sensors

MDPI

Article

Deep Neural Network for 3D Shape Classification Based on Mesh Feature

Mengran Gao [1], Ningjun Ruan [1,*], Junpeng Shi [2] and Wanli Zhou [1]

[1] School of Electronic and Information, Yangtze University, Jingzhou 434023, China
[2] College of Electronic Countermeasure, National University of Defense Technology, Hefei 230037, China
* Correspondence: ruannj@yangtzeu.edu.cn

Abstract: Virtual reality, driverless cars, and robotics all make extensive use of 3D shape classification. One of the most popular ways to represent 3D data is with polygonal meshes. In particular, triangular mesh is frequently employed. A triangular mesh has more features than 3D data formats such as voxels, multi-views, and point clouds. The current challenge is to fully utilize and extract useful information from mesh data. In this paper, a 3D shape classification network based on triangular mesh and graph convolutional neural networks was suggested. The triangular face of this model was viewed as a unit. By obtaining an adjacency matrix from mesh data, graph convolutional neural networks can be utilized to process mesh data. The studies were performed on the ModelNet40 dataset with an accuracy of 91.0%, demonstrating that the classification network in this research may produce effective results.

Keywords: triangular mesh; graph convolutional neural networks; 3D shape classification

check for updates

Citation: Gao, M.; Ruan, N.; Shi, J.; Zhou, W. Deep Neural Network for 3D Shape Classification Based on Mesh Feature. *Sensors* **2022**, *22*, 7040. https://doi.org/10.3390/s22187040

Academic Editors: Marcin Woźniak and Adrian Munteanu

Received: 25 July 2022
Accepted: 14 September 2022
Published: 17 September 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

With the rapid development of 3D data-capturing devices, 3D data collection has become more convenient and faster. Three-dimensional data are foundational to computer graphics and computer vision, and it contains a wealth of geometric, shape, and scale information. The use of 3D models is increasing in daily life, such as in autonomous driving [1,2], virtual reality, and remote sensing mapping [3], all of which require advanced processing [4] and analysis of the collected 3D data. The study of how to effectively classify, identify, and segment 3D models is a hot topic at present.

As computer vision and deep learning have developed rapidly, the study of 3D shapes has shifted from handcraft features [5,6] to deep learning methods. Three-dimensional shapes are available in various representations such as voxels, multi-view, point cloud and polygon meshes, as shown in Figure 1. Early convolutional neural networks have made great strides in the classification of 2D images, but they cannot be applied to unstructured data. MVCNN [7] used the deep learning algorithm on 2D images under a multi-view of the 3D model and obtained the classification results of 3D shapes. However, the multi-view lacked depth information and could not describe the 3D spatial characteristics of objects well. Recently, graph convolutional neural networks have achieved good results on point cloud classification tasks. DGCNN [8] uses the edge convolution module to learn topological features of point clouds by graph convolutional networks, which can better capture local geometric information and achieve better results, but with too many training parameters.

Voxel, multi-view, and point cloud are currently used more frequently than mesh data in most 3D model classification techniques. The point cloud lacks surface information on the objects and is disorganized, having few connections between its points. Information will unavoidably be lost during the voxelization process of 3D models, and large resolutions will necessitate large amounts of memory. Issues with feature learning based on multi-view and voxel data include significant 3D feature loss and numerous processing stages.

Mesh data include information about features including vertices, edges, faces, corners, and normals. Mesh data are more descriptive of objects than other types of data, and it is now being investigated how to effectively extract information from mesh data and to fully utilize it. Based on the advantage of graph convolutional neural networks [9] excelling at processing non-Euclidean structure data, this paper proposes a 3D model classification network based on graph convolutional neural networks combined with triangular mesh, which can learn the features of 3D models directly and achieve the classification of 3D models. The ModelNet40 [10] dataset is used to test the model, and the results for 3D model classification are promising.

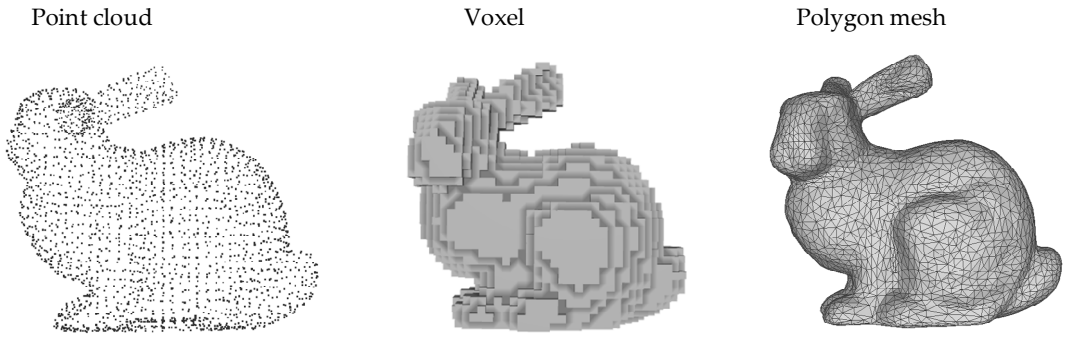


Figure 1. Point cloud, voxel, and polygon mesh representation of 3D models.

2. Related Works

Several methods have been proposed for obtaining features in mesh data. Jiao et al. [11] used several metrics such as dihedral angles, edge angles, and eigenvectors of the vertex normal tensor matrix to identify feature point, feature edge, and feature edge directions. Kim et al. [12,13] calculated the maximum point set to represent the feature points of the 3D mesh by using the average geodesic distance function. Hu et al. [14] proposed a global rarity to describe global saliency: the rarer the part on the 3D mesh, the better the characteristics of the entire mesh.

The following is an introduction to related work on 3D model classification using deep learning.

Multi-view: Initially, scholars represented the projections of the model from different perspectives as 2D images and then used convolutional neural networks to process the projected images. The MVCNN [7], GVCNN [15], and VMVCNN [16] methods transform the 3D model into 2D images obtained by shooting in different view cases and extracting image features to perform the classification. Zhang et al. [17] proposed an effective recognition model based on multi-view convolutional neural networks. Qi et al. [16] added azimuth and elevation angle change features to the training set to improve the performance of the classification model based on the MVCC. Gao et al. [18] systematically evaluated the performance of deep learning features in view-based 3D model retrieval. Projection from multiple angles leads to a lack of global information perception and is difficult to apply in scene segmentation and object detection tasks.

Volumetric: A volumetric occupancy grid is often used in order to represent the environment state as a 3D mesh. Zhang et al. [19] proposed to convert point clouds into regular voxels and to then input them into convolutional neural networks for feature extraction. Riegler et al. [20] transformed the point cloud into an octree format to extract features. Maturana et al. [21] proposed VoxNet, which incorporates voxels and multi-views. Wu et al. [10] developed a 3D-ShapeNets model that handles voxelization directly. Kd-Net [22] constructed the point cloud into a kd-tree and then classified the points. The spherical CNN [23] projects 3D meshes onto closed spheres, confirming spherical convolution's effectiveness in point cloud classification. In comparison with the original model, the voxelized model has a low resolution, which leads to a significant loss of information.

Point clouds: PointNet [24] and PointNet++ [25] network models directly process point cloud data. PointNet can learn from disordered point clouds but ignores the extraction of local features. PointNet++ considers local features of point clouds but ignores the connection between points, and the operation of aggregating point set features is time-consuming. Li et al. [26] proposed PointCNN so that the input order of the point cloud does not affect the convolution operation.

Graph: According to the definition [9,27] of convolution in the graph, this calculation method [28,29] has been employed on point clouds. Simonovsky et al. [30] proposed a network edge ECC applied to any graph structure using maximum sampling to aggregate vertex information. However, there is a problem of high computational cost. KCNet [31] orders local point clouds using graphs and aggregates local features through graph convolution. DGCNN [8] provides a dynamic graph edge convolution module that improves the network's ability to obtain local features, ignoring the vector direction between points, resulting in some information loss. Point GCN [32] recycled edges of the graph within each layer, avoiding unnecessary point cloud grouping and sampling. Grid-GCN [33] proposed a module to reduce theoretical time complexity and to enhance space coverage.

Based on the excellent performance of graph convolutional neural networks in processing unstructured data, in this paper, we propose a 3D shape classification network that combines the features of mesh data. The 3D model is represented as a graph in the model and leverages graph convolutional networks to enhance local feature extraction. The next section describes in detail the model proposed in this paper.

3. Methods

3.1. Overview

We propose a 3D shape classification network based on triangular mesh features and graph convolutional neural networks. Figure 2 shows the flow chart of the algorithm proposed in this paper. The flow chart consists of two parts, one is the simplification and feature extraction of the mesh data, and the other is the 3D shape classification based on graph convolutional neural networks.

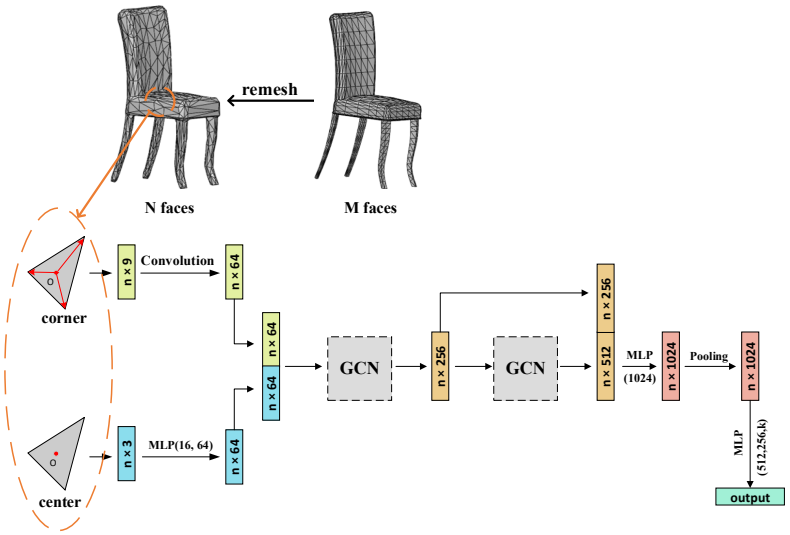


Figure 2. Algorithm flow chart. (After feature extraction of the faces, they are input to the classification network for aggregated vertex information and then pooled to obtain global features. Finally, the classification results are output.)

Triangle meshes are a common way to display 3D models, which consist of three parts: vertices, edges, and faces. A face refers to a triangular face formed by interconnecting three adjacent vertices in the mesh data. The triangular mesh data can be defined as $\boldsymbol{M} = (\boldsymbol{V}, \boldsymbol{F})$, where $\boldsymbol{V} = \{Vi | Vi \in R^3\}$ denotes the set of points and $\boldsymbol{F} = \left\{ f_{ijk} = (v_i, v_j, v_k) \in V, i \neq j, j \neq k \right\}$ de-

notes the set of faces. Triangular mesh data are better equipped to describe 3D models than other data types such as voxels, multi-views, and point clouds. Furthermore, the explicit connection feature of the mesh makes it easier to extract the adjacency matrix of the mesh.

We should simplify the original input 3D model mesh data to obtain a model with no more than 1024 faces. In order to combine the data from nearby vertices, the face is supplied into the model. Multi-scale local feature splicing and pooling then produces the classification results. The model for mesh processing and classification is described in full below.

3.2. Processing Mesh

The triangular mesh contains a rich set of features that can effectively represent the geometric environment. We will extract three initial features from the mesh as the input to the classification model. The three initial features are shown in Figure 3.

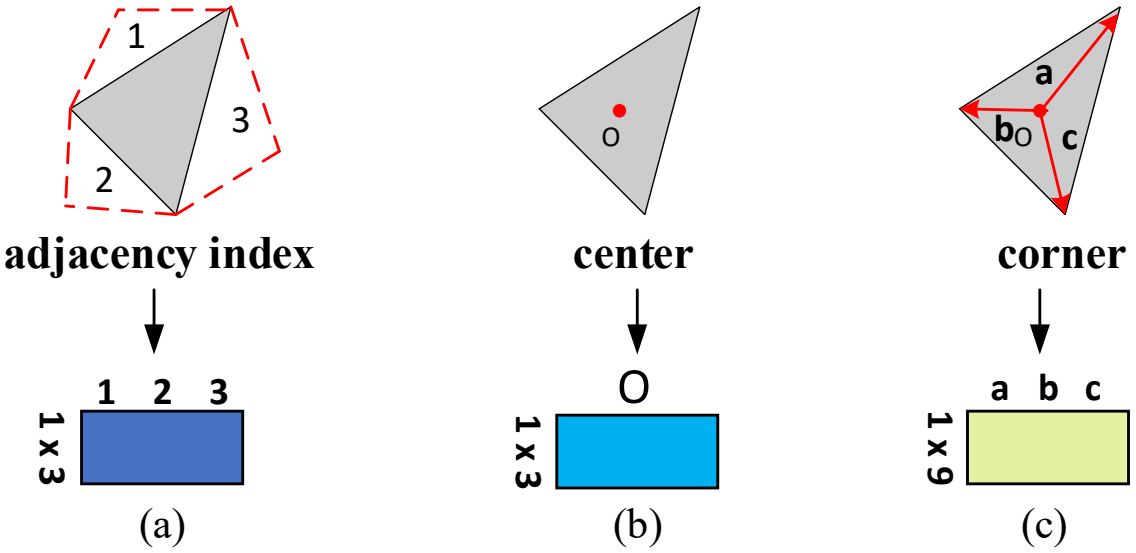


Figure 3. Initial feature extraction: (**a**) indexing of adjacent faces; (**b**) coordinates of the center point of the face; and (**c**) the vector from the center point to each of the three vertices. **a**, **b**, and **c** denote the vectors from the center point of the triangle to the vertices.

We presumed that the meshes representing the 3D models are manifolds [34], that each edge in the mesh is only connected to one or two faces, and that each triangular face is connected to no more than three other triangular faces. We transform the mesh data into a list of faces, treating the face as the sole unit. The list can be used to find each face's adjacency index. A face is filled with its own index if it connects with less than three other faces.

Because the coordinates of the three vertices of the face are known to be A, B, and C in the original data, we can obtain the coordinates of the center point of the face and the corner by using Equations (1) and (2).

$$O = \begin{cases} x_O = (x_A + x_B + x_C) \times \frac{1}{3} \\ y_O = (y_A + y_B + y_C) \times \frac{1}{3} \\ z_O = (z_A + z_B + z_C) \times \frac{1}{3} \end{cases} \tag{1}$$

$$\mathbf{e}_{OA} = \begin{cases} x_{\mathbf{e}} = (x_A - x_O) \\ y_{\mathbf{e}} = (y_A - y_O) \\ z_{\mathbf{e}} = (z_A - z_O) \end{cases} \tag{2}$$

3.3. Model Design

The graph convolutional neural networks are capable of processing unstructured data. The graph features will be learned using the spectral domain graph convolutional neural network proposed by Kipf [9] in this paper. Supported by spectral graph theory, the kernel of the neural network is defined by a filter for graph signal processing, giving it a better filtering capability. The input of the graph convolutional network consists of two parts: First, a feature description $\mathbf{x}_i$ for each node i, which denotes the $N \times M$ feature matrix (N denotes the number of nodes, and M denotes the number of features of the input). The second is the adjacency matrix of the graph. (Construct the adjacency matrix using the adjacency relationship between faces.) The value of each element in the adjacency matrix can be calculated by Equation (3).

$$\mathbf{A}[i,j] = \begin{cases} 1, & (v_i, v_j) \in E(G) \\ 0, & (v_i, v_j) \notin E(G) \end{cases} \tag{3}$$

Suppose that there are L layers of graph convolution and that l denotes the current number of layers. $H^{(l)}$ denotes the output of layer l. $\mathbf{A}$. denotes the adjacency matrix of the graph of N nodes ($\mathbf{A} \notin R^{N \times N}$), so each graph convolutional layer neural network can be represented by the nonlinear function Equation (4).

$$H^{(l+1)} = f(H^l, \mathbf{A}) \tag{4}$$

For a graph convolutional neural network to retain information about the nodes, each node needs to be connected to itself. Next, normalize $\mathbf{A}$ by $\mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$, where $\mathbf{D}$ is the degree diagonal matrix of the nodes and $\mathbf{A}$=$\mathbf{A}$+$\mathbf{I}_N$ ($\mathbf{I}_N$ is the unit matrix). The propagation equation of the graph convolutional neural network can be expressed as

$$H^{(l+1)} = \sigma\left(\mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}\right) \tag{5}$$

Equation (5) shows that constructing the graph's adjacency matrix is the key to using graph convolution. In this paper, the adjacency matrix and the degree matrix can be built by the connection relationship with the face. The graph convolution can be applied to the mesh data.

Before inputting the face features into the graph convolutional network for aggregation, the center point features and the corner vector features need to be processed separately. Referring to the method in MeshNet [35], rotational convolution is used to process the corner vectors of the triangular face, which only works on two corner vectors at a time. Figure 4 shows the diagram of rotational convolution. Suppose that $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$ are the vectors of a face from the center point to the three angles and define its convolution output as Equation (6). Finally, the output is passed through the fully connected layer to obtain a feature with a length of 64.

$$\mathbf{a} \otimes \mathbf{b} + \mathbf{b} \otimes \mathbf{c} + \mathbf{c} \otimes \mathbf{a} \tag{6}$$

where $\otimes$ means a convolution operation and $(\mathbf{a}, \mathbf{b}, \mathbf{c}) \in R^3$.

The $O \in R^{N \times 3}$ denotes the center point feature of the face. Increasing the center point feature to 64 dimensions through a fully connected layer results in obtaining the center point feature dimension as $O \in R^{N \times 64}$. We connect the convolutional output of the corner vectors with the features of the center points to obtain the high-dimensional features and to input them into the graph convolutional neural network for feature aggregation.

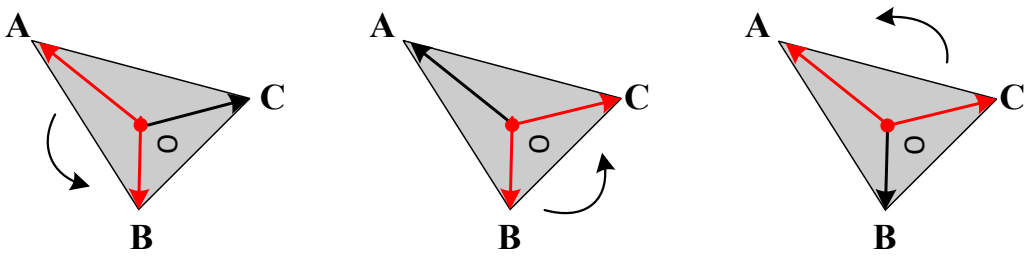


Figure 4. Rotational convolution schematic. **A**, **B**, **C** represent the three vertices of the triangle, O is the center point of the triangle, and the curved arrow outside the triangle represents two adjacent angle vectors for convolution operation.

In this paper, a two-layer graph convolutional neural network is used. We input the high-dimensional features into the first layer of graph convolution to obtain the information on aggregated first-degree neighboring vertices. Then, after the activation function, the output features go through the second layer of graph convolution to obtain the information of the aggregated second-degree neighboring vertices. The information of the aggregated first-degree neighboring vertices and the aggregated second-degree neighboring vertices are stitched together to obtain multi-resolution features, as shown in Figure 5. Degree = 1 denotes the aggregation of first-degree neighboring vertices, and degree = 2 denotes the aggregation of second-degree neighboring vertices.

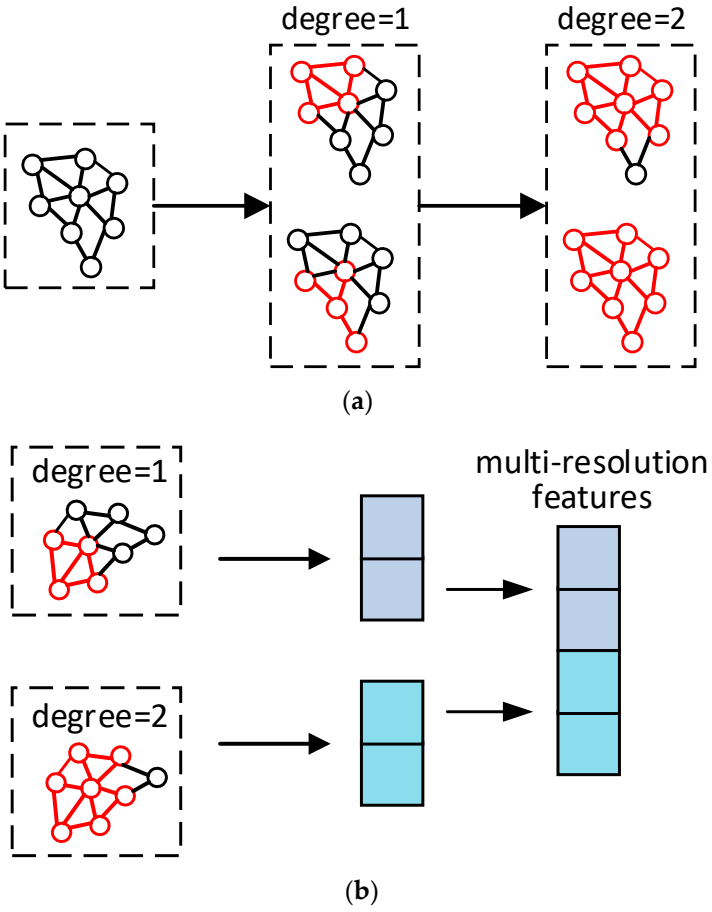


Figure 5. (**a**) Schematic diagram of two-layer graph convolution calculation. (**b**) Feature splicing schematic. Purple represents the information features of the aggregated first-degree neighboring vertices and blue represents the information features of the aggregated second-degree neighboring vertices.

The multi-resolution features are passed through a fully connected layer (1024) to obtain higher-dimensional features, and the information redundancy in the high-dimensional space facilitates the subsequent global pooling operation. In order to obtain the global features, global fusion of the extracted features is required. Max pooling is a nonlinear feature fusion function that is insensitive to the order of elements, and we take the max pooling operation on graphs for graph data. The output features are subjected to max pooling in order to obtain global features, and then, the classification results are obtained through the fully connected layer (512, 256, and 40) and the Softmax layer.

4. Experiments and Results

We conducted experiments to prove the usefulness of our classification model on ModelNet40 [10]. Table 1 shows the parameter settings for model training. We used the PyTorch 1.10.1 (Soumith Chintala, America) deep learning framework and Python 3.6 (Guido van Rossum, Holland). Hardware configuration: CPU: Intel Xeon Platinum 8259CL, GPU: Tesla T4 with 16 G of video memory.

Table 1. Training parameter setting.

Optimizer	Lr	Batch	Weight Decay	Epoch
ADAM	0.001	128	0.0005	250

Many research institutions have opened datasets with 3D models, and we will apply the model proposed in this paper on ModelNet40 for the classification. ModelNet40 has 40 different 3D shape models with 12,311 CAD models, including their mesh information, of which 9843 models are used as the training sets and 2468 models are used as the test sets. Before conducting the experiments, each model needs to be simplified to 1024 faces. If the number of faces is not enough, the existing faces are randomly selected to be filled, which normalizes the model to the unit ball centered at the origin.

Table 2 shows the experimental results of the model in this paper on the ModelNet40 dataset. The overall accuracy is the accuracy of the classification of all 3D models in the dataset, and the mean class accuracy is the average of the accuracy of each category. The overall accuracy of the proposed method compared with existing methods in this paper is 6.3% higher than 3D ShapeNets, 8% higher than VoxNet, and 0.3% higher than PointNet++. We proposed to make full use of the mesh data information by extracting the center points and corner vectors of the faces, which can enhance the ability to describe the local information by considering the spatial position of the faces in the 3D model, the structure of the faces themselves, and the adjacency relationship between the faces. The experimental results show that the model proposed in this paper effectively captures local information from triangular meshes. The classification accuracy of the model in this paper can achieve better results on the representation based on a triangular mesh.

Table 2. Comparison of classification results on ModelNet40.

Model	Representation	Overall Accuracy (%)	Mean Class Accuracy (%)
MVCNN [7]	view	90.1	79.5
VoxNet [21]	volume	83.0	85.9
PointNet [24]	point	89.2	86.2
PointNet++ [25]	point	90.7	-
Kd-Net [22]	point	90.6	88.5
SO-Net [36]	point	90.8	-
Momenet [37]	point	89.3	86.1
LKPO-GNN [38]	point	90.9	88.2
ReebGCN [39]	point	89.9	87.1
Ours	mesh	91.0	89.1

Furthermore, we also experimented with the model of this paper on the ShapeNet dataset [40], which has 16 classes with 16,881 shapes. For the ShapeNet dataset, the overall classification accuracy of this paper's model is 90.8%, which has the potential to achieve good classification results.

An analysis of the spatial complexity and time complexity of the model is shown in Table 3. We used params (number of parameters) to represent the spatial complexity and FLOPs (floating operations conducted for each input sample) to express the time complexity. The classification method proposed in this paper uses only two layers of convolutional neural networks, thus reducing the number of parameters of the model. The point cloud-based classification task algorithm is the most efficient among other data types. The proposed method can achieve comparable results with the point cloud-based method in terms of spatial complexity and time complexity. Therefore, the method proposed in this paper is concise and effective.

Table 3. Space complexity and time complexity for classification.

Model	Params (M)	FLOPs (G)
PointNet [24]	3.48	0.44
PointNet++ [25]	1.48	-
Kd-NET [22]	7.44	-
MVCNN [7]	60.00	62.06
Ours	1.47	1.61

Table 4 compares the classification effects of the model using GCNs with a different number of layers. The overall classification accuracy of the model was 90.2% when using one-layer GCN; the best performance was achieved when using two-layer GCN with an overall classification accuracy of 91%; and the classification ability of the model started to decrease when using three-layer GCN, with an overall classification accuracy of 86.8%. The performance of the model decreased again when using the four-layer GCN, with an overall classification accuracy of 83.6%. The essence of GCN is aggregating neighbor information. Every time the node's features are updated, they aggregate the information of higher order neighbor nodes for any node in the graph. As the number of GCN layers increases, once a certain threshold is reached, the nodes covered by each node converge to the full graph node. This leads to a significant reduction in the diversity of the local network structure of each node, which is very detrimental to the nodes' feature learning. Therefore, we use the two-layer GCN classification model in this paper.

Table 4. Comparisons of overall accuracy when using a different number of GCN layers. We use the ModelNet40 dataset.

Model	GCN Layers			
	1	**2**	**3**	**4**
Ours	90.2%	91.0%	86.8%	83.6%

To investigate the effect of the number of faces in the 3D model on the classification accuracy, we re-simplified the dataset to obtain four kinds of 3D model datasets, with the number of faces being 512, 1024, 2048, and 4096, and input them into the network for the classification experiments. We list the classification results as shown in Table 5. The experimental results show that an increase in the number of faces of the 3D model results in a little improvement in the accuracy of the classification. However, a large number of faces consumes a large amount of computer memory and computing time. To guarantee the network performance and computation speed, we choose to simplify the model to 1024 faces for the experiment.

Table 5. Classification results with a different number of faces on ModelNet40.

Number of Faces	512	1024	2048	4096
Accuracy	90.3%	91.0%	90.8%	91.0%

To analyze the effectiveness of each module in our classification model, we designed ablation experiments to compare the accuracy of ModelNet40 classification under different combinations of modules. Center Module refers to the center point feature of the face and the subsequent fully connected layer. Corner Module refers to the corner vector feature of the face and the rotated convolution layer behind it. GCN refers to the graph convolutional neural network, and after removing the GCN, we pass the initial features directly through the fully connected layer. Table 6 shows the results of the ablation experiments. The experimental results show that only using grid features for classification is not effective, but the addition of the GCN module can help improve the transmission of information in the network and can capture local features better. Finally, we can obtain shape descriptors that have rich features.

Table 6. Classification results of ablation experiments on ModelNet40.

Center Module	√	√	√
Corner Module	√	√	
GCN	√		√
Accuracy	91.0%	89.6%	87.8%

5. Conclusions

In this paper, we proposed a 3D shape classification network that combines the features of mesh data and is capable of learning the mesh data directly. By examining the mesh data's features and by generating an adjacency matrix based on nearby faces, the graph convolution can be applied to the mesh data. Our model is better able to capture 3D model features because of the benefits of using graph convolutional neural networks to analyze non-Euclidean data. The experimental results prove that the classification model in this paper is lighter in terms of the number of parameters and has a better classification effect.

Author Contributions: Supervision, N.R.; Visualization, W.Z.; Writing—original draft, M.G.; Writing—review and editing, J.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ulrich, L.; Nonis, F.; Vezzetti, E.; Moos, S.; Caruso, G.; Shi, Y.; Marcolin, F. Can ADAS Distract Driver's Attention? An RGB-D Camera and Deep Learning-Based Analysis. *Appl. Sci.* **2021**, *11*, 11587. [CrossRef]
2. Sasaki, S.; Premachandra, C. Head Posture Estimation by Deep Learning Using 3-D Point Cloud Data from a Depth Sensor. *IEEE Sens. Lett.* **2021**, *5*, 7002604. [CrossRef]
3. Zheng, G.; Song, Y.; Chen, C. Height Measurement with Meter Wave Polarimetric MIMO Radar: Signal Model and MUSIC-like Algorithm. *Signal Process.* **2022**, *190*, 108344. [CrossRef]
4. Hooda, R.; Pan, W.D.; Syed, T.M. A Survey on 3D Point Cloud Compression Using Machine Learning Approaches. In Proceedings of the SoutheastCon 2022, Mobile, AL, USA, 26 March 2022; pp. 522–529.
5. Guo, W.; Hu, W.; Liu, C.; Lu, T. 3D Object Recognition from Cluttered and Occluded Scenes with a Compact Local Feature. *Mach. Vis. Appl.* **2019**, *30*, 763–783. [CrossRef]
6. Wang, F.; Liang, C.; Ru, C.; Cheng, H. An Improved Point Cloud Descriptor for Vision Based Robotic Grasping System. *Sensors* **2019**, *19*, 2225. [CrossRef] [PubMed]

7. Su, H.; Maji, S.; Kalogerakis, E.; Learned-Miller, E. Multi-View Convolutional Neural Networks for 3D Shape Recognition. In Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV), Santiago, Chile, 7–13 December 2015; pp. 945–953.
8. Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S.E.; Bronstein, M.M.; Solomon, J.M. Dynamic Graph CNN for Learning on Point Clouds. *ACM Trans Graph* **2019**, *38*, 1–12. [CrossRef]
9. Kipf, T.; Welling, M. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv* **2017**, arXiv:abs/1609.02907.
10. Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; Xiao, J. 3D ShapeNets: A Deep Representation for Volumetric Shapes. In Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Boston, MA, USA, 7–12 June 2015; pp. 1912–1920.
11. Jiao, X.; Bayyana, N.R. Identification of C1 and C2 Discontinuities for Surface Meshes in CAD. *Comput Aided Des* **2008**, *40*, 160–175. [CrossRef]
12. Kim, V.G.; Lipman, Y.; Funkhouser, T. Blended Intrinsic Maps. *ACM Trans. Comput. Syst.* **2011**, *30*, 1–12. [CrossRef]
13. Kim, V.G.; Lipman, Y.; Chen, X.; Funkhouser, T. Möbius Transformations for Global Intrinsic Symmetry Analysis. *Comput. Graph. Forum* **2010**, *29*, 1689–1700. [CrossRef]
14. Wu, J.; Shen, X.; Zhu, W.; Liu, L. Mesh Saliency with Global Rarity. *Graph. Models* **2013**, *75*, 255–264. [CrossRef]
15. Feng, Y.; Zhang, Z.; Zhao, X.; Ji, R.; Gao, Y. GVCNN: Group-View Convolutional Neural Networks for 3D Shape Recognition. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 264–272.
16. Qi, C.R.; Su, H.; Nießner, M.; Dai, A.; Yan, M.; Guibas, L.J. Volumetric and Multi-View CNNs for Object Classification on 3D Data. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 5648–5656.
17. Zhang, L.; Sun, J.; Zheng, Q. 3D Point Cloud Recognition Based on a Multi-View Convolutional Neural Network. *Sensors* **2018**, *18*, 3681. [CrossRef] [PubMed]
18. Gao, Z.; Li, Y.; Wan, S. Exploring Deep Learning for View-Based 3D Model Retrieval. *ACM Trans. Multimed. Comput. Commun. Appl.* **2020**, *16*, 1–21. [CrossRef]
19. Zhou, Y.; Tuzel, O. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4490–4499.
20. Riegler, G.; Ulusoy, A.O.; Geiger, A. OctNet: Learning Deep 3D Representations at High Resolutions. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6620–6629.
21. Maturana, D.; Scherer, S. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. 7. In Proceedings of the 2015 IEEE/RSJ international conference on intelligent robots and systems (IROS), Hamburg, Germany, 28 September–2 October 2015.
22. Klokov, R.; Lempitsky, V. Escape from Cells: Deep Kd-Networks for the Recognition of 3D Point Cloud Models. In Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017; pp. 863–872.
23. Cohen, T.; Geiger, M.; Köhler, J.; Welling, M. Spherical CNNs. *arXiv* **2018**, arXiv:abs/1801.10130.
24. Charles, R.Q.; Su, H.; Kaichun, M.; Guibas, L.J. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 77–85.
25. Qi, C.R.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 5105–5114.
26. Li, Y.; Bu, R.; Sun, M.; Wu, W.; Di, X.; Chen, B. PointCNN: Convolution On X-Transformed Points. In Proceedings of the NeurIPS 2018, Montréal, QC, Canada, 3–8 December 2018.
27. Wang, C.; Samari, B.; Siddiqi, K. Local Spectral Graph Convolution for Point Set Feature Learning. In Proceedings of the Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018.
28. Thomas, H.; Qi, C.; Deschaud, J.-E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 6410–6419.
29. Guerrero, P.; Kleiman, Y.; Ovsjanikov, M.; Mitra, N.J. PCPN ET Learning Local Shape Properties from Raw Point Clouds. *Comput. Graph. Forum* **2018**, *37*, 75–85. [CrossRef]
30. Simonovsky, M.; Komodakis, N. Dynamic Edge-Conditioned Filters in Convolutional Neural Networks on Graphs. In Proceedings of the 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 29–38.
31. Shen, Y.; Feng, C.; Yang, Y.; Tian, D. Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling. In Proceedings of the 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 4548–4557.
32. Shi, W.; Rajkumar, R. Point-GNN: Graph Neural Network for 3D Object Detection in a Point Cloud. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020; pp. 1708–1716.
33. Xu, Q.; Sun, X.; Wu, C.-Y.; Wang, P.; Neumann, U. Grid-GCN for Fast and Scalable Point Cloud Learning. In Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); 2020; pp. 5660–5669.

34. Huang, J.; Su, H.; Guibas, L.J. Robust Watertight Manifold Surface Generation Method for ShapeNet Models. *arXiv* **2018**, arXiv:abs/1802.01698.
35. Feng, Y.; Feng, Y.; You, H.; Zhao, X.; Gao, Y. MeshNet: Mesh Neural Network for 3D Shape Representation. *Proc. AAAI Conf. Artif. Intell.* **2019**, *33*, 8279–8286. [CrossRef]
36. Li, J.; Chen, B.M.; Lee, G.H. SO-Net: Self-Organizing Network for Point Cloud Analysis. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA, 18–23 June 2018; pp. 9397–9406.
37. Joseph-Rivlin, M.; Zvirin, A.; Kimmel, R. Momenˆet: Flavor the Moments in Learning to Classify Shapes. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW), Seoul, Korea, 27 October–2 November 2019; pp. 4085–4094.
38. Zhang, W.; Su, S.; Wang, B.; Hong, Q.; Sun, L. Local K-NNs Pattern in Omni-Direction Graph Convolution Neural Network for 3D Point Clouds. *Neurocomputing* **2020**, *413*, 487–498. [CrossRef]
39. Wang, W.; You, Y.; Liu, W.; Lu, C. Point Cloud Classification with Deep Normalized Reeb Graph Convolution. *Image Vis. Comput.* **2021**, *106*, 104092. [CrossRef]
40. Chang, A.X.; Funkhouser, T.A.; Guibas, L.J.; Hanrahan, P.; Huang, Q.; Li, Z.; Savarese, S.; Savva, M.; Song, S.; Su, H.; et al. ShapeNet: An Information-Rich 3D Model Repository. *arXiv* **2015**, arXiv:1512.03012.