聚集

Dynamic Points Agglomeration for Hierarchical Point Sets Learning

Jinxian Liu[1,2] Bingbing Ni[1,2*] Caiyuan Li[1,2] Jiancheng Yang[1,2] Qi Tian[3]
[1]Shanghai Jiao Tong University, Shanghai 200240, China
[2]MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China
[3]Huawei Noah's Ark Lab
{liujinxian, nibingbing, 1161313414, jekyll4168}@sjtu.edu.cn
tian.qi1@huawei.com

Abstract

Many previous works on point sets learning achieve excellent performance with hierarchical architecture. Their strategies towards points agglomeration, however, only perform points sampling and grouping in original Euclidean space in a fixed way. These heuristic and task-irrelevant strategies severely limit their ability to adapt to more varied scenarios. To this end, we develop a novel hierarchical point sets learning architecture, with dynamic points agglomeration. By exploiting the relation of points in semantic space, a module based on graph convolution network is designed to learn a soft points cluster agglomeration. We construct a hierarchical architecture that gradually agglomerates points by stacking this learnable and lightweight module. In contrast to fixed points agglomeration strategy, our method can handle more diverse situations robustly and efficiently. Moreover, we propose a parameter sharing scheme for reducing memory usage and computational burden induced by the agglomeration module. Extensive experimental results on several point cloud analytic tasks, including classification and segmentation, well demonstrate the superior performance of our dynamic hierarchical learning framework over current state-of-the-art methods.

1. Introduction

3D vision analysis has attracted a lot of attention because 3D data contains richer spatial information compared with 2D image/video. 3D point cloud is the simplest 3D data format and consists of irregular and unordered points. With the renaissance of convolution neural network (CNN) in 2D image/video processing, there are many works that have tried to make an adaptation of deep learning framework for point cloud analyses [41, 24, 7, 16].

Recently, some works [26, 28, 22, 10, 32] focus on directly consuming point cloud rather than transforming it into multi-view images or voxel. A well-known pioneering work in this domain is PointNet [26], where point-wise multi-layer perceptrons (MLPs) are used to extract point-wise feature and a max-pooling layer is used to aggregate features of all points. However, every point is treated individually and local dependency structure is not considered. To overcome this problem, PointNet++ [28] proposes a hierarchical architecture for multi-level feature aggregation.

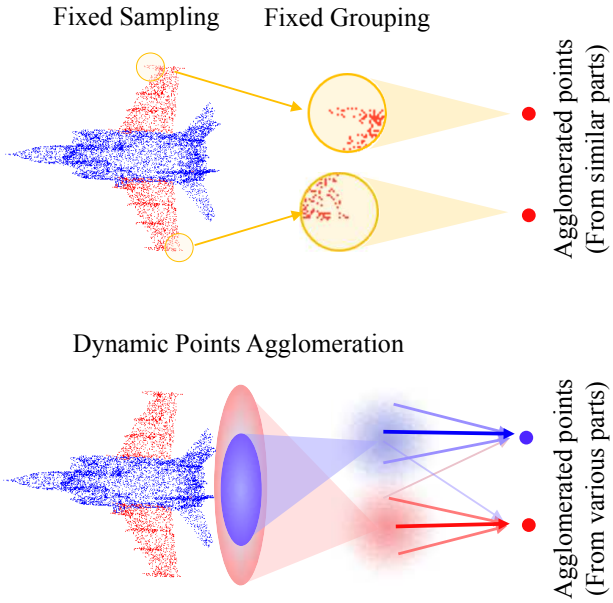

Figure 1. The motivation of our proposed framework. Given a plane, we would like to agglomerate these points into two representative points. If the initial sampling point is from the wing, existing methods are likely to sample two points from two wings of the plane with FPS algorithm and group surrounding points with KNN algorithm. Thereby, information of the fuselage is lost. Our proposed dynamic agglomeration strategy samples, groups and pools points in semantic space, hence points from various local semantic filed will be agglomerated.

在提取语义信息的过程中，如何根据语义信息采样?

*Corresponding author: Bingbing Ni

As demonstrated in [28], hierarchical learning is an effective design principle, and beneficial for the network to gradually enlarge the receptive field of every point and capture features of multiple spatial levels.

PointNet++ achieves hierarchical representation learning by proposed **sampling** layer, **grouping** layer and **pooling** layer. In sampling layer, representative points are generated with iterative Farthest Point Sampling (FPS), where the farthest point (in Euclidean space) from the sampled point set with regard to the rest points is chosen in every iteration. Then for each sampled point, the grouping layer finds K nearest neighbor points within a radius to it (*i.e*., grouping). Finally, a pooling layer (*i.e*., PointNet) aggregates features of K points that belong to the same group as local representations. Bottom-up hierarchical learning is performed by repeating the above steps. In short, we can regard PointNet++ as a hierarchical structure where individual points are agglomerated layer by layer. The limitations of PointNet++ are obvious: 1) the sampling and grouping strategies are heuristic, task-irrelevant and extremely time-consuming, 2) the pooling scheme still does not consider the relation between points. It should be noted that the sampling and grouping operation designed in PointNet++ are based on a hypothesis that points far apart in input 3D space are the most representative (i.e, should be sampled) and points near in input 3D space are semantic similar (*i.e*., should be grouped and pooled). This heuristic strategy is not feasible for learning 3D representations in many scenarios. For example, as shown in Figure 1, given a plane, we want to sample two points and generate groups. The strategy adopted in PointNet++ is likely to sample two points from two wings of the plane respectively, and the generated groups may not contain points from the fuselage. The most important part of the plane is missing. Hence, sampling and grouping points in input Euclidean space with a fixed way cannot be adaptive to various scenarios. However, to the best of our knowledge, most existing hierarchical frameworks take a similar hierarchical learning strategy. PointSIFT [13] designs a SIFT-like module for encoding information of different orientations, yet its hierarchical learning strategy is the same as PointNet++. [22] and [37] develop novel methods to pool features of points, however, they take the same sampling and grouping strategies as PointNet++. It is thus demanding to develop a way that points agglomeration could be performed in semantic space with an adaptive and learnable scheme.

To address this problem, we design a 3D point cloud learning framework where the hierarchical structure of representation processing can be learned instead of being fixed. Namely, for every level of the network hierarchy, points will be sampled, grouped and pooled according to the underlying distribution of the training points and the points features are aggregated with adaptive weights. The motivation of our method is shown in Figure 1. Specifically, we endow the backbone network several dynamic points agglomeration modules. This module is based on graph neural network (GCN) [17], and takes points similarity graph as input and performs message passing among points to learn an agglomeration matrix. The process of points agglomeration (sampling, grouping and pooling) is achieved by only one step, *i.e*., multiplication of the agglomeration matrix and points feature matrix. Moreover, it is a lightweight and flexible module that can be neatly inserted into most existing architectures. In the meantime, we propose a points diversity object function to encourage the sampled points to be more diverse and representative. To further reduce computations and memory usage, we propose a parameter sharing scheme: a whole 3D model is divided into several parts, which are sent into the dynamic points agglomeration module respectively with shared parameters. By these designs, our method achieves state-of-the-art performance in standard classification and segmentation benchmarks (*i.e*., ModelNet10/40 [41], ShapeNet [48], S3DIS [2], *etc*.) with high inference speed.

For the classification task, we plug three points agglomeration module into a backbone, and multi-level features are combined to improve representation capacity. For the segmentation task, we construct an U-net architecture which contains an encoder and a decoder. The encoder is similar to the architecture designed for classification. The decoder utilizes the transposed matrix of the sampling matrix learned in encoder as the corresponding up-sampling matrix.

2. Related Work

Handcrafted Features of 3D data Many handcrafted 3D descriptors are elaborately designed to capture fixed pattern (*i.e*., geometry and shape information) that exists in 3D data. These descriptors can be divided into two categories that include extrinsic and intrinsic descriptors. Extrinsic descriptors such as spin images [14], 3D shape context [8], MeshHOG descriptor [49] are invariant under rigid euclidean transformations, but not under deformations. To address this problem, intrinsic descriptors based on geodesic distances and spectral geometry are proposed, which include kernel signature [34], wave kernel signatures [4], intrinsic shape context [19], *etc*.

Deep Learning on 3D Data Convolution neural network (CNN) has revolutionized the field of 2D image/video processing. Motivated by this, many works have tried to make an adaptation of deep learning to point cloud analysis.

On the one hand, some works explore to directly apply deep learning frameworks designed for 2D image/video analysis on point cloud by transforming its data format. There are two kinds of transformation methods that include multi-view-based [33, 3, 9, 7] and voxel-based [24, 27, 18, 51, 50]. Multi-view-based methods project a 3D model

into a collection of 2D images and standard CNN are used for further processing. One great limitation of multi-view-based methods is that spatial and structure information will be lost through the transformation process. Voxel-based methods convert an irregular 3D point cloud to a regular 3D grid that is named as voxel and then 3D convolution can be applied to process it. However, voxel-based methods waste a large amount of memory and computation usage.

On the other hand, many works [21, 10, 32, 45] focus on directly consuming point cloud by designing applicable architectures. PointNet [26] is the pioneering work in this domain, where point-wise multilayer perceptrons (MLPs) and a max-pooling layer are used to learn global representation. PointNet++ [28] develops a hierarchical learning architecture, where multi-scale local features are captured. However, the sampling and grouping layer designed in PointNet++ take a fixed and time-consuming way. PointCNN [22] proposes a generalization of typical CNNs to learn features from the point cloud. A $\mathcal{X}$ transformation is first operated on point set to transform it into a potentially canonical order, and then perform typical convolution on it. However, the task-irrelevant sampling method (*i.e.*, FPS) is utilized to generate sub-sets. In our work, a learnable sampling and grouping operations are designed to overcome these limitations. Moreover, based on these elaborately designed architectures, there are many works [12, 46, 25, 1, 6] proposed to apply to various applications (*e.g.*, scene segmentation, matching, generation,*etc*.).

Graph Convolution Neural Networks Different from CNN that is performed on regular grid data (*e.g.*, image, video.), graph convolution networks (GCNs) [5] are a class of frameworks designed for non-Euclidean structured data. Many works [15, 23, 36, 39, 44] apply GCNs to image/video processing, biomedical imaging processing, 3D mesh processing, *etc*. Although no explicit graph structure that exists in the point cloud, similarity graph of points can be constructed to facilitate representation learning. There are few works [20, 30, 38, 37] that explore to utilize implicit graph that exists in the point cloud. In [40], geometric deep learning is introduced into point cloud processing, where a graph is constructed to performs message passing among points. However, the scale of point set remains unchanged, which is against the hierarchy principle. In our work, a dynamic points agglomeration module based on GCNs is designed for hierarchical learning.

3. Method

3.1. Motivation and Overview

In this work, we propose a novel hierarchical learning architecture for 3D point cloud analyses that include classification and segmentation. A dynamic points agglomeration module is designed to perform flexible points sampling, grouping and pooling. We name this module as DPAM for clarity. In contrast to previous fixed agglomeration strategy (sampling with FPS and grouping with KNN) that most existing architecture takes, DPAM is a lightweight module that agglomerates points in semantic space dynamically. Based on graph convolution network, DPAM takes points similarity matrix as well as feature matrix as input and performs message passing among these points. Through this way, an agglomeration matrix for points sampling, grouping and pooling is learned. Moreover, a parameter sharing scheme is proposed in our work to reduce memory and computation usage.

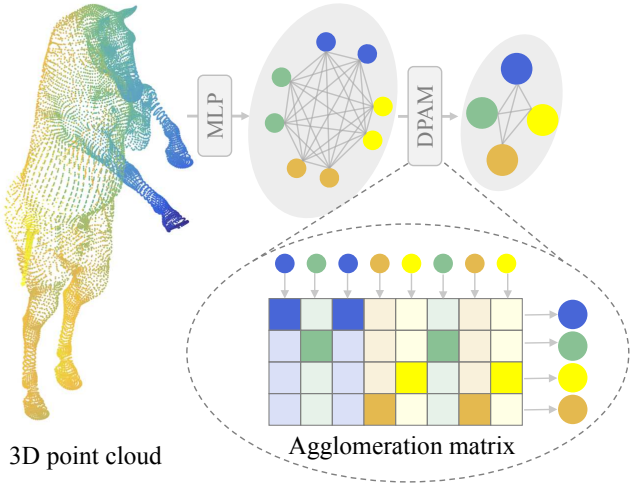

Figure 2. The process of points agglomeration. Taking point similarity graph and points feature matrix as input, DPAM learns an agglomeration matrix. The agglomeration process (*i.e.*, sampling, grouping and pooling) is integrated as a simple matrix multiply operation (equation 3). Best viewed in color.

3.2. PointNet and GCNs Revisit

The backbone of our architecture is similar to PointNet, and the DPAM inserted in our backbone is based on GCN-like network prototype. Therefore, we revisit these two methods in this section.

An unordered point cloud can be represented as $\{P_i|i = 1, 2, ..., n\}$ with $P_i \in \mathbb{R}^d$, where d is the channel of input points and can be represented by coordinate, color, normal, *etc*. PointNet learns a function that maps a set of points to a feature vector, *i.e.*, a few point-wise multi-layer perceptrons (MLPs) are applied to every point individually before a max-pooling layer that aggregates features of all points to a global vector. It is proved that PointNet can arbitrarily approximate any continuous set function and is order-invariance. In this work, a hierarchical learning architecture is constructed by inserting DPAMs into the backbone based on PointNet. In addition to the proposed parameter sharing scheme, our method captures multi-level representations in an efficient way.

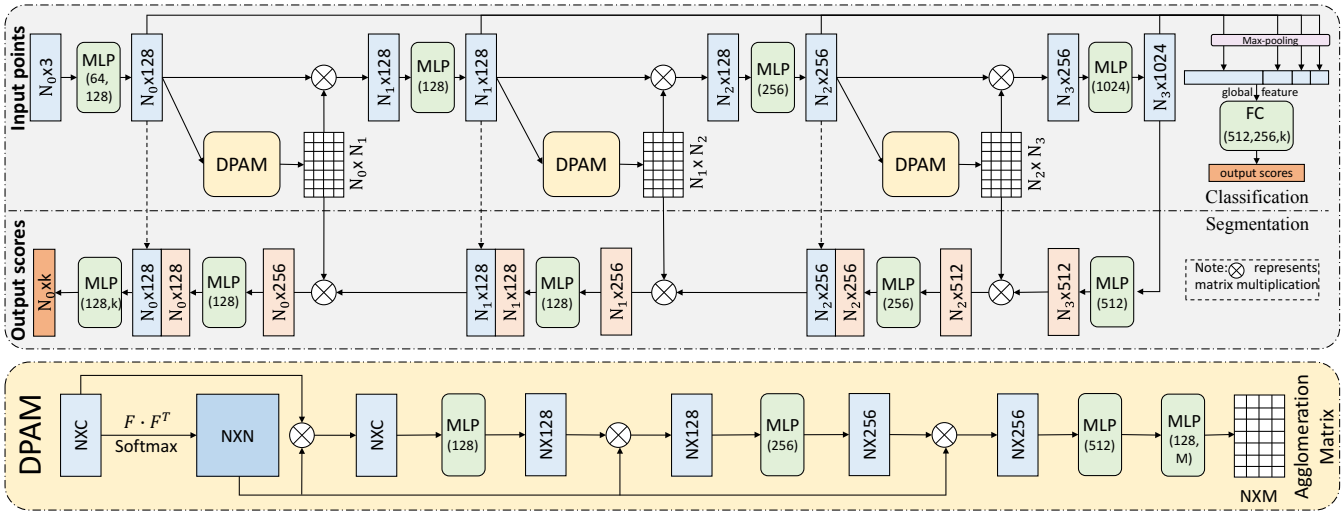

Figure 3. Architecture of our proposed method. For classification, three DPAMs are inserted in the backbone for points agglomeration, and features of multi-levels are concatenated to perform classification. For segmentation, a u-net structure is constructed, where the encoder is the same as the architecture of classification and decoder is symmetrical with the encoder. Note that the DPAMs for encoder and decoder are shared and dot line denotes skip connection. The details of DPAM are also shown in the figure.

The agglomeration module in our proposed architecture is based on graph convolutional networks (GCNs). Different from standard convolutions that are operated on the regular grid (*e.g.*, image, video.), graph convolution is a kind of convolution that is operated on the graph. It computes the response of each node in the graph by aggregating information of its neighbors defined by graph relations (*e.g.*, adjacent matrix, similarity matrix, *etc.*). We adopt the GCN proposed in [17], which gets an adjacent matrix $A \in \mathbb{R}^{N\times N}$, as well as a nodes feature matrix $X_0 \in \mathbb{R}^{N\times c_0}$ as input, and then perform message passing inside the graph. Note that the adjacent matrix $A \in \mathbb{R}^{N\times N}$ represents the relation of N nodes that exists in the graph. Formally, one layer of graph convolution can be represented as:

$$X_h = AX_{h-1}W_h, \tag{1}$$

where h denotes the hth layer of the GCN, $W_h \in \mathbb{R}^{c_{(h-1)}\times c_{(h)}}$ is the weight matrix of layer h and c represents feature channels. The GCN consisted of H graph convolution layers is represented as:

$$X_H = GCN(A, X_0) \in \mathbb{R}^{N\times c_H} \tag{2}$$

3.3. Dynamic Points Agglomeration Module

In this section, we introduce the dynamic points agglomeration module (DPAM) in detail, which is designed for dynamically agglomerating points. By inserting this module into the backbone, we construct a novel hierarchical architecture for point cloud classification and segmentation.

DPAM can be inserted into most existing architectures. For simplicity, we use a simple backbone that is similar to PointNet in this work. Given a set of points $P \in \mathbb{R}^{N\times d}$, we get a feature matrix $F^{(l)} \in \mathbb{R}^{N\times c^{(l)}}$ at layer l of the backbone. DPAM learns an agglomeration matrix $S^{(l)} \in \mathbb{R}^{N\times M}(M < N)$, with which the process of points sampling, grouping and feature aggregating are integrated into one simple step. It is represented as:

$$F_s^{(l)} = S^{(l)T}F^{(l)}, \tag{3}$$

where $F_s^{(l)} \in \mathbb{R}^{M\times c}$ is the output feature matrix of points agglomeration. This points agglomeration process is shown in Figure 2. Note that the agglomerated M points are not corresponding to the points in original input, and they are dynamically sampled based on the underlying distribution of points. In the meantime, our method agglomerates points with soft weights and incorporates relation of points through this agglomeration scheme (*i.e.*, representation of every sampled point is related to all points before agglomeration.).

The key point is to generate the agglomeration matrix. We design a module (*i.e.*, DPAM) to learn this matrix with a GCN. There is no explicit graph that exists in point clouds, hence a similarity graph is constructed based on the embedding of every point:

$$A^{(l)} = softmax(F^{(l)}F^{(l)T}) \in \mathbb{R}^{N\times N}, \tag{4}$$

which can be also regarded as a soft adjacent matrix. The element (i, j) of $A^{(l)}$ represents distance between ith point and jth point in semantic space. Our DPAM is built by stacking multiple layers of graph convolutions. Taking soft adjacent matrix $A^{(l)}$ and feature matrix $F^{(l)}$ as input,

DPAM learns the agglomeration matrix $S^{(l)}$:

$$S^{(l)} = softmax(GCN(A^{(l)}, F^{(l)})) \in \mathbb{R}^{N \times M}, \quad (5)$$

where softmax is performed in a row-wise fashion. By utilizing the similarity relation of points and performing message passing among points, the DPAM learns explicit agglomeration weights of points. Namely, each row of matrix $S^{(l)}$ represents the weight that each point before agglomerating assigns to the M sampled points, and each column represents the weight that each sampled point agglomerated by the points before agglomerating. With this learned agglomeration matrix, points agglomeration can be performed via a simple step mentioned above. The architecture of DPAM is shown in Figure 3.

3.4. Parameter Sharing Scheme

Parameter sharing is an ingenious design in convolution neural network (CNN), especially for image/video processing. In CNN, a small convolution kernel is shared in the whole feature map. Each kernel is served as a filter to detect patterns of a particular type. Motivated by this, we design a sharing scheme to reduce parameters usage and computational burden, as shown in Figure 4.

A 3D point cloud sample usually contains thousands to tens of thousands of points. Hence, the similarity matrix, the feature matrix and the parameter matrix in DPAM would be extremely huge, which makes the agglomeration process take up a lot of memory and computation. Moreover, the huge agglomeration matrix is hard to learn and optimize. To overcome this challenge, we propose to divide a point cloud model into a few parts along an axis (*i.e.*, x-axis, y-axis, z-axis.) and then send them into the shared DPAM for agglomerating representative points of every part respectively. Specifically, we sort all input points from a 3D point cloud along an axis and uniformly divide them into several parts before being sent into shared DPAM.

As mentioned above, the relation of points is incorporated with the proposed DPAM. However, with our designed parameter sharing scheme, every point in every divided part is just related to the points in the same part and do not communicate with the points in other parts. In other words, the size of the receptive field of every point will remain small and unchanged. In general, we hope the receptive field grows from small to large gradually, which is the design philosophy that everyone will follow in CNN designing. In our work, several DPAMs will be inserted into the backbone. Hence, we propose to increase the size of receptive filed through merging parts gradually (*i.e.*, reducing the number of divided parts gradually). Although we merge parts gradually, this design keeps the similarity matrix and weight matrix small because the number of points in every part is reduced after each DPAM.

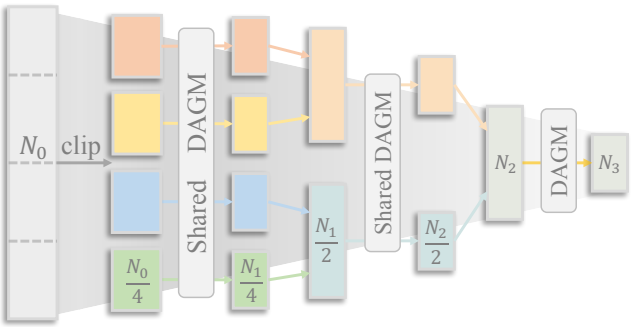

Figure 4. Illustration of our parameter sharing scheme. An ordered point cloud is uniformly divided into several parts and sent into shared DPAM to perform points agglomeration respectively. After each DPAM, the parts will be merged gradually for increasing receptive filed of every point. $N_0, ..., N_3$ denote the number of points, and $N_0 > N_1 > N_2 > N_3$.

3.5. Diversity of Sampled Points

Although we reduce the difficulty of optimizing DPAM through parameter sharing scheme mentioned above, the module still easily converges to local minima early in training. Therefore, the points sampled with learned agglomeration matrix cannot effectively catch representative points among semantic space, and the agglomerated points are likely to be very close. To make the sampled points diverse and representative, we propose to add a constraint to the agglomeration matrix optimization problem. Note that each column of agglomeration matrix $S^{(l)}$ represents the combination weight of each sampled point. Therefore, making sampled points diverse is equivalent to making the combination weight of every sampled point various as much as possible, *i.e.*, making the column vectors of $S^{(l)}$ mutually orthogonal. Hence, we minimize:

$$L_d^{(l)} = ||I - S^{(l)T} S^{(l)}||_F, \quad (6)$$

where every column of $S^{(l)}$ is normalized to a unit vector. This diversity loss is added to the softmax classification loss with weight 0.001 for optimizing.

3.6. Multi-Level Feature Aggregation

With the receptive filed growing gradually, the level of feature learned by the network is also growing. We propose to combine multi-level features for achieving better representation ability. We use max-pooling to get a global feature before every DPAM and then concatenate these multi-level global features for final classification. This design is also similar to the operator proposed in DenseNet [11], which has been proved to facilitate optimizing.

3.7. Network Architecture

Architecture for classification. In our work, we plug three DPAMs into the backbone and the number of divided

parts is reduced with 8-4-1. The number of feature channels and overall classification architecture are shown in Figure 3.

Architecture for segmentation. We construct a U-net structure for segmentation, which contains an encoder and a decoder. The encoder is the same as the architecture for classification. We have tried to plug three new DPAMs in the decoder for up-sampling, however, we find it is not only time-consuming but also hard to optimize. Hence we propose a simple but effective method to perform points up-sampling in the decoder. We directly utilize the transpose matrix of the agglomeration matrix learned in the encoder as the up-sampling matrix. Moreover, skip connection is utilized to construct U-net structure. The whole architecture for segmentation is shown in Figure 3.

4. Experiments

In this section, we evaluate our method on classification and segmentation tasks. Experiments on two object classification benchmarks including ModelNet10, ModelNet40, and 3D object part segmentation dataset ShapeNet, as well as real scene segmentation dataset S3DIS, are carried out. Our method achieves state-of-the-art or comparable performance in these benchmarks. In the meantime, time and space complexity analysis, as well as some important analysis experiments are performed.

4.1. Datasets and Data Pre-Processing

ModelNet10 and ModelNet40 are two standard 3D object classification benchmarks collected from ModelNet [41], which provides CAD models for every object. The point cloud is sampled from CAD models, and we use the ModelNet10 and ModelNet40 dataset generated from [28] in our experiments for fair comparison. The ModelNet10 contains 10 categories and is split into 3,991 training samples and 909 testing samples. The ModelNet40 contains 40 categories, where 9,843 objects belong to the training set and 2,468 samples for testing. Every point in each object is represented with 3D coordinates.

ShapeNetPart is a 3D object part segmentation benchmarks, and it is a subset of ShapeNet [48]. This dataset contains 16681 samples from 16 categories and there are 50 parts in total (each category contains 2-6 parts). Note that the object categories will be given when training and testing for every object. The evaluation metric is mean IoU (mIoU), where IoU is computed for every object and then averaged within the category that the object belongs to.

Large-Scale 3D Indoor Spaces (S3DIS) is a real scene point cloud semantic segmentation benchmark. This dataset contains 3D RDB point clouds of 271 rooms from 6 indoor areas. Each point is annotated with one of 13 semantic categories (*e.g.*, ceiling, floor, chair). We use the mean per-class IoU (mIoU, %) and overall accuracy (OA) as the evaluation metrics.

Data Pre-Processing As mentioned in section 3.4, we divide a point cloud model into several parts for sharing parameters, therefore, the input points are sorted along an axis (x-axis is used in all our experiments) for easy dividing before being sent into DPAM. For object classification and segmentation, we normalize input point clouds to be zero-mean inside a unit sphere. Random jitter, rotation, shift and scale are performed on training objects to improve performance. A T-Net used in [26] is also applied to the input object in our experiments. For scene segmentation, rotation around the z-axis is utilized to augment data.

4.2. Point Cloud Classification

The results of experiments on 3D object dataset ModelNet10 and ModelNet40 are shown in Table 1, where accuracy over class and instance are both displayed. It can be seen that our method achieves state-of-the-art performance except accuracy over instance on ModelNet40 (only 0.3% lower than PointCNN) with all published methods based on point cloud. Note that we only use 1024 points with 3D coordinates for every object as input. We down-sample the points 3 times, and the number of points drops double every time (*i.e.*, 1024-512-256-128). In experiments performed on ModelNet10/40, we use Adam optimizer with initial learning rate 0.001, momentum 0.9 and batch size 16. The learning rate is divided by 2 every 20 epochs. Drop out rate is set to 0.7 after last two layers in the classifier.

Methods	ModelNet10		ModelNet40	
	Class	Instance	Class	Instance
3DShapeNets [41]	83.5	-	77.3	84.7
VoxNet [24]	92.0	-	83.0	85.9
OctNet [29]	90.1	90.9	83.8	86.5
ECC [31]	90.0	90.8	83.2	87.4
Subvolume [27]	-	-	86.0	89.2
Pointnet [26]	-	-	86.2	89.2
Pointnet++ [12]	-	-	-	90.7
SO-Net [21]	93.9	94.1	87.3	90.9
KCNet [30]	-	94.4	-	91.0
SpecGCN [37]	-	-	-	91.5
Kd-Net [18]	93.5	94.0	88.5	91.8
PointCNN [22]	-	-	88.1	**92.2**
Ours	**94.3**	**94.6**	**89.9**	91.9

Table 1. **Classification results on ModelNet10/40.** Our method achieves state-of-the-art performance on ModelNet10 and ModelNet40 compared with methods based on point cloud.

4.3. Part Segmentation on ShapeNetPart

Segmentation on 3D point cloud is formulated as a per-point classification task. The IoU of every category and the overall IoU are shown in Table 2, from which we see that our method achieves state-of-the-art performance. Moreover, our method has low computation cost and high infer-

	Mean	Aero	Bag	Cap	Car	Chair	Ear phone	Guitar	Knife	Lamp	Laptop	Motor	Mug	Pistol	Rocket	Skate board	Table
KD-Net [18]	82.3	80.1	74.6	74.3	70.3	88.6	73.5	90.2	87.2	81.0	94.9	57.4	86.7	78.1	51.8	69.9	80.3
Pointnet [26]	83.7	83.4	78.7	82.5	74.9	89.6	73.0	91.5	85.9	80.8	95.3	65.2	93.0	81.2	57.9	72.8	80.6
A-SCN [42]	84.6	83.8	80.8	83.5	79.3	90.5	69.8	91.7	86.5	82.9	96.0	69.2	93.8	82.5	62.9	74.4	80.8
KCNet [30]	84.7	82.8	81.5	86.4	77.6	90.3	76.8	91.0	87.2	84.5	95.5	69.2	94.4	81.6	60.1	75.2	81.3
RSNet [12]	84.9	82.7	86.4	84.1	78.2	90.4	69.3	91.4	87.0	83.5	95.4	66.0	92.6	81.8	56.1	75.8	82.2
Pointnet++ [28]	85.1	82.4	79.0	87.7	77.3	90.8	71.8	91.0	85.9	83.7	95.3	71.6	94.1	81.3	58.7	76.4	82.6
DGCNN [40]	85.1	84.2	83.7	84.4	77.1	**90.9**	78.5	91.5	87.3	82.9	96.0	67.8	93.3	82.6	59.7	75.5	82.0
SpiderCNN [43]	85.3	83.5	81.0	87.2	77.5	90.7	76.8	91.1	87.3	83.3	95.8	70.2	93.5	82.7	59.7	75.8	82.8
SGPN [38]	85.8	80.4	78.6	78.8	71.5	88.6	78.0	90.9	83.0	78.8	95.8	**77.8**	93.8	**87.4**	60.1	**92.3**	**89.4**
PointCNN [22]	**86.1**	84.1	**86.5**	86.0	**80.8**	90.6	**79.7**	**92.3**	**88.4**	**85.3**	96.1	77.2	**95.3**	84.2	**64.2**	80.0	83.0
Ours	**86.1**	**84.3**	81.6	**89.1**	79.5	**90.9**	77.5	91.8	87.0	84.5	**96.2**	68.7	94.5	81.4	**64.2**	76.2	84.3

Table 2. **Segmentation results (part-wise IoU) on ShapeNetPart Dataset.** Our method achieves state-of-the-art performance.

ence speed. Some segmentation results are visualized in Figure 5. Our method segment the fine-grained details well.

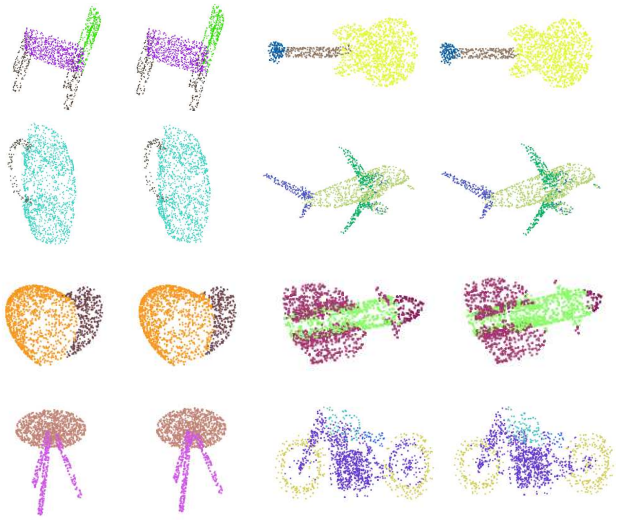

Figure 5. **The segmentation results on ShapeNetPart dataset.** Column 1 and 3: Segmented results. Column 2 and 4: Ground truth. Best viewed in color.

The architecture for part segmentation is shown in Figure 3, which is a U-net structure. As mentioned in Section 3.7 that the up-sampling matrix in the decoder is the transpose of the down-sampling matrix learned in the encoder. 2048 points for every object are used as input. 3 times down-sampling and up-sampling are performed respectively, and the number of points decrease in the encoder and increase in decoder double every time (*i.e.*, 2048-1024-512-256-512-1024-2048.). Adam optimizer is used to optimize per-point cross-entropy loss in this experiment with initial learning rate 0.006, momentum 0.9 and batch size 32. The learning rate is divided by 2 every 20 epochs. Drop out rate is set to 0.5 after the last layer in the classifier.

4.4. S3DIS Indoor Scene Segmentation

We follow the same setting as PointNet [26], where each room is split into blocks of area $1m \times 1m$. Each input point is represented by a 9-dim vector of XYZ, RGB and normalized location as to the room. 4096 points are randomly sampled for each block when training and all points are used for testing. Following the common evaluation setting, a 6-fold cross validation over the 6 ares, with 5 area for training and the left 1 area for validation each time. The test results on Area 5 are reported individually due to the fact that there are overlaps between areas except Area 5. All results on this dataset are shown in Table 3. For 6-fold cross validation, our method achieves superior performance over all methods except PointCNN [22]. On Area 5, our method outperforms all methods in terms of mIoU.

Methods	6-fold CV		Area 5	
	mIoU	OA	mIoU	OA
PointNet [26]	47.6	78.5	41.1	-
SegCloud [35]	-	-	48.9	-
RSNet [12]	56.5	-	-	-
3P-RNN [47]	56.3	86.9	53.4	85.7
SPGraph [20]	62.1	85.5	58.0	**86.4**
PointCNN [22]	**65.4**	**88.1**	57.3	85.9
Ours	64.5	87.6	**60.0**	86.1

Table 3. **Segmentation results on S3DIS.** Mean per-class IoU (mIoU, %) and overall accuracy (OA, %) are shown in table. Our method achieves comparable performance.

The architecture for scene segmentation is shown in Figure 3. The points are down-sampled and up-sampled with 4096-1024-256-32-256-1024-4096. We use Adam optimizer to optimize per-point cross-entropy loss with initial learning rate 0.003, momentum 0.9 and batch size 12. The learning rate is divided by 2 every 20 epochs. Drop out rate is set to 0.4 after the last layer in the classifier.

4.5. Time and Space Complexity Analysis

In this section, we show the time and space complexity of our method in Table 4 to prove that our proposed method does achieve state-of-the-art performance with obviously higher inference speed. This experiment is performed on ModelNet40 with a 1080X GPU. The batch size is also set to 8 for fair comparison. Note that we do not make a comparison with PointCNN [22] in this section due to device limitation. Our method achieves excellent performance with inference speed that is obviously faster than

other methods except PointNet. To further reduce the model size and inference time, we drop the T-net used in our model (*i.e.*, denoted with vanilla in the Table 4). Smaller model size and faster inference time are achieved with only 0.5% drop in accuracy. Although the model size of our method is larger than PointNet++ [28] and SO-Net [21], we can make a trade-off between accuracy and model size by adjusting the number of sampled points and parameters of DPAMs.

Method	Size/Mb	Infer/ms	Acc(%)
PointNet (vanilla) [26]	9.4	11.6	87.2
PointNet [26]	40	25.3	89.2
PointNet++ [28]	12	163.2	90.7
SO-Net [21]	11.5	59.6	90.9
Ours(vanilla)	21.3	18.4	91.4
Ours	29.5	36.6	**91.9**

Table 4. **Time and Space Complexity Analysis on ModelNet40.** Our method achieves excellent performance with high inference speed. Vanilla denotes that the model is trained without T-Net. The model size (Mb), inference time (ms) and accuracy (%) over instance is reported in table.

4.6. Ablation Study

In this section, we study the robustness of our method to random noises and analyze some important hyper-parameters including the number of divided parts in every DPAM and the axis that the parts divided along. All experiments in this section are performed on ModelNet40.

Robustness to random noises. We randomly replace input points with Gaussian noise during testing, where mean value μ is set to 0 and the standard deviation σ is set to 0.1 and 0.05 respectively. The standard deviation represents noise intensity. Compared with Pointnet++, our method is more robust to noises as shown in Figure 6. The horizontal axis shows the number of replaced noise points and the vertical axis shows the accuracy on Modelnet40.

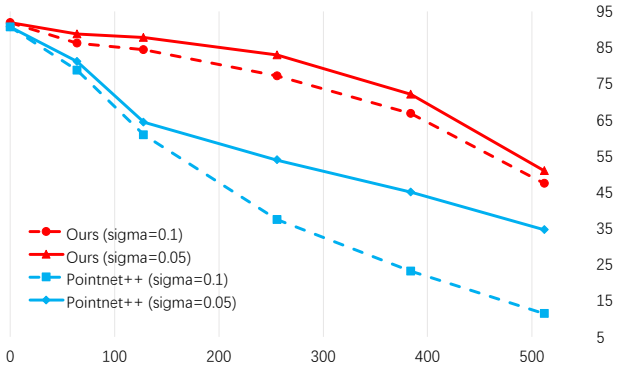

Figure 6. Analysis of noises. Our method shows better robustness compared with Pointnet++.

Effectiveness of our proposed parameter sharing scheme. We have mentioned in Section 3.4 that we propose to divide input point cloud into several parts for sharing parameters and merge divided parts gradually for increasing receptive field of every point. In this part, we analyze the effectiveness of this scheme, as well as the robustness to various combinations of the number of divided parts. We conduct classification experiments on ModelNet40 with 5 combinations (*i.e.*, 1-1-1, 8-8-8, 4-2-1,16-4-1, 8-4-1). Note that the combination 1-1-1 denotes that we train the model **without** our proposed parameter sharing scheme. The results are shown in Table 5. Comparing the results of combination 1-1-1 with other combinations, we can conclude that dividing an input object into several parts do bring significant performance improvement. The effectiveness of merging parts gradually is also proved by comparing combinations 8-8-8 with other decreasing combinations.

Combinations	1-1-1	8-8-8	4-2-1	16-4-1	8-4-1
Accuracy (%)	90.9	91.2	91.7	91.7	91.9

Table 5. Effectiveness of our proposed parameter sharing scheme. The accuracy (%) over instance is reported in table.

Robustness to different axes which the point cloud is divided along. Table 6 shows that our proposed parameter sharing scheme is not sensitive to the axis, along which input point cloud is divided. Our method achieves similar performance when we divide input along the x-axis, y-axis, z-axis respectively. However, the performance drops a little when we divide the input randomly.

	Random	X-axis	Y-axis	Z-axis
Acc (%)	90.9	91.9	91.6	91.8

Table 6. Robustness to the different axis which point cloud is divided along. The accuracy (%) over instance is reported in table.

5. Conclusion

In this work, a dynamic points agglomeration module is proposed to construct an efficient hierarchical point sets learning architecture. This GCN-based module is applied to learning a points agglomeration matrix with points relation and feature. In contrast to fixed agglomeration strategy that samples and groups points in a fixed way, our proposed dynamic agglomeration strategy can dynamically adapt to various situations. Moreover, a parameter sharing scheme is proposed to reduce memory and computation usage. Our dynamic agglomeration architecture achieves better performance on several benchmarks with high inference speed compared with fixed points agglomeration strategy.

Acknowledgements This work was supported by National Science Foundation of China U1611461 and China's Thousand Youth Talents Plan, STCSM 17511105401, 18DZ2270700 and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China. This work was sponsored by CCF-Tencent Open Fund. This work was also jointly supported by SJTU-Minivision joint research grant.

References

[1] Mikaela Angelina Uy and Gim Hee Lee. Pointnetvlad: Deep point cloud based retrieval for large-scale place recognition. In *CVPR*, June 2018.

[2] Iro Armeni, Ozan Sener, Amir R. Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *CVPR*, June 2016.

[3] Amir Arsalan Soltani, Haibin Huang, Jiajun Wu, Tejas D. Kulkarni, and Joshua B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In *CVPR*, July 2017.

[4] Mathieu Aubry, Ulrich Schlickewei, and Daniel Cremers. The wave kernel signature: A quantum mechanical approach to shape analysis. In *ICCV Workshops*, pages 1626–1633, 2011.

[5] Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. Spectral networks and locally connected networks on graphs. *CoRR*, abs/1312.6203, 2013.

[6] Haowen Deng, Tolga Birdal, and Slobodan Ilic. Ppfnet: Global context aware local features for robust 3d point matching. In *CVPR*, June 2018.

[7] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *CVPR*, June 2018.

[8] Andrea Frome, Daniel Huber, Ravi Kolluri, Thomas Bülow, and Jitendra Malik. Recognizing objects in range data using regional point descriptors. In *ECCV*, pages 224–237, 2004.

[9] Xinwei He, Yang Zhou, Zhichao Zhou, Song Bai, and Xiang Bai. Triplet-center loss for multi-view 3d object retrieval. In *CVPR*, June 2018.

[10] Binh-Son Hua, Minh-Khoi Tran, and Sai-Kit Yeung. Pointwise convolutional neural networks. In *CVPR*, June 2018.

[11] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *CVPR*, July 2017.

[12] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *CVPR*, June 2018.

[13] Mingyang Jiang, Yiran Wu, and Cewu Lu. Pointsift: A siftlike network module for 3d point cloud semantic segmentation. *CoRR*, abs/1807.00652, 2018.

[14] Andrew Edie Johnson and Martial Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Trans. Pattern Anal. Mach. Intell.*, 21(5):433–449, 1999.

[15] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *CVPR*, June 2018.

[16] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *CVPR*, June 2018.

[17] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.

[18] Roman Klokov and Victor S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *ICCV*, pages 863–872, 2017.

[19] Iasonas Kokkinos, Michael M Bronstein, Roee Litman, and Alex M Bronstein. Intrinsic shape context descriptors for deformable shapes. In *CVPR*, pages 159–166, 2012.

[20] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, June 2018.

[21] Jiaxin Li, Ben M. Chen, and Gim Hee Lee. So-net: Self-organizing network for point cloud analysis. In *CVPR*, June 2018.

[22] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, pages 828–838, 2018.

[23] Or Litany, Alex Bronstein, Michael Bronstein, and Ameesh Makadia. Deformable shape completion with graph convolutional autoencoders. In *CVPR*, June 2018.

[24] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, pages 922–928, 2015.

[25] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum pointnets for 3d object detection from rgb-d data. In *CVPR*, June 2018.

[26] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, July 2017.

[27] Charles Ruizhongtai Qi, Hao Su, Matthias Nießner, Angela Dai, Mengyuan Yan, and Leonidas J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In *CVPR*, pages 5648–5656, 2016.

[28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NIPS*, pages 5105–5114, 2017.

[29] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, pages 6620–6629, 2017.

[30] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *CVPR*, June 2018.

[31] Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *CVPR*, July 2017.

[32] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. Splatnet: Sparse lattice networks for point cloud processing. In *CVPR*, June 2018.

[33] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, pages 945–953, 2015.

[34] Jian Sun, Maks Ovsjanikov, and Leonidas J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Comput. Graph. Forum*, 28(5):1383–1392, 2009.

[35] Lyne P. Tchapmi, Christopher Bongsoo Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *3DV*, pages 537–547, 2017.

[36] Nitika Verma, Edmond Boyer, and Jakob Verbeek. Feastnet: Feature-steered graph convolutions for 3d shape analysis. In *CVPR*, June 2018.

[37] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *ECCV*, September 2018.

[38] Weiyue Wang, Ronald Yu, Qiangui Huang, and Ulrich Neumann. Sgpn: Similarity group proposal network for 3d point cloud instance segmentation. In *CVPR*, June 2018.

[39] Xiaolong Wang and Abhinav Gupta. Videos as space-time region graphs. In *ECCV*, pages 413–431, 2018.

[40] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *CoRR*, abs/1801.07829, 2018.

[41] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, pages 1912–1920, 2015.

[42] Saining Xie, Sainan Liu, Zeyu Chen, and Zhuowen Tu. Attentional shapecontextnet for point cloud recognition. In *CVPR*, June 2018.

[43] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *ECCV*, September 2018.

[44] Yichao Yan, Qiang Zhang, Bingbing Ni, Wendong Zhang, Minghao Xu, and Xiaokang Yang. Learning context graph for person search. In *CVPR*, June 2019.

[45] Jiancheng Yang, Qiang Zhang, Bingbing Ni, Linguo Li, Jinxian Liu, Mengdie Zhou, and Qi Tian. Modeling point clouds with self-attention and gumbel subset sampling. In *CVPR*, pages 3323–3332, 2019.

[46] Yaoqing Yang, Chen Feng, Yiru Shen, and Dong Tian. Foldingnet: Point cloud auto-encoder via deep grid deformation. In *CVPR*, June 2018.

[47] Xiaoqing Ye, Jiamao Li, Hexiao Huang, Liang Du, and Xiaolin Zhang. 3d recurrent neural networks with context fusion for point cloud semantic segmentation. In *ECCV*, September 2018.

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

[49] Andrei Zaharescu, Edmond Boyer, Kiran Varanasi, and Radu Horaud. Surface feature detection and description with applications to mesh matching. In *CVPR*, pages 373–380, 2009.

[50] Wei Zhao, Jiancheng Yang, Bingbing Ni, Dexi Bi, Yingli Sun, Mengdi Xu, Xiaoxia Zhu, Cheng Li, Liang Jin, Pan Gao, Peijun Wang, Yanqing Hua, and Ming Li. Toward automatic prediction of egfr mutation status in pulmonary adenocarcinoma with 3d deep learning. *Cancer Medicine*, 2019.

[51] Wei Zhao, Jiancheng Yang, Yingli Sun, Cheng Li, Weilan Wu, Liang Jin, Zhiming Yang, Bingbing Ni, Pan Gao, Peijun Wang, Yanqing Hua, and Ming Li. 3d deep learning from ct scans predicts tumor invasiveness of subcentimeter pulmonary adenocarcinomas. *Cancer Research*, 78(24):6881–6889, 2018.