MGMA: MESH GRAPH MASKED AUTOENCODERS FOR SELF-SUPERVISED LEARNING ON 3D SHAPE

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce a self-supervised learning model to extract face nodes and global graph embeddings on meshes. We define a model with graph masking on a mesh graph composed of faces to pre-train on self-supervised tasks. We evaluate our pre-trained model on shape classification and segmentation benchmarks. The results suggest that our model outperforms prior state-of-the-art mesh encoders: In ModelNet40 classification task, it achieves an accuracy of 89.8%, and in ShapeNet segmentation task, it performs a mean Intersection-over-Union (mIoU) of 78.5. Further, we explore and explain the correlation between test and training masking ratios on Mesh Graph Masked Autoencoders (MGMA). And we find best performances are obtained when mesh graph masked autoencoders are trained and evaluated under different masking ratios. Our work may open up new opportunities to address label scarcity and improve the learning power in geometric deep learning research.

1 INTRODUCTION

Mesh is a data format widely used in computer graphics and is used more and more frequently in computer vision tasks as additional supervision or inference targets. It provides an accurate, efficient, and irregular representation of three-dimensional shapes. These properties make it a popular format for capturing continuous underlying surfaces.

Many commonly used datasets, such as ModelNet (Wu et al., 2015), ShapeNet (Chang et al., 2015), ScanNet (Dai et al., 2017), and Pix3D (Sun et al., 2018), utilize meshes as the core or intermediate agent. A number of 3D data formats can be derived from the mesh structure, such as voxel grids, point clouds, and implicit surfaces. Researchers customize a series of methods to analyze those regular data formats using deep learning, like using 3D convolution to parse 3D voxel grids (Wu et al., 2016), using symmetric functions (Qi et al., 2017a) to process point clouds, and using signed distance fields to represent the implicit surfaces (Cruz et al., 2021; Park et al., 2019).

Mesh representation itself could provide excellent quality and computational efficiency while preserving sharp shape features. Deep learning with data formats extracted from meshes have gained more and more success in 3D shape analysis, while analyzing their original data format with deep learning approaches is still an open problem. So studies on developing deep learning methods on mesh data attract lots of interest. Traditional approaches treat a mesh as a graph with vertices as nodes (Hanocka et al., 2019b; Verma et al., 2018) and develop methods akin to CNN, which contains convolution and pooling operations, to learn shared filters to extract features from edges in meshes. However, such approaches ignore the rich manifold structure meshes can represent, such as topology and Riemannian metric. On the other hand, most of the current mesh-based networks validate themselves on small or synthetic datasets. The dearth of studies that demonstrate the effectiveness of mesh on large datasets limits the development of deep learning applications on meshes. Moreover, the compact and efficient essence of mesh data representation should also be well utilized in ongoing geometric deep learning research. A powerful tool to analyze 3D meshes would benefit computer graphics and computer vision researchers.

There are significant challenges in developing mesh-based geometric deep learning methods. The first challenge is passing mesh, an irregular data format, forward in a neural network. In our work, we take the mesh as a graph composed of multiple faces as nodes of the graph. The emergence of success in graph processing provides us with a model to handle graph data. Thus, the mesh is another data format that a graph model naturally processes. Further, we design an attention mechanism along graph convolution on meshes to leverage its excellent feature extraction ability.

Meanwhile, because of the high cost and high variability associated with manual data labeling, there are more and more unlabeled 3D data. Traditional studies do not consider unlabeled data, which induces a huge sacrifice of untapped information. Therefore, unsupervised learning attracts more attention and has become an important concept for extracting information from unlabeled data. When we review the trend and development of artificial intelligence, self-supervised training on large datasets and producing pre-trained models for downstream tasks is becoming a predominant power in processing and extracting important features from billions and millions of data (Chen et al., 2020b;a; Dosovitskiy et al., 2021; Brown et al., 2020). Training an autoencoder with masking (Devlin et al., 2019) on the input data during training has been proved to be an effective method for image classification (He et al., 2022). In this paper, benefiting from using the mesh data representation, we propose to apply graph masking and point cloud reconstruction to support our self-supervised learning architecture and advance 3D deep learning research.

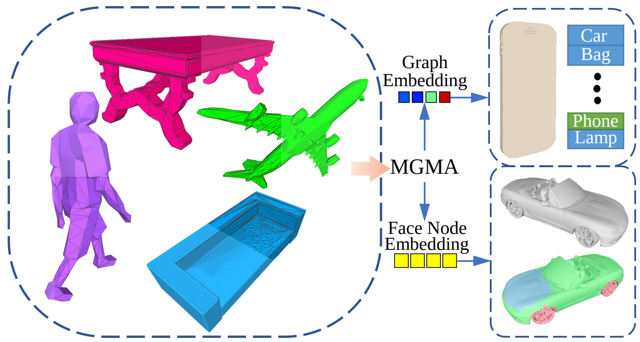


Figure 1: **Data and applications of MGMA.** We propose a new deep net architecture that analyzes meshes as a graph composed of faces using an attention mechanism. It is pre-trained with self-supervisions and provides face node embeddings and global graph embeddings for 3D recognition tasks like classification and part segmentation.

In our paper, we present a mesh-based framework, Mesh Graph Masked Autoencoder (MGMA), which is pre-trained on self-analyzing the mesh data, and apply the pre-trained model to large-scale 3D imaging datasets. Our network is designed to be suitable for different kinds of mesh representations to increase flexibility and support a variety of available data. MGMA exhibits state-of-the-art performance on supervised tasks. Furthermore, it could perform unsupervised and semi-supervised classification and segmentation tasks. We show in Figure 1 that a mesh could be considered as a graph with faces as nodes and pre-trained to have a model which could be applied to multiple tasks in recognition tasks. To demonstrate the effectiveness of our method, we perform a variety of experiments and show state-of-the-art performance among the mesh-based shape feature extractors. The key contributions of our work are as follows: **1.** We introduce a mesh graph autoencoder and train it with graph masking. **2.** With our novel MGMA encoder, our self-supervised learning model incorporates unlabeled data into the training stage and enhances the 3D data learning power. **3.** We comprehensively evaluate our model under various learning benchmarks on SHREC11, ModelNet40 supervised and unsupervised classification, and ShapeNetPart semi-supervised segmentation tasks and show that our model achieves state-of-the-art results w.r.t prior mesh-based neural network models. **4.** We explore and explain the correlation between test and training masking ratios on MGMA. And we find best performances are obtained when mesh graph masked autoencoders are trained and evaluated under different masking ratios. This gained insight may guide future self-supervised learning algorithm development.

2 Relate Work

Deep Learning on Meshes Treating a polygon mesh as a graph would accordingly apply graph-based methods on it. There are two existing categories for graph methods: spectral methods (Bruna et al., 2013; Henaff et al., 2015; Defferrard et al., 2016; Kipf & Welling, 2016; Levie et al., 2019) and spatial methods (Micheli, 2009; Atwood & Towsley, 2016; Niepert et al., 2016; Gilmer et al., 2017; Fey et al., 2018; Masci et al., 2015; Monti et al., 2017; Huang et al., 2019). Moreover, the convolution in the spectral domain is non-localized filtering (Defferrard et al., 2016). Chebyshev polynomial expansion is a method to solve the non-localization problem (Defferrard et al., 2016).

On the other hand, there is no easy way to induce the weight sharing across different locations of the graph due to the difficulty of matching local neighborhoods in the spatial domain (Bruna et al., 2013). Nevertheless, Atwood & Towsley (2016) proposed a spatial filtering method that assumes information is transferred from a vertex to its adjacent vertex with a specific transition probability. The power of the transition probability matrix implies that farther adjacent vertices provide little information for the central vertex. Furthermore, Geodesic CNN (Masci et al., 2015), MoNet (Monti et al., 2017), and SplineCNN (Fey et al., 2018) deal with the weight sharing problem by designing local coordinate systems for the central vertex in a local patch. They apply a set of weighting functions to aggregate the characteristics at the adjacent vertices. Next, they calculate a weighted mean of these aggregates. However, these methods are informatically expensive and require pre-defined local coordinate systems. In addition, Neural3DMM (Bouritsas et al., 2019) introduces the spiral convolution operation by enforcing a local ordering of vertices through the spiral operator. An initial point for each spiral is a vertex with the shortest geodesic path to a fixed reference point on a template shape. The remaining vertices of the spiral are ordered in the clockwise or counterclockwise directions inductively. However, finding a reference point for an arbitrary shape is challenging. Moreover, the initial point is not unique once two adjacent vertices have the same shortest path to the reference point.

FeaStNet (Verma et al., 2018) proposes a graphical neural network in which the neighborhood of each peak for the convolution operation is not preset but instead calculated dynamically. Tangent convolution is introduced in (Tatarchenko et al., 2018), where a small neighborhood around each vertex is used to reconstruct the local function upon which convolution is applied. Some generative models have also been tried on the mesh. Litany et al. (2018) perform shape completion via a graph autoencoder. MeshCNN (Hanocka et al., 2019b) utilizes the particular property of edge in a triangle mesh to extract edge features. Yang et al. (2021) apply continuous convolution on a geodesic region of mesh.

Self-Supervised Learning Self-supervised learning is to define some tasks from the data itself, and those human-defined tasks are used to pre-train the model. It is used in computer vision with proxy tasks such as predicting order in time (Wei et al., 2018), finding missing pixels (Pathak et al., 2016), location of patches (Doersch et al., 2015), image orientations (Gidaris et al., 2018), human-made artifacts (Jenni & Favaro, 2018), clusters of images (Caron et al., 2018), camera locations (Agrawal et al., 2015), jaggle puzzle (Noroozi & Favaro, 2016), color of videos (Vondrick et al., 2018), and tracking of image patches(Wang & Gupta, 2015). These works demonstrate promising results in transferring visual features from proxy tasks to other tasks.

Thus, defining proxy tasks that are related enough to the downstream task is quite important (Jenni & Favaro, 2018). On the other hand, supervisions, like density estimation or clustering, are not domain-specific (Caron et al., 2018). Deep clustering models(Aljalbout et al., 2018; Min et al., 2018; Yang et al., 2017; Hershey et al., 2016; Xie et al., 2016; Ghasedi Dizaji et al., 2017; Shaham et al., 2018; Yang et al., 2016; Hsu & Lin, 2018) come up to jointly train with a network-specific loss.

There are many works exploring self-supervised learning on point clouds. They use multi-tasks learning (Hassani & Haley, 2019), reconstruction (Achlioptas et al., 2018; Yang et al., 2018;?) contrast learning (Zhang & Zhu, 2019), restoring point cloud (Shi et al., 2020), point cloud autoregression (Sun et al., 2019b), the orientation prediction (Poursaeed et al., 2020; Han et al., 2019), and approximating convex decomposition (Gadelha et al., 2020) to pre-train the model and achieve state-of-the-art results on point cloud classification and segmentation tasks. Recently, masked autoencoders are used for self-supervised learning on image classification tasks(He et al., 2022).

Transformer Applications Transformer, which is proposed by Vaswani et al. (2017), has been widely used in natural language processing (NLP) and then computer vision. In NLP, large Transformer-based models are often pre-trained on large datasets and then fine-tuned for the downstream tasks, like BERT (Devlin et al., 2019) and GPT (Radford et al., 2018; 2019; Brown et al., 2020). In computer vision, applying Transformer on image processing experiences the local-to-global and low-to-high resolution process. Image Transformer (Parmar et al., 2018) applies self-attention to local neighborhoods. And this local attention replaces convolutions (Hu et al., 2019; Ramachandran et al., 2019; Zhao et al., 2020). Sparse Transformers (Child et al., 2019) use scalable approximations to global self-attention for images. Another way to apply attention to blocks of varying sizes (Weissenborn et al., 2019), in this particular case, along individual axes (Ho et al., 2019; Wang et al., 2020).

Some models (Cordonnier et al., 2020; Dosovitskiy et al., 2021; Bello et al., 2019; Wu et al., 2020; Chen et al., 2020a) extract patches of size 2×2 or 7×7 from the input image then apply CNN and Transformer sequentially. These works make Transformer achieve state-of-the-art results on small and medium resolution images. Instead of just classification, Transformer is also used in video processing(Wang et al., 2018; Sun et al., 2019a), object detection(Hu et al., 2018; Carion et al., 2020), unsupervised object discovery(Locatello et al., 2020), and unified text-vision tasks(Chen et al., 2020c; Lu et al., 2019; Li et al., 2019). Recently, Liang et al. (2022) use Transformer as an autoencoder network for mesh reconstruction and self-supervised learning.

3 METHOD

MGMA is a masked autoencoder that interprets the mesh as a graph, and each graph node is a face on the mesh. The features on the face nodes are randomly masked first and passed through multiple face graph attention layers. Then max-pooling is applied to obtain the global graph embedding, which is passed to a point cloud decoder for reconstruction pre-train tasks.

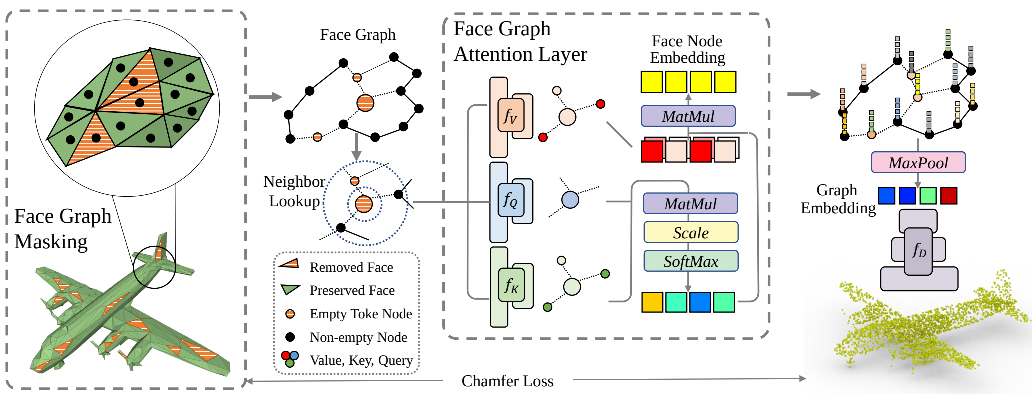


Figure 2: **Architecture of MGMA.** MGMA is a masked autoencoder structure that extracts global information from mesh and decodes the information into a point cloud. The structure interprets the mesh as a graph, and each node of the graph is a face on the mesh. For the node that is selected as a masked node, its feature is replaced with the mask embedding. The features on the face node are passed through multiple face graph attention layers. The layer aggregates the information from neighing nodes to the center node using an attention mechanism (detailed in Section 3). *MatMul, Scale,* and *SoftMax* is defined in Equation 1. Then max-pooling is applied across each face node and passes the global graph embedding to a point cloud decoder. Chamfer distance is computed between the decoded point cloud and the points sampled from the surface of the mesh to train the autoencoder. For detail structure of the network, see Section A.2

Masking on face graph is achieved by randomly selecting nodes on the graph according to the masking ratio. After one node is selected as the masked node, a learnable masking embedding takes the place of the original embedding, which is adopted from (He et al., 2022; Devlin et al., 2019).

Face graph attention layer is the core of our network, as shown in Figure 2. The layer takes a graph and the features on each node of the graph as input. For each node in the graph, the layer first gathers its neighbors according to an adjacency matrix which could be an n-ring neighbor adjacency matrix in our architecture. We denote r as the feature of the root node and n as the gathered features of the root node and its neighbors. Three linear layers f_V, f_Q, and f_K take n, r, and n as input to compute V, Q, and K. In our work, we keep the output dimension of key, value, and query fixed to 64.

$$FaceNodeEmbedding = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

After obtaining V, Q, and K, we use Equation 1to get the embedding of each face node. In Equation 1, d_k stands for the dimensional size of K. Details of composing the layers into an encoder are in Section A.2.

Method:	SHREC11 Split 16	SHREC11 Split 10
MeshGraphNet(Song et al., 2020)	28.9%	16.0%
MeshNet(Feng et al., 2019)	55.6%	44.7%
MeshCNN(Hanocka et al., 2019b)	98.6%	91.0%
PD-MeshNet(Milano et al., 2020)	99.7%	99.1%
MeshWalker(Lahav & Tal, 2020)	98.6%	97.1%
MeshNet++(Singh et al., 2021b)	100%	99.8%
ExMeshCNN(Kim & Chae, 2022)	100%	99.3%
SubdivNet(Hu et al., 2022)	100%	100%
MGMA(Ours)	**100%**	**100%**

Table 1: Classification accuracy for SHREC11 dataset.

Method:	ModelNet40
MeshNet(Feng et al., 2019)	88.9%
MeshWalker(Lahav & Tal, 2020)	88.9%
MeshGraphNet(Song et al., 2020)	89.8%
MVCNN(Su et al., 2015)	90.1%
PointNet++(Qi et al., 2017b)	90.7%
MeshNet++(Singh et al., 2021b)	91.6%
SDMC(Singh et al., 2021a)	92.2%
MeshMAE(Liang et al., 2022)	92.5%
ExMeshCNN(Kim & Chae, 2022)	**93.0%**
MGMA(Ours)	92.95%

Table 2: Classification accuracy for ModelNet40 dataset.

Reconstruction loss In the reconstruction loss function, we create a reconstruction decoder for this function. The input to this decoder is the graph embedding of the mesh. The expected output is the point cloud sampled from the mesh. Like the paper(Achlioptas et al., 2018), we use a similar network architecture f_D for decoding a point cloud. So we choose the point cloud as the target for the decoder to generate. And the loss function is the Chamfer Distance (CD), as shown in Equation 2.

$$\mathcal{L}_{CD} = \frac{1}{N}\sum_{n=1}^{N}\min_{\hat{p}\in\hat{s}}\|p_n - \hat{p}\|_2^2 + \frac{1}{M}\sum_{m=1}^{M}\min_{p\in s}\|\hat{p}_m - p\|_2^2 \tag{2}$$

where s and $\hat{s}$ are the ground truth and predicted point sets. M and N denote the number of points in the ground truth and predicted point sets. p_n and $\hat{p}_m$ are points in point set s and $\hat{s}$.

4 Experiments and Results

In this section, we introduce experiments to validate the effectiveness of our neural networks. First, we demonstrate the effectiveness of the encoder part of our networks on two supervised classification tasks. Then, we verify our work by pre-training the network for an unsupervised classification task. Finally, we conduct a semi-supervised experiment for part segmentation on 3D shapes.

4.1 Supervised Classification

we first verify that our network's encoder could outperform other networks. By using the designed mesh graph attention encoder, we achieve state-of-the-art performance on SHREC11 and ModelNet40 when the mesh is the input data modality.

SHREC11 is a dataset introduced in (Lian et al., 2011) that contains 30 classes, with 20 3D objects in each class. We follow the setup in which split 16 and 10 are the numbers of training 3D objects in each class, making split 10 a harder classification task than split 16. We use the meshes processed by, (Hanocka et al., 2019a) and each mesh contains 500 faces. Our results are reported in Table 1. We train our encoder 300 epochs with Adam optimizer, (Kingma & Ba, 2015) which is with β equal to 0.9 and 0.999, ϵ equal to 1^{-8}, learning rate 0.0002 and weight decay equal to 0.0. We compare our mesh graph attention encoder against eight methods that also take meshes as the input to their networks. It turns out that our encoder is able to get 100% accuracy on both setups.

Because SHREC11 is a relatively small dataset for supervised classification and some methods have reached 100% accuracy, we further validate our mesh graph attention encoder on ModelNet40 (Wu et al., 2015).

ModelNet40 is a dataset that contains 40 classes, and there are 9840 meshes for training and 2468 meshes for testing. Because the meshes in ModelNet40 have different numbers of faces. To fit the meshes onto GPU and to improve the GPU utilization, we follow the method in (Huang et al., 2018) to first make the mesh watertight, then simplify the meshes into 2048 faces. We train our encoder 300 epochs with the same optimizer settings as for SHREC11. The learning rate is decayed by a multiplicative factor of 0.1 at steps 30 and 60. Our method achieved 92.95% test accuracy on

ModelNet40. The results are reported in Table 2. We compare our encoder with other night methods. Our results are on par with state-of-the-art classification on ModelNet40.

These experiments validate that our encoder could get state-of-the-art performances on 3D shape classification tasks. The next experiments are to validate the model's performance on unsupervised tasks.

4.2 Unsupervised Classification

We pre-train the model across all the provided training data in ModelNet40. We keep the pre-trained model's weight and use it for the classification tasks. We do not perform fine-tuning when using the pre-trained model for downstream tasks. After obtaining the graph embedding, we use a linear SVM as the unsupervised tool for classification on ModelNet40.

The process of our unsupervised learning is stated as follows. We first pre-train the masked autoencoder with training data with the same training hype-parameter setting as in Section 4.1. After pre-training the model, we pick the model with the lowest Chamfer Distance on provided test data. Because the data used for pre-training do not contain label information, we do not consider computing test data's Chamfer Distance as information leaking. We use the best model to extract global embeddings from the training and test data, a vector with dimension 1024. Once we obtain the global embeddings, we use linear SVMs to train on ModelNet40 training data's global embeddings. We use 5-fold cross-validation to compute the average validation accuracy on the data split from training data. We also perform a logarithm search on the regularization parameter C of SVM from 1 to 1000 with the number of steps equal to 10. Then we pick the SVM model with the best average accuracy on validation data to compute the test accuracy. In Section A.1, we visualize graph embeddings using t-SNE(Van der Maaten & Hinton, 2008).

In Table 3, our method performs best compared with other mesh-based neural networks on unsupervised learning on ModelNet40. There are two reasons our method outperforms other mesh-based methods. The first reason is our encoder utilizes an attention mechanism to pick important points while ignoring the noisy information by assigning lower weight to the noisy neighboring. The second reason could contribute to the masking mechanism. It provides more data augmentation to our model and forces the model to focus less on the details of the shapes than the general information in the graph. And three methods (Han et al., 2019; Chen et al., 2021; Poursaeed et al., 2020) that outperform our methods are point cloud-based methods. The possible reason could be that data augmentation, like rotation (Han et al., 2019), is not considered when designing our framework. Adding such design components to our framework will be explored in future work.

Method	Modality	Accuracy
LGAN(Achlioptas et al., 2018)	Point	84.5
PointDistShi et al. (2020)	Point	84.7
PointGrowSun et al. (2019b)	Point	85.8
MRTNetGadelha et al. (2018)	Point	86.4
PCGANLi et al. (2018)	Point	87.8
FoldingNetYang et al. (2018)	Point	88.4
NSamplerRemelli et al. (2019)	Point	88.7
3D-PointCapsNetZhao et al. (2019)	Point	88.9
Multi-taskHassani & Haley (2019)	Point	89.1
ACDGadelha et al. (2020)	Point	89.8
MAP-VAEHan et al. (2019)	Point	90.2
GSIRChen et al. (2021)	Point	90.4
PointOEPoursaeed et al. (2020)	Point	**90.8**
ContrastNetZhang & Zhu (2019)	Voxel	86.8
AnyPoint(Zhang et al., 2021)	Point+Voxel	86.4
SPH Kazhdan et al. (2003)	Mesh	68.2
FeaStNetVerma et al. (2018)	Mesh	74.4
MeshCNNHanocka et al. (2019b)	Mesh	76.8
ContConvYang et al. (2021)	Mesh	76.5
MeshMAE(Liang et al., 2022)	Mesh	89.2
MGMA(Ours)	Mesh	**89.8**

Table 3: **Accuracy of unsupervised methods for classification on ModelNet40.** We compare with multiple methods taking different modalities of 3d data, including point cloud, voxel, and mesh as the input.

In Figure 3, we show the reconstruction results on ModelNet40 test data. To some extent, the autoencoder ignores the input mesh's detailed features while preserving the input mesh's overall structure. Those detailed features, like the airplane's engine, the chair's arm, and the leg style of a table, are ignored during the reconstruction. Ignoring those detailed features means that the encoder encodes the information that is good for decoding into an average shape in the class but forgets the detail. For reconstruction tasks, this is not desired. But for classification, this process is like cleaning redundant information from the input shape. More reconstruction visualization results are shown in Figure 10.

4.3 PART SEGMENTATION

Part segmentation is a fine-grained point-wise classification task that aims to predict each point's part category label in a given shape. In our work, we need to predict the part category label for each face in a mesh. We evaluate the learned point features on the ShapeNetPart dataset (Yi et al., 2016), which contains 16,881 objects from 16 categories (12149 for training, 2874 for testing, and 1858 for validation). Each object consists of 2 to 6 parts with a total of 50 distinct parts among all categories. We use the mean Intersection-over-Union (mIoU) as the measurement calculated by averaging the IoUs of the different parts occurring in one shape.

For the segmentation result, we follow the protocol from (Hassani & Haley, 2019). The results are shown in Table 4. In the original dataset, only point clouds and their corresponding point-wise labels are provided. To get ground truth for meshes, we need to first align the mesh with the point cloud by sampling points on the mesh and align the centers of the sampled point clouds with the provided point clouds. After the alignments, we first sample points on the face uniformly for each face on the mesh. Then we compute the nearest point in the ground truth point cloud. After that, the face's label is determined by the major vote of all the sampled points' labels.

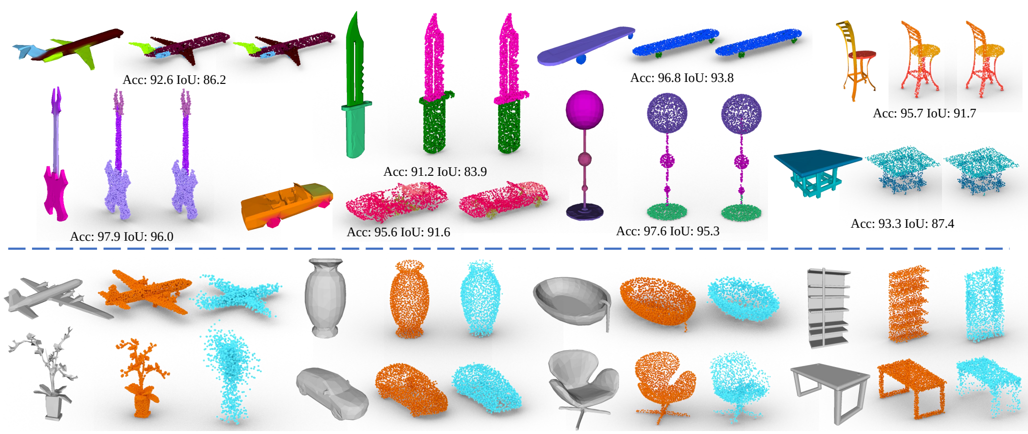


Figure 3: **Visual results.** The top part shows the training results of the semi-supervised part segmentation task. For each object, the label of the face is computed from the face embedding. Then we project the face label from mesh to the provided point cloud to compute accuracy and IoU. The predicted label for each point is in the middle, and the ground truth is on the right. The bottom part shows the reconstruction results of objects in the test dataset. From left to right, each object is the input mesh, the ground truth point cloud, and the predicted point cloud.

After the processing, we follow (Zhao et al., 2019) to randomly use 5% and 1% of the ShapeNetPart training data to evaluate the segment part task in a semi-supervised setting. We use the same pre-trained model to extract the face features of the sampled training data, along with validation and test samples without any finetuning. Following (Hassani & Haley, 2019), We then train a 4-layer MLP [2048, 4096, 1024, 50] on the sampled training sets and evaluate it on all test data. The input feature to the MLP is the concatenation of face node embeddings and global graph embeddings which makes the input features have a dimension size of 2048. We train the model with Adam optimizer with a fixed learning rate of 0.002. This training process takes 30 epochs and converges very fast. Because the features are clear for the MLP to distinguish, the entire process takes about 15 minutes, including the testing after each epoch's training.

Model	%train data	Cat.mIoU	Ins. mIoU	Aero	Bag	Cap	Car	Chair
Multi-Task	**5%**	**72.1**	77.7	**78.4**	**67.7**	78.2	66.2	**85.5**
MGMA-5	**5%**	69.5	**78.5**	77.8	66.4	46.8	**69.4**	81.7
MGMA-1	**1%**	49.3	72.5	77.5	31.3	0.0	61.6	80.1

Earphone	Guitar	Knife Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate	Table
52.6	**87.7**	**81.676.3**	**93.7**	56.1	80.1	70.9	44.7	60.7	73.0
50.4	83.8	66.5 70.2	92.5	**57.0**	**80.4**	**74.2**	**47.0**	**65.4**	**81.9**
28.3	84.3	36.3 55.2	91.6	0.0	65.9	61.5	34.2	0.0	80.2

Table 4: Comparison between our semi-supervised model and other model (Hassani & Haley, 2019) on ShapeNetPart segmentation task. Average mIoU over instances (Ins.) and categories (Cat.) are reported. MGMA-5 stands for training the appended MLP with 5% of the training data. And MGMA-1 stands for 1%.

During testing, we project the label computed on mesh's faces back to the provided point clouds according to the distance between the points and faces. Results shown in Table 4 suggest that our method is able to perform on par with the point cloud baselines and on ShapeNetPart semi-supervised learning segmentation task. In Figure 3, we show the visualization result of our semi-supervised learning segmentation. More segmentation visualization results are shown in Figure 9.

5 Discussion

5.1 *Is 0 masking ratio the Best Choice for Evaluating MGMA*

In He et al. (2022), the masking ratio at testing is fixed at 0. This is under the assumption that providing as much information to the trained masked autoencoder is the best choice. We explore the effect of test masking ratios on the unsupervised classification task. In our experiments, the test masking ratio is not fixed but also variable when evaluating the pre-trained model. In Figure 4 (a), we fix the test masking ratio to 0.0 and vary the training masking ratio from 0.1 to 0.9. And it demonstrates that varying training masking ratios could change the performance on unsupervised learning tasks. In Figure 4 (b) and (c), we vary the masking ratio not only during training but also during testing and validation. It turns out that the maximum test accuracy is obtained when the training masking ratio is 0.6 and the test masking ratio is 0.1 or 0.3. This result suggests that choosing 0 as the test masking ratio is not the only choice for evaluating a model trained with masking.

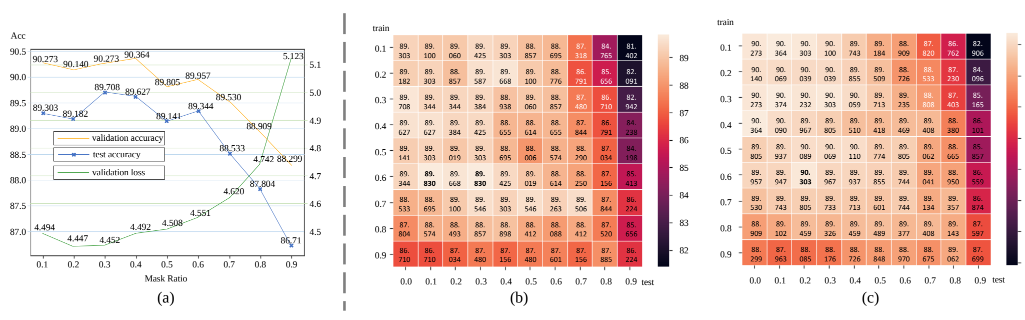


Figure 4: **Visualization of test and validation accuracy under different training and test masking ratio on the graph.** (a) plots the curve of test accuracy, validation accuracy, and validation loss (with unit 10^{-3}) by fixing the masking ratio at testing to 0 and varying the training masking ratio from 0.1 to 0.9. (b) and (c) are the heat maps of test accuracy and validation accuracy. The lighter the color, the higher the accuracy. The highest test accuracy (89.830%) is masked in bold in (b).

For the convenience of delivery, we denote a 2D coordinate (a, b) as the situation when the training masking ratio is a, and the test masking ratio is b. In Figure 5, we investigate why the best test accuracy happens at $(0.6, 0.1)$ and $(0.6, 0.3)$. We compute the difference between validation accuracy and test accuracy. This difference is usually taken as the symbol of overfitting or underfitting. It turns out that in most cases, our model overfitted the task. But those maximum test accuracy points happen to points less overfitting. Another point that exhibits such property is $(0.7, 0.7)$ in the difference map. But at that point, more information about the mesh is lost. There are totally three regions on the heat map in Figure 5 exhibiting the less overfitting property. The last one is at $(0.2, 0.6)$. But the testing ratio is too high that the model is not overfitting but also extracts less useful information. Even though in MaskMAE, 0.75 is the best choice for masking, our 3D mesh dataset differs from the image dataset. In 3D space, this masking ratio becomes lower, which means a face in a mesh participating in classification plays a more important role than each pixel in an image.

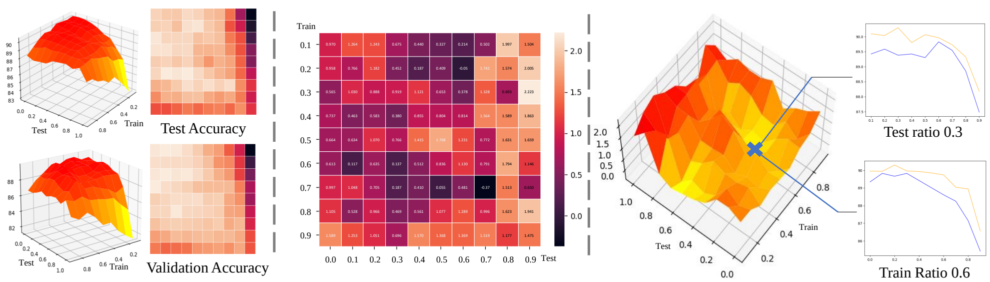


Figure 5: **Analysis of masking ratio.** The test and validation accuracy heat map on the left is visualized as a 3D patch. In the middle, the difference between validation and test accuracy is visualized in a heat map. The difference heat map is visualized on the right in a 3D patch. And two sub-graphs show the curve of test accuracy (in blue) and validation accuracy (in yellow) by fixing different test and training masking ratios.

Also, the point at position $(0.5, 0.5)$ makes the model most overfitting. There are two possible reasons. First, training with a masking ratio of 0.5 gives the input model the most freedom, making validation easier and testing harder. Second, having the same masking ratio could make the model rely on finding masking information from the mesh. An opposite example is (0.6, 0.1) points. The model is trained at a masking ratio of 0.6 but tested at a masking ratio of 0.1. At this time, the masking still helps purge out the redundant information unrelated to the classification. But also, the training and test difference make the model force itself to discard information on masking but find common details. For more accuracy curves under different training and test masking ratios, see Section A.3.

6 Conclusion

We propose a self-supervised mesh encoding approach to learn point and shape features on meshes that use three self-supervised losses, including context, COD, and autoencoding multi-scale graph-based encoder. We thoroughly evaluated our model on mesh classification and segmentation benchmarks. The results suggest that the learned block-level and class-level features outperform prior state-of-the-art models in self-supervised representation learning. For instance, in ModelNet40 shape classification tasks, our model achieved the state-of-the-art (among self-supervised mesh encoders) accuracy of 89.8%. We also find that different combinations of test and training masking ratios in MGMA could provide varying information to downstream tasks. In the ShapeNetPart segmentation task, it achieved a mIoU of 78.5, which outperforms the state-of-the-art mesh encoders. We hope our work could provide a new direction at mesh deep learning analysis and self-supervised learning on mesh data.

References

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds, 2018.

Pulkit Agrawal, Joao Carreira, and Jitendra Malik. Learning to see by moving. In *The IEEE International Conference on Computer Vision (ICCV)*, pp. 37–45, December 2015.

Elie Aljalbout, Vladimir Golkov, Yawar Siddiqui, and Daniel Cremers. Clustering with deep learning: Taxonomy and new methods. *arXiv preprint arXiv:1801.07648*, 2018.

James Atwood and Donald F. Towsley. Diffusion-convolutional neural networks. In *NIPS*, 2016.

I. Bello, B. Zoph, Q. Le, A. Vaswani, and J. Shlens. Attention augmented convolutional networks. In *ICCV*, 2019.

Giorgos Bouritsas, Sergiy V. Bokhnyak, Stylianos Ploumpis, Michael M. Bronstein, and Stefanos Zafeiriou. Neural 3d morphable models: Spiral convolutional networks for 3d shape representation learning and generation. *ICCV*, 2019.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv*, 2020.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, 2013.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *The European Conference on Computer Vision (ECCV)*, pp. 132–149, September 2018.

Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository, 2015.

Haolan Chen, Shitong Luo, Xiang Gao, and Wei Hu. Unsupervised learning of geometric sampling invariant representations for 3d point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 893–903, 2021.

Mark Chen, Alec Radford, Rewon Child, Jeff Wu, and Heewoo Jun. Generative pretraining from pixels. In *ICML*, 2020a.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In Hal Daumé III and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 1597–1607. PMLR, 13–18 Jul 2020b. URL `https://proceedings.mlr.press/v119/chen20j.html`.

Yen-Chun Chen, Linjie Li, Licheng Yu, Ahmed El Kholy, Faisal Ahmed, Zhe Gan, Yu Cheng, and Jingjing Liu. UNITER: UNiversal Image-TExt Representation Learning. In *ECCV*, 2020c.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv*, 2019.

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. In *ICLR*, 2020.

Rodrigo Santa Cruz, Leo Lebrat, Pierrick Bourgeat, Clinton Fookes, Jurgen Fripp, and Olivier Salvado. Deepcsr: A 3d deep learning approach for cortical surface reconstruction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 806–815, 2021.

Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 2017.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019.

Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Unsupervised visual representation learning by context prediction. In *The IEEE International Conference on Computer Vision (ICCV)*, pp. 1422–1430, December 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale, 2021.

Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, and Yue Gao. Meshnet: Mesh neural network for 3d shape representation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 8279–8286, 2019.

M. Fey, J. E. Lenssen, F. Weichert, and H. Müller. Splinecnn: Fast geometric deep learning with continuous b-spline kernels. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018.

Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing, 2018.

Matheus Gadelha, Aruni RoyChowdhury, Gopal Sharma, Evangelos Kalogerakis, Liangliang Cao, Erik Learned-Miller, Rui Wang, and Subhransu Maji. Label-efficient learning on point clouds using approximate convex decompositions, 2020.

Kamran Ghasedi Dizaji, Amirhossein Herandi, Cheng Deng, Weidong Cai, and Heng Huang. Deep clustering via joint convolutional autoencoder embedding and relative entropy minimization. In *The IEEE International Conference on Computer Vision (ICCV)*, pp. 5736–5745, Oct 2017.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. In *International Conference on Learning Representations (ICLR)*, 2018.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *ArXiv*, 2017.

Zhizhong Han, Xiyang Wang, Yu-Shen Liu, and Matthias Zwicker. Multi-angle point cloud-vae: Unsupervised feature learning for 3d point clouds from multiple angles by joint self-reconstruction and half-to-half prediction, 2019.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn: a network with an edge. *ACM Trans. Graph.*, 2019a.

Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. Meshcnn. *ACM Transactions on Graphics*, 38(4):1–12, Jul 2019b. ISSN 1557-7368. doi: 10.1145/3306346.3322959. URL `http://dx.doi.org/10.1145/3306346.3322959`.

Kaveh Hassani and Mike Haley. Unsupervised multi-task feature learning on point clouds, 2019.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16000–16009, 2022.

Mikael Henaff, Joan Bruna, and Yann LeCun. Deep convolutional networks on graph-structured data. *CoRR*, 2015.

John R Hershey, Zhuo Chen, Jonathan Le Roux, and Shinji Watanabe. Deep clustering: Discriminative embeddings for segmentation and separation. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 31–35, March 2016.

Jonathan Ho, Nal Kalchbrenner, Dirk Weissenborn, and Tim Salimans. Axial attention in multidimensional transformers. *arXiv*, 2019.

C. Hsu and C. Lin. Cnn-based joint clustering and representation learning with feature drift compensation for large-scale image data. *IEEE Transactions on Multimedia*, 20(2):421–429, Feb 2018.

Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *CVPR*, 2018.

Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *ICCV*, 2019.

Shi-Min Hu, Zheng-Ning Liu, Meng-Hao Guo, Jun-Xiong Cai, Jiahui Huang, Tai-Jiang Mu, and Ralph R Martin. Subdivision-based mesh convolution networks. *ACM Transactions on Graphics (TOG)*, 41(3):1–16, 2022.

Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018.

Jingwei Huang, Haotian Zhang, Li Yi, Thomas Funkhouser, Matthias Nießner, and Leonidas J Guibas. Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019.

Simon Jenni and Paolo Favaro. Self-supervised feature learning by learning to spot artifacts. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2733–2742, June 2018.

Michael Kazhdan, Thomas Funkhouser, and Szymon Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, SGP '03, pp. 156–164, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association. ISBN 1-58113-687-0. URL http://dl.acm.org/citation.cfm?id=882370.882392.

Seonggyeom Kim and Dong-Kyu Chae. Exmeshcnn: An explainable convolutional neural network architecture for 3d shape analysis. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 795–803, 2022.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

Thomas Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *ArXiv*, 2016.

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

Ron Levie, Federico Monti, Xavier Bresson, and Michael M. Bronstein. Cayleynets: Graph convolutional neural networks with complex rational spectral filters. *IEEE Transactions on Signal Processing*, 2019.

Chun-Liang Li, Manzil Zaheer, Yang Zhang, Barnabas Poczos, and Ruslan Salakhutdinov. Point cloud gan, 2018.

Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. In *Arxiv*, 2019.

Z Lian, A Godil, B Bustos, M Daoudi, J Hermans, S Kawamura, Y Kurita, G Lavoua, P Dp Suetens, et al. Shape retrieval on non-rigid 3d watertight meshes. In *Eurographics workshop on 3d object retrieval (3DOR)*. Citeseer, 2011.

Yaqian Liang, Shanshan Zhao, Baosheng Yu, Jing Zhang, and Fazhi He. Meshmae: Masked autoencoders for 3d mesh data analysis. *arXiv preprint arXiv:2207.10228*, 2022.

Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1886–1895, 2018.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. *arXiv*, 2020.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *Advances in neural information processing systems*, 32, 2019.

Jonathan Masci, Davide Boscaini, Michael M. Bronstein, and Pierre Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *ICCVW*, 2015. doi: 10.1109/ICCVW.2015.112.

A. Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 2009.

Francesco Milano, Antonio Loquercio, Antoni Rosinol, Davide Scaramuzza, and Luca Carlone. Primal-dual mesh convolutional neural networks. *Advances in Neural Information Processing Systems*, 33:952–963, 2020.

Erxue Min, Xifeng Guo, Qiang Liu, Gen Zhang, Jianjing Cui, and Jun Long. A survey of clustering with deep learning: From the perspective of network architecture, 2018.

F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. In *CVPR*, 2017. doi: 10.1109/CVPR.2017.576.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. *ArXiv*, 2016.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European Conference on Computer Vision (ECCV)*, pp. 69–84. Springer, 2016.

Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 165–174, 2019.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In *ICML*, 2018.

Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A. Efros. Context encoders: Feature learning by inpainting. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2536–2544, June 2016.

Omid Poursaeed, Tianxing Jiang, Han Qiao, Nayun Xu, and Vladimir G. Kim. Self-supervised learning of point clouds via orientation estimation, 2020.

Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding with unsupervised learning. *Technical Report*, 2018.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *Technical Report*, 2019.

Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jon Shlens. Stand-alone self-attention in vision models. In *NeurIPS*, 2019.

Edoardo Remelli, Pierre Baque, and Pascal Fua. Neuralsampler: Euclidean point cloud auto-encoder and sampler, 2019.

Uri Shaham, Kelly Stanton, Henry Li, Ronen Basri, Boaz Nadler, and Yuval Kluger. Spectralnet: Spectral clustering using deep neural networks. In *International Conference on Learning Representations (ICLR)*, 2018.

Yi Shi, Mengchen Xu, Shuaihang Yuan, and Yi Fang. Unsupervised deep shape descriptor with point distribution learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9353–9362, 2020.

Vinit Veerendraveer Singh, Shivanand Venkanna Sheshappanavar, and Chandra Kambhamettu. Mesh classification with dilated mesh convolutions. In *2021 IEEE International Conference on Image Processing (ICIP)*, pp. 3138–3142. IEEE, 2021a.

Vinit Veerendraveer Singh, Shivanand Venkanna Sheshappanavar, and Chandra Kambhamettu. Meshnet++: A network with a face. In *Proceedings of the 29th ACM International Conference on Multimedia*, pp. 4883–4891, 2021b.

An Ping Song, Xin Yi Di, Xiao Kang Xu, and Zi Heng Song. Meshgraphnet: An effective 3d polygon mesh recognition with topology reconstruction. *IEEE Access*, 8:205181–205189, 2020. doi: 10.1109/ACCESS.2020.3037236.

Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pp. 945–953, 2015.

Chen Sun, Austin Myers, Carl Vondrick, Kevin Murphy, and Cordelia Schmid. Videobert: A joint model for video and language representation learning. In *ICCV*, 2019a.

Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B Tenenbaum, and William T Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

Yongbin Sun, Yue Wang, Ziwei Liu, Joshua E. Siegel, and Sanjay E. Sarma. Pointgrow: Autoregressively learned point cloud generation with self-attention, 2019b.

Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017.

Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2598–2606, 2018.

Carl Vondrick, Abhinav Shrivastava, Alireza Fathi, Sergio Guadarrama, and Kevin Murphy. Tracking emerges by colorizing videos. In *The European Conference on Computer Vision (ECCV)*, pp. 391–408, September 2018.

Huiyu Wang, Yukun Zhu, Bradley Green, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Axial-deeplab: Stand-alone axial-attention for panoptic segmentation. In *ECCV*, 2020.

Xiaolong Wang and Abhinav Gupta. Unsupervised learning of visual representations using videos. In *The IEEE International Conference on Computer Vision (ICCV)*, pp. 2794–2802, December 2015.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *CVPR*, 2018.

Donglai Wei, Joseph J. Lim, Andrew Zisserman, and William T. Freeman. Learning and using the arrow of time. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8052–8060, June 2018.

Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. In *ICLR*, 2019.

Bichen Wu, Chenfeng Xu, Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Masayoshi Tomizuka, Kurt Keutzer, and Peter Vajda. Visual transformers: Token-based image representation and processing for computer vision. *arxiv*, 2020.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pp. 82–90, 2016.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes, 2015.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *Proceedings of The 33rd International Conference on Machine Learning (ICML)*, volume 48, pp. 478–487. PMLR, June 2016.

Bo Yang, Xiao Fu, Nicholas D. Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, volume 70, pp. 3861–3870. PMLR, August 2017.

Jianwei Yang, Devi Parikh, and Dhruv Batra. Joint unsupervised learning of deep representations and image clusters. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5147–5156, June 2016.

Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation, 2018.

Zhangsihao Yang, Or Litany, Tolga Birdal, Srinath Sridhar, and Leonidas Guibas. Continuous geodesic convolutions for learning on 3d shapes. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 134–144, 2021.

Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6):210:1–210:12, November 2016.

Ling Zhang and Zhigang Zhu. Unsupervised feature learning for point cloud by contrasting and clustering with graph convolutional neural network, 2019.

Zaiwei Zhang, Rohit Girdhar, Armand Joulin, and Ishan Misra. Self-supervised pretraining of 3d features on any point-cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10252–10263, 2021.

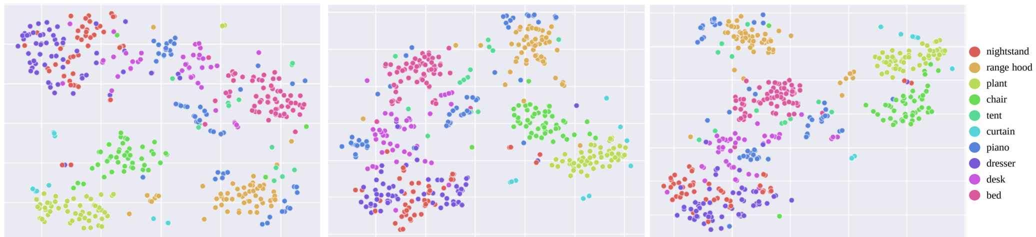
Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *CVPR*, 2020.

Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks, 2019.

A Appendix

A.1 t-SNE visualization

We visualize graph embeddings obtained by fixing the test masking ratio to 0 using t-SNE (Van der Maaten & Hinton, 2008). We could observe that plant and chair are two classes clustering close but easy to distinguish. The reason could be that both of them are tall cuboids. But chairs have a more regular appearance than plants. The piano and range hood are also the same cases. They have a similar outlook but are different when looking in detail. Usually, the mesh of the range hood has a hole inside its body. Another confusion to the model is the nightstand and dresser, two potentially similar objects. The t-SNE plot at ratios 0.3 and 0.6 are quite similar in clustering. While the plot at a ratio of 0.9 begins to confuse objects like desks and pianos (see the bed category move from corner to the center).



(a) Train mask ratio 0.3 (b) Train mask ratio 0.6 (c) Train mask ratio 0.9

Figure 6: **Visualization of t-SNE on ModelNet40.** We fix the test masking ratio to 0 and choose 10 random classes for rendering the figures.

A.2 NETWORK ARCHITECTURE

The overall architecture of our network is shown in Figure 7. It has a heavier encoder than the decoder. It follows the design logic in MaskMAE since, after pre-training, we no longer need the decoder. The reason we did not use batch normalization in the decoder is to follow (Achlioptas et al., 2018). And decoder is not the mean focus of our paper. The input features to the graph are computed using descriptors defined in (Singh et al., 2021b), which are 320-dimension vectors for face nodes. For the first two mesh graph attention blocks, we use 1-ring neighbors for neighboring lookup. For the third mesh graph attention block, we use 2-ring neighbors.

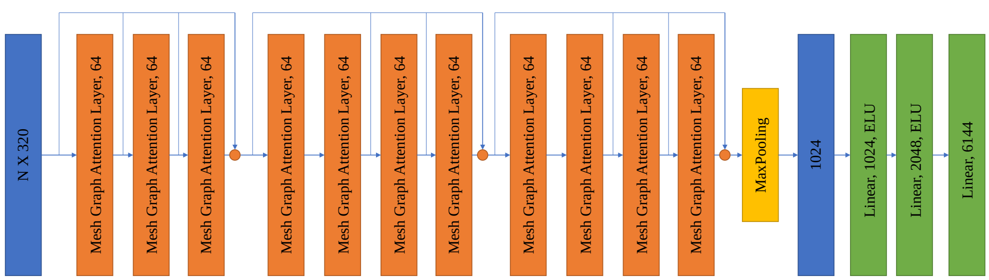


Figure 7: **Neural Network Architecture.** N stands for the number of face nodes. The number after each layer stands for the dimension of the output embedding. The orange dot stands for concatenating the forward embeddings from previous layers. Each mesh graph attention layer has batch normalization and ReLU layer following.

A.3 MASKING RATIO ANALYSIS

In Figure 8, we plot the accuracy curve under different training and test masking ratio. Three patterns of accuracy curve are found when the test masking ratios are fixed.

The first happens at test masking ratios of 0.0, 0.1, and 0.2. The accuracy goes up and down. The second one is at test masking ratios of 0.3, 0.4, 0.5, and 0.6. The accuracy goes up and done and up again. The last one happened at test masking ratios of 0.7, 0.8, and 0.9. The accuracy goes up. The reason is straightforward for the first and third patterns. For the first pattern, the models are trained with low masking ratios. When the training masking ratio increases, the models focus on extracting information other than just masking, which explains why there is an increasing curve at the beginning. And when the ratio is too high, there is not enough information. Thus, the curve begins to drop.

The third pattern is caused by test masking ratios being too high such that the models trained with low masking ratios could efficiently capture the information of testing meshes. And only models trained under high masking ratios could capture information from testing meshes.

The second pattern is generated when the first and third patterns merge.

One pattern of accuracy curve is found when the training masking ratios are fixed.

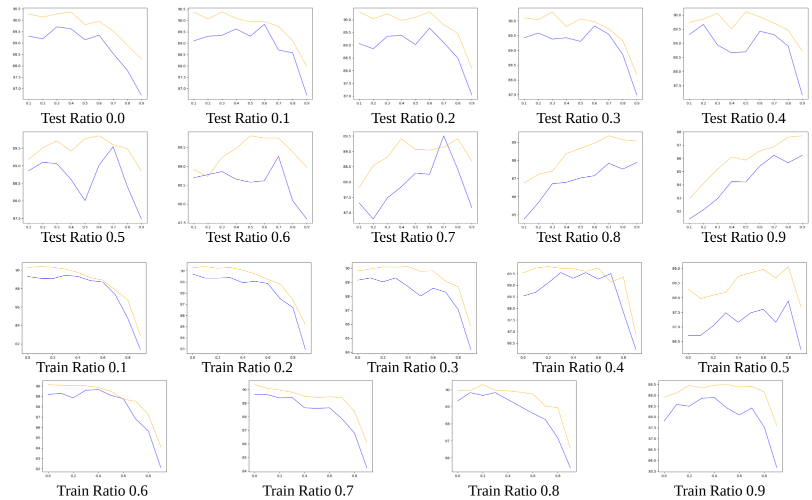


Figure 8: **Masking ratio's effect on accuracy curves.** The validation accuracy curve (in yellow) and test accuracy curve (in blue) are plotted by fixing different test and training masking ratios.

A.4 Segmentation Visualization Results

More segmentation results are shown in Figure 9

A.5 Reconstruction Visualization Results

More reconstruction results are shown in Figure 10.

Figure 9: **Segmentation results.**

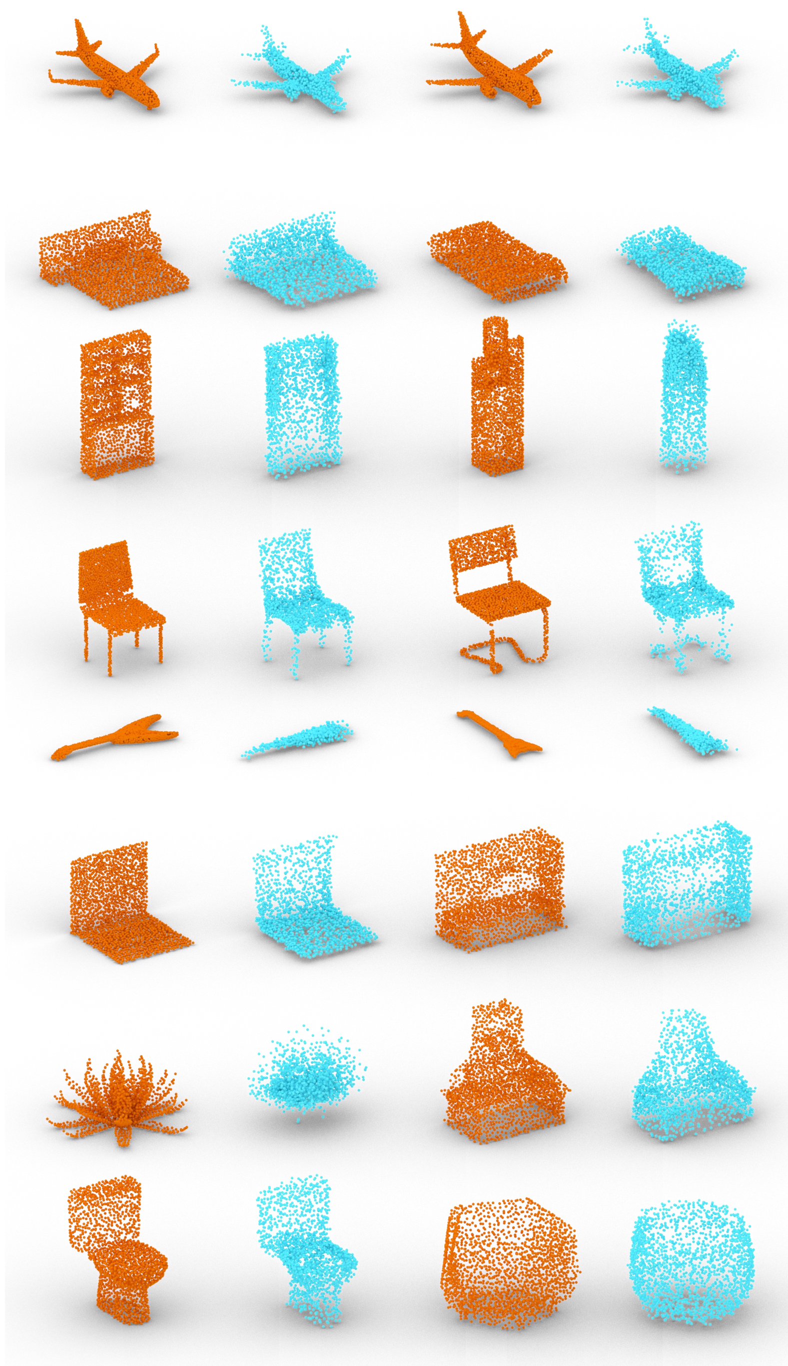


Figure 10: **Reconstruction results.**