TopoGAT: Plug-and-Play Topological Graph Attention for Fine-Grained 3D Segmentation

Xinyu Jiang[1], Lech Szymanski[1], and Steven Mills[1]*

Ōtākou Whakaihu Waka / University of Otago
Ōtepoti / Dunedin, Aotearoa / New Zealand
jiaxi917@student.otago.ac.nz
lech.szymanski@otago.ac.nz steven.mills@otago.ac.nz

Abstract. Fine-grained 3D point cloud segmentation is essential for robot manipulation and CAD editing. A key unresolved challenge is that different parts of the same object often share similar local geometry, making part boundaries difficult to distinguish. Existing methods construct local patches via K-nearest search and rely on deeper and larger networks to learn discriminative features. However, these approaches mainly focus on local geometry and lack global structural awareness, which leads to ambiguous predictions under sampling noise and partial observations. In this work, we propose Topological Graph Attention Network (TopoGAT), a lightweight, plug-and-play backbone refinement module that integrates topological data analysis with Graph Attention Networks to introduce global structural information into point-wise feature learning. When combined with existing backbones, TopoGAT improves segmentation accuracy with a slight parameter increase. Extensive experiments on fine-grained 3D part segmentation validate the effectiveness of the proposed TopoGAT and show up to 1.0% improvement on ShapeNetPart dataset and 3.3% improvement on S3DIS dataset.

Keywords: Point Cloud Segmentation · Topological Data Analysis · Graph Learning

1 Introduction

Fine-grained 3D semantic segmentation is the task of assigning a semantic label to every point in a point cloud, focused on capturing subtle, part-level details. This capability is fundamental for applications requiring precise 3D geometric understanding, including robotic manipulation and augmented reality [1].

A key challenge in fine-grained 3D part segmentation is that different parts of the same object often share very similar geometry. In objects like chairs or airplanes, many parts are self-similar, making it hard to tell them apart—especially near boundaries, thin connections, or when the point cloud is noisy or incomplete. In these situations, getting the part labels right often requires understanding the

* Corresponding author.

global structure of the whole object, not just local geometry. However, most existing methods [9, 20, 21, 32] build local patches with KNN and focus on local features, so they struggle with this problem. Graph-based methods [19, 33, 34] like GNNs and GATs do better by capturing geometric structure, but they only model pairwise point relationships and miss higher-order structural patterns in complex 3D shapes.

Geometric and topological information in 3D point clouds provides important global structural information for fine-grained 3D segmentation. Topological Data Analysis (TDA) can extract such global features from complex data. In TDA, a process called **filtration** builds a series of topological structures at different scales from an input point cloud, allowing the model to capture relationships beyond local geometry. **Persistent homology** then can compute multi-scale topological features from filtration, typically represented as persistence diagrams. In these representations, features of different dimensions describe different structural properties of the point cloud, such as connected components (0D), loops or cycles (1D), and higher-order structures like cavities (2D).

In this work, we combine TDA [4] with deep learning in a novel Topological Graph Attention Network (TopoGAT), which incorporates persistent homology to regress Graph Attention [23]. This module is plug-and-play to any point cloud segmentation backbone, refining the backbone and enhancing ability to capture global structural features of 3D objects.

To the best of our knowledge, our work is the first to combine geometric learning and persistent homology to tackle the fine-grained 3D semantic segmentation problem. The main contributions of our work are:

- TopoEdge Attention, a topology-conditioned graph attention layer that uses the vectorized persistence diagram to modulate pairwise attention weights via FiLM, guiding local feature aggregation by global topological structure.
- TopoGAT, a novel 3D segmentation module, consists of multiple TopoEdge attention layers and designs a topology-aware feature fusion that combines persistent homology with geometric learning, capturing global topological structure to improve the segmentation performance of part boundaries.

The proposed method can be integrated into any 3D semantic segmentation backbone; with only a small number of additional parameters, it fine-tunes existing backbone networks and consistently improves segmentation performance. Experiments on a range of datasets demonstrate that TopoGAT-enhanced systems achieve state-of-the-art results on fine-grained semantic segmentation tasks.

2 Related Works

2.1 Deep Learning-based Methods

Deep neural networks have been very successful in 3D point cloud semantic segmentation tasks. PointNet [20] was the first to process raw point clouds as input directly, using shared MLPs on each point and max-pooling to get a global

feature for classification. However, it treats each point independently and ignores local structures. PointNet++ [21] fixes this by grouping nearby points with farthest point sampling and ball query, enabling multi-scale local feature learning. Recently, Transformer-based methods have been adopted for 3D point cloud processing. PointTransformer [32] applies vector self-attention on local point neighborhoods, improving segmentation and classification. Point-MAE [17] masks point patches and trains the model to reconstruct them, learning representations without labels. Point-M2AE [30] extends this with a multi-scale pyramid architecture for better capturing both local and global information. PointGPT [5] takes a different route by ordering point patches spatially and predicting them auto-regressively, akin to next-token prediction in language models.

However, all these methods rely on K-nearest neighbor (KNN) search, enlarging the receptive field through deeper networks. But in fine-grained segmentation, where different parts often share local similarities, these strategies can lead to false predictions.

2.2 Geometric Learning-based Methods

Many methods use graph-based networks to capture a 3D object's structure. DGCNN [26] builds dynamic graphs in feature space and proposes Edge Convolution to aggregate local information. GACNet [25] adds attention weights based on both spatial positions and learned features. GTNet [34] combines graph attention with global self-attention in one framework. These methods have made good progress in extracting local geometric features, but graph message passing only captures pairwise relations between points. Moreover, they rely on KNN to construct graphs, which fails to encode high-order structural information such as loops and cavities — a critical limitation for fine-grained object segmentation. Other works try to define convolution-like operations on irregular point clouds. KPConv [22] places learnable kernel points in 3D space to increase the receptive field and capture global structural information. TopoGAT addresses self-similarity and part-boundary distinguish issues by using persistent homology to improve semantic segmentation performance in the backbone.

2.3 Topology-driven point cloud learning

Recently, Topology-based methods have been used to improve point cloud segmentation and shape understanding. Beksi and Papanikolopoulos [3] used persistent homology to segment point clouds by tracking connected components, which is robust to noise but mainly works as a hand-crafted clustering method and does not learn semantic features. TopoSeg [15] and PHGCN [27] further combine persistent homology with neural networks through topological losses or PH-based descriptors, which helps reduce broken parts and improve topology consistency. However, these methods usually depend on fixed topological descriptors or training-time PH losses, which can be costly and may keep noisy or less useful persistence points. Compared with them, our TopoGAT learns to select important H_1 and H_2 persistent features from precomputed topological

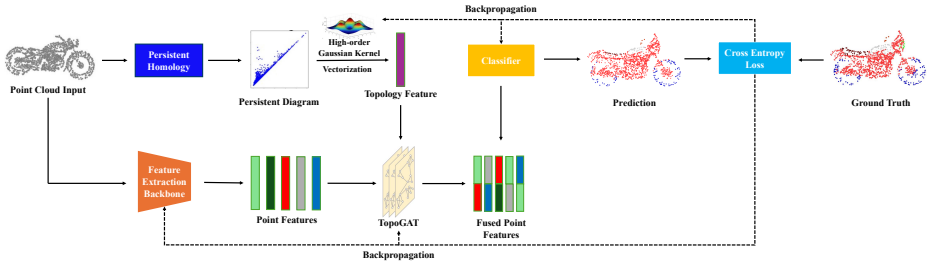

Fig. 1: The pipeline of the TopoGAT. The point cloud is fed into a feature extraction backbone (e.g. PointNet++) to generate point features; independently, the point cloud is used to generate the topology vector from the persistence diagram through persistent homology. Point features are concatenated with the topology vector and fed into TopoGAT as node features. Topology-informed point features produced by TopoGAT are fed into classifier layer for the final parts segmentation. The whole system is trained end-to-end, enabling the use of TopoGAT to refine the backbone.

signatures, producing compact topology-aware features that better complement geometric features for classification and part segmentation.

3 Method

The pipeline of our proposed TopoGAT network is shown in Fig. 1. Given an input point cloud $X = \{x_i\}_{i=1}^{N} \subset \mathbb{R}^{3\times N}$, we first employ a backbone (e.g. PointNet++ or Point-MAE) to extract local point-wise features h_i. Meanwhile, a persistent homology that captures global topological structure is computed from X and vectorized into a fixed-length vector. Instead of treating topology as an auxiliary global descriptor [8], TopoGAT integrates the vectorized persistence diagram into the graph attention computation, where topological information regulates the pairwise attention weights between points. This design enables topology-guided feature propagation and establishes a direct interaction between global structural and local geometric representations.

3.1 Persistence Diagram Generation

For a 3D point cloud set X, we compute persistent homology induced by an α-complex filtration. We first triangulate point cloud $Del(X)$, yielding a simplicial complex over the points. For a radius $r \geq 0$, define balls

$$B(x_i, r) = \left\{ y \in \mathbb{R}^3 \middle| \, \|y - x_i\| \leq r \right\} \tag{1}$$

where $y \in \mathbb{R}^3$ is any point in the 3D space (xyz coordinate). The alpha complex at scale r, denoted by $\mathcal{K}(r)$, is a subcomplex of $Del(X)$ that matches the shape of the union of balls $\bigcup_i B(x_i, r)$. We evaluate r on a discrete set $\{r_t\}_{t=0}^{T}$ with $r_0 < r_1 < ... < r_T$, which gives a nested filtration

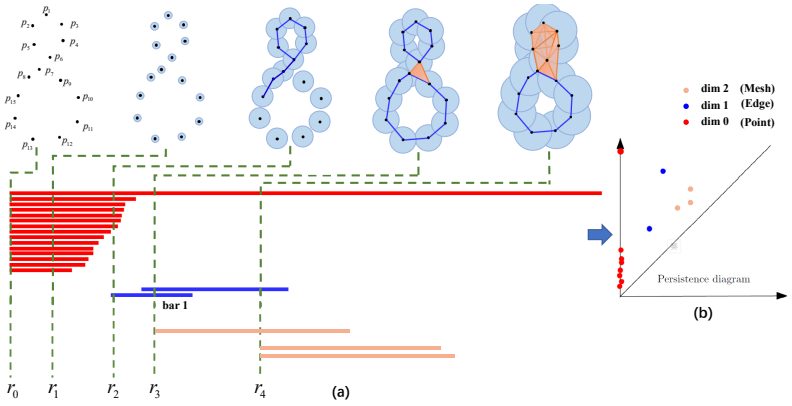

Fig. 2: The illustration of Persistent Homology. Dim 0/1/2 correspond to the simplicies of points, edges, meshes. (a) Filtration: each point is expanded with an equal-radius ball. At $r = 0$, only dim-0 components exist. As the radius grows, a dim-1 edge is born when two balls first touch and dies when the corresponding components merge or form a higher-order simplex (e.g., bar1 is born at r_2 and dies after r_3). Similarly, a dim-2 mesh is born ($\triangle p_6 p_7 p_9$) and it dies when the triangle is filled at a larger radius. Dim-0 features persist for all radii. (b) The (birth, death) pairs of all features forms the persistence diagram (PD_k).

$$\emptyset = \mathcal{K}(r_0) \subseteq \mathcal{K}(r_1) \subseteq \cdots \subseteq \mathcal{K}(r_T). \tag{2}$$

From this filtration, we compute persistent homology for $k \in S \subseteq \{0, 1, 2\}$, which represent the simplicies of point, edge, and mesh generation respectively. Let $C_k(r_t)$ be the k-chain group generated by the k-simplices in $\mathcal{K}(r_t)$ over a coefficient field $\mathbb{F}$. We obtain the boundary maps ∂_k: $C_k(r_t) \to C_{k-1}(r_t)$, and homology group:

$$H_k(\mathcal{K}(r_t); \mathbb{F}) = \ker(\partial_k)/\mathrm{im}(\partial_{k+1}). \tag{3}$$

Here, $\ker(\dots)$ and $\mathrm{im}(\dots)$ represent the kernel and image of a linear transformation respectively.

As r increases, new classes appear (birth) that later get merged with others (death). This gives persistence intervals $\left\{ \left[b_i^{(k)}, d_i^{(k)} \right) \right\}_{i=1}^{M_k}$, where $b_i^{(k)}$ and $d_i^{(k)}$ are the birth and death scales of a k-dimensional class. In the final stage, we generate a persistence diagram as 2D birth-death point sets:

$$\mathrm{PD}_k = \{(b_i^{(k)}, d_i^{(k)})\}_{i=1}^{M_k}, \quad k \in \mathcal{S}. \tag{4}$$

Each $\mathrm{PD}_k \subset \mathbb{R}^2$ is a birth-death point cloud.

3.2 Vectorization of Persistence Diagram

Filtration produces persistence diagram as 2D point sets, but we still cannot extract topological features directly from PD_k, because: 1) each 3D object produces a diagram with a different number of points M_k, yielding a variable-length inputs to networks; 2) the points in PD_k form an unordered set and most networks are order-sensitive; 3) the generated PD_k is noisy, as many birth–death points cluster near the diagonal, representing simplicial complexes that die immediately after birth and carry no useful information.

To address these issues, we convert the PD_k into a fixed-length, order-invariant vectors followed by a permutation-invariant pooling over the whole set. We write each persistence diagram point as $p_i = (b_i, d_i) \in \mathbb{R}^2$, and denote the number of points by $m := M_k$. We define a set of point-to-scalar functions $\varphi_j : \mathbb{R}^2 \to \mathbb{R}$, $j = 1, ..., K$. For each point p_i, the j-th function gives

$$f_{ij}^{p} = \varphi_j\left(p_i\right), \quad i = 1, ..., m. \tag{5}$$

Next, we use a higher-order Gaussian kernel for φ_j, expressed as $\varphi_{\mathrm{HOG}} : \mathbb{R}^2 \to \mathbb{R}$, where p_i is mapped to a scalar as follows:

$$\varphi_{\mathrm{HOG}}(p_i) = \exp\left\{-\left(\frac{(b_i-\mu_1)^2}{\sigma_1^2}\right)^{\rho_1} - \left(\frac{(d_i-\mu_2)^2}{\sigma_2^2}\right)^{\rho_2}\right\}, \tag{6}$$

where $\mu_i \in \mathbb{R}^2$, σ_i, $\rho_i \in \mathbb{R}_+^2$, $i = 1, 2$, in our implementation, we set these as learnable parameters.

We then aggregate the unordered set $\left\{f_{ij}^{p}\left(p_i\right)\right\}_{i=1}^{m}$ into a fixed-length vector using a weight function $\omega\left(p_i\right)$ and a permutation-invariant max pooling:

$$\mathbf{t} \ = \max\Big(\left\{\omega(p_i) \cdot f_{ij}^{p}(p_i)\right\}_{p_i \in \mathrm{PD}_k}\Big). \tag{7}$$

For the weight function $\omega\left(p_i\right)$, we use $\omega\left(p_i\right) = d_i - b_i \geq 0$ (persistence interval), which gives higher weights to points with longer lifetime. This filters out the points along the diagonal of the persistence diagram.

3.3 Topological Graph Attention Network

Vectorization of the persistence diagram produces fixed-length topology embedding $\mathbf{t}$, which we take to be a global representation of a 3D object. To impart that global representation into the backbone, we propose TopoGAT. We first feed the input points $\mathbf{X} = [x_1, ..., x_N] \in \mathbb{R}^{3\times N}$ into a backbone network $\mathcal{B}(\cdot)$. The backbone outputs a local feature for each point:

$$\mathbf{H} = \mathcal{B}(X) \in \mathbb{R}^{C\times N}, \quad \mathbf{H} = [\mathbf{h}_1, ..., \mathbf{h}_N], \tag{8}$$

where $\mathbf{h}_i \in \mathbb{R}^{C_b}$ is the feature vector of x_i. To let every point be aware of the global topological structure, we broadcast the topology embedding $\mathbf{t}$ to all points:

$$\mathbf{T} = \mathbf{t}\mathbf{1}_N^{\top} \in \mathbb{R}^{C_t\times N}. \tag{9}$$

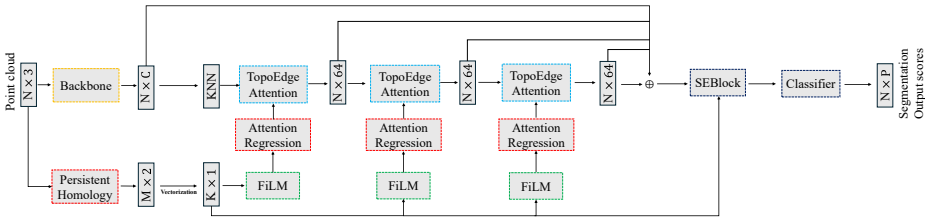

Fig. 3: Model architectures. TopoGAT takes N points as input, and the backbone computes $N \times C$ graph node features. Then a K-nearest search constructs graph, and features within each neighborhood are aggregated to compute TopoEdge Attention. The Vectorized Topology features regress the TopoEdge Attention via FiLM module. Finally, the aggregated features and vectorized topology features are fed into a SEBlock and a classifier to produce the final segmentation result.

We then concatenate $\mathbf{H}$ and $\mathbf{T}$ along the feature dimension from the initial graph node features, where each node carries both local and global topology:

$$\mathbf{Z}^{(0)} = [\mathbf{H}; \mathbf{T}] \tag{10}$$

TopoEdge Attention Design TopoEdge Attention is a topology conditioned graph attention layer. Its goal is to let the global topology embedding $\mathbf{t}$ guide how each point aggregates information from its local layers. Each layer ℓ consists of three steps: graph construction, topology-conditioned attention and neighborhood aggregation.

In each layer ℓ, we build a KNN graph by computing Euclidean distances in the feature space. For the first layer, the neighbors are determined by the initial node features $Z^{(0)}$. For the following layers, we use the updated features Z^{ℓ}. The KNN result $\mathcal{N}_i^{(\ell)}$ can be obtained from:

$$\mathcal{N}_i^{(\ell)} = kNN(z_i^{\ell}) \tag{11}$$

For an edge (i, j) with $j \in \mathcal{N}_i^{\ell}$, we construct edge features by concatenating the center node feature, the relative feature difference, and the relative spatial position:

$$\mathbf{e}_{ij}^{(\ell)} = \left[\mathbf{z}_i^{(\ell)};\ \mathbf{z}_j^{(\ell)} - \mathbf{z}_i^{(\ell)};\ \mathbf{x}_j - \mathbf{x}_i\right] \tag{12}$$

However, the edge features above only capture local pairwise relationships and cannot capture topological structure. To address this, we adopt Feature-wise Linear Modulation (FiLM) [18] to condition the edge features on the topology embedding $\mathbf{t}$, applying a learned per-channel scale and shift that allows topology to amplify or suppress certain feature dimensions.

Specifically, for each layer $\ell > 1$, a small conditioner network maps $\mathbf{t}$ to a scale vector γ and a shift vector β:

$$[\boldsymbol{\gamma}^{(\ell)};\boldsymbol{\beta}^{(\ell)}] = \phi_{\text{film}}^{(\ell)}(\mathbf{t}) \in \mathbb{R}^{2d\times 1}, \qquad \boldsymbol{\gamma}^{(\ell)},\boldsymbol{\beta}^{(\ell)} \in \mathbb{R}^{d\times 1}. \tag{13}$$

where $\phi_{\text{film}}^{(\ell)}$ is a 2-layer MLP

$$\phi_{\text{film}}^{(\ell)}(\mathbf{t}) = W_2^{(\ell)}\,\delta\Big(W_1^{(\ell)}\mathbf{t} + \mathbf{b}_1^{(\ell)}\Big) + \mathbf{b}_2^{(\ell)}, \tag{14}$$

and $\delta(\cdot)$ is the ReLU function. Using γ and β, we modulate each edge feature vector with a channel-wise affine transform:

$$\tilde{\mathbf{e}}_{ij}^{(\ell)} = \mathbf{e}_{ij}^{(\ell)} \odot \big(1 + \boldsymbol{\gamma}^{(\ell)}\big) + \boldsymbol{\beta}^{(\ell)}. \tag{15}$$

The modulated edge features $\tilde{\mathbf{e}}_{ij}$ now encode local geometry and global topology. We convert them into attention weights α_{ij}^{ℓ} by mapping each $\tilde{\mathbf{e}}_{ij}$ to a scalar logit and normalizing over all neighbors via softmax:

$$s_{ij}^{(\ell)} = \phi_{\text{out}}^{(\ell)}\Big(\tilde{\mathbf{e}}_{ij}^{(\ell)}\Big), \qquad \alpha_{ij}^{(\ell)} = \frac{\exp\Big(s_{ij}^{(\ell)}\Big)}{\sum\limits_{j'\in\mathcal{N}_i^{(\ell)}} \exp\Big(s_{ij'}^{(\ell)}\Big)} \tag{16}$$

With the topology-informed attention weights, we aggregate neighbor features for each point. This is the core message-passing step where global topology guides which neighbors contribute more to the updated representation:

$$\hat{\mathbf{z}}_i^{(\ell)} = \sum \alpha_{ij}^{(\ell)}\,\mathbf{z}_j^{(\ell)} \quad \mathbf{z}_j \in \mathcal{N}_i^{(\ell)}. \tag{17}$$

Also, to preserve each point's own information during aggregation, we add a residual connection through a linear projection:

$$\mathbf{r}_i^{(\ell)} = W_r^{(\ell)}\,\mathbf{z}_i^{(\ell)} \tag{18}$$

We combine the attended features and the residual term, then apply both max and mean pooling over the neighborhood to capture aggregation statistics:

$$\mathbf{z}_i^{(\ell+1)} = \Big[\max_{j\in\mathcal{N}_i^{(\ell)}} \big(\hat{\mathbf{z}}_i^{(\ell)} + \mathbf{r}_i^{(\ell)}\big)\ ;\ \text{mean}_{j\in\mathcal{N}_i^{(\ell)}}\big(\hat{\mathbf{z}}_i^{(\ell)} + \mathbf{r}_i^{(\ell)}\big)\Big], \tag{19}$$

the max pooling highlights the most salient feature in each channel, while mean pooling captures the overall distribution of the neighborhood. Concatenating both provides a richer representation than either alone. This gives $\mathbf{Z}^{(\ell+1)} = \big[\mathbf{z}_1^{(\ell+1)},\ldots,\mathbf{z}_N^{(\ell+1)}\big]$. We stack L layers and concatenate all outputs:

$$\mathbf{G} = \text{Concat}\big(\mathbf{Z}^{(1)},\ldots,\mathbf{Z}^{(L)}\big) \in \mathbb{R}^{C_g\times N} \tag{20}$$

Topology-Aware Feature Fusion After L TopoEdge Attention layers, we have three complementary representations: the backbone features $\mathbf{H}$, the topology embedding $\mathbf{T}$, and the graph features $\mathbf{G}$. We fuse them by concatenation:

$$\mathbf{U} = \big[\mathbf{H}; \mathbf{T}; \mathbf{G}\big] \in \mathbb{R}^{C_u \times N}, \quad \mathbf{U} = \big[\mathbf{u}_1, \ldots, \mathbf{u}_N\big]. \tag{21}$$

Since not all feature channels are equally important for segmentation, we apply a Squeeze-and-Excitation [10] (SE) block to learn channel-wise importance weights, we first update squeeze $\mathbf{U}$ by global average pooling across all points:

$$\mathbf{u}_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} \mathbf{u}_i \in \mathbb{R}^{C_u} \tag{22}$$

We flatten the topology embedding as $\mathbf{t}_{\text{flat}} = \text{vec}(\mathbf{t}) \in \mathbb{R}^{C_t}$, then concatenate and predict channel weights:

$$\mathbf{s} = \sigma(W_2\, \delta(W_1\, [\mathbf{u}_{\text{avg}}; \mathbf{t}_{\text{flat}}])) \in \mathbb{R}^{C_u} \tag{23}$$

where $\delta(\cdot)$ and $\sigma(\cdot)$ are the ReLU and Sigmoid functions respectively. The SE output is

$$\mathbf{U}' = \mathbf{U} \odot \big(\mathbf{s}\mathbf{1}_N^\top\big). \tag{24}$$

We then feed the SE output $\mathbf{U}'$ to classifier layer and get the segmentation prediction $\hat{y} \in \mathbb{R}^{N \times P}$, where P is the number of parts of 3D objects. Finally, we use the Cross-Entropy (CE) loss to train the backbone and TopoGAT.

4 Experiments

We evaluate TopoGAT on two standard benchmarks: ShapeNetPart [28] for object part segmentation, and S3DIS [2] for semantic segmentation.

4.1 Experimental Setup

To demonstrate the effectiveness of TopoGAT, we tested it with five backbone models: PointNet++ [21], DGCNN [26], Point-MAE [17], TopoPointNet++ [8] and PointMLP [16]. We train the backbone and our proposed TopoGAT with AdamW optimizer, an initial learning rate of $lr = 0.003$, weight decay 10^{-4}, with Cosine scheduler, and a batch size of 16 with 4×3090 GPUs. For ShapeNetPart segmentation, point clouds were randomly down-sampled to 2048 points. We set the KNN search $k = 30$ in all of the three graph layers. We ran the training and test 3 times and report the averaged results. The standard deviation is lower than $5e-3$. Moreover, we adopt two classical metrics **Instance mIOU**(Inst mIOU) and **Class Average mIOU** (Class mIOU) [20] to evaluate the segmentation accuracy of different part segmentation strategies. **To balance computational cost and effectiveness, we downsample each object to 256 points (ShapeNetPart) or 1024 points (S3DIS) for generating** PD_k.

Table 1: Overall mean Inter-over-Union (mIOU) on ShapeNetPart (the standard deviation is lower than 5e-3).

Method	Inst mIOU	Class mIOU	aero	bag	cap	car	chair	ear	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table
PointNet++	84.8	82.5	82.6	85.2	84.3	78.2	90.4	**75.4**	**91.5**	86.7	83.4	94.9	71.3	94.5	83.1	**59.1**	**77.2**	81.8
+TopoGAT	**85.8**	**83.0**	**83.6**	**87.1**	**86.3**	**80.0**	**91.0**	73.1	**91.5**	**86.8**	**84.9**	**95.6**	**72.3**	**94.8**	**83.6**	**59.1**	76.1	**82.6**
DGCNN	85.0	82.0	84.0	83.4	86.7	77.8	90.6	**77.7**	91.2	87.5	82.8	95.7	66.3	**95.8**	81.1	56.5	74.5	82.8
+TopoGAT	**85.9**	**82.5**	**84.1**	**84.2**	**88.0**	**78.4**	**91.1**	74.3	**91.7**	**88.1**	**85.1**	**96.0**	**66.9**	94.2	**81.4**	**57.3**	**75.4**	**83.4**
TopoPointNet++	84.7	**82.8**	81.8	**84.4**	87.0	77.7	89.2	**75.5**	91.3	85.7	79.9	95.4	71.0	**95.0**	83.0	**63.0**	**81.2**	83.4
+TopoGAT	**85.3**	**82.8**	**83.5**	81.5	**87.7**	**79.7**	**90.0**	70.0	**91.8**	**86.1**	**80.7**	**95.7**	**72.2**	94.9	**84.7**	61.8	81.1	**83.7**
PointMLP	85.7	83.7	83.7	85.1	84.6	80.7	91.1	72.4	**91.6**	87.0	85.0	95.8	**78.5**	**95.1**	**84.1**	**66.9**	76.7	80.9
+TopoGAT	**86.2**	**84.0**	**84.6**	84.2	**89.4**	**80.9**	**91.4**	**79.8**	91.4	**87.9**	**85.2**	**95.9**	75.1	95.0	81.8	62.6	**77.1**	**82.3**
PointMAE	86.1	84.4	84.9	**84.1**	88.9	80.5	91.6	78.5	**91.7**	**88.2**	85.7	96.2	75.9	94.9	84.7	65.5	77.2	**81.6**
+TopoGAT	**86.7**	**84.7**	**85.4**	82.6	**89.4**	**80.7**	**91.9**	**80.4**	91.6	88.0	**85.8**	**96.5**	**76.4**	**95.8**	**85.0**	**66.6**	**78.2**	**81.6**

4.2 3D Object Part Segmentation in ShapeNetPart

ShapeNetPart [28] is a benchmark for object-level part segmentation, comprising 16,880 models across 16 shape categories with 50 part labels in total. As shown in Table 1, TopoGAT consistently improves all baseline methods on both Instance mIOU and Class mIOU, regardless of backbone architecture—from classic point-based methods (PointNet, DGCNN) to MLP-based (PointMLP) and transformer-based (PointMAE) models. The gains are particularly pronounced on categories with complex geometry, such as cap, knife, and motorbike, where local features alone are insufficient for part discrimination. The best variant, PointMAE-TopoGAT, achieves 86.7% Instance mIOU, the highest among all compared methods. Overall, these results show that TopoGAT is a flexible and effective plug-and-play module, and the best variant built upon PointMAE reaches 86.7% Instance mIOU, which is the highest among all compared methods on ShapeNetPart dataset.

Table 2 further reveals an accuracy–efficiency trade-off. TopoGAT introduces only 1.4M additional parameters, nearly constant across backbones. With this small overhead, the performance even achieves higher than large transformer-based models. For example, PointNet++ has only **4.9M** parameters, and PointNet++ - TopoGAT becomes **4.9M + 1.4M** already approaches the Instance mIOU of transformer-based models with **19.5M − 27.1M** parameters, such as PointGPT, PointMAE, and ACT [7]. On stronger backbones, the benefit persists at the same modest cost: PointMAE-TopoGAT adds only **1.4M** parameters atop **27.1M** and achieves the best Instance mIOU of **86.7%**. These results confirm that TopoGAT is a lightweight, effective plug-and-play module capable of achieving state-of-the-art performance with minimal parameter overhead.

In Fig.4, we show some segmentation results on ShapeNetPart with and without TopoGAT. It can be observed that the baseline model tends to produce noisy and fragmented predictions near part boundaries, especially in regions where different parts share similar local geometry. With TopoGAT, the results are much cleaner, and the boundaries match the ground truth better. This shows that the topological information from persistent homology helps the model classify the part that are hard to distinguish by local geometry alone.

Table 2: Comparison of part segmentation performance on ShapeNetPart between existing state-of-the-art methods and their TopoGAT-enhanced variants

Methods	Reference	Cls. mIOU	Inst. mIOU	Params
PointNet [20]	CVPR 2017	80.4	83.7	3.6M
PointNet++ [21]	NeurIPS 2017	82.5	84.8	4.9M
DGCNN [26]	TOG 2019	82.0	85.0	1.3M
PointMLP [16]	ICLR 2022	83.7	85.7	12.6M
Point-BERT [29]	CVPR 2022	84.1	85.6	27.1M
Point-MAE [17]	ECCV 2022	84.4	86.1	27.1M
PointGPT-S [5]	NeurIPS 2023	84.1	86.2	19.5M
PointGPT-B [5]	NeurIPS 2023	84.5	86.5	82.1M
ACT-PointGST [14]	TPAMI 2025	84.0	85.8	32.6M
PointNet++-TopoGAT	**Ours**	83.0	85.8	4.9M + 1.4M
DGCNN-TopoGAT	**Ours**	82.5	85.9	1.3M + 1.4M
TopoPointNet++-TopoGAT	**Ours**	82.8	85.3	1.9M + 1.4M
PointMLP-TopoGAT	**Ours**	84.0	86.2	12.6M + 1.4M
PointMAE-TopoGAT	**Ours**	**84.7**	**86.7**	27.1M + 1.4M

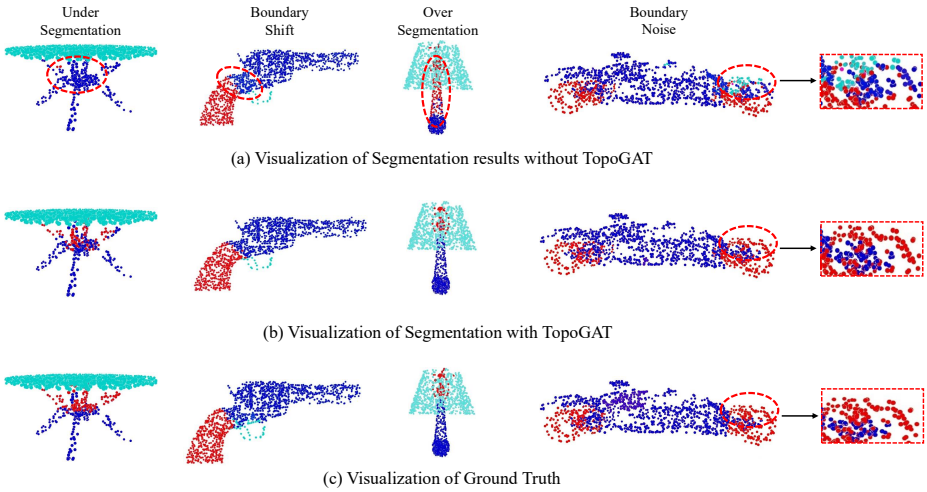

Fig. 4: The visualization of Segmentation Result. (a) The visualization of segmentation results without TopoGAT. (b) The visualization of segmentation results with TopoGAT. (c) The visualization of ground truth.

4.3 3D Semantic Segmentation in S3DIS dataset

To further validate the generalization of TopoGAT beyond part segmentation dataset on ShapeNetPart, we conduct additional experiments on S3DIS scene segmentation benchmark. We remove all per-point attributes like RGB color and only use **xyz coordinates as input**, so that any improvement comes purely from better geometric feature extraction. As shown in Table 3, TopoGAT brings clear mIOU improvement across all four backbones, ranging from 1.5 to 3.3 points. For example, PointNet++ and PointMLP, which use hierarchical sam-

Table 3: Overall mean Inter-over-Union (mIOU) on S3DIS dataset (the standard deviation is lower than 1e-2).

Method	Class Avg mIOU
PointNet++	52.5
PointNet++ - TopoGAT	**55.8**
DGCNN	53.6
DGCNN - TopoGAT	**55.5**
PointMLP	53.7
PointMLP - TopoGAT	**56.8**
PointMAE	60.4
PointMAE- TopoGAT	**61.9**

pling and simple affine transformations without building any topological structure, improve by 3.3 and 3.1 points respectively. DGCNN, which already builds dynamic graphs over local neighborhoods, sees a smaller but still meaningful gain of 1.9 points.

It is also worth noting that PointMAE, which uses self-supervised masked autoencoder pre-training to learn geometric features, still gets a 1.5 point boost from TopoGAT and reaches the best overall mIOU of 61.9%. This means the topological information provided by our module further refines pre-trained networks. The fact that TopoGAT works well with four very different backbone designs, from hierarchical point sampling to dynamic graph convolution to self-supervised pre-training, confirms that it is a general module that can extract useful geometric priors from coordinates alone.

Comparison with state-of-the-art plug-and-play modules To further validate the design of TopoGAT, we compare it against existing backbone refining networks as an ablation study, including PointWOLF [11], PointMixup [6], RSMix [12], SageMix [13], SN-Adapter [31], and PA Pooling [24]. As shown in Table 4, TopoGAT achieves 83.0% and 82.5% Class mIOU on PointNet++ and DGCNN respectively, doing better than PointWOLF on both backbones. The per-category results further show that TopoGAT does particularly well on categories like cap, motorbike, and table, which again suggests that topological modeling is especially helpful when the shape is complex or the part boundaries are hard to distinguish from local features alone. Table 5 provides a more direct comparison against existing plug-and-play methods, where TopoGAT reaches 85.8% and 85.9% Instance mIOU on PointNet++ and DGCNN respectively, outperforming the second-best method SN-Adapter 85.5%. This shows that introducing topological structure is a more effective way to boost part segmentation performance than other data augmentation-based approaches. In summary, table 4 and 5 confirm that TopoGAT is a strong and general plug-in module, and that leveraging topological structure is a well-justified design choice.

Table 4: Overall mean Inter-over-Union (mIOU) on ShapeNetPart Dataset and compared with PointWOLF.

Method	Class mIOU	aero	bag	cap	car	chair	ear	guitar	knife	lamp	laptop	motor	mug	pistol	rocket	skate	table
PointNet++	82.5	82.6	85.2	84.3	78.2	90.4	75.4	**91.5**	86.7	83.4	94.9	71.3	94.5	83.1	59.1	**77.2**	81.8
+PointWOLF	82.4	82.0	83.9	**87.3**	77.6	90.6	**78.4**	91.1	**87.6**	84.7	95.2	62.0	94.5	81.3	**62.5**	75.7	**83.2**
+TopoGAT	**83.0**	**83.6**	**87.1**	86.3	**80.0**	**91.0**	73.1	**91.5**	86.8	**84.9**	**95.6**	**72.3**	**94.8**	**83.6**	59.1	76.1	82.6
DGCNN	82.0	84.0	83.4	86.7	77.8	90.6	**77.7**	91.2	87.5	82.8	95.7	66.3	**95.8**	81.1	56.5	74.5	82.8
+PointWOLF	80.4	82.9	73.3	83.5	76.7	90.8	76.7	91.4	**89.2**	**85.2**	95.8	53.7	94.0	80.1	54.9	74.3	**83.4**
+TopoGAT	**82.5**	**84.1**	**84.2**	**88.0**	**78.4**	**91.1**	74.3	**91.7**	88.1	85.1	**96.0**	**66.9**	94.2	**81.4**	**57.3**	**75.4**	**83.4**

Table 5: Complete part segmentation results (Instance mIOU) on ShapeNetPart Dataset.

Model	Base	+SN-Adapter	+PointMixup	+RSMix	+SageMix	+PointWOLF	+ PA Pooling	+TopoGAT(Ours)
PointNet++	84.8	85.3	85.5	85.4	85.7	85.2	85.3	**85.8**
DGCNN	85.0	85.5	85.3	85.2	85.4	85.2	85.6	**85.9**

4.4 Experiments for Hyperparameter Sensitivity

Experiments for different k selection In TopoGAT, the neighborhood size k in the KNN graph controls the receptive field for attention computation. A different k means each point sees a different range of surrounding points. To study this effect, we fix $\ell = 3$ and vary k from 20 to 40; results are reported in Table 6. For PointNet++-TopoGAT, Instance mIOU is 85.4% at $k = 20$ and rises to 85.7% at $k = 25$, since a larger neighborhood enables each point to aggregate wider local context, making topology-guided attention more informative. Performance further improves to 85.8% at $k = 35$ and remains stable beyond that point. DGCNN-TopoGAT exhibits a similar trend but at a slightly higher level, starting at 85.5% (k=20) and reaching 85.8% by $k = 25$, with a marginal gain to 85.9% at $k = 30$ and $k = 40$. The consistently higher scores of DGCNN-TopoGAT can be attributed to its dynamic graph construction, which already captures richer local structure before TopoGAT is applied. Overall, both backbones show that moderate values of k (25-30) already capture effective local topology, and going beyond that does not help much.

Experiments for different Layer ℓ selection Another key factor in our TopoGAT is the number of TopoEdge Attentions ℓ. We fix $k = 30$ and set ℓ from 1 to 5. The results are shown in Table 7. For PointNet++-TopoGAT, with a single layer, the model gets 85.3% Instance mIOU, as one message-passing step limits each point to its direct neighbors, preventing topological information from propagating further. Performance steadily improves as ℓ increases, reaching 85.8% at $\ell = 3$, since stacking layers allows each point to aggregate information from a larger receptive field, resulting in a better topology-guided understanding of global structure. At $\ell = 4$ and $\ell = 5$ performance stays at 85.8%, suggesting that three layers are already enough to capture useful topological context for this backbone. DGCNN-TopoGAT follows a similar upward trend. However, unlike PointNet++-TopoGAT, it continues to improve to 86.0% at $\ell = 4$ and remains

Table 6: Complete part segmentation results (Instance mIOU) on ShapeNetPart

Model	k = 20	k = 25	k = 30	k = 35	k = 40
PointNet++-TopoGAT	85.4	85.7	85.8	85.8	85.8
DGCNN-TopoGAT	85.5	85.8	85.9	85.9	85.9

Table 7: Complete part segmentation results (Instance mIOU) on ShapeNetPart with different l selection.

Model	$\ell = 1$	$\ell = 2$	$\ell = 3$	$\ell = 4$	$\ell = 5$
PointNet++-TopoGAT	85.3	85.6	85.8	85.8	85.8
DGCNN-TopoGAT	85.5	85.6	85.8	86.0	86.0

stable at $\ell = 5$, suggesting that DGCNN benefits from deeper message passing, likely because its dynamic graph computation at each layer helps maintain feature diversity and avoids over-smoothing.

4.5 Ablation Study

To verify the contribution of each component in TopoGAT, we incrementally add each module on top of PointNet++ and DGCNN and report Instance mIOU in Table 8. Simply concatenating the topology embedding with backbone features (Eq. 10) yields 84.7% and 84.8%, showing almost no gain over the base models. This confirms that naively appending a global descriptor is insufficient — the network needs a mechanism to actively leverage topological information during feature aggregation. This also corroborates the findings in TopoPointNet++ [8], where directly injecting topological descriptors as auxiliary features yields limited Instance mIOU improvement, further confirming that a structured interaction mechanism such as FiLM-based attention regression is essential for the network to effectively leverage topological information.

Adding FiLM-based attention regression (Eq. 13–15) produces the largest gain, reaching 85.6% and 85.8%, because it allows topology to directly modulate pairwise attention weights rather than serving as passive input features. The SE block alone (Eq. 22–23) brings only modest improvement (84.9%/85.2%), but combining all three components in the full TopoGAT achieves the best results of 85.8% and 85.9%, indicating that FiLM conditioning is the primary driver while the SE block provides complementary channel-level refinement.

4.6 Computational Cost and Scalability

We further evaluate the computational cost of TopoGAT on ShapeNetPart. Table 9 reports the inference resources of different backbones, where the inference time excludes PH/PD generation. TopoGAT adds a stable overhead of about 6–7 ms per frame and 1–2 GB peak VRAM across different backbones. This

Table 8: Ablation study of TopoGAT components on ShapeNetPart (Instance mIOU).

Model	Base	+Topology Embedding Concat	+FiLM	+SE Block	+TopoGAT
PointNet++	84.8	84.7	85.6	84.9	**85.8**
DGCNN	85.0	84.8	85.8	85.2	**85.9**

overhead mainly comes from the extra KNN operations in the TopoGAT module. Even with this overhead, all tested models run within 9–19 ms per frame, showing that TopoGAT remains efficient for practical inference.

Table 9: Efficiency and PH-point trade-off on ShapeNetPart.

Inference resource (excludes PH generation)				
Backbone	VRAM (GB)	Time (ms/frame)	+TopoGAT VRAM (GB)	+TopoGAT Time (ms/frame)
PointNet++	3.59	10	4.96	16
DGCNN	3.34	5	4.61	12
PointMLP	3.71	3	5.52	9
PointMAE	4.34	12	6.11	19
PH generation				
PH points	128	256	512	1024
PH time (ms/sample)	5.82	7.13	14.26	36.81

5 Conclusion

In this paper, we proposed **TopoGAT**, a topology-aware refinement module that integrates persistent homology into graph attention for 3D point segmentation. TopoGAT extracts persistent homology from point clouds and encodes it as fixed-length vectorized representations, providing explicit topological information that complements local geometric features from the backbone network. Rather than directly concatenating topology descriptors with point-wise features, TopoGAT incorporates vectorized topological information into attention regression, enabling topology-guided feature propagation and adaptive reweighting of pairwise point interactions. Experiments show that TopoGAT serves as an effective plug-and-play module that consistently improves various backbone architectures with limited computational overhead, highlighting the value of explicitly modeling topological priors in attention-based geometric learning.

Acknowledgments

This work was supported by the Marsden Fund Council from Government funding, managed by Royal Society Te Apārangi through the project *3D Shape Analysis with Geometric Declarative Networks*.

References

1. Aksoy, E.E., Baci, S., Cavdar, S.: SalsaNet: Fast road and vehicle segmentation in LiDAR point clouds for autonomous driving. In: 2020 IEEE intelligent vehicles symposium (IV). pp. 926–932. IEEE (2020)
2. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3D semantic parsing of large-scale indoor spaces. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1534–1543 (2016)
3. Beksi, W.J., Papanikolopoulos, N.: 3D point cloud segmentation using topological persistence. In: 2016 IEEE International Conference on Robotics and Automation (ICRA). pp. 5046–5051. IEEE (2016)
4. Carrière, M., Chazal, F., Ike, Y., Lacombe, T., Royer, M., Umeda, Y.: PersLay: A neural network layer for persistence diagrams and new graph topological signatures. In: International Conference on Artificial Intelligence and Statistics. pp. 2786–2796. PMLR (2020)
5. Chen, G., Wang, M., Yang, Y., Yu, K., Yuan, L., Yue, Y.: PointGPT: Auto-regressively generative pre-training from point clouds. Advances in Neural Information Processing Systems **36**, 29667–29679 (2023)
6. Chen, Y., Hu, V.T., Gavves, E., Mensink, T., Mettes, P., Yang, P., Snoek, C.G.: PointMixup: Augmentation for point clouds. In: European Conference on Computer Vision. pp. 330–345. Springer (2020)
7. Dong, R., Qi, Z., Zhang, L., Zhang, J., Sun, J., Ge, Z., Yi, L., Ma, K.: Autoencoders as cross-modal teachers: Can pretrained 2D image transformers help 3D representation learning? In: The Eleventh International Conference on Learning Representations (2023), https://openreview.net/forum?id=8Oun8ZUVe8N
8. Guan, Z., Du, S., Liu, Q.: TopoLayer: A universal neural network layer for topological feature learning on point clouds using persistent homology. In: 2025 IEEE International Conference on Multimedia and Expo (ICME). pp. 1–6. IEEE (2025)
9. Guo, M.H., Cai, J.X., Liu, Z.N., Mu, T.J., Martin, R.R., Hu, S.M.: PCT: Point cloud transformer. Computational visual media **7**(2), 187–199 (2021)
10. Hu, J., Shen, L., Sun, G.: Squeeze-and-excitation networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7132–7141 (2018)
11. Kim, S., Lee, S., Hwang, D., Lee, J., Hwang, S.J., Kim, H.J.: Point cloud augmentation with weighted local transformations. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 548–557 (2021)
12. Lee, D., Lee, J., Lee, J., Lee, H., Lee, M., Woo, S., Lee, S.: Regularization strategy for point cloud via rigidly mixed sample. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15900–15909 (2021)
13. Lee, S., Jeon, M., Kim, I., Xiong, Y., Kim, H.J.: SageMix: Saliency-guided mixup for point clouds. Advances in Neural Information Processing Systems **35**, 23580–23592 (2022)
14. Liang, D., Feng, T., Zhou, X., Zhang, Y., Zou, Z., Bai, X.: Parameter-efficient fine-tuning in spectral domain for point cloud learning. IEEE Transactions on Pattern Analysis and Machine Intelligence (2025)
15. Liu, W., Guo, H., Zhang, W., Zang, Y., Wang, C., Li, J.: TopoSeg: Topology-aware segmentation for point clouds. In: IJCAI. pp. 1201–1208 (2022)
16. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In: International Conference on Learning Representations (2022), https://openreview.net/forum?id=3Pbra-_u76D

17. Pang, Y., Wang, W., Tay, F.E., Liu, W., Tian, Y., Yuan, L.: Masked autoencoders for point cloud self-supervised learning. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II. pp. 604–621. Springer (2022)
18. Perez, E., Strub, F., De Vries, H., Dumoulin, V., Courville, A.: FiLM: Visual reasoning with a general conditioning layer. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
19. Phan, A.V., Le Nguyen, M., Nguyen, Y.L.H., Bui, L.T.: DGCNN: A convolutional neural network over large-scale labeled graphs. Neural Networks **108**, 533–543 (2018)
20. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: 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 (2017)
21. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: PointNet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
22. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: KPConv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6411–6420 (2019)
23. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., Bengio, Y., et al.: Graph attention networks. In: International Conference on Learning Representations. vol. 6 (2018)
24. Wang, J., Xu, T., Song, L., Ding, L., Li, H., Jiang, P., Han, Y., Li, J.: PAPooling: Graph-based position adaptive aggregation of local geometry in point clouds. ACM Transactions on Multimedia Computing, Communications and Applications **21**(4), 1–18 (2025)
25. Wang, L., Huang, Y., Hou, Y., Zhang, S., Shan, J.: Graph attention convolution for point cloud semantic segmentation. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 10296–10305 (2019)
26. Wang, Y., Sun, Y., Liu, Z., Sarma, S.E., Bronstein, M.M., Solomon, J.M.: Dynamic graph CNN for learning on point clouds. ACM Transactions on Graphics (tog) **38**(5), 1–12 (2019)
27. Wong, C.C., Vong, C.M.: Persistent homology based graph convolution network for fine-grained 3D shape segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 7098–7107 (2021)
28. Yi, L., Kim, V.G., Ceylan, D., Shen, I.C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., Guibas, L.: A scalable active framework for region annotation in 3D shape collections. ACM Transactions on Graphics (ToG) **35**(6), 1–12 (2016)
29. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-BERT: Pre-training 3D point cloud transformers with masked point modeling. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 19313–19322 (2022)
30. Zhang, R., Guo, Z., Gao, P., Fang, R., Zhao, B., Wang, D., Qiao, Y., Li, H.: Point-M2AE: multi-scale masked autoencoders for hierarchical point cloud pre-training. Advances in neural information processing systems **35**, 27061–27074 (2022)
31. Zhang, R., Wang, L., Guo, Z., Shi, J.: Nearest neighbors meet deep neural networks for point cloud analysis. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 1246–1255 (2023)
32. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 16259–16268 (2021)

33. Zheng, Q., Qi, Y., Wang, C., Zhang, C., Sun, J.: PointViG: A lightweight GNN-based model for efficient point cloud analysis. arXiv preprint arXiv:2407.00921 (2024)
34. Zhou, W., Wang, Q., Jin, W., Shi, X., He, Y.: Graph transformer for 3d point clouds classification and semantic segmentation. Computers & Graphics **124**, 104050 (2024)