NoMeFormer: Non-Manifold Mesh Transformer

Mohammadreza Heidarianbaei, Mareike Dorozynski, Max Mehltretter , Franz Rottensteiner

Institute of Photogrammetry and GeoInformation, Leibniz University Hannover, Germany
(heidarianbaei, dorozynski, mehltretter, rottensteiner)@ipi.uni-hannover.de

Keywords: semantic segmentation, 3D texture mesh , transformer, non-manifold geometry

Abstract

Semantic segmentation of textured 3D meshes, i.e. the assignment of a class label to each triangle of such a mesh, is an important task in various fields. Existing deep learning models face problems when processing meshes with non-manifold structures. Most methods for 3D mesh classification rely on the assumption of manifold structure, which limits their applicability in real-world scenarios. To address this limitation, we propose NoMeFormer, a transformer-based framework specifically designed to handle any type of 3D mesh without imposing structural constraints, making it particularly suited for non-manifold mesh segmentation. A key innovation in our approach is the introduction of Local-Global (L-G) transformer blocks, which address the quadratic complexity of transformers. Initially, features are aggregated within spatial clusters of faces, followed by capturing long-range dependencies between faces via global attention. This architecture enables the model to effectively leverage both low- and high-frequency contextual information. Our experiments show that a variant of NoMeFormer based on geometrical features achieves a mean F1 score of 58.9% on the Hessigheim 3D benchmark dataset. Our framework overcomes the limitations of manifold-based approaches, offering a robust solution for semantic segmentation on non-manifold 3D meshes.

1. Introduction

The semantic segmentation of 3D meshes is the process of assigning class labels to the faces of the mesh. Compared to other 3D representations, 3D meshes offer distinct advantages, including geometric connectivity, memory efficiency, clear surface definition, and in case of a textured mesh detailed texture representation. The significance of 3D mesh segmentation and its applications has lead to the generation of datasets consisting of textured meshes, e.g. (Kölle et al., 2021; Gao et al., 2021).

In recent years, deep learning (DL) has solidified its position as a leading methodology for semantic segmentation and a range of other tasks (Ronneberger et al., 2015; Russakovsky et al., 2015). While most DL architectures in Photogrammetry and Computer Vision have been primarily developed for structured 2D data, the extension of these models to effectively learn representations from 3D data has emerged as a rapidly advancing area of research (Ioannidou et al., 2017; Ahmed et al., 2018; Mildenhall et al., 2021). Unlike images, which possess a universal grid-based representation, 3D data encompasses a variety of forms, including volumetric grids, point clouds, and meshes. Consequently, the adoption of deep learning frameworks for processing 3D data is inherently dependent on the chosen representation. Each type offers unique advantages and limitations concerning computational efficiency, data sparsity, and the capacity to capture geometric and other details.

Volumetric representations, akin to pixel-based image formats, present a straightforward data type that can be processed by traditional deep learning models. However, these representations are burdened by significant memory consumption, particularly at high resolutions, which constrains their scalability and overall resolution capabilities. This limitation makes these representations suboptimal for tasks that demand high surface detail and fine-grained information. In contrast, point clouds and meshes offer more efficient storage and greater surface fidelity. Point clouds represent discrete sets of points distributed in 3D space, capturing the geometric structure of an object's surface. In addition, meshes connect these points with edges and define faces to represent continuous surfaces, thereby enhancing the representation of the surface and mitigating ambiguities inherent in the surface representation by point clouds. This makes them ideal for 3D tasks, as they offer a detailed geometric structure while maintaining low memory overhead. Beyond that, meshes can also incorporate texture information through texture mapping, which might provide additional support for semantic segmentation.

While textured 3D meshes provide rich geometric and radiometric information, their irregular nature poses challenges for deep learning models, which are primarily designed for regular, grid-like input data. Consequently, most advancements in 3D mesh processing have focused on manifold meshes (Hanocka et al., 2019; Feng et al., 2019; Milano et al., 2020; Liang et al., 2022), which adhere to strict topological constraints. A manifold mesh is a mesh in which each edge is shared by exactly two faces, and in which vertices create a fan-like structure (Edelsbrunner and Harer, 2010). This constraint establishes a well-defined neighborhood relationship, which enables the adaptation of conventional DL architectures. However, the methods relying on manifolds cited above struggle to generalize to real-world scenarios in which non-manifold meshes, characterized by irregular connections and complex geometries, are prevalent. This is, for instance, the case in the meshes provided in the Hessigheim 3D dataset (Kölle et al., 2021).

To address the limitations just mentioned, we propose the Non-Manifold Mesh Transformer (NoMeFormer). It directly processes textured non-manifold meshes, leveraging the flexibility of transformer architectures (Vaswani et al., 2017). This opens up new possibilities for tasks such as semantic segmentation, classification, and generative modeling on non-manifold mesh data. In our framework, each face of a mesh is represented as a token. Initially, a feature vector is computed for each face, which is derived from a combination of geometrical

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-G-2025-365-2025 | © Author(s) 2025. CC BY 4.0 License.

and textural properties of the face. To learn the feature representation of faces in a 3D mesh, we rely on transformer networks (Vaswani et al., 2017). This choice is driven by their inherent nature of being order-invariant and supporting variable-length inputs, which is beneficial for irregular data such as non-manifold meshes. By using transformers, we can disregard the rigid structure and topological constraints of the mesh, treating each face independently while capturing spatial and topological relationships through positional encodings. This allows the model to learn meaningful representations without directly relying on the mesh's topological relationships.

However, the quadratic complexity of transformer architectures inhibits interactions among all tokens. To mitigate this problem, inspired by (Chu et al., 2021), we propose a structure we call a Local-Global (L-G) transformer block which consists of two sub-blocks. In the the first one, *local block*, features are aggregated within spatial clusters of adjacent faces in the 3D mesh. The local block also gathers information of each cluster into a single learnable token, called *cluster token*. In the *global block*, each cluster token interacts with face tokens to capture global contextual information integrating long-range dependencies between spatially distant regions. Our contributions can be summarized as follows:

- We propose a new framework for the semantic segmentation of 3D meshes, utilizing a transformer network capable of processing arbitrary 3D meshes without the necessity of imposing manifold or other constraints.
- To address the quadratic complexity of transformers, we present a new approach that first clusters faces into patches based on their spatial proximity and then learns feature aggregation within local patches. Subsequently, a global block is implemented to capture global context and to consider long-range interactions.
- We design a distinct network branch to determine per-face feature vectors, integrating both geometrical and textural information.
- We evaluate our model using a publicly available benchmark dataset, conducting a comprehensive ablation study to assess the effectiveness of its individual components.

2. Related Work

In this section, we first present a brief review of deep learning-based 3D data processing techniques that utilize representations other than meshes, such as point clouds and voxels. Afterwards, we provide a detailed survey of research focused on deep learning-based 3D mesh analysis and processing, with particular emphasis on semantic segmentation.

Early approaches for processing 3D data commonly represent data as a voxel grid. Analogously to 2D image structures, this representation enables the straightforward application of Convolutional Neural Networks (CNNs) for 3D data analysis (Maturana and Scherer, 2015; Wu et al., 2015). However, such approaches are affected by high memory consumption and computational inefficiency, especially when dealing with data of high resolution. While some efforts have been made to address these challenges, e.g. by hierarchically partitioning 3D space into octrees (Riegler et al., 2017), more recent work has focused on processing point clouds, a more memory-efficient and flexible way of representing the geometry of 3D objects (Qi et al., 2017a,b; Hu et al., 2020; Qian et al., 2022). The success of transformers across various domains along with their inherent order-invariance, which eliminates the need to define the order of point cloud data, has led researchers to explore their application in point cloud processing (Guo et al., 2021; Yu et al., 2022).

The aforementioned methods are tailored for 3D representations other than meshes. In contrast, textured meshes integrate geometric and radiometric data, capturing both an object's shape and detailed surface characteristics, such as color and texture. This richer representation can improve the prospects of success for tasks such as semantic segmentation in complex environments. It enables a context-aware analysis, reducing ambiguities in learning geometric and radiometric features, and provides a more informative representation than point clouds or voxel grids for tasks requiring high-fidelity surface information.

However, due to the irregular structure of meshed data, traditional DL models originally designed for regular grid-based inputs cannot be applied directly. This limitation has driven significant interest in adapting these models, as well as advancing geometric deep learning to address such data modalities (Bronstein et al., 2021). Whereas Laupheimer (2022) proposes to transfer the problem to point cloud or image classification by specific transfer functions, there have also been significant efforts to extend the concepts used in CNNs to direct processing of meshes. Masci et al. (2015) introduced a notion of convolutions for non-Euclidean domains. They extend traditional CNNs to curved surfaces, represented as Riemannian manifolds, by employing geodesic polar coordinates for local patches instead of the standard grid structure used in Euclidean space. MeshNet (Feng et al., 2019) is a DL network designed specifically to operate on 3D mesh faces. It incorporates two key descriptors: a spatial descriptor, which captures positional information via the center of gravity of each face, and a structural descriptor, which extracts geometric features. MeshNet also enhances spatial feature aggregation through mesh convolution layers that expand the receptive field by leveraging neighboring face indices.

Hu et al. (2022) continue the effort to adapt CNNs for mesh processing, with a particular focus on pooling layers to expand the receptive field. Their approach utilizes subdivision surfaces to construct a fine-to-coarse hierarchy, analogous to pooling operations in CNNs. MeshCNN (Hanocka et al., 2019) presents a different paradigm in mesh-based CNNs by focusing on edges as the primary entities for classification, extending the concept of convolutions to mesh structures. MeshCNN employs an edge collapse pooling mechanism, in which the network learns which edges to collapse, thereby dynamically adjusting the topology to improve the receptive field and hierarchical feature learning.

PD-Mesh (Milano et al., 2020) considers a mesh as a graph and extends point-wise convolution to mesh processing. The authors construct two mesh graphs: one in which nodes represent faces and another one in which nodes represent edges. Features from adjacent nodes in both graphs are aggregated using a Graph Attention Network (GAT). To mimic pooling operations, a mesh simplification technique is applied. MeshWalker (Lahav and Tal, 2020) defines a random walk on the vertices of the mesh and leverages the sequential nature of this process by using recurrent neural networks to learn mesh representations.

All the methods just mentioned impose the manifold constraints on input meshes, restricting the networks' ability to process arbitrary 3D meshes. Laplacian2Mesh (Dong et al., 2022) tries to

alleviate this problem by transforming the mesh into the spectral domain using the Laplacian matrix of the mesh. After processing the mesh in this spectral domain, the intermediate output is transformed back to the spatial domain. Although this method can handle non-manifold meshes, it is primarily designed for vertex rather than face segmentation. Additionally, the transformation to the spectral domain may lead to a loss of fine-grained spatial details. DiffusionNet (Sharp et al., 2022) is designed around three key components: pointwise perceptrons, learned diffusion, and spatial gradient features. Additionally, a Multi-Layer Perceptron (MLP) is employed to process the raw features of each vertex alongside spatial gradient features, enhancing directional filtering. While this network can process various types of meshes, feature aggregation is limited to continuous and local fields of view due to its diffusion equation-based approach. Tutzauer et al. (2019) integrate feature engineering and feature learning for semantic segmentation of urban triangle meshes. For each face, a multi-scale feature vector is computed and fed into a 1D CNN. However, instead of relying on convolutional kernels to capture spatial and contextual information, this method leverages features in different spherical neighbourhoods to encode spatial context. The convolutional kernels serve as feature embedding mechanisms rather than primary spatial aggregators.

Inspired by the success of Transformer models in Natural Language Processing (NLP), other disciplines have also adopted such architectures. This has resulted in the development of masked autoencoders (He et al., 2022) for self-supervised training based on mesh data (Liang et al., 2022). To address the quadratic complexity associated with attention mechanisms, this work leverages patches of data similar to those used in the Vision Transformer (ViT) (Dosovitskiy et al., 2021). However, due to the irregular structure of meshes, traditional patching methods based on pixel grids cannot be directly applied; the algorithm for merging mesh faces described in (Lee et al., 1998) is used to generate a coarser representation. The merged faces are dealt with as single patches, concatenating their features and using patch embeddings to create tokens. However, this mesh simplification method introduces constraints on the input mesh and faces challenges when processing non-manifold meshes.

To the best of our knowledge, no prior work has adapted deep neural networks for processing non-manifold meshes. While considerable research has been conducted on the semantic segmentation and classification of 3D meshes with promising accuracy, nearly all methods impose the manifold constraint on the input meshes. Methods that do bypass this requirement often suffer from limitations, including spectral domain information loss, constrained local feature aggregation, or discontinuities in feature representation. The most closely related work that integrates transformers is (Liang et al., 2022), but it also relies on manifoldness for generating patches. To address these limitations, we leverage the unique property of transformers that permits order-invariant processing, eliminating the need for structural constraints on the input meshes. Further, to make the transformers applicable to large meshes, we draw inspiration from the two-step processing strategy for efficient image processing based on transformers used in (Chu et al., 2021).

3. Non-Manifold Mesh Transformer

In this section, we present our proposed methodology for semantic segmentation of textured non-manifold meshes. Section 3.1 presents an overview of our method. The two main components of the architecture, the feature extraction and the Local-Global transformer branches, are explained in Sections 3.2 and 3.3, respectively. Section 3.4 presents the classification head, whereas Section 3.5 outlines the training procedure.

3.1 Overview

The input of our method consists of a textured 3D triangulated mesh $\mathcal{M} = (\mathcal{V}, \mathcal{F}, \mathcal{T}, \mathcal{I})$, where $V = \{v_i \mid v_i \in \mathbb{R}^3\}$ is the set of vertices, each corresponding to a point in 3D space, and $F = \{f_i \mid f_i = (v_{a,i}, v_{b,i}, v_{c,i}),\ v_{a,i}, v_{b,i}, v_{c,i} \in V\}$ denotes the set of triangular faces, where each face f_i is defined by three vertices $v_{a,i}, v_{b,i}, v_{c,i}$. The texture is contained in a texture image $\mathcal{I} \in \mathbb{R}^{H \times W \times 3}$, and the faces are linked to their texture by texture coordinates, defined as a set of triplets $\mathcal{T} = \{[(x_{a,i}, y_{a,i}), (x_{b,i}, y_{b,i}), (x_{c,i}, y_{c,i})] \, \forall f_i \in \mathcal{F}\}$; each triplet contains the coordinates of the vertices of the corresponding face in the texture image $\mathcal{I}$, and the pixels inside the triangle given by these coordinates represent the texture of the face. The primary objective of our model is to perform semantic segmentation, i.e. to assign each face of the mesh to one of a set of predefined classes. The model output is represented as $\hat{\mathbf{Y}} \in \mathbb{R}^{|F| \times N_C}$, where $|F|$ is the number of faces and N_C is the number of predefined classes; $\hat{\mathbf{Y}}$ thus contains a vector of class scores for every face of the mesh, and the class label is defined as the class having the maximum score.

Transformers require inputs in the form of sequences of tokens, but directly using all faces of a 3D mesh as tokens is computationally infeasible due to the quadratic complexity $O(n^2)$ of the attention mechanism, where n is the number of tokens. Handling large meshes becomes prohibitively expensive, necessitating strategies to reduce the number of tokens or to sparsify the attention mechanisms. Methods such as the ViT (Dosovitskiy et al., 2021) split an image into patches (i.e., small windows of pixels) and consider each patch to be a token. However, this approach is not directly applicable to 3D meshes due to their irregular structure, and it also leads to a loss of important local information. As an alternative, we propose a combination of local and global processing, applying attention-based processing to tokens corresponding to faces in local clusters and then exchanging information between clusters in a global processing step. Thus, before processing the mesh, the local clusters have to be defined. For that purpose, the mesh vertices V are partitioned into K clusters based on their spatial proximity according to the Euclidean distance using k-means clustering, e.g. (Bishop, 2006). After that, each face is assigned to the cluster containing the majority of its vertices; in cases where each vertex belongs to a different cluster, the face is randomly assigned to one of the clusters. This results in a representation of the mesh faces by

$$F = \bigcup_{k=1}^{K} C_k,$$

where $C_k = \{f_{k1}, f_{k2}, \ldots, f_{kN_k}\}$, f_{kj} is the j^{th} face in cluster C_k, and N_k is the number of faces in that cluster.

Our proposed network is illustrated in Figure 1. It consists of two main branches: the feature extraction branch and the Local-Global (L-G) transformer branch. The feature extraction branch extracts a feature vector F_{Fet} of dimensionality Fet for every face. This vector encodes both geometric and textural information related to a face, and the resultant vectors are collected in a tensor of dimension $|F| \times F_{Fet}$, where $|F|$ denotes the number of faces. The L-G transformer branch receives as an input the

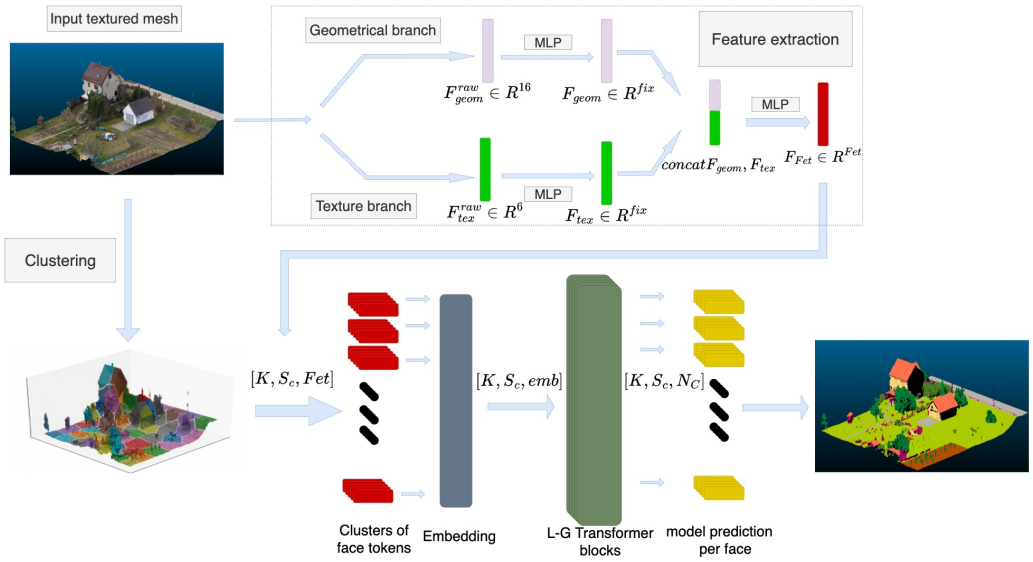

Figure 1. General architecture of our model. The feature extraction branch extracts a feature vector for every face, considering textural and geometrical information. K-means is used to generate clusters of faces. The face feature vectors are structured according to the clusters, and then they are passed on to a series of L-G transformer blocks. The output of the final block is processed by a classification head to yield class predictions. Numbers in brackets indicate the dimensionality of the tensors passed on to the subsequent blocks.

face clusters $C_1, \ldots, C_K$ generated in the way described above, each face being characterized by a feature vector F_{Fet} determined by the feature extraction branch. The data are subsequently formatted for input into the L-G transformer branch by structuring them in clusters, which allows a combination of local and global information exchange. The L-G transformer branch consists of multiple L-G blocks, the output of each block forming the input of the subsequent one. The output of the last L-G block is used by a classification head to predict the class scores and determine the final class labels. Details about all components are presented in the subsequent sections.

3.2 Feature Extraction Branch

This branch is designed to extract a feature vector for each face, considering both geometry and texture. Thus, it consists of two sub-branches: the first branch is dedicated to deriving geometrical features F_{geom} from original hand-crafted features, while the second one focuses on extracting textural features F_{tex}. The outputs of the two branches are concatenated along the feature dimension to form a combined vector $[F_{tex}, F_{geom}]$, which is then processed by a MLP to produce the final feature vector $F_{Fet} \in \mathbb{R}^{Fet}$ for every face. The geometric and texture branches are discussed in Sections 3.2.1 and 3.2.2, respectively.

3.2.1 Geometric Branch: This branch is designed to capture the geometrical properties of every face. We start by defining a set of 7 handcrafted features for each face, namely the face's area, its normal vector, and the angles between the face and its neighbours; the resultant feature vectors are denoted by $F_{geom}^{hc} \in \mathbb{R}^7$. Additionally, the raw 3D vertex coordinates, normalized relative to the center of each cluster and represented as a nine-dimensional vector, are concatenated with the handcrafted geometric features to form the 16-dimensional feature vectors $F_{geom}^{raw} \in \mathbb{R}^{16}$. The inclusion of the vertex coordinates is crucial to incorporate positional information as an inductive bias in the attention mechanism, but they may also help in extracting high-level geometric features beyond the handcrafted ones. The feature vector is processed by a MLP to yield a final geometric feature vector $F_{geom} \in \mathbb{R}^{fix}$ of dimension fix.

3.2.2 Texture Branch: As pointed out in Section 3.1, the texture coordinates $(x_{a,i}, y_{a,i}), (x_{b,i}, y_{b,i}), (x_{c,i}, y_{c,i})$ define a triangular region within the texture image for every face f_i, which encloses the set of pixels P corresponding to that face. The texture features are to be computed from this set of pixels. In our current implementation, we also use handcrafted features to represent the texture of each face. We calculate the mean and standard deviation for each channel from all pixels associated with a face, resulting in a six-dimensional feature vector $F_{tex}^{raw} \in \mathbb{R}^6$ per face. This feature vector is subsequently processed by a MLP to achieve a final texture feature vector $F_{tex} \in \mathbb{R}^{fix}$ having the same dimensionality as the output of the geometrical branch. Using hand-crafted features is not optimal for representing the rich information contained in the texture of a face. In the future, we intend to use transformers to extract the texture features, which could not yet be implemented due to time constraints.

3.3 L-G Transformer Branch

The input to the L-G transformer branch is represented as

$$X = \left\{ C_k^F = \{F_{Fet,k1}, F_{Fet,k2}, \ldots, F_{Fet,kN_k}\} \right\}_{k=1}^{K},$$

where C_k^F is a tensor containing the feature vectors of all faces in cluster C_k of the K clusters and $F_{Fet,ki}$ is the feature vector associated with the face f_{ki} in that cluster. While transformers can theoretically process sequences of arbitrary lengths, their implementation requires consistent dimensions across a batch. To address this, we pad the input within each cluster to match the length of the longest sequence of faces. This approach prevents interactions between padded tokens and face tokens by using a binary attention mask, assigning a value of 1 to face tokens and 0 to padded tokens. Therefore, we can represent the input as a tensor $X \in \mathbb{R}^{K \times S_c \times Fet}$, where S_c denotes the maximum number of faces in any cluster. This input X is passed through an embedding block, consisting of an MLP to transform each token from the dimension Fet into an embedding dimension emb, yielding a tensor $E \in \mathbb{R}^{K \times S_c \times emd}$ (cf. Fig. 1).

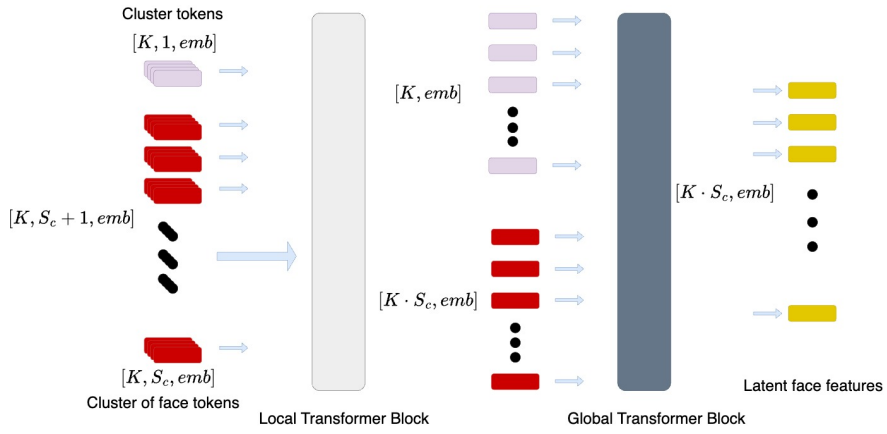

Figure 2. The L-G transformer block is comprised of two distinct components. The local transformer block focuses on learning fine-grained details within face clusters, generating two sets of sequences: cluster tokens (depicted in pink) and face tokens (depicted in red). Subsequently, the global transformer block processes these sequences through cross-attention mechanisms to effectively capture the global context.

This tensor E is subsequently passed through a series of L-G transformer blocks; the total number of blocks can be considered to be a hyperparameter (we use six blocks in our experiments). Each L-G block is specifically designed to extract both local and global contextual information from the input mesh. This is achieved through a dual structure comprising a local transformer sub-block and a global transformer sub-block, each optimized for its unique function, as elaborated in Sections 3.3.1 and 3.3.2, respectively. We define each L-G block as a pairing of one local and one global sub-block. The structure of an L-G block is shown in Figure 2.

3.3.1 Local Block: This block is designed to aggregate features by employing self-attention to model interactions between the faces of a cluster in a local manner. Additionally, inspired by (Devlin et al., 2019), a single learnable *cluster token* is concatenated with the face tokens in each cluster, transforming the input from $E \in \mathbb{R}^{K \times S_c \times emd}$ to $E_{cont} \in \mathbb{R}^{K \times (S_c+1) \times emb}$. Thus, the local transformer block can simultaneously learn high-level features of the faces inside a cluster while summarizing them into this dedicated *cluster token*, which is later used in the global block for long-range feature aggregation.

Each local block applies a standard transformer block (Vaswani et al., 2017), linearly projecting the input to key, query, and value matrices and applying multi-head self-attention within the tokens in each cluster. The self-attention captures intra-cluster dependencies and aggregates local features of faces, but there is no interaction between faces from different clusters at this stage. The output of this block passes through layer normalization and a MLP consisting of two fully connected layers with a ReLU activations, along with residual connections. The output of the local block retains the input shape as $Z_{local} \in \mathbb{R}^{K \times (S_c+1) \times emb}$, representing higher-level latent local features.

3.3.2 Global Block: This block complements the local one by learning long-range dependencies across clusters, enabling the model to capture broader spatial patterns. First, the output Z_{local} of the local block is re-arranged by splitting it into two parts: a block of cluster tokens $Z_{clusters} \in \mathbb{R}^{K \times emb}$ and a block of face tokens $Z_{faces} \in \mathbb{R}^{(K \cdot S_c) \times emb}$ (cf. Fig. 2). In the global block, cross-attention is applied, allowing the model to compute attention between cluster and face tokens, thus capturing inter-cluster dependencies. In the cross-attention block, the sequence $Z_{clusters}$ of cluster tokens is used to generate the key and value matrices, while the sequence Z_{faces} of face tokens provides queries. Just like in the local block, the architecture follows the original transformer model (Vaswani et al., 2017), but it uses cross-attention instead of self-attention. The result is a tensor $Z^o_{global} \in \mathbb{R}^{(K \cdot S_c) \times emb}$. This tensor is re-shaped to the original cluster-based partitioning, which can be done because we know which faces belong to which cluster, resulting in an output tensor $\mathbf{Z}_{global} \in \mathbb{R}^{K \times S_c \times emb}$, which forms the input for the subsequent block.

The cross-attention mechanism used in the global block significantly reduces the computational complexity. With $|F|$ denoting the total number of faces, the complexity is reduced from $O(|F|^2)$ to $O(|F| \cdot K)$. This reduction is achieved through the use of cluster tokens, making the approach more efficient. This allows the model to efficiently capture global context across the entire mesh, whereas the local block still allows information flow and thus the consideration of context at cluster level.

3.4 Classification

The output Z_{global} of the last L-G transformer block is processed by a classification head. It consists of a MLP that transforms each feature vector corresponding to a face to a vector of raw class scores, which are normalized using the softmax function. The resultant normalized class scores are represented by a tensor of shape $K \times S_c \times N_C$. Removing the padding tokens in accordance with the attention mask, a representation in the original cluster-based structure is generated, which is flattened to obtain the result $\hat{\mathbf{Y}}$ as defined in Section 3.1.

3.5 Network Training

Training is based on the categorical cross-entropy loss L_{ce} to measure the divergence between the predicted probabilities for each class associated with individual faces and their corresponding true labels. It aggregates the loss across all faces in the mesh and all classes:

$$L_{ce} = \sum_{i=1}^{|F|} \sum_{c=1}^{N_C} y_{ic} \cdot \log(\hat{y}_{ic}), \tag{1}$$

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-G-2025-365-2025 | © Author(s) 2025. CC BY 4.0 License.

where $|F|$ represents the total number of faces in the mesh, N_C denotes the total number of classes, y_{ic} is the ground truth binary indicator for face i to belong to class c ($y_{ic} = 1$) or not ($y_{ic} = 0$), and $\hat{y}_{ic}$ is the predicted probability for the face i to belong to class c.

4. Experiments and Results

4.1 Dataset

To evaluate our proposed model, we used the Hessigheim 3D (H3D) dataset (Kölle et al., 2021). It includes high-resolution point clouds and non-manifold textured 3D meshes. Our study focuses solely on the mesh modality (H3D-Mesh). The dataset consists of three subsets for training, validation, and testing, respectively. However, as the test set was not publicly available at the time of writing, we only used the training and validation sets, comprising 9,236,637 and 2,577,554 faces, respectively; the training set is used for training our model, whereas the validation set of the benchmark is used for evaluation, i.e. as our test set. The faces of the dataset are annotated according to a class structure with 11 semantic classes: *Low Vegetation, Impervious Surface, Vehicle, Urban Furniture, Roof, Façade, Shrub, Tree, Soil, Vertical Surface*, and *Chimney*. However, approximately 40% of the mesh faces remain unlabeled, primarily in regions where the mesh exceeds the annotated point cloud area. The data are split into tiles, which we process individually.

4.2 Experimental Protocol

The deep learning model is implemented using the PyTorch library (Paszke et al., 2017). All experiments are conducted on an NVIDIA A100 80GB GPU. The network hyperparameters were defined empirically. The number of L-G blocks is set to 6, with 2 attention heads per block and an embedding dimension of $emb = 256$. Additionally, $K = 300$ clusters are generated by k-means clustering. Given the extensive number of faces in these tiles, we opted to split the tiles after clustering, processing a fixed number of 50 clusters per forward pass, due to hardware limitations. In all experiments, we trained the model using the tiles from the training set of the H3D dataset, minimizing the loss defined in Section 3.5. The model parameters were initialized randomly using the approach described in (Glorot and Bengio, 2010), and minimization of the loss is performed using Adam (Kingma and Ba, 2014) with an initial learning rate of 0.0001, paired with a step-based scheduler that has a step size of 1000 and a decay factor of 0.9. Data augmentation was used to mitigate overfitting, applying random rotations by angles of up to ±45° about all three axes, random scaling with factors between 0.5 and 2, and adding Gaussian noise with a standard deviation of 0.01 to vertex positions to enhance the model's robustness against small geometric perturbations.

The trained model is used to classify the faces of the mesh tiles of the test set (i.e., the validation set of the benchmark - see above). To evaluate the model's performance, we compared the output of the classification to the given reference labels and determined two evaluation metrics: the mean F1 score $mF1$ and the overall accuracy OA. The *mean F1 score* is particularly useful for assessing classifier performance on imbalanced datasets, as it combines precision and recall of all classes into a single score. The F1 score for class i is:

$$\mathrm{F1}_i = \frac{2 \cdot \mathrm{TP}_i}{2 \cdot \mathrm{TP}_i + \mathrm{FP}_i + \mathrm{FN}_i}, \tag{2}$$

where TP_i, FP_i and FN_i are the numbers of true positives, false positives and false negatives for class i. The mean F1 score is the arithmetic mean of the $F1_i$ values over all classes i. The overall accuracy OA measures the proportion of correctly classified faces:

$$OA = \frac{\sum_{i=1}^{N_C} \mathrm{TP}_i}{|F|}. \tag{3}$$

4.3 Experimental Setup

We conducted a comprehensive series of experiments to evaluate our model and explore the efficacy of various of its components. Table 1 gives an overview about these experiments and settings. There are two groups of experiments involving our NoMeFormer: in one group, only the 6 local attention blocks were used in the L-G Transformer branch (experiments named NMF-L in Table 1), whereas the remaining ones used both local and global attention blocks (experiments named NMF-LG in Table 1); a comparison allows to assess the relevance of using the global blocks and, thus, to allow an exchange of information between clusters.

Within each group, we conduct four experiments, varying the definition of the feature vectors for the mesh faces. The first variants in each group, NMF-L_1 and NMF-LG_1, rely solely on the seven hand-crafted geometric features $F_{geom}^{hc} \in \mathbb{R}^7$, which is subsequently processed by a MLP to yield F_{Fet} with a dimensionality of $Fet = 64$. The second variants in each group, NMF-L_2 and NMF-LG_2, expand upon the first by incorporating the nine vertex coordinates into the geometric feature vectors. The resulting raw feature vectors $F_{geom}^{raw} \in \mathbb{R}^{16}$ are also processed by a MLP generating F_{Fet} with a dimensionality of $Fet = 64$. The results will show the relevance of the positional encoding for the attention mechanism.

In the third variants in each group, NMF-L_3 and NMF-LG_3, we enhance the feature representation by concatenating the raw hand-crafted texture features defined in Section 3.2.2 with the geometric features from the second variants (NMF-L_2 and NMF-LG_2). This integration enables the model to leverage both geometric and radiometric information. Specifically, we concatenate the two vectors, $F_{geom}^{raw} \in \mathbb{R}^{16}$ and $F_{tex}^{raw} \in \mathbb{R}^6$, resulting in a merged vector that is processed through an MLP, again producing F_{Fet} with a dimensionality of $Fet = 64$. Lastly, the feature extraction branch described in Section 3.2 is applied in the final variants (NMF-L_4 and NMF-LG_4) to extract the features for each face, with both fix and Fet set to 64.

As the test set of the benchmark was not publicly available at the time of writing, it was difficult to compare our results to the state-of-the art. In order to nevertheless allow a comparative analysis of our model's performance against existing methodologies, we also trained a Random Forest (RF) classifier, reported to achieve the highest scores according in (Kölle et al., 2021), and subsequently evaluated it on the validation set as a baseline for comparison with our results. The RF was trained using a combination of handcrafted geometrical and textural features that is identical to the combination of hand-crafted features used in the experiments NMF-L_3 and NMF-LG_3, resulting in a 22-dimensional feature vector per face. This is different from the features used in (Kölle et al., 2021). We selected these particular features as they are also integral to our proposed model. Whereas this does not really allow a comparison to the state of the art, it allows for a comparison of the results of our model to a classical machine learning approach based on the

Name	Model	Features
NMF-L_1	NMF-L	$F_{geom}^{hc} \in \mathbb{R}^7$
NMF-L_2	NMF-L	$F_{geom}^{raw} \in \mathbb{R}^{16}$
NMF-L_3	NMF-L	$[F_{geom}^{raw}, F_{tex}^{raw}] \in \mathbb{R}^{22}$
NMF-L_4	NMF-L	$FEX(F_{geom}^{raw}, F_{tex}^{raw})$
NMF-LG_1	NMF-L+G	$F_{geom}^{hc} \in \mathbb{R}^7$
NMF-LG_2	NMF-L+G	$F_{geom}^{raw} \in \mathbb{R}^{16}$
NMF-LG_3	NMF-L+G	$[F_{geom}^{raw}, F_{tex}^{raw}] \in \mathbb{R}^{22}$
NMF-LG_4	NMF-L+G	$FEX(F_{geom}^{raw}, F_{tex}^{raw})$
RF	RF	$[F_{geom}^{raw}, F_{tex}^{raw}] \in \mathbb{R}^{22}$

Table 1. Overview of the conducted experiments. Name: the name by which an experiment is referred to in the text. Model: the model used (NMF-L: NoMeFormer with local blocks only, NMF-L+G: NoMeFormer with local and global blocks; RF: Random Forest. Features: features used as input. $FEX(\cdot)$ indicates the use of the feature extraction branch to generate a feature vector for each face.

same set of raw features. We employed a RF with 100 trees and a maximum depth of 20, setting the number of features considered for the split at each node to 5. Nodes receiving fewer than 10 samples were not split further in the training process, and the Gini index was used to select the optimal separating surface in each node in the training procedure.

4.4 Results

Table 2 presents the mean F1 scores and the overall accuracy achieved in the experiments defined in Section 4.3. Comparing the results for the two groups of experiments with and without the global blocks in the L-G transformer, the positive effect of incorporating the global blocks is obvious: in all four variants involving different definitions of the feature vectors, the networks using both the local and global blocks (NMF-LG) outperform the networks only using local blocks (NMF-L) by a large margin (between 4.4% and 9.7% in $mF1$). This highlights the importance of considering long-range interactions in these global blocks and cluster tokens. By encoding the representation of feature sets into a single token and propagating it across the remaining faces in the mesh, the model captures global context efficiently, which leads to an improvement in the classification task. This result also confirms the suitability of the L-G Transformer blocks as described in Section 3.3 for the semantic segmentation of 3D meshes.

Name	$mF1$ [%]	OA [%]
NMF-L_1	45.9	52.0
NMF-L_2	49.2	53.1
NMF-L_3	42.7	48.9
NMF-L_4	46.8	50.6
NMF-LG_1	50.3	53.9
NMF-LG_2	**58.9**	**61.1**
NMF-LG_3	49.5	53.7
NMF-LG_4	52.4	56.8
RF	31.1	39.3

Table 2. Mean F1 score ($mF1$) and Overall Accuracy (OA) of the results obtained in the experiments involving different classification models. Name: name of the experiment according to Table 1.

As far as the ordering of the variants involving different definitions of the face features is concerned, it is identical for the two groups of experiments, so that in the following we only analyse the results achieved for the better network variant, i.e. the one using both local and global blocks (NMF-LG). The baseline model only using 7 hand-crafted features (NMF-LG_1) achieves competitive results, with a mean F1 score of 50.3%. Adding the 3D vertex coordinates results in the best performing model, yielding an improvement of 8.6% in $mF1$ and outperforming all other model variants by a large margin. It is worth noting that for this particular set of features, the improvement due to the use of global blocks (9.7% in $mF1$) is also considerable. This underscores the effectiveness of incorporating positional information as a means of imparting inductive bias to the model. This improvement may also be attributed to the model's capacity to learn more complex features from these positions, as opposed to relying mainly on relative geometric attributes.

The results in Table 1 show that integrating handcrafted textural features did not yield the expected results. While it was anticipated that textural features would improve the classification, particularly in cases where geometrical features are insufficient for distinguishing between certain classes, such as *Soil* and *Grass*, the inclusion of textural data actually resulted in a reduction of the mF1 score. Specifically, concatenating the hand-crafted geometrical and textural features (NMF-LG_3) leads to a notable 9.4% drop in $mF1$ performance compared to the version trained without textural information, NMF-LG_2; the OA is also reduced by 6.4%. While introducing the feature extraction branch (NMF-LG_4) partially mitigates this decline, improving performance by 2.9% in $mF1$ and 3.1% in OA, these metrics still remain substantially lower than those achieved by the variant NMF-LG_2 excluding textural information (6.5% in $mF1$, 4.3% in OA). This can probably be attributed to the method of incorporating textural information into the model. Specifically, textural data were represented by few statistical features, which captures the intricate and informative aspects of the texture inadequatly. This problem needs to be tackled by an improved textural feature extraction branch in future work.

As shown in Table 2, the classification results achieved by all variants of NoMeFormer are better than those of the RF by a large margin. The best NoMeFormer variant, NMF-LG_2, outperforms the RF by 25.8% in mean F1 score and by 21.8% in OA. This performance gain is consistent with expectations, as NoMeFormer's transformer-based architecture, equipped with L-G transformer blocks, enables effective feature aggregation across both local and global scales. In contrast, the RF, relying on handcrafted features only, lacks the capacity for representation learning and robust feature aggregation. It is anticipated that the RF would exhibit improved performance as feature complexity increases, but this is beyond the scope of this paper.

The most frequent misclassification errors occurred between geometrically similar classes, such as *Soil*, *Low-vegetation* and *Shrub* with *Urban Furniture*. These misclassifications are likely to be due to the close similarity of geometrical features of these classes and an imbalance in class representation within the training set. The first problem results in overlapping feature representation, whereas the class imbalance biases the model toward classes with a greater abundance of examples, ultimately leading to a higher rate of misclassification. Nevertheless, these most frequent classification errors are an additional indication that more work on obtaining expressive textural features is required, because the mentioned examples all occurred between vegetation and other objects.

5. Conclusions and Outlook

In this paper, we introduced NoMeFormer, a transformer-based network designed to process arbitrary 3D meshes without im-

posing manifold constraints. To effectively capture fine-grained details, high-frequency patterns, and global context while addressing the quadratic complexity of the attention mechanism, we propose the L-G block, a dual-transformer module that sequentially considers local and global dependencies. Using a minimal set of geometric input features, our experimental results demonstrate that NoMeFormer achieves a mean F1 score of 58.9% on the Hessigheim 3D benchmark dataset. This framework addresses key limitations of approaches imposing the manifold constraint on the input mesh, offering a framework for processing non-manifold 3D meshes, suitable for applications such as semantic segmentation.

There are several limitations that warrant further exploration. Incorporating textural information has been observed to inadvertently reduce the network classification capabilities, indicating a need for enhanced integration strategies, such as utilizing a distinct transformer-based network that encapsulates high-level features from pixel sets of each face into single vector representation. Additionally, due to the data-intensive nature of transformers, pretraining on large and diverse datasets is crucial for enabling the model to develop more generalizable features. Investigating self-supervised pretraining methods, such as those proposed in (He et al., 2022), may greatly enhance the model's ability to identify important patterns and adapt to various mesh types. This approach is expected to bolster its effectiveness for downstream applications, particularly in semantic segmentation tasks where task-specific data may be scarce.

Acknowledgments

The research reported in this paper was performed in the context of the project ChemiNova. ChemiNova has received funding from the European Union's Horizon Europe Framework Programme under grant agreement 101132442.

References

Ahmed, E., Saint, A., Shabayek, A. E. R., Cherenkova, K., Das, R., Gusev, G., Aouada, D., Ottersten, B. E., 2018. A survey on deep Learning advances on different 3D data representations. *arXiv: Computer Vision and Pattern Recognition*.

Bishop, C. M., 2006. *Pattern Recognition and Machine Learning*. 1^{st} edn, Springer, New York (NY), USA.

Bronstein, M. M., Bruna, J., Cohen, T., Velivckovi'c, P., 2021. Geometric Deep Learning: grids, groups, graphs, geodesics, and gauges. *ArXiv*, abs/2104.13478.

Chu, X., Tian, Z., Wang, Y., Zhang, B., Ren, H., Wei, X., Xia, H., Shen, C., 2021. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in Neural Information Processing Systems*, 34, 9355–9366.

Devlin, J., Chang, M.-W., Lee, K., Toutanova, K., 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *North American Chapter of the Association for Computational Linguistics*.

Dong, Q., Wang, Z., Gao, J., Chen, S., Shu, Z., Xin, S., 2022. Laplacian2Mesh: laplacian-based mesh understanding. *IEEE Transactions on Visualization and Computer Graphics*, 30, 4349-4361.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N., 2021. An image is worth 16x16 words: transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*.

Edelsbrunner, H., Harer, J. L., 2010. *Computational Topology: An Introduction*. American Mathematical Society, Providence, RI.

Feng, Y., Feng, Y., You, H., Zhao, X., Gao, Y., 2019. Meshnet: mesh neural network for 3d shape representation. *AAAI Conference on Artificial Intelligence*, 33(1), 8279–8286.

Gao, W., Nan, L., Boom, B., Ledoux, H., 2021. SUM: A benchmark dataset of semantic urban meshes. *ISPRS Journal of Photogrammetry and Remote Sensing*, 179, 108-120.

Glorot, X., Bengio, Y., 2010. Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, JMLR Workshop and Conference Proceedings, 249–256.

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R. R., Hu, S.-M., 2021. PCT: point cloud transformer. *Computational Visual Media*, 7, 187–199.

Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohen-Or, D., 2019. Meshcnn: a network with an edge. *ACM Transactions on Graphics (ToG)*, 38(4), 1–12.

He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R., 2022. Masked autoencoders are scalable vision learners. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16000–16009.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A., 2020. Randla-net: efficient semantic segmentation of large-scale point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11108–11117.

Hu, S.-M., Liu, Z.-N., Guo, M.-H., Cai, J.-X., Huang, J., Mu, T.-J., Martin, R. R., 2022. subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3), 1–16.

Ioannidou, A., Chatzilari, E., Nikolopoulos, S., Kompatsiaris, I., 2017. Deep learning advances in computer vision with 3d data: A survey. *ACM computing surveys (CSUR)*, 50(2), 1–38.

Kingma, D. P., Ba, J., 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rottensteiner, F., Wegner, J. D., Ledoux, H., 2021. The hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from uav LiDAR and multi-view-stereo. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 1, 11.

Lahav, A., Tal, A., 2020. Meshwalker: deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6), 1–13.

Laupheimer, D., 2022. On the information transfer between imagery, point clouds, and meshes for multi-modal semantics utilizing geospatial data. PhD thesis, Institute of Photogrammetry and Remote Sensing, University of Stuttgart, Germany.

Lee, A. W., Sweldens, W., Schröder, P., Cowsar, L., Dobkin, D., 1998. Maps: multiresolution adaptive parameterization of surfaces. *Proceedings of the 25th annual conference on Computer Graphics and Interactive Techniques*, 95–104.

Liang, Y., Zhao, S., Yu, B., Zhang, J., He, F., 2022. Meshmae: masked autoencoders for 3d mesh data analysis. *European Conference on Computer Vision*, 37–54.

Masci, J., Boscaini, D., Bronstein, M. M., Vandergheynst, P., 2015. geodesic convolutional neural networks on riemannian manifolds. *2015 IEEE International Conference on Computer Vision Workshop (ICCVW)*, 832-840.

Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition. *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 922–928.

Milano, F., Loquercio, A., Rosinol, A., Scaramuzza, D., Carlone, L., 2020. Primal-dual mesh convolutional neural networks. *Advances in Neural Information Processing Systems*, 33, 952–963.

Mildenhall, B., Srinivasan, P. P., Tancik, M., Barron, J. T., Ramamoorthi, R., Ng, R., 2021. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1), 99–106.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., Lerer, A., 2017. Automatic differentiation in pytorch. *NIPS 2017 Workshop on Autodiff*.

Qi, C. R., Su, H., Mo, K., Guibas, L. J., 2017a. Pointnet: deep learning on point sets for 3d classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–660.

Qi, C. R., Yi, L., Su, H., Guibas, L. J., 2017b. PointNet++: deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30.

Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B., 2022. PointNext: revisiting pointnet++ with improved training and scaling strategies. *Advances in Neural Information Processing Systems*, 35, 23192–23204.

Riegler, G., Osman Ulusoy, A., Geiger, A., 2017. Octnet: Learning deep 3d representations at high resolutions. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3577–3586.

Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation. *Medical Image Computing and Computer-assisted Intervention–MICCAI 2015, part III*, 234–241.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. et al., 2015. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115, 211–252.

Sharp, N., Attaiki, S., Crane, K., Ovsjanikov, M., 2022. Diffusionnet: discretization agnostic learning on surfaces. *ACM Transactions on Graphics (TOG)*, 41(3), 1–16.

Tutzauer, P., Laupheimer, D., Haala, N., 2019. Semantic urban mesh enhancement utilizing a hybrid model. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, 175–182.

Vaswani, A., Shazeer, N. M., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. *Neural Information Processing Systems*.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes. *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 1912–1920.

Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J., 2022. Point-bert: pre-training 3d point cloud transformers with masked point modeling. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19313–19322.