surfaces, through a discrete approximation. The mesh is an efficient way to represent non-uniform surfaces, from simple shapes that generally require only a small number of polygons, up to arbitrarily complex objects, where the number of required polygons may increase significantly. The advantages presented by a mesh are particularly evident when compared to other forms of representation, like point clouds, which fall short when higher quality and preservation of sharp shape features are required.

With the increasing spread of deep learning techniques in many fields, research has tried to apply approaches from computer vision to 3D shape analysis. Convolutional neural networks (CNNs), in particular, have demonstrated outstanding performance on a variety of images-related tasks such as classification (Krizhevsky et al., 2012; Szegedy et al., 2016; He et al., 2016) and semantic segmentation (Ronneberger et al., 2015; Jégou et al., 2017; Chen et al., 2017). However, CNNs are designed to work on images, which are represented on a regular grid of discrete values, far from the irregular representation of 3D shapes. On the other hand, representing 3D objects through volumetric grids, e.g. mapping 3D shapes to multiple 2D projections (Su et al., 2015) or 3D voxel grids (Wu et al., 2015), is extremely inefficient and leads to computational costs that increase exponentially with higher resolution.

Recent approaches have tried to directly apply CNNs to the sparse point cloud representation (Qi et al., 2017; Achlioptas et al., 2018). These approaches have a substantial gain in terms of efficiency, but present an ill-defined notion of neighborhoods and connectivity and are inherently oblivious to the local surface. This issue makes the application of convolution and pooling operations non-trivial. To overcome this limitation, several works have recently tried to generalize CNN architectures to non-Euclidean domains such as graphs, and incorporate neighborhood information (Monti et al., 2017; Wang et al., 2019; Li et al., 2018a). Other approaches have tried to apply deep neural networks to 3D meshes (Boscaini et al., 2016; Xu et al., 2017;

* Corresponding author.
E-mail address: giuseppe.vecchio@phd.unict.it (G. Vecchio).

https://doi.org/10.1016/j.cviu.2023.103773
Received 26 January 2023; Received in revised form 17 April 2023; Accepted 3 July 2023
Available online 13 July 2023

Hanocka et al., 2019). One recent example is MeshCNN (Hanocka et al., 2019), which obtained state-of-the-art results on several segmentation datasets.

A recent trend in computer vision revolves around the use of transformer-based architectures, originally born for NLP, Vaswani et al. (2017) for vision tasks (Dosovitskiy et al., 2020; Touvron et al., 2021). The success of transformers lies in their extensive attention mechanism, which allows the network to learn global correlations between inputs. This property makes transformers able to intrinsically operate on fully-connected graphs. However, when dealing with sparse graphs, transformers show evident limitations, mainly because of the sinusoidal positional encoding that is not able to exploit graph topology and to the lack of local attention operators. Recently, Dwivedi and Bresson (2020) proposed an approach to extend the transformer architecture for arbitrary graphs. It introduces a graph transformer architecture which leverages the graph connectivity inductive bias, exploiting the graph topology. In particular, they (1) propose a new attention mechanism, (2) replace the positional encoding with the Laplacian eigenvectors, (3) re-introduce batch normalization layers, and (4) take into account edge feature representation.

Inspired by this work, and leveraging the structure of a mesh, which can be represented as a graph where the nodes correspond to vertices connected by polygon edges, we propose MeT, a transformer-based architecture for semantic mesh segmentation. In particular, our approach embeds locality features by means of the Laplacian operator (as in Dwivedi and Bresson (2020)) and by combining polygon features with clustering-based features into a novel two-stream transformer layer architectures, where features from the two modalities are extracted through self-attention and combined through cross-attention. Additionally, we ensure that graph structure inferred by the input mesh affects the attention operators, by injecting adjacency and clustering information as attention masks.

We evaluate our method on a variety of publicly-available mesh datasets of 3D objects and human bodies; in our experiments, the proposed approach is able to outperform previous works, both qualitatively and quantitatively.

To sum up, the key contributions of this work are:

- We enforce graph locality in the transformer by a combination of clustering information operator with Laplacian positional encoding in place of positional encoding.
- We introduce novel self-attention and cross-attention mechanisms, specifically designed for mesh segmentation, that take into account adjacency and clustering information to mask elements and further impose locality.
- Experimental results on multiple standard benchmarks with different type of meshes showing that our model outperforms, both quantitatively and qualitatively, existing mesh segmentation methods, setting new state-of-the-art performance on the task.

2. Related work

Meshes represent a way to describe 3D objects. They consist of vertices, edges and faces that defines the shape of a polyhedral object. In this work we will focus on triangular meshes, i.e., a mesh where all the faces are triangles.

2.1. Mesh segmentation

The semantic segmentation of 3D meshes is the process of assigning a label to each face. The task of semantic segmentation for meshes has applications in many fields such as robotics, autonomous driving, augmented reality and medical images analysis. Following the success of deep learning, several CNN-based methods have been applied 3D meshes to tackle the task of mesh segmentation (Rodrigues et al., 2018; He et al., 2021). We hereby present an overview of relevant work on 3D data analysis using neural networks, grouped by input representation type.

Volumetric. A common approach to represent the 3D shape into a binary voxel form that is the 3D analogous to a 2D grid such as an image. This allows for extending to 3D grids operations that are applied on 2D grids, thus applying any common image-based approaches to the shape domain. This concept was first introduced by Wu et al. (2015), who present a CNN that processes voxelized shapes for classification and completion. Following this approach, Brock et al. (2016) introduce a shape reconstruction method, using a voxel-based variational autoencoder. In 2019, Hanocka et al. (2018) present Alignet which used a voxel representations estimated applied the deformation on the original mesh.

Although being easy to process and extend existing method to voxels, this kind of representation is computationally and memory expensive. Resource efficient methods to process volumetric representations are an open research field with several approaches being proposed (Li et al., 2016; Riegler et al., 2017). Sparse convolutions allows to further reduce computational and memory requirements, leading to more efficient approaches (Yan et al., 2018; Graham et al., 2018; Choy et al., 2019; Lang et al., 2019; Yin et al., 2021), but suffer from inaccurate position information due to voxelization.

Graph. Another family of approaches leverages the ability to represent meshes as a graph structure. We distinguish between two main approaches for graph processing, one relies on the spectral properties of graphs (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Kostrikov et al., 2018); the second one is to directly process graphs extracting locally connected regions and transforming them into a canonical form for a neural network (Niepert et al., 2016). In 2017, Xu et al. (2017) propose a new architecture called Directionally Convolutional Network (DCN) that extends CNNs by introducing a rotation-invariant convolution and a pooling operation on the surface of 3D shapes. In particular, they propose a two-stream segmentation framework: one stream uses the proposed DCN with face normals as the input, while the other one is implemented by a neural network operating on the face distance histogram. The learned shape representations from the two streams are fused by an element-wise product. Finally, Conditional Random Field (CRF) is applied to optimize the segmentation. Yi et al. (2017) propose SyncSpecCNN, a spectral CNN with weight sharing in the spectral domain spanned by graph laplacian eigenbases, to tackle the task of 3D segmentation. Kostrikov et al. (2018) propose a Graph Neural Network (GNN) which exploits the Dirac operator to leverage extrinsic differential geometry properties of three-dimensional surfaces. These methods generally operate on the vertices of a graph.

Manifold. Masci et al. (2015), with the Geodesic Convolutional Neural Networks, and Boscaini et al. (2016) with the Anisotropic Convolutional Neural Networks, proposed two different CNNs-based architectures for triangular mesh segmentation.

In 2019, MeshNet was proposed by Hanocka et al. (2019). This architecture differs from the previous by working on mesh edges rather than faces. MeshCNN combines specialized convolution and pooling layers that operate on the mesh edges by leveraging their intrinsic geodesic connections. Convolutions are applied on edges and the four edges of their incident triangles, and pooling is applied via an edge collapse operation that retains surface topology, thereby, generating new mesh connectivity for the subsequent convolutions. MeshCNN learns which edges to collapse, thus forming a task-driven process where the network exposes and expands the important features while discarding the redundant ones.

Other approaches, like Lahav and Tal (2020), Smirnov and Solomon (2021) and Sharp et al. (2022a) propose alternative solutions to the segmentation task. MeshWalker (Lahav and Tal, 2020) represents mesh's geometry and topology by a set of random walks along the surface; these walks are fed to a recurrent neural network. HodgeNet (Smirnov

and Solomon, 2021), instead, tackles the problem relying on spectral geometry, and proposes parallelizable algorithms for differentiating eigencomputation, including approximate backpropagation without sparse computation. Finally, DiffusionNet (Sharp et al., 2022a) introduces a general-purpose approach to deep learning on 3D surfaces, using a simple diffusion layer to agnostically represent any mesh.

2.2. Graph transformers

Since their introduction, Transformers (Vaswani et al., 2017) have demonstrated their wide applicability to many different tasks, from NLP to Computer Vision. The original transformer was designed for handling sequential data in NLP, and operates on fully connected graphs representing all connections between the words in a sentence. However, when dealing with sparse graph, transformers perform poorly. Recently, several attempts to adapt transformers to graphs have been proposed (Li et al., 2018b; Nguyen et al., 2019; Zhang et al., 2020) focusing on heterogeneous graphs, temporal networks and generative modeling (Yun et al., 2019; Xu et al., 2021; Hu et al., 2020; Zhou et al., 2020). In 2019, Li et al. (2018b) introduce a model employing attention an all graph nodes, instead of a node's local neighbors, to capture global information. This approach limits the exploitation of sparsity, which is a good inductive bias for learning on graph datasets as shown in Dwivedi and Bresson (2020). To learn global information other approaches involve the use of a graph-specific positional features (Zhang et al., 2020), node Laplacian position eigenvectors (Belkin and Niyogi, 2003; Dwivedi and Bresson, 2020), relative learnable positional information (You et al., 2019) and virtual nodes (Li et al., 2015).

Dwivedi and Bresson (2020), propose an approach to extend the transformer architecture for arbitrary graphs. It introduces a graph transformer architecture with four new properties compared to the standard model, which are: (1) an attention mechanism which is a function of the neighborhood connectivity for each node in the graph; (2) positional encoding represented by the Laplacian eigenvectors, which naturally generalize the sinusoidal positional encoding often used in NLP; (3) a batch normalization layer in contrast to the layer normalization; (4) edge feature representation.

MeshFormer (Li et al., 2022) propose a mesh segmentation method based on graph transformers, which uses a boundary-preserving simplification to reduce the data size, a Ricci flow-based clustering algorithm for constructing hierarchical structures of meshes, and a graph transformer with cross-resolution convolutions, which extracts richer high-resolution semantic. Recently Zhuang et al. (2022) introduced a novel method for 3D mesh segmentation named Navigation Geodesic Distance Transformer (NGD-Transformer). It exploit the manifold properties of the mesh through a novel positional encoding called navigation geodesic distance positional encoding, which encodes the geodesic distance between vertices. Our work takes inspiration from Dwivedi and Bresson (2020) and proposes a transformer-based architecture for tackling 3D meshes represented as graphs. As in Dwivedi and Bresson (2020) we employ a positional encoding represented by the Laplacian eigenvectors of the adjacency matrix and a pre-layer batch normalization. However, we extend the original approach by adapting the architecture to 3D meshes, particularly, by proposing two cross-attention modules (similarly to decoder layers) learning local and global representations on 3D meshes and clusters thereof.

3. Method

In this work, we propose a novel transformer-based architecture for semantic segmentation of 3D meshes. The proposed method takes inspiration from recent vision transformer architectures (Guo et al., 2019) and spectral methods for graphs, in order to create an embedding in a Euclidean space with the topological features of the mesh.

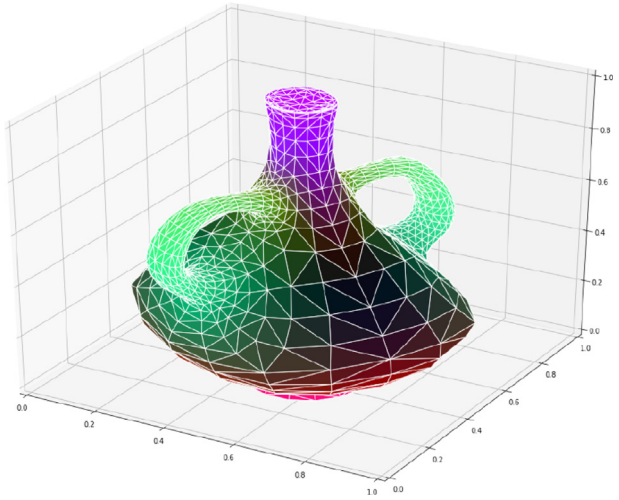

Fig. 1. Visualization of the first three eigenvectors of the Laplacian of the mesh dual graph.

Given a triangular mesh, described as a set of V vertices $\{\mathbf{v}_k = (x_k, y_k, z_k)\}_{k=1,\dots,V}$ and a set of N triangles $\{\mathbf{f}_i = (k_{i,1}, k_{i,2}, k_{i,3}, \mathbf{n}_i)\}_{i=1,\dots,N}$, where each triangle is defined by its three vertices and the normal direction $\mathbf{n}_i$ of its surface, the goal is to assign a class $c_i \in C$ to each triangle $\mathbf{f}_i$, representing the dominant class on the surface of the triangle.

3.1. Feature extraction

For each triangle, we initially extract a set of features based on spectral properties of the triangle graph (where triangles are nodes, and shared sides are edges), which is the dual of the mesh (where vertices are nodes, and triangle sides are edges).

The process starts by building the adjacency matrix $\mathbf{A}$, of size $N \times N$ (N being the number of triangles in the mesh), such that $A_{ij} = 1$ if the ith and the jth triangles share an edge, and $A_{ij} = 0$ otherwise. From the adjacency matrix $\mathbf{A}$, we then compute the symmetric normalized Laplacian matrix $\mathbf{L}$ as:

$$\mathbf{L} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}, \tag{1}$$

where $\mathbf{I}$ is the identity matrix and $\mathbf{D}$ is the degree matrix for $\mathbf{A}$, i.e., a diagonal matrix such that D_{ii} is the number of edges connected to i (equivalently, the sum of the elements in the ith row of $\mathbf{A}$). Then, we identify the $(E+1)$th (with $E \leq N$) eigenvector with the smallest non-zero eigenvalue. The ith components of the E remaining eigenvectors, corresponding to vector $\mathbf{l}_i$, are then used to encode the location of the ith triangle within the mesh. We employ these features as a positional encoding in the transformer, as described by Dwivedi and Bresson (2020), and to identify local neighborhood by means of clustering (described in the next section).

Formally, given a triangle $\mathbf{f}_i = (k_1, k_2, k_3, \mathbf{n}_i)$, where $\mathbf{n}_i$ is its normal vector direction, obtained by computing the vector product $\tilde{\mathbf{n}}_i = (\mathbf{v}_{k_2} - \mathbf{v}_{k_1}) \times (\mathbf{v}_{k_3} - \mathbf{v}_{k_1})$ and then normalizing it as $\mathbf{n}_i = \tilde{\mathbf{n}}_i / \|\tilde{\mathbf{n}}_i\|$, we obtain the feature representation $\mathbf{t}_i$ for triangle i as $\mathbf{t}_i = (\mathbf{v}_{k_1}, \mathbf{v}_{k_2}, \mathbf{v}_{k_3}, \mathbf{n}_i, \mathbf{l}_i)$.

Fig. 1 shows the visualization of the Laplacian eigenvectors for a mesh.

3.2. Clustering

Before being processed by the network, triangles' features are clustered in $M = V/\lambda$ clusters, where λ is a configurable parameter, controlling the average number of mesh vertices per cluster. As we describe in detail when presenting our transformer architecture, we

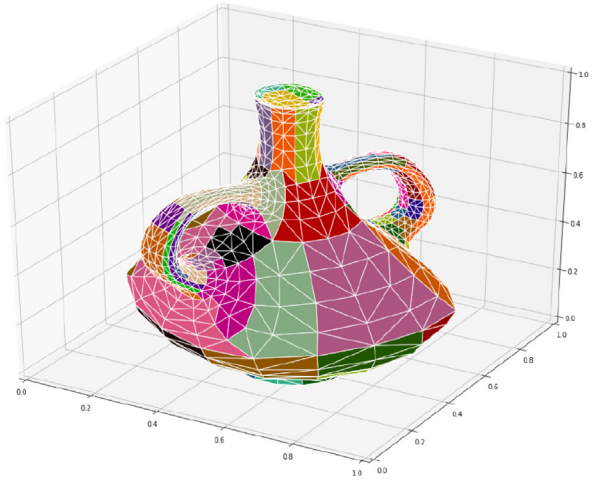

Fig. 2. Example of triangles clustering with $\lambda = 8$.

introduce clustering as an additional and more explicit way than positional encoding to enforce locality on the features extracted for each triangle. Clustering is carried out using the Ward method (Ward, 1963), which applies constraints on the connectivity dictated by the dual graph adjacency matrix, generating clusters geometrically and topologically connected and cohesive.

The result is a matrix $\mathbf{J}$ with shape $N \times M$, such that $J_{im} = 1$ if the ith triangle is in the mth cluster, and $J_{im} = 0$ otherwise. *Each row $\mathbf{j}_i$ in $\mathbf{J}$ can be interpreted as the one-hot cluster representation for the ith triangle.* Fig. 2 shows an example of mesh triangles clustering.

3.3. Network architecture

Mesh Transformer. The proposed MeT architecture implements a transformer model with two internal feature extraction streams, one for triangle features and one for cluster features, organized in matching sets (i.e., the ith element of the triangle set corresponds to the ith element of the cluster set). The two sets of features are processed by a cascade of transformer layers; only features from the triangle stream are finally used for prediction through a two-layer feedforward network, which predicts for each triangle a score vector $\mathbf{s}_i$, of size equal to the number of segmentation classes.

Given the extracted features $\mathbf{t}_i$ and cluster identifier $\mathbf{j}_i$ for each mesh triangle, we first convert them into two sequences of *tokens*, to be provided as input to the transformer layers. Triangle tokens $\mathbf{e}_i$ are obtained as:

$$\mathbf{e}_i = \mathrm{FF}_t\left(\mathbf{t}_i\right) \tag{2}$$

where FF_t is a feedforward layer with ReLU activation,[1] of output size d_t. Cluster tokens $\mathbf{p}_i$, of size d_p, are obtained by a learnable embedding layer on the corresponding one-hot cluster identifier $\mathbf{j}_i$. Matrices $\mathbf{E} \in \mathbb{R}^{N \times d_t}$ and $\mathbf{P} \in \mathbb{R}^{N \times d_p}$ are defined by laying each token as a row in the corresponding matrix.

Each network layer, illustrated in Fig. 3, can thus be defined as a function $L_i(\cdot,\cdot)$ on token sequences:

$$L_i(\mathbf{E},\mathbf{P}) = \left(\mathrm{R}_i\left(\mathrm{SA}_{t,i}\left(\mathrm{TC}_i(\mathbf{E},\mathbf{P})\right)\right), \mathrm{R}_i\left(\mathrm{SA}_{p,i}\left(\mathrm{CT}_i(\mathbf{E},\mathbf{P})\right)\right)\right) \tag{3}$$

where $\mathrm{SA}_{t,i}$ and $\mathrm{SA}_{p,i}$ are, respectively, the triangle and cluster self-attention functions, R_i is a residual connection function, and TC_i and CT_i are, respectively, the function updating triangle tokens from cluster tokens and vice versa. The output of each layer has the same dimensions as the input, allowing for arbitrary length of encoder sequences.

[1] All feedforward layers in our model have ReLU activations.

Multi-head attention. Before introducing the details of the encoder layers, let us present a general formulation of multi-head attention, which is extensively employed in the proposed architecture. An attention function A receives three matrices $\mathbf{Q} \in \mathbb{R}^{N_q \times d_k}$ (query), $\mathbf{K} \in \mathbb{R}^{N_k \times d_k}$ (key) and $\mathbf{V} \in \mathbb{R}^{N_k \times d_v}$ (value), and returns a matrix $\mathbf{O} \in \mathbb{R}^{N_q \times d_v}$, where each row is computed as a linear combination of rows from $\mathbf{V}$, weighted by normalized dot-product similarity between rows of $\mathbf{Q}$ and $\mathbf{K}$, as follows:

$$A(\mathbf{Q},\mathbf{K},\mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_k}}\right)\mathbf{V} \tag{4}$$

where softmax is applied on rows of the input matrix. In multi-head attention, in order to capture several possible attention patterns between elements, $\mathbf{Q}$, $\mathbf{K}$ and $\mathbf{V}$ are usually computed by linearly projecting a set of input matrices $\hat{\mathbf{Q}} \in \mathbb{R}^{N_q \times \hat{d}_k}$, $\hat{\mathbf{K}} \in \mathbb{R}^{N_k \times \hat{d}_k}$ and $\hat{\mathbf{V}} \in \mathbb{R}^{N_k \times \hat{d}_v}$ through multiple sets of projection matrices $\left\{\left(\mathbf{W}_{q,i}, \mathbf{W}_{k,i}, \mathbf{W}_{v,i}\right)\right\}_{i=1,\dots,h}$, with h being the number of heads. The attention outputs for each set of projection matrices are then concatenated and linearly projected to produce the final output, as follows:

$$\mathrm{MA}(\hat{\mathbf{Q}},\hat{\mathbf{K}},\hat{\mathbf{V}}) = \mathrm{concat}\left(H_1,\dots,H_h\right)\mathbf{W}_O \tag{5}$$

where $\mathbf{W}^O \in \mathbb{R}^{N \times d_o}$ is a linear projector to the desired output dimension, and H_i is the output of the ith attention head:

$$H_i = A\left(\hat{\mathbf{Q}}\mathbf{W}_{q,i}, \hat{\mathbf{K}}\mathbf{W}_{k,i}, \hat{\mathbf{V}}\mathbf{W}_{v,i}\right) \tag{6}$$

The amount of computation required for multi-head attention is approximately the same as in single-head attention, by uniformly splitting dimensions d_q, d_k and d_v among the h heads. In this work, for simplicity, we set $d_q = d_k = d_v = d_o = d$, whose specific value depends on where multi-head attention is employed in the network, as described below.

Self-attention for cluster tokens. The architecture of the self-attention module for cluster tokens is presented in Fig. 4. The module receives the set of cluster tokens $\mathbf{P}$ and applies a function SA_p defined as:

$$\mathbf{P}_n = \mathrm{PLN}(\mathbf{P}) \tag{7}$$

$$\mathrm{SA}_p(\mathbf{P}) = \mathrm{MA}\left(\mathbf{P}_n,\mathbf{P}_n,\mathbf{P}_n\right) + \mathbf{P} \tag{8}$$

where PLN is Pre-Layer Normalization (Xiong et al., 2020), which has been shown to improve training of transformer architectures, and query, key and values matrices are all set to $\mathbf{P}_n$, as is typical of self-attention. A final residual connection is applied to improve gradient flow. The d size is set to d_p, i.e., the size of the input cluster tokens.

Self-attention for triangles tokens, illustrated in Fig. 5, shares the same architecture as the self-attention module for clusters, but it employs the adjacency matrix $\mathbf{A}$ as a mask for multi-head attention computation. The choice to adopt a adjacency-based attention masking mechanism is due to the need to preserve the capacity of the model to capture both local composition and long-range dependency (Guo et al., 2019) and to reduce computation requirements for high-resolution meshes exploiting the sparsity of the $\mathbf{A}$ matrix. To carry out masked multi-head attention, the attention function in Eq. (4) is modified by subtracting infinity from masked positions of the query-key similarity vector, in order to nullify the corresponding softmax terms. The resulting attention function A_{mask} is defined as:

$$A_{\mathrm{mask}}(\mathbf{Q},\mathbf{K},\mathbf{V},\mathbf{M}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top - \mathbf{M}}{\sqrt{d_k}}\right)\mathbf{V} \tag{9}$$

where elements of $\mathbf{M}$ are either 0 or $-\infty$. We can thus define our self-attention module for triangles as:

$$\mathbf{E}_n = \mathrm{PLN}(\mathbf{E}) \tag{10}$$

$$\mathrm{SA}_t(\mathbf{E}) = \mathrm{MA}_{\mathrm{mask}}\left(\mathbf{E}_n,\mathbf{E}_n,\mathbf{E}_n,\hat{\mathbf{A}}\right) + \mathbf{E} \tag{11}$$

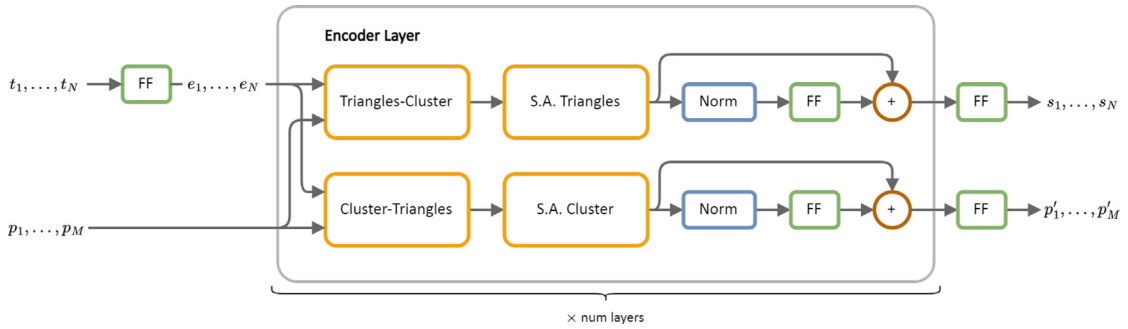

Fig. 3. Representation of an encoder layer of the Mesh Transformer.

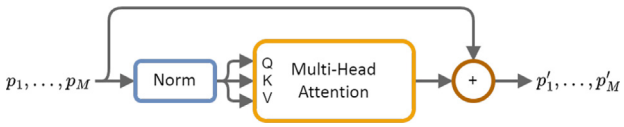

Fig. 4. Architecture of the self-attention module for cluster tokens.

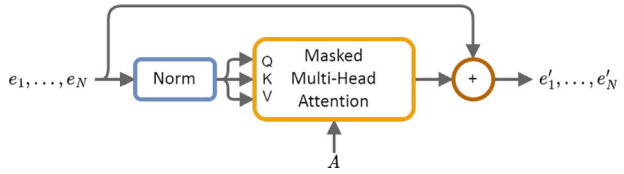

Fig. 5. Representation of the self-attention module for triangles.

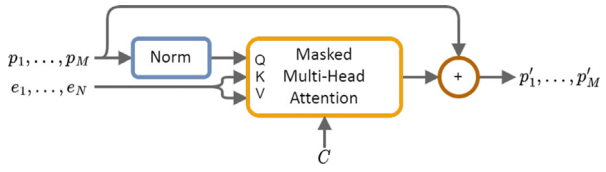

Fig. 6. Representation of the cluster-triangle update module.

where $\mathrm{MA}_{\mathrm{mask}}$ is the variant MA employing A_{mask} as attention function, and $\hat{\mathbf{A}} = \log \mathbf{A}$, so that $\hat{A}_{ij} = -\infty$ where $A_{ij} = 0$, and $\hat{A}_{ij} = 0$ where $A_{ij} = 1$. The d size for multi-head attention is set to d_t, i.e., the size of the input triangle tokens.

Updating cluster representation from triangle tokens. The cluster-triangle update module is introduced to update the clusters' representation w.r.t. the triangles', thus allowing the network to exchange information between the two different modalities employed for modeling graph structure, i.e., Laplacian eigenvectors and clustering. To this aim, we employ masked multi-head attention using cluster tokens for computing query vectors, and triangle tokens to compute keys and values; in order to aggregate, for each cluster, only information of the triangles contained in it, we compute a symmetric matrix $\mathbf{C}$ from the $\mathbf{J}$ clustering matrix by setting $C_{ij} = 1$ if triangles i and j belong to the same cluster, i.e., $\mathbf{j}_i = \mathbf{j}_j$, and $C_{ij} = 0$ otherwise. The architecture of the cluster-triangle update module is presented in Fig. 6, and implements the following function:

$$\mathbf{P}_n = \mathrm{PLN}(\mathbf{P}) \tag{12}$$

$$\mathrm{CT}(\mathbf{E}, \mathbf{P}) = \mathrm{MA}_{\mathrm{mask}}\left(\mathbf{P}_n, \mathbf{E}, \mathbf{E}, \hat{\mathbf{C}}\right) + \mathbf{P} \tag{13}$$

where mask $\hat{\mathbf{C}} = \log \mathbf{C}$, as above. The d dimension for multi-head attention is set to d_p, i.e., the size of input cluster tokens.

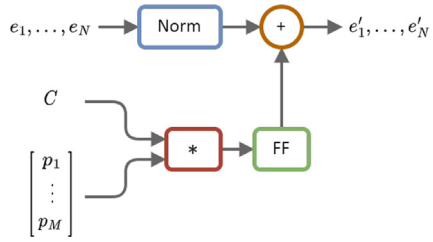

Fig. 7. Representation of the triangle-cluster update module.

Updating triangle representations from cluster tokens. A triangle-cluster update module is also used to update triangle representation with respect to clusters. Similarly to the cluster-triangle case, each triangle is affected only by elements belonging to the same cluster. The cross-attention module computes the sum between each triangle token and a projection of the average of the corresponding cluster tokens through a single feed-forward layer, as follows:

$$\mathbf{E}_n = \mathrm{PLN}(\mathbf{E}) \tag{14}$$

$$\mathrm{TC}(\mathbf{E}, \mathbf{P}) = \mathbf{E}_n + \mathrm{FF}_{\mathrm{TC}}(\mathbf{CP}) \tag{15}$$

where $\mathrm{FF}_{\mathrm{TC}}$ is a single feedforward layer. The architecture of the triangle update module is presented in Fig. 7. This operation can be interpreted as a form of cross-attention between triangle tokens and cluster tokens, where the former attend to the latter by means of a constant attention factor defined by cluster membership.

Layer residual connection. The output of each token stream of a network layer finally undergoes a feedforward residual transformation, to independently transform each token, as follows:

$$\mathbf{S}_n = \mathrm{PLN}(\mathbf{S}) \tag{16}$$

$$(\mathbf{S}) = \mathrm{FF}_R(\mathbf{S}_n) + \mathbf{S} \tag{17}$$

where $\mathbf{S}$ is either $\mathbf{E}$ or $\mathbf{P}$, and FF_R is a feedforward layer. The architecture of the triangle update module is presented in Fig. 7.

4. Experimental results

In this section, we first introduce the datasets employed in our work: the COSEG Shapes dataset (Wang et al., 2012), and the Human segmentation datasets proposed by Maron et al. (2017). Then, we evaluate the accuracy of our approach on the two different datasets. First, we assess how the model performs in three categories of the COSEG dataset, namely, *Chairs*, *Vases* and *Tele-Aliens*; afterwards, we evaluate our method on the segmentation of human body meshes as well as on the ShapeNet dataset (Chang et al., 2015). Ablation study then follows to substantiate the choices on the architecture components. As a methodical note on the evaluation, for the comparison to MeT, with existing methods, we report the performance values reported in their original papers on the considered benchmarks.

Table 1
Quantitative results (classification accuracy in percentage) on the COSEG dataset. Values are reported in terms of accuracy between the predicted and ground-truth segmentation.

Method	Chairs	Vases	Tele-Aliens
Xie et al. (2014)	85.9	87.1	83.3
Kim et al. (2013)	91.2	85.6	–
Guo et al. (2015)	95.5	88.5	–
LaplacianNet (Qiao et al., 2019)	94.2	92.2	93.9
Wang et al. (2018)	95.9	91.2	93.0
Xu et al. (2017)	95.7	90.9	–
MeshCNN (Hanocka et al., 2019)	99.6	97.3	97.6
MeshWalker (Lahav and Tal, 2020)	99.6	98.7	99.1
NGD (Zhuang et al., 2022)	95.2	91.8	94.3
MeshFormer (Li et al., 2022)	98.9	**99.1**	99.1
MeT (Ours)	**99.8**	98.9	**99.3**

4.1. Datasets and metrics

We test the performance of MeT and compare it with those yielded by existing models on three standard benchmarks, namely, the Shape COSEG dataset (Wang et al., 2012), the Human dataset (Maron et al., 2017) and the ShapeNet dataset (Chang et al., 2015).

The Shape COSEG dataset (Wang et al., 2012) consists of 11 sets of shapes with a consistent ground-truth segmentation and labeling: 8 sets are rather small and come from the dataset by Sidi et al. (2011), while the 3 remaining ones contain, respectively, tele-alines, vases and chairs. Given the scale of tele-alines, vases and chairs sets compared to the other eight sets, we used only them to evaluate the performance of MeT. Train and test splits are the same defined in MeshCNN (Hanocka et al., 2019) for a fair comparison. As validation set we use 6% of the training set.

We also evaluate our method on human segmentation dataset introduced by Maron et al. (2017). It consists of human meshes from several datasets, in particular SCAPE, FAUST, MIT Animation and SHREC 2007. The latest is used as test set, as in the MeshCNN (Hanocka et al., 2019) paper.

ShapeNet (Chang et al., 2015) is a large-scale repository of shapes represented by 3D models of objects categorized following the WordNet taxonomy. ShapeNet contains semantic annotations about object parts as well as for rigid alignments, bilateral symmetry planes, physical sizes and other annotations.

4.2. Model training and evaluation

We train our model with mini-batch gradient descent, using the AdamW (Loshchilov and Hutter, 2017) optimizer and a batch size of 12. Learning rate is set to $5 \cdot 10^{-5}$ with a weight decay of 0.01. Dropout with probability 0.1 is used after each feedforward layer and multi-head attention in the transformer encoder, and after each feedforward layer in the classification network. The value of the λ parameters controlling the features clustering, described in Section 3, is 8 for all the experiments, while token dimensions are set as $d_t = 512$ and $d_p = 1024$. All these parameters we set by measuring performance on a validation set extracted from each the COSEG dataset. Cross-entropy loss function is used and weighted for each triangle based on its surface (larger triangles have more weight). We perform data augmentation by applying random translation, rotation and scaling for each mesh in a mini-batch.

Accuracy is computed, as in DCN (Xu et al., 2017), as the total surface of triangles correctly classified over the entire surface.

Data preprocessing. Similarly to MeshCNN, each mesh in the dataset is preprocessed reducing the number of vertices to a maximum of 1200 using the algorithm proposed by Garland and Heckbert (1997). Duplicated vertices are merged and "padding" triangles are added to allow batched processing of meshes. After preprocessing, each mesh consists of 2412 triangles. Padding triangles are not adjacent to any mesh triangle and do not influence the final prediction. Vertex coordinates are standardized between −1 and 1. The **A** and **P** matrices are extended to include the padding triangles.

Table 2
Quantitative results (classification accuracy in percentage) on the Human dataset. Values are reported in terms of accuracy between the predicted and ground-truth segmentation.

Method	Laplacian encoding
MeshCNN (Hanocka et al., 2019)	92.3
MeshWalker (Lahav and Tal, 2020)	92.7
Sharp et al. (2022b)	91.7
MeT (Ours)	**93.6**

Table 3
Quantitative results (classification accuracy in percentage) on the ShapeNet dataset. Values are reported in terms of accuracy between the predicted and ground-truth segmentation.

Method	ShapeNet
Shapeboost (Kalogerakis et al., 2010)	77.2
Guo et al. (2015)	77.6
ShapePFCN (Kalogerakis et al., 2017)	85.7
LaplacianNet (Qiao et al., 2019)	91.5
MeshTransformer (Li et al., 2022)	92.6
MeT (Ours)	**94.2**

Table 4
Ablation study of input features. We ablate the transformer network on the input to the model.

Model version	Chairs	Vases	Tele-Aliens
w/o coordinates	94.4	92.1	93.8
w/o Laplacian embedding	97.9	97.0	97.4
w/o normals	98.8	97.4	98.6
Full MeT input	**99.8**	**98.9**	**99.3**

We first evaluate our models on the Chairs, Vases and Tele-Aliens subsets of the COSEG dataset. For each set, we report the performance, in terms of accuracy. Table 1 shows that our approach achieves a higher global accuracy on all the COSEG sets w.r.t. state of the art methods. Fig. 8 shows segmentation examples for each mesh set.

Mesh segmentation performances on the Human dataset (Maron et al., 2017) are reported in Table 2, showing better performance of our approach also on this benchmark when comparing with three state-of-the-art algorithms. Fig. 9 shows qualitative results for the predicted segmentation.

Finally, we compute mesh segmentation performances on the ShapeNet dataset (Maron et al., 2017), which are showed in Table 3. Also on this benchmark, MeT yields better accuracy than state-of-the-art methods.

4.3. Ablation study

We perform an ablation study, on the three subsets of the *COSEG* dataset, to substantiate our design choices. We first assess how each component in the triangle representation affects performance, namely, triangle coordinates, surface normal and Laplacian positional encoding. Results in Table 4 show that all the input features positively affect accuracy. However, the highest contribution to the final performance is provided by the Laplacian.

We then assess the importance of the cluster-related stream described in Section 3, i.e., cluster self-attention and cluster-triangle cross-attention . A comparison of the model accuracy with and without the cluster modules is presented in Table 5, where we can see that the cluster modules lead to gain of 3.6 percent points over the baseline, i.e., the model using only triangle information.

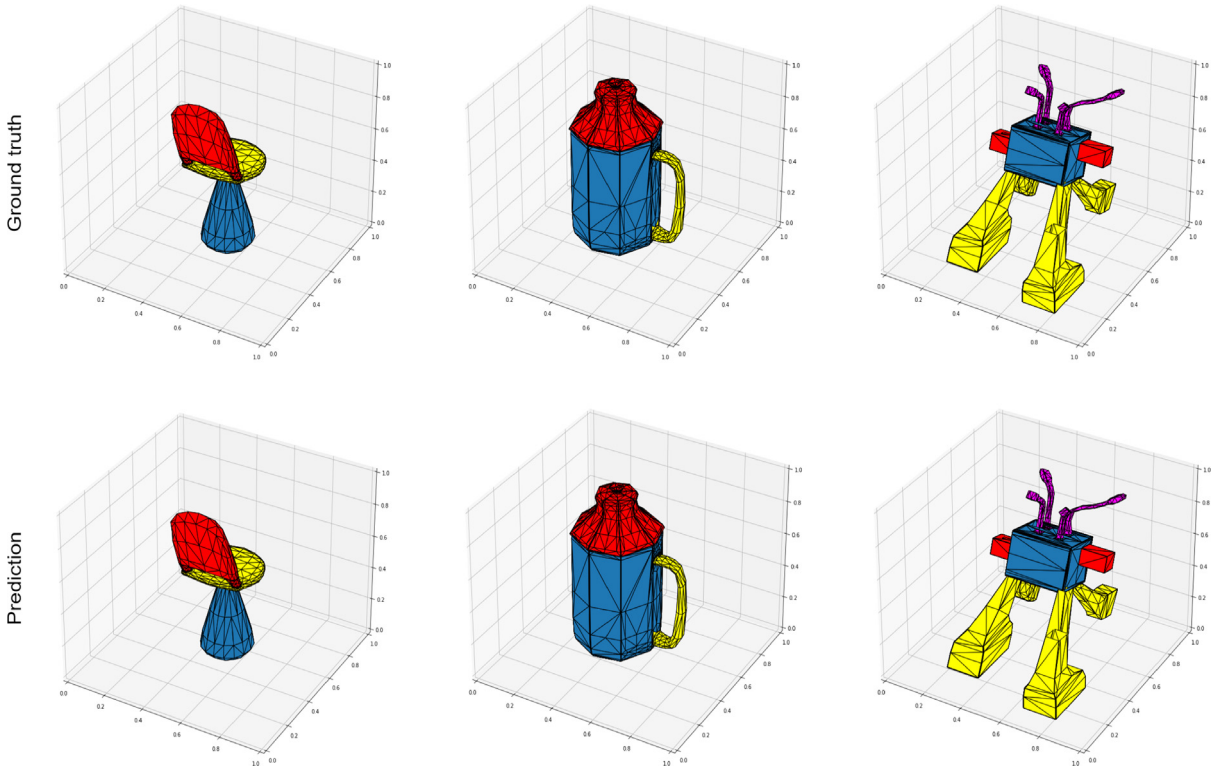

Fig. 8. Output segmentation examples by MeT on COSEG meshes. At the top, the ground truth; at the bottom, the network prediction. Left to right: a chair example; a vase example and a tele-alien example.

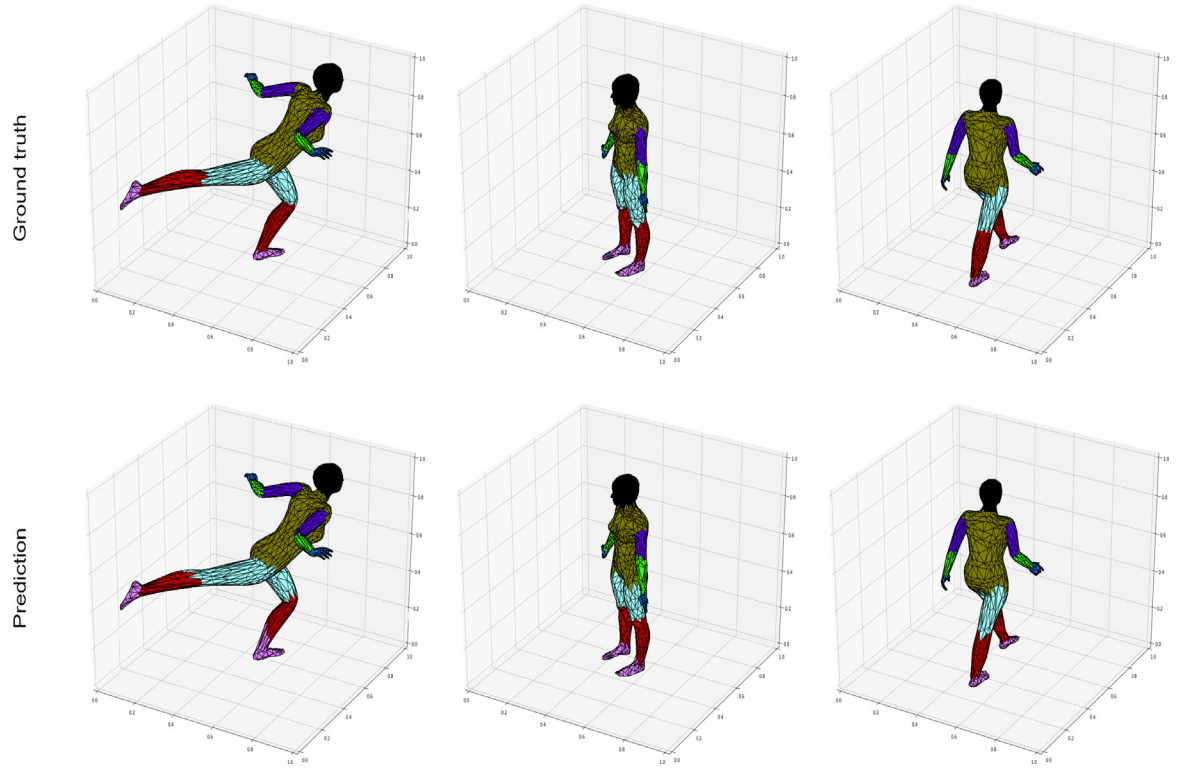

Fig. 9. Output segmentation samples on human meshes. At the top the segmentation ground truth; at the bottom the network prediction.

Table 5
Ablation study of design choices. We ablate the transformer network, comparing performance with and without the cluster modules.

Model version	Chairs	Vases	Tele-Aliens
w/o cluster modules	96.4	95.3	96.1
Full MeT	**99.8**	**98.9**	**99.3**

5. Conclusion

In this work, we introduce a novel transformer-based architecture for 3D mesh segmentation. Our approach successfully and significantly extends standard transformers with features specifically designed for the task at hand. First, we introduce a two-stream processing pipeline with each transformer layer, designed to enforce locality through the combination between mesh triangle features and clustering-based features, and by integrating spectral graph properties, through Laplacian vectors, to replace classic sinusoidal positional encoding. Additionally, we adapt typical attention mechanisms in transformers, by taking into account graph properties, and in particular by using adjacency matrix and triangle clustering to explicitly mask multi-head self- and cross-attention.

Experimental results, evaluated on multiple object categories, show that the resulting approach is able to outperform state-of-the-art methods on mesh segmentation, and demonstrate the positive impact of our architectural novelties by means of extended ablation studies.

To conclude, we show that transformer models — in spite of their characteristics for global processing and limitations with representing locality in sparse graphs — can be successfully adapted to mesh analysis, by carefully integrating methodological adjustments designed to capture mesh properties in a complex task such as segmentation.

CRediT authorship contribution statement

Giuseppe Vecchio: Idea design, Software development, Experiments execution, Paper writing. **Luca Prezzavento:** Code development, Experiments execution. **Carmelo Pino:** Idea design, Paper writing. **Francesco Rundo:** Idea design, Paper writing. **Simone Palazzo:** Guided the research in all steps from idea generation to paper writing. **Concetto Spampinato:** Guided the research in all steps from idea generation to paper writing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This research is supported by the project Future Artificial Intelligence Research (FAIR) PNRR MUR Cod. PE0000013- CUP: E63C2200 1940006 and by the project “LEGO.AI: LEarning the Geometry of knOwledge in AI systems”, n. 2020TA3K9N.

References

Achlioptas, P., Diamanti, O., Mitliagkas, I., Guibas, L., 2018. Learning representations and generative models for 3d point clouds. In: International Conference on Machine Learning. PMLR, pp. 40–49.
Belkin, M., Niyogi, P., 2003. Laplacian eigenmaps for dimensionality reduction and data representation. Neural Comput. 15 (6), 1373–1396.
Boscaini, D., Masci, J., Rodolà, E., Bronstein, M., 2016. Learning shape correspondence with anisotropic convolutional neural networks. Adv. Neural Inf. Process. Syst. 29.
Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., Lévy, B., 2010. Polygon Mesh Processing. CRC Press.
Brock, A., Lim, T., Ritchie, J.M., Weston, N., 2016. Generative and discriminative voxel modeling with convolutional neural networks. arXiv preprint arXiv:1608.04236.
Bruna, J., Zaremba, W., Szlam, A., LeCun, Y., 2013. Spectral networks and locally connected networks on graphs. arXiv preprint arXiv:1312.6203.
Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F., 2015. ShapeNet: An information-rich 3D model repository. URL: http://arxiv.org/abs/1512.03012, cite arxiv:1512.03012.
Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L., 2017. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. 40 (4), 834–848.
Choy, C., Gwak, J., Savarese, S., 2019. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3075–3084.
Defferrard, M., Bresson, X., Vandergheynst, P., 2016. Convolutional neural networks on graphs with fast localized spectral filtering. Adv. Neural Inf. Process. Syst. 29.
Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al., 2020. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.
Dwivedi, V.P., Bresson, X., 2020. A generalization of transformer networks to graphs. arXiv preprint arXiv:2012.09699.
Garland, M., Heckbert, P.S., 1997. Surface simplification using quadric error metrics. In: Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques. pp. 209–216.
Graham, B., Engelcke, M., Van Der Maaten, L., 2018. 3D semantic segmentation with submanifold sparse convolutional networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 9224–9232.
Guo, Q., Qiu, X., Liu, P., Shao, Y., Xue, X., Zhang, Z., 2019. Star-transformer. arXiv preprint arXiv:1902.09113.
Guo, K., Zou, D., Chen, X., 2015. 3D mesh labeling via deep convolutional neural networks. ACM Trans. Graph. 35 (1), 1–12.
Hanocka, R., Fish, N., Wang, Z., Giryes, R., Fleishman, S., Cohen-Or, D., 2018. Alignet: Partial-shape agnostic alignment via unsupervised learning. ACM Trans. Graph. 38 (1), 1–14.
Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D., 2019. Meshcnn: a network with an edge. ACM Trans. Graph. 38 (4), 1–12.
He, Y., Yu, H., Liu, X., Yang, Z., Sun, W., Wang, Y., Fu, Q., Zou, Y., Mian, A., 2021. Deep learning based 3D segmentation: A survey. arXiv preprint arXiv:2103.05423.
He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778.
Henaff, M., Bruna, J., LeCun, Y., 2015. Deep convolutional networks on graph-structured data. arXiv preprint arXiv:1506.05163.
Hu, Z., Dong, Y., Wang, K., Sun, Y., 2020. Heterogeneous graph transformer. In: Proceedings of the Web Conference 2020. pp. 2704–2710.
Jégou, S., Drozdzal, M., Vazquez, D., Romero, A., Bengio, Y., 2017. The one hundred layers tiramisu: Fully convolutional densenets for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. pp. 11–19.
Kalogerakis, E., Averkiou, M., Maji, S., Chaudhuri, S., 2017. 3D shape segmentation with projective convolutional networks. In: CVPR. IEEE Computer Society, pp. 6630–6639, URL: http://dblp.uni-trier.de/db/conf/cvpr/cvpr2017.html#KalogerakisAMC17.
Kalogerakis, E., Hertzmann, A., Singh, K., 2010. Learning 3D mesh segmentation and labeling. ACM Trans. Graph. 29 (4), http://dx.doi.org/10.1145/1778765.1778839.
Kim, V.G., Li, W., Mitra, N.J., Chaudhuri, S., DiVerdi, S., Funkhouser, T., 2013. Learning part-based templates from large collections of 3D shapes. ACM Trans. Graph. 32 (4), 1–12.
Kostrikov, I., Jiang, Z., Panozzo, D., Zorin, D., Bruna, J., 2018. Surface networks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2540–2548.
Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. Adv. Neural Inf. Process. Syst. 25.
Lahav, A., Tal, A., 2020. MeshWalker: Deep mesh understanding by random walks. ACM Trans. Graph. 39 (6), http://dx.doi.org/10.1145/3414685.3417806.
Lang, A.H., Vora, S., Caesar, H., Zhou, L., Yang, J., Beijbom, O., 2019. Pointpillars: Fast encoders for object detection from point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 12697–12705.

Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B., 2018a. Pointcnn: Convolution on x-transformed points. Adv. Neural Inf. Process. Syst. 31.
Li, Y., He, X., Jiang, Y., Liu, H., Tao, Y., Hai, L., 2022. MeshFormer: High-resolution mesh segmentation with graph transformer.
Li, Y., Liang, X., Hu, Z., Chen, Y., Xing, E.P., 2018b. Graph transformer.
Li, Y., Pirk, S., Su, H., Qi, C.R., Guibas, L.J., 2016. Fpnn: Field probing neural networks for 3d data. Adv. Neural Inf. Process. Syst. 29.
Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R., 2015. Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493.
Loshchilov, I., Hutter, F., 2017. Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101.
Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V.G., Lipman, Y., 2017. Convolutional neural networks on surfaces via seamless toric covers. ACM Trans. Graph. 36 (4), 71.
Masci, J., Boscaini, D., Bronstein, M., Vandergheynst, P., 2015. Shapenet: Convolutional Neural Networks on Non-Euclidean Manifolds. Technical Report.
Monti, F., Boscaini, D., Masci, J., Rodola, E., Svoboda, J., Bronstein, M.M., 2017. Geometric deep learning on graphs and manifolds using mixture model cnns. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 5115–5124.
Nguyen, A., Le, B., 2013. 3D point cloud segmentation: A survey. In: 2013 6th IEEE Conference on Robotics, Automation and Mechatronics. RAM, IEEE, pp. 225–230.
Nguyen, D., Nguyen, T., Phung, D., 2019. Universal self-attention network for graph classification. arXiv preprint arXiv:1909.11855.
Niepert, M., Ahmed, M., Kutzkov, K., 2016. Learning convolutional neural networks for graphs. In: International Conference on Machine Learning. PMLR, pp. 2014–2023.
Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 652–660.
Qiao, Y.-L., Gao, L., Yang, J., Rosin, P.L., Lai, Y.-K., Chen, X., 2019. LaplacianNet: Learning on 3D meshes with Laplacian encoding and pooling. arXiv preprint arXiv:1910.14063.
Riegler, G., Osman Ulusoy, A., Geiger, A., 2017. Octnet: Learning deep 3d representations at high resolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 3577–3586.
Rodrigues, R.S., Morgado, J.F., Gomes, A.J., 2018. Part-based mesh segmentation: a survey. In: Computer Graphics Forum, Vol. 37. Wiley Online Library, pp. 235–274.
Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. Springer, pp. 234–241.
Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M., 2022a. Diffusionnet: Discretization agnostic learning on surfaces. ACM Trans. Graph. 41 (3), 1–16.
Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M., 2022b. DiffusionNet: Discretization Agnostic Learning on Surfaces. ACM Trans. Graph. 41 (3), 1–16. http://dx.doi.org/10.1145/3507905, URL: https://hal.science/hal-03938034.
Sidi, O., van Kaick, O., Kleiman, Y., Zhang, H., Cohen-Or, D., 2011. Unsupervised co-segmentation of a set of shapes via descriptor-space spectral clustering. In: Proceedings of the 2011 SIGGRAPH Asia Conference. pp. 1–10.
Smirnov, D., Solomon, J., 2021. HodgeNet: learning spectral geometry on triangle meshes. ACM Trans. Graph. 40 (4), 1–11.
Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3d shape recognition. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 945–953.
Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2818–2826.
Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H., 2021. Training data-efficient image transformers & distillation through attention. In: International Conference on Machine Learning. PMLR, pp. 10347–10357.
Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I., 2017. Attention is all you need. Adv. Neural Inf. Process. Syst. 30.
Wang, Y., Asafi, S., Van Kaick, O., Zhang, H., Cohen-Or, D., Chen, B., 2012. Active co-analysis of a set of shapes. ACM Trans. Graph. 31 (6), 1–10.
Wang, P., Gan, Y., Shui, P., Yu, F., Zhang, Y., Chen, S., Sun, Z., 2018. 3D shape segmentation via shape fully convolutional networks. Comput. Graph. 76, 182–192.
Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M., 2019. Dynamic graph cnn for learning on point clouds. Acm Trans. Graph. (Tog) 38 (5), 1–12.
Ward, J.H., 1963. Hierarchical grouping to optimize an objective function. J. Amer. Statist. Assoc. 58 (301), 236–244.
Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3D shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1912–1920.
Xie, Z., Xu, K., Liu, L., Xiong, Y., 2014. 3D shape segmentation and labeling via extreme learning machine. In: Computer Graphics Forum, Vol. 33. Wiley Online Library, pp. 85–95.
Xiong, R., Yang, Y., He, D., Zheng, K., Zheng, S., Xing, C., Zhang, H., Lan, Y., Wang, L., Liu, T., 2020. On layer normalization in the transformer architecture. In: International Conference on Machine Learning. PMLR, pp. 10524–10533.
Xu, H., Dong, M., Zhong, Z., 2017. Directionally convolutional networks for 3D shape segmentation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2698–2707.
Xu, P., Joshi, C.K., Bresson, X., 2021. Multigraph transformer for free-hand sketch recognition. IEEE Trans. Neural Netw. Learn. Syst..
Yan, Y., Mao, Y., Li, B., 2018. Second: Sparsely embedded convolutional detection. Sensors 18 (10), 3337.
Yi, L., Su, H., Guo, X., Guibas, L.J., 2017. Syncspeccnn: Synchronized spectral cnn for 3d shape segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2282–2290.
Yin, T., Zhou, X., Krahenbuhl, P., 2021. Center-based 3d object detection and tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11784–11793.
You, J., Ying, R., Leskovec, J., 2019. Position-aware graph neural networks. In: International Conference on Machine Learning. PMLR, pp. 7134–7143.
Yun, S., Jeong, M., Kim, R., Kang, J., Kim, H.J., 2019. Graph transformer networks. Adv. Neural Inf. Process. Syst. 32.
Zhang, J., Zhang, H., Xia, C., Sun, L., 2020. Graph-bert: Only attention is needed for learning graph representations. arXiv preprint arXiv:2001.05140.
Zhou, D., Zheng, L., Han, J., He, J., 2020. A data-driven graph generative model for temporal interaction networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. pp. 401–411.
Zhuang, J., Liu, X., Zhuang, W., 2022. NGD-transformer: Navigation geodesic distance positional encoding with self-attention pooling for graph transformer on 3D triangle mesh. Symmetry 14 (10), 2050.