MeshMAE: Masked Autoencoders for 3D Mesh Data Analysis

Yaqian Liang[1]*, Shanshan Zhao[2], Baosheng Yu[3], Jing Zhang[3], and Fazhi He[1]

[1] School of Computer Science, Wuhan University, China
[2] JD Explore Academy, China
[3] School of Computer Science, The University of Sydney, Australia
{yqliang, fzhe}@whu.edu.cn, {sshan.zhao00}@gmail.com
{baosheng.yu, jing.zhang1}@sydney.edu.au

Abstract. Recently, self-supervised pre-training has advanced Vision Transformers on various tasks *w.r.t.* different data modalities, *e.g.*, image and 3D point cloud data. In this paper, we explore this learning paradigm for 3D mesh data analysis based on Transformers. Since applying Transformer architectures to new modalities is usually non-trivial, we first adapt Vision Transformer to 3D mesh data processing, *i.e.,* Mesh Transformer. In specific, we divide a mesh into several non-overlapping local patches with each containing the same number of faces and use the 3D position of each patch's center point to form positional embeddings. Inspired by MAE, we explore how pre-training on 3D mesh data with the Transformer-based structure benefits downstream 3D mesh analysis tasks. We first randomly mask some patches of the mesh and feed the corrupted mesh into Mesh Transformers. Then, through reconstructing the information of masked patches, the network is capable of learning discriminative representations for mesh data. Therefore, we name our method MeshMAE, which can yield state-of-the-art or comparable performance on mesh analysis tasks, *i.e.,* classification and segmentation. In addition, we also conduct comprehensive ablation studies to show the effectiveness of key designs in our method.

Keywords: Transformer, Masked autoencoding, 3D mesh analysis, Self-supervised pre-training

1 Introduction

In recent years, Transformers [55] have been the technological dominant architecture in NLP community [7,31,16]. With autoregressive language modeling [46,47] or masked autoencoding [16], Transformer architectures can be trained on a very large unlabeled corpus of text. In computer vision, to pre-train a generalizable vision model, many strategies have been intensively studied, such as contrastive

* This work was done during Y. Liang's internship at JD Explore Academy.

learning [27], egomotion prediction [2], and geometric transformation recognition [20]. Recently, inspired by the success of the masked autoencoding strategy in NLP, some works [5,64] also apply such a strategy to images or point clouds by reconstructing the masked tokens. Instead of predicting the tokens, a latest work, MAE [26], proposes to directly reconstruct raw pixels with a very impressive performance achieved, which provides a new self-supervised pre-training paradigm. Inspired by this, in this paper, we further explore the masked autoencoding strategy on 3D mesh data, whose structure is different from either 2D images or 3D point clouds.

As an effective 3D representation, 3D mesh has been widely exploited in computer graphics, such as 3D rendering, model reconstruction, and animation [28,22,51]. Along with the development of deep learning, remarkable achievements have been made in various mesh analysis tasks by adopting deep neural networks, such as 3D mesh classification [19], segmentation [29], and 3D human reconstruction [36]. In comparison with a image, the 3D mesh is composed of vertices and faces, which have not the specific order. While the connection between vertices makes 3D mesh not completely discrete data as point cloud. In light of the distinct structure of mesh data, there are some key challenges that need to be solved before applying the masked autoencoding strategy to the 3D mesh.

In this paper, we start from adapting the vanilla Vision Transformer [18] to process mesh data, which remains unexplored. As there are lots of faces in a mesh, *e.g.,* from 10^3 to 10^5 in the original ModelNet40 (or ShapeNet), it is almost infeasible to apply the self-attention mechanism in Transformers over all faces. Inspired by [18], we split the original mesh into a set of non-overlapping mesh patches, which are regarded as the basic unit processed by the Transformer block. In addition, we calculate the positional embeddings in a naive manner since 3D mesh data inherently contains positional information. Specifically, the 3D coordinate of the center point of each mesh patch is used as the positional embedding. Finally, built upon the vanilla Vision Transformers, our Mesh Transformer processes these mesh embeddings via the multi-head self-attention layer and feedforward network sequentially. Basically, exploiting these operations, we can apply the Transformer architecture for 3D mesh data, *i.e.,* Mesh Transformer.

Inspired by MAE [26], we wonder whether performing masked autoencoding on the unlabeled 3D mesh data could also promote the ability of the network. To arrive at this, we are required to design an efficient reconstruction objective first. Unlike MAE [26], which reconstructs the raw pixels naturally, we choose to recover the geometric information of the masked patches. In detail, we achieve the reconstruction at two levels, *i.e.,* point-level and face-level. For point-level, we aim to predict the position of all points belonging to the masked patch, while for face-level, we regress the features for each face. Taking advantage of the two reconstruction objectives, the model can learn discriminative mesh representation in a self-supervised manner from large-scale unlabeled 3D mesh data.

On the basis of the above analysis, we present MeshMAE, a masked autoencoder network for Transformer-based 3D mesh pre-training. To show the effectiveness of the proposed pre-text task, we visualize the reconstructed meshes. We observe that our MeshMAE model correctly predicts the shape of masked mesh patches and infers diverse, holistic reconstructions through our decoder. We conduct the pre-training on ModelNet40 and ShapeNet respectively and conduct the fine-tuning and linear probing experiments for mesh classification. In addition, we also finetune the pre-trained network for the part segmentation task. The experimental results and comprehensive ablations demonstrate the effectiveness of our method. The main contributions of our method are as follows:

- We explore adapting the masked-autoencoding-based pre-training strategy to 3D mesh analysis. To achieve this, we design a feasible Transformer-based network, Mesh Transformer, and reconstruction objectives in light of the distinctive structure of 3D mesh data.
- Our method achieves new state-of-the-art or comparable performance on the mesh classification and segmentation tasks. We also conduct intensive ablations to better understand the proposed MeshMAE.

2 Related works

2.1 Transformer

Transformers [55] were first introduced as an attention-based framework in NLP and currently have become the dominant framework in NLP [16,47] due to its salient benefits, including massively parallel computing, long-distance characteristics, and minimal inductive bias. Along with its remarkable success in NLP, the trend towards leveraging Transformer architecture into the computer vision community has also emerged. The Vision Transformer (ViT) [18] makes a significant attempt to exploit the self-attention mechanism, and has obtained many state-of-the-art (SOTA) records. So far, Transformers have already been extensively studied in various computer tasks, especially 2D image analysis tasks, such as image classification [9,18,52,61,65], object detection [68,8,14,42], semantic segmentation [58,67,11,17], pose estimation [36,30,34], and depth estimation [35,62]. However, the application of ViT in 3D data still remains limited.

Almost simultaneously, [23,63,66,53] proposes to extend Transformer architectures into the disordered and discrete 3D point cloud. Among them, [23] proposes an offset attention (OA) module to sharpen the attention weights and reduce the influence of noise. To save the computational costs, [66] proposes a local vector self-attention mechanism to construct a point Transformer layer. In comparison, to facilitate Transformer to better leverage the inductive bias of point clouds, [63] devises a geometry-aware block to model the local geometric relationships. Nevertheless, those prior efforts all involve more or less inductive biases, making them out of line with standard Transformers. As for meshes, to the best of our knowledge, there are no existing papers that directly apply Transformers to process the irregular mesh data. Instead, some works only use images

as the input of the Transformer and output mesh data. For example, Mesh-Graphormer [37] combines the graph convolutions and self-attentions to model both local and global interactions, METRO [36] proposes to utilize Transformer to learn the correlation among long-range joints and vertices in images. PolyGen [44] proposes to generate the vertices and faces in sequence with Transformer. In this work, we would explore how to apply the standard Transformer to 3D mesh data.

2.2 Self-supervised Learning

Self-supervised learning could learn the meaningful feature representations via pretext tasks that do not require extra annotations. For example, contrastive learning learns feature representations by increasing intra-class distance and decreasing extra-class distance [59,45]. In comparison, autoencoding pursues a conceptually different direction by mapping an input to a latent representation in an encoder and reconstructing the input using a decoder. In NLP, BERT [16] first attempts to pre-train bidirectional representations from the unlabeled text in a self-supervised scheme. Since then, the pretext of Masked Language Modeling (MLM) has arisen significant interest in NLP [7,46,47]. Motivated by the success of BERT in NLP, there are many attempts in the computer vision area [5,10,12,31,39,54]. Among them, iGPT [10] proposes to operate on sequences of pixels and predict the unknown pixels, while ViT [18] proposes to reshape the images into patches, and then adapts the standard Transformer to process images. Recently, BEiT [5] proposes a pretext task that tokenizes the input images into discrete visual tokens firstly and then recovers the masked discrete tokens, while MAE [26] encourages the model to reconstruct those missing pixels directly without the tokenization. For 3D data analysis, PointBERT [64] generalizes the concept of BERT into 3D point clouds by devising a masked point reconstruction task to pre-train the Transformers. Variation autoencoder (VAE) has been applied to implement 3D mesh deformation and pose transfer [13,4]. Our work is greatly inspired by MAE and PointBERT. However, the distinctive characteristics of 3D meshes hinder the straightforward use of BERT on 3D mesh analysis. In this paper, we aim to explore how to adapt mask autoencoding pre-training to 3D meshes with minimal modification.

2.3 Mesh Analysis

The polygon mesh is an effective 3D representation, which can depict the geometric context of a 3D object preciously. However, the vertices in 3D meshes do not have the same number of adjacent vertices, leading to the general convolution and pooling operations cannot be applied to the mesh data directly. Accompanied by the development of deep learning methods on 3D data, how to adapt the neural network to 3D mesh processing is always the highlight. Initially, Masci *et al.* [15] makes the first effort to generalize convolutional neural networks to 3D mesh based on localized frequency analysis. More recently, researchers extensively study how to implement feature aggregation and downsampling operations

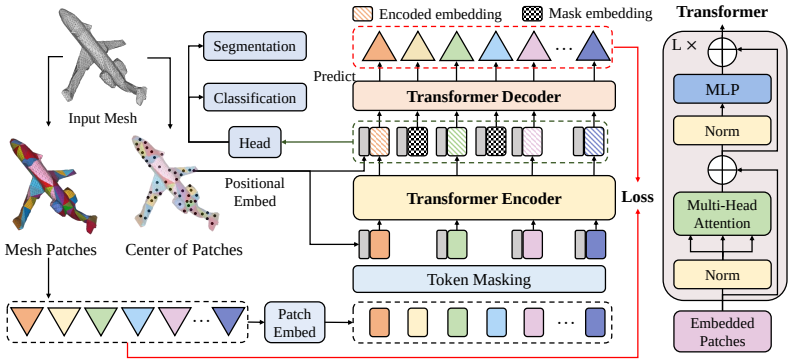

Fig. 1: The overall architecture of the proposed MeshMAE framework. The input mesh is divided into several non-overlapping patches, which would be embedded by the MLP. During pre-training, patch embeddings will be randomly masked, and only the visible embeddings will be fed into the Transformers. Then, the mask embeddings are introduced and sent to the decoder together with the combination of encoded embeddings. The targets of the decoder are to reconstruct the vertices and face features of masked patches. After pre-training, the decoder is discarded and the encoder is applied to downstream tasks.

on irregular structures. For example, MeshCNN [25] explores classical methods of Graphics, which defines an ordering invariant convolution operation for edges and utilizes the edge collapse operation to implement pooling. To overcome the problem that the unfixed number of vertex neighborhoods in the mesh, some methods introduce graph neural networks (GNNs) [41,56,43,49] to implement the convolution on vertices, where the feature aggregation is conducted on the vertices and their 1-ring neighbors. While each face must have three adjacent faces in the manifold mesh, this kind of stable structure makes feature aggregation of faces easier. MeshNet learns the spatial and structural features of a face by aggregating its three adjacent faces. SubdivNet [29] processes the meshes to obtain the fine-to-coarse hierarchical structure, which could be processed by the convolution-like operation. Actually, the attention-based Transformer does not require convolution or pooling operation and would not be affected by the data structure. Therefore, it may be more suitable for irregular mesh data.

3 Method

In this paper, we aim to extend the Vision Transformer into mesh analysis, Mesh Transformer, and apply the masked-autoencoding-based self-supervised pre-training strategy to Mesh Transformer. Here, we introduce the details of our Transformer-based mesh analysis framework, Mesh Transformer, and the MAE-based pre-training strategy. Figure 1 shows the framework of MeshMAE.

3.1 Mesh Transformer

A triangle mesh $M = (V; F)$ is defined by vertices and triangular faces. To apply Transformer on mesh, we need to address patch split, patch embedding, and positional embedding, which are introduced in the following parts.

Mesh Patch Split. In comparison with 3D point cloud data containing a set of discrete points, the faces of 3D mesh provide the connection relationship between vertices. As a result, we can use the geometric information of each face to represent the feature. In detail, as SubdivNet [29] does, for face f_i, we define its feature as a 10-dimensional vector, consisting of the face area (1-dim), three interior angles of the triangle (3-dim), face normal (3-dim), and three inner products between the face normal and three vertex normals (3-dim).

The self-attention-based architectures of Transformers ease the pains of designing especial feature aggregation operations for 3D meshes. However, if we apply the self-attention mechanism over all faces, the huge computational cost of quadratic complexity would be unbearable. Therefore, before applying Transformers to mesh, we first group the faces of a mesh into a set of non-overlapping patches. However, unlike image data, which is regular and could be divided into a grid of square patches, mesh data is irregular and the faces are usually unordered, which makes the patch split challenging. To address this issue, we propose to first 'remesh' the original mesh to make the structure regular and hierarchical. Specifically, we adopt MAPS algorithm [33] to implement the remeshing operation, which simplifies the mesh to a base mesh with N faces (96 N 256 in our experiments). After the remeshing process, the base mesh, *simpli ed mesh*, is coarser than the original mesh and is incapable of representing the shape accurately. Therefore, we further subdivide all faces in the base mesh t times in a 1-to-4 manner and get a refined mesh called t mesh. We can divide t mesh into non-overlapping patches by grouping the faces corresponding to the same face in the base mesh into a patch. In practice, we subdivide 3 times and thus each patch contains 64 faces. The process is illustrated in Figure 2.

Transformer Backbone. In this paper, we adopt the standard Transformer as the backbone network, consisting of a series of multi-headed self-attention layers and feedforward network (FFN) blocks.

To represent each patch, for image data, we can concatenate the RGB values of raw pixels located in the same patch in a natural order. For mesh data, thanks to remeshing operation which subdivides the face orderly, we can also use the concatenation of the 10-dim feature vectors of all faces belonging to the same patch as the patch's feature. As shown in Figure 2, we can find that the face in the base mesh is always divided in a fixed order starting from a selected vertex. Although the selected vertex might impact the order, we experimentally find that the impact is very slight (Sec. 4.3). Therefore, we can concatenate the faces' features according to the order generated by the remeshing process as the patch's representation (as shown in Figure 5). Then, we adopt an MLP to

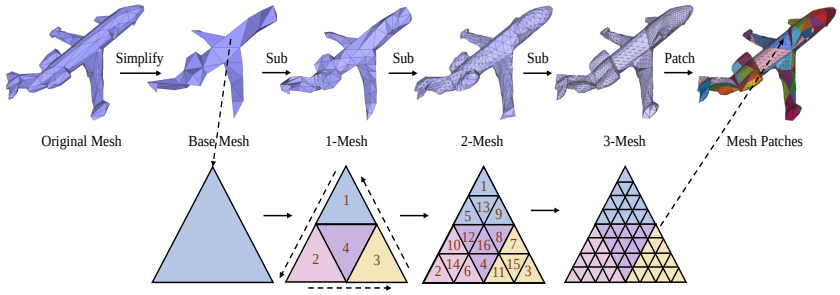

Fig. 2: The process of remeshing operation. The input mesh is simplified first, and a bijection is constructed between the original mesh and the base mesh. Then the base mesh is subdivided 3 times and new vertices are projected back onto the input.

project the feature vector of those patches into representations $\{\boldsymbol{e}_i\}_{i=1}^{g}$, where g denotes the number of patches. The representations are thus regarded as the input into the Transformer. The input features only describe the shape information of faces. Generally, Transformer-based methods utilize the positional embedding to provide the position information of patches. In comparison against natural language or image where the serial number is taken as the position, the mesh data contains 3D space position of each face. In this paper, we adopt the center 3D coordinates of faces to compute the positional embeddings, which would be more suitable for the geometric data and unordered patches. In practical, we first calculate the center point coordinates $\{\boldsymbol{c}_i\}_{i=1}^{g}$ of each patch, and then apply an MLP to $\{\boldsymbol{c}_i\}_{i=1}^{g}$ to obtain the positional embedding $\{\boldsymbol{p}_i\}_{i=1}^{g}$ for patches.

Formally, we define the input embeddings $\boldsymbol{X} = \{\boldsymbol{x}_i\}_{i=1}^{g}$ as the combination of patch embeddings $\boldsymbol{E} = \{\boldsymbol{e}_i\}_{i=1}^{g}$ and positional embeddings $\boldsymbol{P} = \{\boldsymbol{p}_i\}_{i=1}^{g}$. In this way, the overall input sequence is defined as $\boldsymbol{H}^0 = \{\boldsymbol{x}_1; \boldsymbol{x}_2; \cdots; \boldsymbol{x}_g\}$. There are L layers of Transformer block in the encoder network, and the output of the last layer $\boldsymbol{H}^L = \{\boldsymbol{h}_1^L; \cdots; \boldsymbol{h}_g^L\}$ represents the encoded representation of the input patches.

3.2 Mesh Pre-training Task

Motivated by BERT [16] and MAE [26], we study the masked modeling strategy for mesh representation learning based on the introduced Mesh Transformer. Specifically, we devise a masked mesh modeling task that aims at reconstructing the geometric structure of the masked patches from partially visible sub-meshes. Our method is based on autoencoders, where an encoder is utilized to map the visible sub-meshes into latent representations and the decoder reconstructs the geometric structure from the latent representations. By modeling the masked parts, the model attains the geometric understanding of the mesh. Below, we provide more details.

Encoder and Decoder. In the pre-training task, the encoder and decoder networks are both composed of several Transformer blocks. We set the encoder, *i.e.,* our Mesh Transformer, as 12 layers and a lightweight decoder with 6 layers. In our method, according to a pre-defined masking ratio, some patches of the input mesh are masked, and the remaining visible patches are fed into the encoder. Before feeding into the decoder, we utilize a shared mask embedding to replace all the masked embeddings, which indicates the presence of the missing patches that need to be predicted. Then, the input to the decoder is composed of encoded visible embeddings and mask embeddings. Here, we add the positional embeddings to all embeddings once again, which provide the location information for both masked and visible patches. It is noted that the decoder is only used during pre-training to perform mesh reconstruction tasks, while only the encoder is used in the downstream tasks.

Masked Sequence Generation. The complete mesh embeddings are denoted by $E = \{e_1, \ldots, e_g\}$ and their indices are denoted by $I = \{1, \ldots, g\}$. Following MAE, we first randomly mask a subset of patches, where the indices I_m of masked embeddings are sampled from I randomly with the ratio r. In this way, we denote the masked embeddings as $E_m = E[I_m]$ and the unmasked embeddings as $E_{um} = E[I \quad I_m]$. Next, we replace the masked embeddings E_m with a shared learnable mask embedding E_{mask} while keeping their positional embedding unchanged. Finally, the corrupted mesh embeddings $E_c = E_{um} \; [\; \{E_{mask} + p_i : i \in I_m\}$ are fed into the encoder. When the masking ratio is high, the redundancy can be eliminated largely and the masked parts would not be solved easily by extrapolation from visible neighboring patches. Therefore, the reconstruction task is relatively challenging.

Reconstruction Targets. The targets of the pre-training task are also crucial. In NLP, BERT [16] proposes the pretext of Masked Language Modeling (MLM), which first masks the input randomly and then recovers a sequence of input tokens. Following it, there are also similar works on images, *e.g.*, BEiT [5], and on point clouds, *e.g.,* PointBert [64], both of which train the models by recovering the tokens learned via dVAE [48]. However, these kinds of dVAE tokenizer reconstruction require one more pre-training stage. Recently, MAE [26] skips the dVAE training process and proposes to reconstruct the pixel values directly, which is simpler and saves much computation overhead.

Inspired by MAE [26], our method directly recovers the input patches. Specifically, we define a reconstruction target as the shape of the masked patches. As shown in Figure 2, there are only 45 unique vertices in the 64 faces of a mesh patch. To recover the shape of patches, these 45 vertices are required to be located in the corresponding ground truth positions. Therefore, we propose to predict the 3D relative coordinate (x, y, z) of them directly (relative to the center point of the patch), and the output of the decoder can be written as $P_r = \{p_{r_i}\}_{i=1}^{45}$. During the training phase, we adopt the l_2-form Chamfer distance to calculate the reconstruction loss between the relative coordinates of

predicted vertices and that of ground truth vertices, which is shown as follows:

$$L_{CD}(P_r; G_r) = \frac{1}{jP_rj} \sum_{p \in P_r} \min_{g \in G_r} jjp \quad gjj + \frac{1}{jG_rj} \sum_{g \in G_r} \min_{p \in P_r} jjg \quad pjj; \tag{1}$$

where G_r denotes the relative coordinates of the 45 ground truth vertices in a patch.

In the real mesh, the vertices are connected with edges to compose the faces, which are the important unit of the mesh data. However, due to the disorder of the vertices, it is intractable to recover the faces' information if we only predict the vertices' position, which might degrade the network's capability of recovering the geometric structure. In order to enhance the restraint and predict the local details, we propose to additionally predict the face-wise features, *i.e.,* the input representation of the face. To achieve this, we add another linear layer behind the decoder to predict all faces' features for each patch. We denote the predicted features by $J\ 2\ R^{64\times10}$. Here, we organize the faces' features in the order generated by the remeshing operation. We adopt the face-wise MSE loss L_{MSE} to evaluate the reconstruction effect of features. The overall optimization object is combined by both L_{CD} and L_{MSE}, as follows,

$$L = L_{MSE} + \quad L_{CD}; \tag{2}$$

where is the loss weight.

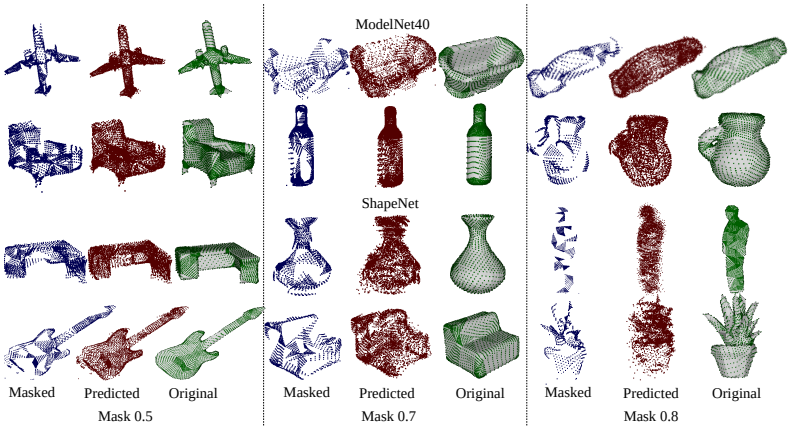

Fig. 3: Reconstruction results on ModelNet40 test set. Top two lines are the results of model trained on ModelNet40, and bottom two lines are the results of model trained on ShapeNet. Here, we show the effects when the mask ratio is 0.5, 0.7 and 0.8 respectively. More results will be demonstrated in the Supplementary.

It is noted that the input features do not contain coordinates of the three vertices of each face, while we propose to predict the vertices coordinates of

Table 1: Classification accuracy on ModelNet40. The first row is a point cloud method.

Method	Acc (%)
PCT [23]	92.4
MeshWalker [32]	90.5
MeshNet [19]	88.4
SubdivNet [29]	91.4
Transformer (w/o PT)	91.5
MeshMAE (PT M)	91.7
MeshMAE (PT S)	92.5

the masked patches. Here, we illustrate the shape reconstruction effect on the ModelNet40 in Figure 3, where the models are pre-trained on ModelNet40 and ShapeNet, respectively. We directly show the vertices in mesh instead of faces, because the vertices are disordered and it is difficult to define the edge connection between them, which has also no impact on the pre-training task. It can be seen that our model can recover the shape of original meshes precisely by predicting vertices, and the performance of our model is still well even if the mask ratio is 80%, suggesting that the proposed model has indeed learned the geometric information of 3D meshes.

4 Experiments

Here, we first describe the data pre-processing methods and introduce the settings of the proposed pre-training scheme. The learned representation would be evaluated through doing supervised training under two settings: end-to-end fine-tuning and linear probing. The experiments are conducted on two downstream tasks, including object classification and part segmentation. We provide more experimental results, analyses, and visualizations in the Supplementary.

4.1 Implementation Details

Data Pre-processing. In the original mesh datasets,e.g., ShapeNet and ModelNet40, the number of faces varies dramatically among meshes and most 3D meshes are not watertight or 2-manifold, that is, there are holes or boundaries on the surface. In order to facilitate the subsequent mesh preprocessing, we first reconstruct the shapes to build the corresponding manifold meshes and simplify the manifold meshes into 500 faces uniformly. Then, we remesh all meshes in the datasets using MAPS algorithm [33,38] to obtain the meshes with regular structures. In this process, the input mesh (500 faces) would be simplified to a base mesh with a smaller resolution (96-256 faces), and the base mesh would be further subdivided 3 times. After pre-processing, the remeshed meshes contain $n \times 64$ faces, where n denotes the face number in the base mesh. To improve the data diversity, we generate 10-times remeshed meshes for each input mesh by collapsing random edges in the remeshing process.

Data Augmentation. We apply random anisotropic scaling with a normal distribution = 1 and = 0 :1 to reduce network sensitivity to the size of meshes. According to the geometrical characteristics of 3D meshes, we additionally utilize shape deformation. The shape deformation is based on the free form deformation (FFD) [50], driven by moving the position of external lattice control points.

Training Details. For pre-training, we utilize ShapeNet and ModelNet40 as the dataset respectively to explore the di erence between small datasets and large datasets. Among them, ShapeNet covers about 51, 000 3D meshes from 55 common object categories and ModelNet40 contains 12, 311 3D meshes from 40 categories. We utilize ViT-Base [18] as the encoder network with very slight modi cation, e.g., the number of input features' channels. And following [26], we set a lightweight decoder, which has 6 layers. We employ an AdamW optimizer, using an initial learning rate of 1e-4 with a cosine learning schedule. The weight decay is set as 0.05 and the batch size is set as 32. We set the same encoder network that of pre-training in downstream tasks. For the classi cation task, we exploit the max-pooling operation behind the encoder and append a linear classi er. While for the segmentation task, we utilize two segmentation heads to provide a two-level feature aggregation, which would be introduced in the following part. We set the batch size as 32, and employ AdamW optimizer with an initial learning rate of 1e-4. The learning rate is decayed by a factor of 0.1 at 30 and 60 epochs in classi cation (80 and 160 epochs in segmentation).

Fig. 4: The experimental results under di erent masking ratios, where the pre-trained models are obtained on ModelNet40 in the left two gures while that are obtained on ShapeNet in the right two gures. The x-axes denotes the masking ratio and y-axes denotes the classi cation accuracy.

4.2 Downstream Tasks

In this subsection, we present the experimental results of classi cation and segmentation tasks. Here, we demonstrate the results of Mesh Transformer without

pre-training (`w/o PT'), the results of MeshMAE with pre-training on ModelNet (`PT M') and ShapeNet (`PT S').

Object Classi cation. For the classi cation task, we utilize the Manifold40 [29] (generated from ModelNet40) as the dataset. To demonstrate the e ectiveness of the proposed method, we compare it with several recent mesh classi cation methods and a recent point cloud-based method. Experimental results are reported in Table 1. In this part, we denote the results of mesh Transformer as `Transformer'. Here, we list the ne-tuning results of models pre-trained on ModelNet40 and ShapeNet, respectively. The results of them both outperform the baseline, demonstrating that the proposed mask autoencoding strategy is also e ective in the mesh analysis task. Compared with the result of `ModelNet40', the result of `ShapeNet' is higher and obtains the state-of-the-art performance, which shows that larger data sets can promote to learn a better feature representation in the proposed pre-training task. This is consistent with the conclusion in the CV.

Figure 4 shows the linear probing and ne-tuning results under di erent masking ratios on ModelNet40. Though the number of meshes in ShapeNet is almost 5 times that in ModelNet40, the linear probing results of ShapeNet are always lower than that of ModelNet40. We think this is due to the domain gaps between ShapeNet and ModelNet40. And the ne-tuning results of ShapeNet are much higher than that of ModelNet40, demonstrating a larger pre-training dataset could indeed bring better results. From Figure 4, we nd that masking 50% patches works well for both ne-tuning and linear probing settings.

Part Segmentation. In this part, we utilize two datasets to conduct segmentation experiments, Human Body [40] and COSEG-aliens [60]. Speci cally, the Human Body dataset consists of 381 training meshes from SCAPE [3], FAUST [6], MIT [57], and Adobe Fuse [1], and 18 test meshes from SHREC07 [21]. Each mesh sample is segmented into 8 parts. We also evaluate our method on the COSEG-aliens dataset with 200 meshes labeled with 4 parts. Both two datasets are processed by the remeshing operation, and the face labels are obtained from the mapping between the remeshed data and the raw meshes using the nearest-face strategy. Compared with classi cation, the segmentation task is more challenging, since it needs to predict dense labels for each face while faces within a patch might belong to di erent categories. In practice, we utilize two segmentation heads to provide a two-level feature aggregation. Speci cally, we concatenate the output of the encoder with the feature embedding of each face to provide a ne-grained embedding. Besides, we do not design any other network structure speci c for segmentation. Tables 2 and 3 show the segmentation results on Human and COSEG-aliens datasets. Our method achieves comparable performance with recent state-of-the-art methods in segmentation tasks. Considering that the proposed method only uses patch embeddings, it might be di cult for the network to learn detailed structure information. Therefore, the performance

improvement on the segmentation task is very limited compared with it on the classi cation task. Despite this, pre-training can still bring improvements.

Table 2: Mesh segmentation accuracy on the COSEG-aliens dataset.

Method	Acc (%)
MeshCNN [25]	94.4
PD-MeshNet [41]	89.0
SubdivNet [29]	98.5
Transformer (w/o PT)	97.6
MeshMAE (PT M)	97.9
MeshMAE (PT S)	98.0

Table 3: Mesh segmentation accuracy on the Human Body dataset.

Method	Acc (%)
Toric Cover [40]	88.0
MeshCNN [25]	87.7
SNGC [24]	91.3
SubdivNet [29]	90.8
Transformer (w/o PT)	90.1
MeshMAE (PT M)	90.1
MeshMAE (PT S)	90.3

4.3 Ablation Studies

Here, we present several ablation studies to further analyze the components in our method. To facilitate the analysis, we only conduct pre-training on ModelNet40 in this subsection.

Reconstruction Target. In the proposed masked autoencoding task, we propose to recover the input feature and predict the vertices of masked patches simultaneously. In this part, we would verify their e ectiveness separately and nd an appropriate loss weight between them. In Table 4, we list the classification performance under several loss settings when the mask ratio is set as 0.5. It is found that the best classi cation performance can be obtained when $L_{MSE} = L_{CD} = 1 : 0:5$.

Table 4: Comparison of reconstruction targets. The values of L_{MSE} and L_{CD} denote the weights of them, respectively. `Fine' indicates the results of ne-tuning, while `Line' indicates the results of linear probing.

L_{MSE}	L_{CD}	Fine (%)	Line (%)
1	0	91.2	89.8
0	1	90.6	89.7
1	0.1	90.8	90.4
1	0.25	90.8	90.7
1	0.5	91.7	91.1
1	1	91.5	90.3
1	2	91.5	89.7

Fig. 5: The order of face features, where the serial number of the face denotes the location of its feature vector in the concatenated feature vector.

Face Order. In comparison with images with regular structure, the faces in patches are triangular and they cannot be arranged by row and column. In this

paper, we set the order of face features in a patch as shown in Figure 5, which is de ned according to the remeshing process. Here, we conduct several test experiments to verify whether the other orders of faces would lower the performance. In Table 5, `Original' denotes the faces order in our method, `Rotate' denotes the structure of faces remains unchanged while starting from another two angles, e.g., starting from face 2 (Rotate-l) or face 3 (Rotate-r), and `Random' denotes the face features are arranged randomly. In Table 5, we repeat the experiments 5 times for each setting. According to the results, we nd that even if the order of faces is randomly disturbed, the performance decreases very slightly and remains basically stable, which demonstrates the our selection of the order works well for patch representation.

Table 5: The e ect of di erent orders of face features in the patches.

Sorting Method	Acc (%)	Std.
Original	91.7	4e-2
Rotate-l	91.7	5e-2
Rotate-r	91.6	2e-2
Random	91.6	4e-2

Table 6: Comparison of di erent positional embedding strategies.

Strategy	Mask	Acc (%)	Mask	Acc (%)
a)	0.25	42.6	0.5	46.1
b)	0.25	88.2	0.5	89.3
c)	0.25	75.0	0.5	77.5
d)	0.25	83.7	0.5	89.7

Positional Embedding. Each patch contains 64 faces, and how to embed positional information of 64 faces is a problem worth exploring. Here, we propose several positional embedding strategies: a) utilizing learnable parameters as MAE [26] does; b) rst embedding the center coordinates of each face, then applying max-pooling; c) rst reshaping the center coordinates of 64 faces (64, 3) into a one-dimension vector (64 3), then embedding this vector directly; d) rst calculating the center coordinates of the whole patches, then embedding the center of the patch directly (ours). Table 6 lists the classi cation results of the above strategies, which are all obtained by linear probing. We nd that embedding the center of the patch directly could obtain the best results.

5 Conclusions

In this paper, we introduce Transformer-based architecture (Mesh Transformer) into mesh analysis and investigate the masked-autoencoding-based pre-training method for Mesh Transformer pre-training. By recovering the features and shape of the masked patches, the model could learn the geometric information of 3D meshes. And the comprehensive experimental results and ablations prove that the proposed MeshMAE could learn a more e ective feature representation. The study shows the feasibility of the Transformer-based method on mesh analysis. In future, we would like to explore its applications in other mesh-related tasks.

Acknowledgements: This work is supported by the National Natural Science Foundation of China under Grant No. 62072348. Dr. Baosheng Yu and Dr. Jing Zhang are supported by ARC Project FL-170100117.

References

1. Adobe.com: Animate 3d characters for games, film, and more. Retrieved Janurary 24, 2021 from https://www.mixamo.com (2021)
2. Agrawal, P., Carreira, J., Malik, J.: Learning to see by moving. In: ICCV. pp. 37–45 (2015)
3. Anguelov, D., Srinivasan, P., Koller, D., Thrun, S., Rodgers, J., Davis, J.: Scape: shape completion and animation of people. ACM Transactions on Graphics (TOG) pp. 408–416 (2005)
4. Aumentado-Armstrong, T., Tsogkas, S., Jepson, A., Dickinson, S.: Geometric disentanglement for generative latent shape models. In: CVPR. pp. 8181–8190 (2019)
5. Bao, H., Dong, L., Wei, F.: Beit: Bert pre-training of image transformers. arXiv preprint arXiv:2106.08254 (2021)
6. Bogo, F., Romero, J., Loper, M., Black, M.J.: Faust: Dataset and evaluation for 3d mesh registration. In: CVPR. pp. 3794–3801 (2014)
7. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. NeurIPS **33**, 1877–1901 (2020)
8. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: ECCV. pp. 213–229. Springer (2020)
9. Chen, C.F., Fan, Q., Panda, R.: Crossvit: Cross-attention multi-scale vision transformer for image classification. arXiv preprint arXiv:2103.14899 (2021)
10. Chen, M., Radford, A., Child, R., Wu, J., Jun, H., Luan, D., Sutskever, I.: Generative pretraining from pixels. In: ICML. pp. 1691–1703. PMLR (2020)
11. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. In: NeurIPS. pp. 17864–17875 (2021)
12. Conneau, A., Lample, G.: Cross-lingual language model pretraining. In: NeurIPS. pp. 7059–7069 (2019)
13. Cosmo, L., Norelli, A., Halimi, O., Kimmel, R., Rodola, E.: Limp: Learning latent shape representations with metric preservation priors. In: ECCV. pp. 19–35. Springer (2020)
14. Dai, Z., Cai, B., Lin, Y., Chen, J.: Up-detr: Unsupervised pre-training for object detection with transformers. In: CVPR. pp. 1601–1610 (2021)
15. Davide, B., Jonathan, M., Simone, M., Michael, M.B., Umberto, C., Pierre, V.: Learning class-specific descriptors for deformable shapes using localized spectral convolutional networks. Computer Graphics Forum **34**(5), 13–23 (2015)
16. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
17. Ding, L., Lin, D., Lin, S., Zhang, J., Cui, X., Wang, Y., Tang, H., Bruzzone, L.: Looking outside the window: Wide-context transformer for the semantic segmentation of high-resolution remote sensing images. arXiv preprint arXiv:2106.15754 (2021)
18. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. In: ICLR (2020)
19. Feng, Y., Feng, Y., You, H., Zhao, X., Gao, Y.: Meshnet: Mesh neural network for 3d shape representation. In: AAAI. pp. 8279–8286 (2019)
20. Gidaris, S., Singh, P., Komodakis, N.: Unsupervised representation learning by predicting image rotations. In: ICLR (2018)

21. Giorgi, D., Biasotti, S., Paraboschi, L.: Shape retrieval contest 2007: Watertight models track. SHREC competition **8**(7) (2007)
22. Guan, S., Xu, J., Wang, Y., Ni, B., Yang, X.: Bilevel online adaptation for out-of-domain human mesh reconstruction. In: CVPR. pp. 10472–10481 (2021)
23. 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)
24. Haim, N., Segol, N., Ben-Hamu, H., Maron, H., Lipman, Y.: Surface networks via general covers. In: ICCV. pp. 632–641 (2019)
25. Hanocka, R., Hertz, A., Fish, N., Giryes, R., Fleishman, S., Cohenor, D.: Meshcnn: a network with an edge. ACM Transactions on Graphics (TOG) **38**(4), 90 (2019)
26. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: CVPR. pp. 16000–16009 (2022)
27. Hjelm, R.D., Fedorov, A., Lavoie-Marchildon, S., Grewal, K., Bachman, P., Trischler, A., Bengio, Y.: Learning deep representations by mutual information estimation and maximization. In: ICLR (2018)
28. Hu, S.M., Liang, D., Yang, G.Y., Yang, G.W., Zhou, W.Y.: Jittor: a novel deep learning framework with meta-operators and unified graph execution. Science China Information Sciences **63**(222103), 1–21 (2020)
29. Hu, S.M., Liu, Z.N., Guo, M.H., Cai, J.X., Huang, J., Mu, T.J., Martin, R.R.: Subdivision-based mesh convolution networks. ACM Transactions on Graphics (TOG) (2021)
30. Huang, L., Tan, J., Liu, J., Yuan, J.: Hand-transformer: non-autoregressive structured modeling for 3d hand pose estimation. In: ECCV. pp. 17–33. Springer (2020)
31. Joshi, M., Chen, D., Liu, Y., Weld, D.S., Zettlemoyer, L., Levy, O.: Spanbert: Improving pre-training by representing and predicting spans. Transactions of the Association for Computational Linguistics **8**, 64–77 (2020)
32. Lahav, A., Tal, A.: Meshwalker: Deep mesh understanding by random walks. ACM Transactions on Graphics (TOG) **39**(6), 1–13 (2020)
33. Lee, A.W., Sweldens, W., Schröder, P., Cowsar, L., Dobkin, D.: Maps: Multiresolution adaptive parameterization of surfaces. In: ACM SIGGRAPH. pp. 95–104 (1998)
34. Li, W., Liu, H., Tang, H., Wang, P., Van Gool, L.: Mhformer: Multi-hypothesis transformer for 3d human pose estimation. arXiv preprint arXiv:2111.12707 (2021)
35. Li, Z., Liu, X., Drenkow, N., Ding, A., Creighton, F.X., Taylor, R.H., Unberath, M.: Revisiting stereo depth estimation from a sequence-to-sequence perspective with transformers. In: CVPR. pp. 6197–6206 (2021)
36. Lin, K., Wang, L., Liu, Z.: End-to-end human pose and mesh reconstruction with transformers. In: CVPR. pp. 1954–1963 (2021)
37. Lin, K., Wang, L., Liu, Z.: Mesh graphormer. arXiv preprint arXiv:2104.00272 (2021)
38. Liu, H.T.D., Kim, V.G., Chaudhuri, S., Aigerman, N., Jacobson, A.: Neural subdivision. arXiv preprint arXiv:2005.01819 (2020)
39. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019)
40. Maron, H., Galun, M., Aigerman, N., Trope, M., Dym, N., Yumer, E., Kim, V.G., Lipman, Y.: Convolutional neural networks on surfaces via seamless toric covers. ACM Transactions on Graphics (TOG) **36**(4), 71–1 (2017)
41. Milano, F., Loquercio, A., Rosinol, A., Scaramuzza, D., Carlone, L.: Primal-dual mesh convolutional neural networks. NeurIPS **33**, 952–963 (2020)

42. Misra, I., Girdhar, R., Joulin, A.: An end-to-end transformer model for 3d object detection. In: CVPR. pp. 2906–2917 (2021)
43. Monti, F., Shchur, O., Bojchevski, A., Litany, O., Günnemann, S., Bronstein, M.M.: Dual-primal graph convolutional networks. arXiv preprint arXiv:1806.00770 (2018)
44. Nash, C., Ganin, Y., Eslami, S.A., Battaglia, P.: Polygen: An autoregressive generative model of 3d meshes. In: PMLR. pp. 7220–7229. PMLR (2020)
45. Pathak, D., Girshick, R., Dollár, P., Darrell, T., Hariharan, B.: Learning features by watching objects move. In: CVPR. pp. 2701–2710 (2017)
46. Radford, A., Narasimhan, K., Salimans, T., Sutskever, I.: Improving language understanding by generative pre-training (2018)
47. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
48. Rolfe, J.T.: Discrete variational autoencoders. arXiv preprint arXiv:1609.02200 (2016)
49. Saleh, M., Wu, S.C., Cosmo, L., Navab, N., Busam, B., Tombari, F.: Bending graphs: Hierarchical shape matching using gated optimal transport. In: CVPR (2022)
50. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. ACM SIGGRAPH computer graphics pp. 151–160 (1986)
51. Tianyu, L., Yali, W., Junhao, Z., Zhe, W., Zhipeng, Z., Yu, Q.: PC-HMR: pose calibration for 3d human mesh recovery from 2d images/videos. In: AAAI. pp. 2269–2276. AAAI Press (2021)
52. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICLR. pp. 10347–10357 (2021)
53. Trappolini, G., Cosmo, L., Moschella, L., Marin, R., Melzi, S., Rodolà, E.: Shape registration in the time of transformers. In: NeurIPS. pp. 5731–5744 (2021)
54. Trinh, T.H., Luong, M.T., Le, Q.V.: Selfie: Self-supervised pretraining for image embedding. arXiv preprint arXiv:1906.02940 (2019)
55. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: NeurIPS. pp. 5998–6008 (2017)
56. Verma, N., Boyer, E., Verbeek, J.: Feastnet: Feature-steered graph convolutions for 3d shape analysis. In: CVPR. pp. 2598–2606 (2018)
57. Vlasic, D., Baran, I., Matusik, W., Popović, J.: Articulated mesh animation from multi-view silhouettes. In: ACM SIGGRAPH. pp. 1–9 (2008)
58. Wang, H., Zhu, Y., Adam, H., Yuille, A., Chen, L.C.: Max-deeplab: End-to-end panoptic segmentation with mask transformers. In: CVPR. pp. 5463–5474 (2021)
59. Wang, X., Gupta, A.: Unsupervised learning of visual representations using videos. In: ICCV. pp. 2794–2802 (2015)
60. Wang, Y., Asafi, S., Van Kaick, O., Zhang, H., Cohen-Or, D., Chen, B.: Active co-analysis of a set of shapes. ACM Transactions on Graphics (TOG) **31**(6), 1–10 (2012)
61. Xu, Y., Zhang, Q., Zhang, J., Tao, D.: Vitae: Vision transformer advanced by exploring intrinsic inductive bias. In: NeurIPS. pp. 28522–28535 (2021)
62. Yang, G., Tang, H., Ding, M., Sebe, N., Ricci, E.: Transformer-based attention networks for continuous pixel-wise prediction. In: CVPR. pp. 16269–16279 (2021)
63. Yu, X., Rao, Y., Wang, Z., Liu, Z., Lu, J., Zhou, J.: Pointr: Diverse point cloud completion with geometry-aware transformers. In: ICCV. pp. 12498–12507 (2021)

64. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. arXiv preprint arXiv:2111.14819 (2021)
65. Zhang, Q., Xu, Y., Zhang, J., Tao, D.: Vitaev2: Vision transformer advanced by exploring inductive bias for image recognition and beyond. arXiv preprint arXiv:2202.10108 (2022)
66. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: ICCV. pp. 16259–16268 (2021)
67. Zheng, S., Lu, J., Zhao, H., Zhu, X., Luo, Z., Wang, Y., Fu, Y., Feng, J., Xiang, T., Torr, P.H., et al.: Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In: CVPR. pp. 6881–6890 (2021)
68. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)