HGNet: Learning Hierarchical Geometry from Points, Edges, and Surfaces

Ting Yao, Yehao Li, Yingwei Pan, Tao Mei
HiDream.ai Inc.
{tingyao.ustc, yehaoli.sysu, panyw.ustc}@gmail.com, tmei@hidream.ai

Abstract

Parsing an unstructured point set into constituent local geometry structures (e.g., edges or surfaces) would be helpful for understanding and representing point clouds. This motivates us to design a deep architecture to model the hierarchical geometry from points, edges, surfaces (triangles), to super-surfaces (adjacent surfaces) for the thorough analysis of point clouds. In this paper, we present a novel Hierarchical Geometry Network (HGNet) that integrates such hierarchical geometry structures from super-surfaces, surfaces, edges, to points in a top-down manner for learning point cloud representations. Technically, we first construct the edges between every two neighbor points. A point-level representation is learnt with edge-to-point aggregation, i.e., aggregating all connected edges into the anchor point. Next, as every two neighbor edges compose a surface, we obtain the edge-level representation of each anchor edge via surface-to-edge aggregation over all neighbor surfaces. Furthermore, the surface-level representation is achieved through super-surface-to-surface aggregation by transforming all super-surfaces into the anchor surface. A Transformer structure is finally devised to unify all the point-level, edge-level, and surface-level features into the holistic point cloud representations. Extensive experiments on four point cloud analysis datasets demonstrate the superiority of HGNet for 3D object classification and part/semantic segmentation tasks. More remarkably, HGNet achieves the overall accuracy of 89.2% on ScanObjectNN, improving PointNeXt-S by 1.5%.

1. Introduction

With the growing popularity of 3D acquisition technologies (such as 3D scanners, RGBD cameras, and LiDARs), 3D point cloud analysis has been a fast-developing topic in the past decade. Practical point cloud analysis systems have great potential for numerous applications, e.g., autonomous driving, robotics, and virtual reality. In comparison to 2D RGB image data, 3D data of point clouds has an additional dimension of depth, leading to a three dimensional world as

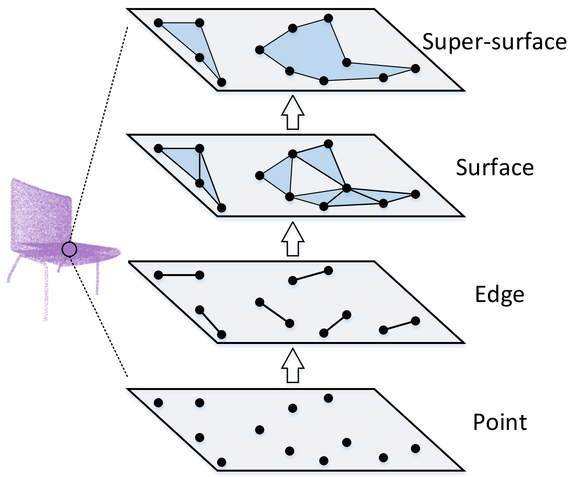

Figure 1. The progressive development of a four-level hierarchical geometry structure among points, edges, surfaces, and super-surfaces. Such hierarchical geometry structure further in turn strengthens surface-level, edge-level, and point-level features in a top-down manner.

in our daily life. The 3D point clouds are armed with several key merits, e.g., preserving rich geometry information and being invariant to lighting conditions. Nevertheless, considering that point clouds are naturally unstructured without any discretization, it is not trivial to directly applying the typical CNN operations widely adopted in 2D vision over the primary point set for analyzing 3D data.

To alleviate this limitation, a series of efforts have attempted to interpret the geometry information rooted in unstructured point clouds, such as the regular point-based [30, 38], edge/relation-based [32, 46], or surface-based [39] representations. For instance, PointNet [30] builds up the foundation of regular point-based feature extraction via stacked Multi-Layer Perceptrons (MLPs) plus symmetric aggregation. The subsequent works (e.g., PointNet++ [32] and DGCNN [46]) further incorporate edge/relation-based features by using set abstraction or graph models for aggregating/propagating points and edges, pursuing the mining of finer geometric structures. More recently, RepSurf [39] learns surface-based features over triangle meshes to enable an amplified expression of local geometry. Despite showing encouraging performances, most existing point cloud analysis techniques lack a thorough understanding of the compositional local geometry structures in the point clouds.

In this work, we propose to mitigate this issue from the standpoint of parsing an unstructured point set into a hierarchical structure of constituent local geometry structures to comprehensively characterize the point clouds. Our launching point is to construct a bottom-up local hierarchical topology from the leaf level of irregular and unordered points, to the intermediary levels of edges and surfaces, and the root level of super-surfaces (i.e., adjacent surfaces). Figure 1 conceptually depicts the progressive development of a hierarchical geometry structure for a local set of points. Such hierarchical structure in turn triggers the reinforcement of 3D model capacity for capturing comprehensive geometry information. The learning of multi-granular geometry features, i.e., surface-level, edge-level, and point-level features, is enhanced by aggregating geometry information from super-surfaces, surfaces, or edges derived from the hierarchical topology in a top-down fashion.

By consolidating the idea of delving into the local hierarchical geometry in point clouds, we present a new Hierarchical Geometry Network (HGNet) to facilitate the learning of point cloud representations. Specifically, we construct a four-level hierarchy among points, edges, surfaces, and super-surfaces. The edges are first built between two neighbor points and each is extended to a surface (triangle) by connecting another neighbor point. The adjacent surfaces are then grouped as a super-surface. Next, HGNet executes top-down feature aggregation from super-surfaces, surfaces, and edges, leading to three upgraded geometry features of surfaces, edges, and points, respectively. Finally, we capitalize on a Transformer structure to unify these refined surface-level, edge-level, and point-level geometry features with holistic contextual information, thereby improving point cloud analysis.

We analyze and evaluate our HGNet through extensive experiments over multiple 3D vision tasks for point cloud analysis (e.g., 3D object classification and part/semantic segmentation tasks), which empirically demonstrate its superior advantage against state-of-the-art backbones. More remarkably, on the real-world challenging benchmark of ScanObjectNN, HGNet achieves to-date the best published overall accuracy of 89.2%, which absolutely improves PointNeXt-S (87.7%) with 1.5%. For part and semantic segmentation on PartNet and S3DIS datasets, HGNet surpasses PosPool and PointNeXt by 1.8% and 1.8% in mIoU.

2. Related Work

Projection-based Methods aim to project the 3D point cloud to image planes or 3D voxels, while leaving the detailed geometry information unexploited. There are two main streams to formulate projections: view-based and voxel-based projections. Inspired by the breakthrough performance in 2D CNNs [10, 11, 41, 43], early works [3, 7, 31, 42] project the 3D point cloud to image planes from different views, avoiding the process of irregular and unordered point cloud. In general, they utilize 2D CNNs to extract features from image planes and fuse the multi-view features as the point cloud representations. Therefore, the point cloud analysis is framed as the well-explored 2D problem, and thus benefits from the mature techniques of CNNs. Another direction is to project the irregular 3D point cloud into regular 3D voxels, followed by standard 3D CNNs [5, 26, 48]. Due to the cubic growth in the number of voxels w.r.t. the resolution, voxel-based methods suffer from huge computation and memory costs. To tackle this issue, OctNet [40] uses a set of unbalanced octrees to hierarchically partition the 3D space. SSCNs [6] utilizes sparse convolutions to process spatially-sparse data more efficiently.

Point-based Methods mines the geometry structure among primary points to boost point cloud analysis. The pioneering work PointNet [30] utilizes multi-layer perceptrons followed by a max-pooling layer to extract global information. PointNet++ [32] further applies PointNet recursively on a nested partitioning of input point cloud for learning hierarchical representations. PointMLP [25] designs a pure residual MLP network coupled with a lightweight yet effective geometric affine module. PointNorm [60] proposes the DualNorm module after the sampling-grouping layer to reduce the irregularity of point cloud. PointNeXt [34] upgrades PointNet++ with an inverted residual bottleneck design and a set of improved training strategies to enable efficient and effective model scaling.

Moreover, recent works learn to capture more detailed geometry structures (e.g., edges/relaitons between points). For example, RS-CNN [17] proposes relation-shape convolution to explicitly encode geometric relation of points. DGCNN [46] introduces the edges in the local aggregation layer which connect a point and its neighbors. Then the local graph is dynamically constructed to capture local geometric features. PosPool [18] simplifies the local aggregation operator and removes learnable weights. In contrast to aggregating points within a local region, PointASNL [53] introduces the adaptive sampling and local-nonlocal module to capture neighbor and long-range dependencies of the sampled point. CurveNet [50] develops the curve aggregation to group both local and non-local points from a curve generated by the guided walk strategy. To enrich the local geometry information, surface curvature is proposed and calculated based on normal vector and neighbors coordinate [4]. RepSurf [39] further extends PointNet++ by replacing the point position with surface representation in the stem layer. Recently, inspired by Transformer used in various vision tasks [14, 15, 21, 22, 28, 29, 54–56], Transformer structure [8, 59] is applied to 3D point clouds, aiming to mine point relationship in local regions via self-attention.

Our work also falls into the latter category of exploiting detailed geometry structures among points. Compared to existing techniques that commonly take one or two kinds of geometry information into account, our HGNet delves into more detailed local hierarchical geometry among points, edges, surfaces, and even super-surfaces. Such design pursues a thorough point cloud understanding.

3. Approach

We design Hierarchical Geometry Network (HGNet) to exploit comprehensive hierarchical geometry in point clouds for facilitating point cloud representation learning. HGNet first builds a four-level hierarchy from the leaf level of points, to the intermediary levels of edges and surfaces, and the root level of the super-surfaces. Such way leads to a multi-granular and structured modeling of comprehensive geometry information in point clouds. Next, top-down feature aggregation is performed between every two adjacent levels to contextually strengthen point cloud representations at surface-level, edge-level, and point-level. A Transformer structure is finally leveraged to holistically unify all the enhanced surface-level, edge-level, and point-level representations for point cloud analysis.

3.1. Multi-stage Scheme

Formally, let $P = \{p_i | i = 1, ..., N\} \in \mathbb{R}^{N\times 3}$ denote a set of points, where N is the number of points in the Cartesian space. The target of point-based methods is to design a deep neural network to learn the representations of P. Most existing architectures adopt the typical multi-stage scheme based on PointNet++ [32], which consists of a stem layer and multiple stages with different point resolutions. The stem layer is to generate the representation f_i for each point p_i, and all the point features are further fed into the following stages. Each stage contains a local aggregation layer followed by several point-wise transformation layers. The module between every two stages employs the Farthest Point Sampling (FPS) algorithm to downsample the points, and meanwhile increase the number of channels. Note that the point resolution is preserved in the following layers within the same stage.

3.2. Point-level Representation

We start from the learning of point-level representations, which is formulated under the same multi-stage scheme as in PointNet++. In particular, each point is first encoded via point embedding in stem layer. Next, we perform the edge-to-point aggregation from edge-level to point-level on the basis of our constructed hierarchical topology, yielding the point-level representation.

Point Embedding. The point embedding in stem layer aims to transform each point p_i into a higher dimensional feature. Technically, let $\widetilde{f}_i$ denote the input feature of each point p_i for the stem layer. This input point feature $\widetilde{f}_i$ is commonly set as the position (*i.e.*, p_i=(x,y,z)) in the Cartesian space. A point-wise transformation layer is employed to map the input point feature into higher dimensional feature $f_i^p \in \mathbb{R}^d$, where d is the dimension of f_i^p:

$$f_i^p = F_p(\widetilde{f}_i). \tag{1}$$

F_p denotes the point-wise transformation layer (implemented as fully-connected layer). Next, the output point features of the stem layer are fed into the following layers in each stage, including the edge-to-point aggregation layer and several point-wise transformation layers.

Edge-to-Point Aggregation for Points. Given each transformed point feature, the edge-to-point aggregation layer is to aggregate the locally connected edges derived from its neighbor points. Here the edge-to-point aggregation layer is implemented as the typical point-wise MLP aggregation [18,32]. In particular, by taking the input point p_i as the anchor point, each connected edge between p_i and its neighbor point p_j is first represented as the relative location in between:

$$\Delta p_{ij} = p_j - p_i = [\Delta x_{ij}, \Delta y_{ij}, \Delta z_{ij}]. \tag{2}$$

Next, the relative location Δp_{ij} and the corresponding neighbor point feature f_j^p are jointly fed into a transform function $G^p(\cdot,\cdot)$, leading to an upgraded edge feature g_{ij}^p. Finally, all the upgraded features of connected edges are aggregated to form the output point-level feature g_i^p of the anchor point p_i through a reduction function R. Note that the reduction function is typically implemented as the average pooling, max pooling or self-attention. In this way, the whole procedure of edge-to-point aggregation for the anchor point p_i is formulated as:

$$\begin{aligned} g_{ij}^p &= G^p(\Delta p_{ij}, f_j^p) = G_{pos}^p(\Delta p_{ij}) + G_{feat}^p(f_j^p), \\ g_i^p &= R(\{g_{ij}^p \mid j \in N(i)\}), \end{aligned} \tag{3}$$

where $N(i)$ represents the set of neighbor points of the anchor point p_i according to the constructed hierarchical topology, G_{pos}^p and G_{feat}^p denotes the position and feature transformation of points, respectively. Accordingly, by applying edge-to-point aggregation over points, the output point-level feature g_i^p is nicely endowed with the edge geometry information.

3.3. Edge-level Representation

In analogy to point-level representation learning, we frame the learning of edge-level representation learning in the similar multi-stage scheme. Specifically, the stem layer is leveraged to transform the input edge through edge embedding. The surface-to-edge aggregation is then utilized from surface-level to edge-level based on our hierarchical topology, obtaining edge-level representations.

Edge Embedding. In this context, each input edge is defined as the connection between the point p_i and its nearest neighbor point p'_i. By denoting this input edge as $e_i = (p_i, p'_i)$, we also represent it as the relative location between two points, and a point-wise transformation layer is further utilized to encode this edge feature $e_i \in \mathbb{R}^3$ into higher dimensional feature $f_i^e \in \mathbb{R}^d$:

$$f_i^e = F_e(e_i) = F_e(p'_i - p_i), \tag{4}$$

where F_e is the point-wise transformation layer and the output edge features are regarded as the inputs of the following surface-to-edge aggregation layer.

Surface-to-Edge Aggregation for Edges. Based on each transformed edge feature, surface-to-edge aggregation layer aims to aggregate the locally built surfaces to form edge-level representations, where each surface is composed of the anchor edge and the neighbor point. Technically, given the edge $e_i = (p_i, p'_i)$ and the corresponding neighbor point p_j of the anchor point p_i, we directly construct a surface, which is denoted as $s_{ij} = (p_i, p'_i, p_j)$. Here we calculate the surface center c_{ij} and normal vectors n_{ij} to represent this surface s_{ij}:

$$\begin{aligned} c_{ij} &= (p_i + {p'}_i + p_j)/3, \\ n_{ij} &= \mathbf{CrossProduct}({p'}_i - p_i, p_j - p_i), \end{aligned} \tag{5}$$

where $\mathbf{CrossProduct}$ is the cross product operation. After that, the surface center c_{ij}, normal vector n_{ij}, and the edge feature derived from neighbor point f_j^e are fed into a transform function $G^e(\cdot,\cdot,\cdot)$, yielding the upgraded surface feature g_{ij}^e. Furthermore, we aggregate the upgraded features of all the constructed surfaces on the hierarchical topology, leading to the edge-level representation g_i^e for the anchor edge e_i via reduction function R:

$$\begin{aligned} g_{ij}^e &= G^e(c_{ij}, n_{ij}, f_j^e) = G_{pos}^e(c_{ij}, n_{ij}) + G_{feat}^e(f_j^e), \\ g_i^e &= R(\{g_{ij}^e \mid j \in N(i)\}), \end{aligned} \tag{6}$$

where G_{pos}^e and G_{feat}^e is the position and feature transformation of edges, respectively. As such, the output edge-level feature g_i^e is strengthened with additional surface geometry information via surface-to-edge aggregation.

3.4. Surface-level Representation

Here we go one step further to learn surface-level representations with more complex geometry information among adjacent surfaces. Similarly, the learning of surface-level representation is also formulated with multi-stage scheme. Concretely, we first encode each surface with surface embedding in stem layer. The super-surface-to-surface aggregation is further leveraged from super-surface-level to surface-level based on the hierarchical topology, aiming to achieve surface-level representations.

Surface Embedding. Each input surface in the stem layer is constructed as the surface consisting of the point p_i and its two nearest neighbor points (p'_i and p''_i). In this way, we denote each input surface as $s_i = (p_i, p'_i, p''_i)$, and represent it as the concatenation of the corresponding surface center c_i and normal vector n_i. A point-wise transformation layer is further leveraged to transform this surface feature $s_i \in \mathbb{R}^6$ into high dimension representation $f_i^s \in \mathbb{R}^d$. The overall surface embedding operates as follows:

$$\begin{aligned} c_i &= (p_i + {p'}_i + {p''}_i)/3, \\ n_i &= \mathbf{CrossProduct}({p'}_i - p_i, {p''}_i - p_i), \\ f_i^s &= F_s(s_i) = F_s([c_i, n_i]), \end{aligned} \tag{7}$$

where F_s is the point-wise transformation layer.

Super-Surface-to-Surface Aggregation for Surfaces. Given each transformed surface feature, super-surface-to-surface aggregation is devised to aggregate the locally constructed super-surfaces. Each super-surface is built as a group of adjacent surfaces derived from the anchor surface and the neighbor point. Formally, conditioned on the anchor surface $s_i = (p_i, p'_i, p''_i)$ and one neighbor point p_j of the anchor point p_i, we construct the super-surface containing three adjacent surfaces ($s_{ij}^1 = (p_i, p'_i, p_j)$, $s_{ij}^2 = (p_i, p''_i, p_j)$, and $s_{ij}^3 = (p'_i, p''_i, p_j)$). For each surface s_{ij}^1/s_{ij}^2/s_{ij}^3, we calculate its surface representation $f_{ij}^{s^1}$/$f_{ij}^{s^2}$/$f_{ij}^{s^3}$ as the concatenation of surface center and normal vector as in Eq. (5). Next, we transform both super-surface feature (the concatenation of three adjacent surface features) and the surface feature derived from neighbor point f_j^s into the upgraded super-surface feature g_{ij}^f via transform function $G^s(\cdot,\cdot,\cdot,\cdot)$. All the upgraded features of super-surfaces based on hierarchical topology are further aggregated to produce the surface-level feature g_i^f of the anchor surface s_i through reduction function R:

$$\begin{aligned} g_{ij}^f &= G^s(f_{ij}^{s^1}, f_{ij}^{s^2}, f_{ij}^{s^3}, f_j^s) = G_{pos}^s(f_{ij}^{s^1}, f_{ij}^{s^2}, f_{ij}^{s^3}) + G_{feat}^s(f_j^s), \\ g_i^f &= R(\{g_{ij}^f \mid j \in N(i)\}), \end{aligned} \tag{8}$$

where G_{pos}^s and G_{feat}^s is the position and feature transformation of surfaces, respectively. In comparison to edge-level feature g_i^e and point-level feature g_i^p, this output surface-level feature g_i^f reflects more complex geometry information of super-surfaces.

3.5. HG Block

We proceed to present our core proposal, i.e., Hierarchical Geometry block (HG block), that holistically unifies all the point-level, edge-level, and surface-level features via a Transformer structure.

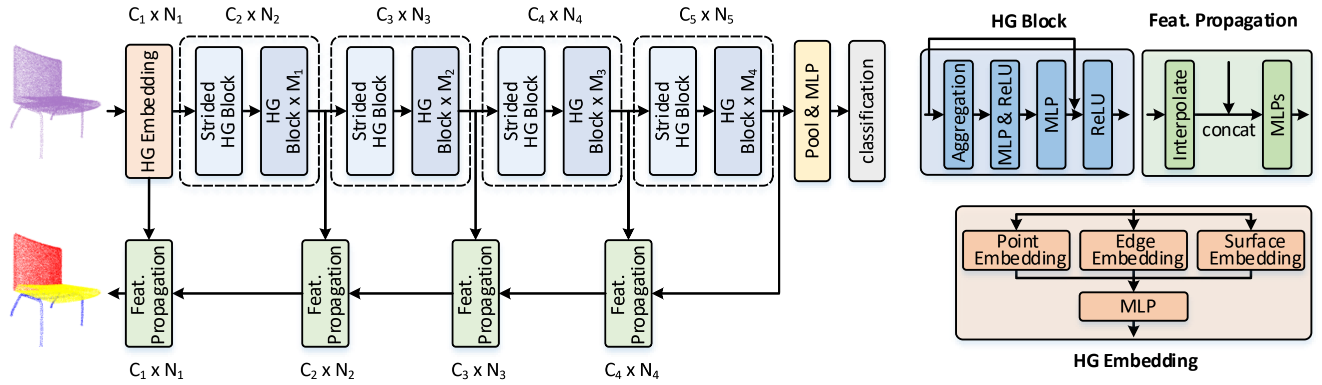

Figure 2. The detailed architecture of the proposed Hierarchical Geometry Network (HGNet) for point cloud analysis.

HG Embedding. HG block first integrates the input point, edge, and surface representations, and then encodes them through HG embedding in the stem layer. Specifically, the primary input point feature f_i^p, edge feature f_i^e, and surface feature f_i^s are obtained as in Eq.(1), (4), and (7). A fully-connected layer F_{hg} is then employed to jointly encode these features into multi-granular geometry features:

$$f_i^{hg} = F_{hg}([f_i^p, f_i^e, f_i^s]). \tag{9}$$

Transformer-based Aggregation. After that, HG block performs Transformer-based aggregation to dynamically aggregate the multi-granular geometry features of neighbor points. Technically, we calculate the output point cloud representations of HG block with self-attention layer:

$$\begin{aligned}
\Delta f_{ij}^{hg} &= f_j^{hg} - f_i^{hg},\\
\alpha_{ij} &= \rho(\gamma(\phi(\Delta f_{ij}^{hg}) + \delta(i,j))),\\
v_{ij} &= \beta(\varphi(\Delta f_{ij}^{hg}) + \delta(i,j)),\\
g_i^{hg} &= \sum_{j\in N(i)} \alpha_{ij} \odot v_{ij},
\end{aligned} \tag{10}$$

where Δf_{ij}^{hg} denotes the subtraction between multi-granular geometry feature f_i^{hg} and f_j^{hg} that basically reflects the relative geometry information in between. γ, β, ϕ, and φ are point-wise transformations, δ is a position encoding function and ρ is the normalization function (e.g., softmax). As such, the self-attention layer measures the attention weight for each neighbor point p_j of query point p_i, which is further utilized to aggregate the values (i.e., the features transformed by β). Here we directly utilize the summation of position transformation output in point-level, edge-level, and surface-level representations (in Eq.(3), (6), (8)) as the position encodings:

$$\delta(i,j) = G_{pos}^p(\Delta p_{ij}) + G_{pos}^e(c_{ij}, n_{ij}) + G_{pos}^s(f_{ij}^{s^1}, f_{ij}^{s^2}, f_{ij}^{s^3}). \tag{11}$$

In an effort to reduce the computational cost of self-attention, we reuse the values v_{ij} to compute the attention weight. Finally, the Transformer-based aggregation in HG block operates as follows:

$$\begin{aligned}
\Delta f_{ij}^{hg} &= f_j^{hg} - f_i^{hg},\\
v_{ij} &= \beta(\varphi(\Delta f_{ij}^{hg}) + \delta(i,j)),\\
g_i^{hg} &= \sum_{j\in N(i)} \rho(\gamma(v_{ij})) \odot v_{ij}.
\end{aligned} \tag{12}$$

3.6. HGNet

Following the basic configuration of PointNeXt [34], we construct our HGNet based on HG blocks. The entire architecture of HGNet is composed of one stem layer and four stages. The stem layer utilizes the HG embedding described in Sec.3.5 to encode the point, edge, surface representations. Each stage contains one strided HG block and multiple stacked HG blocks. The strided HG block contains one subsampling layer to downsample the points from previous stage and one Transformer-based aggregation layer. Next, inspired by ConvNeXt [19], the following HG block includes a Transformer-based aggregation layer and two layers of MLPs. For the part/semantic segmentation task, we adopt a U-net design and couple HGNet with a symmetric decoder. Transition up modules are additionally utilized to connect the consecutive stages in the decoder. Figure 2 shows an overview of HGNet architecture.

4. Experiments

We evaluate the effectiveness of HGNet via various empirical evidences on four challenging point cloud analysis datasets: ScanObjectNN [45] and ModelNet40 [49] for 3D classification, PartNet [27] for part segmentation, and S3DIS [1] for semantic segmentation.

4.1. 3D Classification on ScanObjectNN

Settings. ScanObjectNN is a real-world 3D object dataset, which is commonly adopted in recent point cloud analysis works [9, 25, 34]. It consists of about 15,000 real-world objects derived from 15 classes with 2,902 unique

Table 1. Comparison results of HGNet with other state-of-the-art methods on ScanObjectNN for 3D classification.

Method	OA	mAcc	Params.	FLOPs	Throughput
PointNet [30]	68.2	63.4	3.5M	0.9	4212
PointNet++ [32]	77.9	75.4	1.5M	1.7	1872
DGCNN [46]	78.1	73.6	1.8M	4.8	402
PointCNN [13]	78.5	75.1	-	-	-
BGA-DGCNN [45]	79.7	75.7	-	-	-
DRNet [36]	80.3	78.0	-	-	-
GBNet [37]	80.5	77.8	-	-	-
PRANet [2]	82.1	79.1	-	-	-
RepSurf-U [39]	84.3	81.3	1.5M	3.1	294
PointMLP [25]	85.4	83.9	13.2M	31.3	191
Point-M2AE [58]	86.4	-	-	-	-
Pix4Point [35]	86.8	84.9	-	-	-
PointNorm [60]	86.8	85.6	12.6M	-	140
PointNeXt-S [34]	87.7	85.8	1.4M	1.6	2040
HGNet	**89.2**	**87.5**	1.5M	3.1	1489

object instances. Note that this dataset includes noise, occlusions and the existence of background, thereby making it more challenging. Following [34], we conduct experiments on the hardest variant (PB_T50_RS) of ScanObjectNN, and report the class-average accuracy (mAcc) and overall accuracy (OA) for evaluation. During training, we adopt random scaling, random rotation and point resampling data augmentations as in [34]. We employ AdamW optimizer [24] on a single V100 GPU. The whole optimization includes 500 epochs with cosine decay learning rate scheduler [23] and label smoothing. The batch size, learning rate and weight decay is set as 64, 0.002, and 0.05, respectively.

Performance Comparison. Table 1 details the performance comparisons between our HGNet and other state-of-the-art methods. In general, our HGNet consistently exhibits better performances than the state-of-the-art methods, including point-based approach (e.g., PointNet), edge/relation-based method (e.g., PointNet++ and DGCNN), and surface-based technique (e.g., RepSurf-U), in terms of both evaluation metrics. In particular, the overall accuracy of HGNet reaches 89.2%, leading to the absolute improvement of 1.5% against the best competitor PointNeXt-S (87.7%). This generally demonstrates the key advantage of holistically integrating hierarchical geometry structures to facilitate point cloud analysis. More specifically, by additionally performing edge-to-point aggregation and using set abstraction to obtain joint representations of points and edges, PointNet++ outperforms PointNet that solely explores geometry information among points. DGCNN and BGA-DGCNN further boosts up the performances by leveraging graph models to exploit the edges/relations between points, that highlight the merit of edge-level geometry information. Moreover, when building an amplified expression of local geometry at surface-level, RepSurf-U exhibits better 3D classification performances than BGA-DGCNN. However, the performances of RepSurf-U are lower than PointNeXt-S that strengthens the 3D model capacity of edge-based method (PointNet++) with high-quality designs (e.g., an inverted residual bottleneck and improved training strategies). By further exploiting more detailed local hierarchical geometry among points, edges, surfaces, and super-surfaces, our HGNet achieves the best performances. The result confirms that framing unstructured point sets in hierarchical topology is an effective way to enhance point cloud representation learning.

Table 2. Comparison results of HGNet with other state-of-the-art methods on ModelNet40 for 3D classification.

Method	Inputs	OA	mAcc	Params.	FLOPs	Throughput
PointNet [30]	1k P	89.2	86.0	3.5M	0.9	4212
PointNet++ [32]	1k P	90.7	-	1.5M	1.7	1872
PointCNN [13]	1k P	92.5	88.1	0.6M	-	44
PointConv [47]	1k P	92.5	-	-	-	-
KPConv [44]	7k P	92.9	-	14.3M	-	-
DGCNN [46]	1k P	92.9	90.2	2.2M	3.9	263
RS-CNN [17]	1k P	92.9	-	-	-	-
PointASNL [53]	1k P	92.9	-	-	-	-
PCT [8]	1k P	93.2	-	2.9M	2.3	-
DensePoint [16]	1k P	93.2	-	-	-	-
PosPool [18]	5k P	93.2	-	19.4M	-	-
DeepGCN [12]	-	93.6	90.9	2.2M	3.9	263
GBNet [37]	1k P	93.8	91.0	8.4M	16.3	112
GDANet [52]	1k P	93.8	-	0.93M	26.3	14.0
PA-DGC [51]	1k P	93.9	-	-	-	-
Point Trans. [59]	1k P	93.7	90.6	-	-	-
PointNeXt-S [34]	1k P	93.7	90.9	4.5M	6.5	859
PointMLP [25]	1k P	94.1	91.3	13.2M	31.3	191
RepSurf-U [39]	1k P	94.4	91.4	1.5M	3.1	294
HGNet	1k P	**94.5**	**91.9**	1.4M	2.3	1550

4.2. 3D Classification on ModelNet40

Settings. ModelNet40 benchmark contains 12.3K meshed CAD models derived from 40 categories. The dataset is officially split into 9,843 and 2,468 CAD models for training and testing, respectively. We report the class-average accuracy (mAcc) and overall accuracy (OA) on testing set for evaluation as in [34]. During training, we follow the data pre-processing pipeline in [34], including data augmentations of random scaling and translation. The whole architecture is optimized with AdamW optimizer [24] (batch size: 32, learning rate: 0.001, weight decay: 0.05) on a single V100 GPU. The overall optimization process is composed of 600 epochs with cosine decay learning rate scheduler [23] and label smoothing loss.

Performance Comparison. Table 2 summarizes the performances. For fair comparison against most baselines with the inputs of 1K sampled points, we also implement our HGNet with the same inputs (1K points). As shown in this table, the performance trends on ModelNet40 are simi-

Table 3. Comparison results of HGNet with other state-of-the-art methods on PartNet for part segmentation.

Method	val	test	bed	bottle	chair	clock	dishw.	disp.	door	earph.	fauc.	knife	lamp	micro.	fridge	st. furn.	table	tr. can	vase
PointNet [30]	-	35.6	13.4	29.5	27.8	28.4	48.9	76.5	30.4	33.4	47.6	32.9	18.9	37.2	33.5	38.0	29.0	34.8	44.4
PointNet++ [32]	-	42.5	30.3	41.4	39.2	41.6	50.1	80.7	32.6	38.4	52.4	34.1	25.3	48.5	36.4	40.5	33.9	46.7	49.8
DeepGCN [12]	-	45.1	35.9	49.3	41.1	33.8	56.2	81.0	31.1	45.8	52.8	44.5	23.1	51.8	34.9	47.2	33.6	50.8	54.2
PointCNN [13]	-	46.4	41.9	41.8	43.9	36.3	58.7	82.5	37.8	48.9	60.5	34.1	20.1	58.2	42.9	49.4	21.3	53.1	58.9
point-wise MLP [18]	48.1	51.5	44.5	**52.6**	46.0	38.4	**68.2**	82.5	46.9	47.1	58.7	43.8	26.4	59.2	48.7	52.5	41.3	55.4	57.3
pseudo grid [18]	50.8	53.0	47.5	50.9	**49.2**	44.8	67.0	**84.2**	49.1	49.9	62.7	38.3	27.0	59.4	54.3	54.1	44.5	57.4	60.7
adapt weights [18]	50.1	53.5	46.1	47.9	47.2	42.7	64.4	83.7	55.6	49.5	61.7	49.5	27.4	59.3	**57.7**	53.5	45.1	57.5	60.9
PosPool [18]	50.6	53.8	49.5	49.4	48.3	**49.0**	65.6	**84.2**	**56.8**	53.8	62.4	39.3	24.7	61.3	55.5	54.6	44.8	56.9	58.2
HGNet	**52.7**	**55.6**	**51.0**	51.7	48.2	47.8	66.3	83.9	54.8	**54.9**	**64.3**	**51.9**	**30.1**	**61.9**	57.0	**54.7**	**46.4**	**58.4**	**61.2**

lar to those on ScanObjectNN. Concretely, HGNet slightly surpasses the current state-of-the-art technique (RepSurf-U). It is worthy to note that ModelNet40 is almost saturated (around 94% in OA metric for a long time), and it is difficult to introduce large margin of performance gains. Nevertheless, our HGNet still manages to obtain 0.1% improvement in OA metric against RepSurf-U. The results basically validate the effectiveness of unifying multi-granular geometry features among points, edges, and surfaces in our HGNet for point cloud analysis.

4.3. Part Segmentation on PartNet

Settings. For part segmentation task, we utilize PartNet benchmark which consists of 26,671 CAD shape models (70% models for training, 10% models for validation, and 20% models for testing) over 24 object categories. Each CAD shape model is annotated with 18 parts on average. We follow the setting in [18] and use the 10,000 points provided by this dataset as the inputs for each shape model. Random scaling and jittering are leveraged as data augmentations. We utilize AdamW optimizer [24] to train HGNet for 300 epochs with multi-step learning rate decay on four V100 GPUs. We fix the batch size, learning rate, weight decay, and decay rate as 8, 0.00125, 0.02, and 0.1, respectively. We report the part-category mIoU on both validation and testing sets (over 17 object categories with fine-grained annotations) for evaluation.

Performance Comparison. Table 3 shows the performances of our HGNet for part segmentation task. HGNet attains the highest performances for 10 of 17 object categories on testing (test) set. In particular, the part-category mIoU across all the 17 object categories reach 52.7% and 55.6% on validation (val) and test set, making the absolute improvement over PosPool by 2.1% and 1.8%. The results basically verify that the exploitation of hierarchical geometry among points, edges, and surfaces improves the 3D model capacity for part segmentation task. Figure 3 further showcases the part segmentation ground truths and results of different approaches for two CAD shape models. Compared to other baselines, the predictions of our HGNet are the closest to the ground truths.

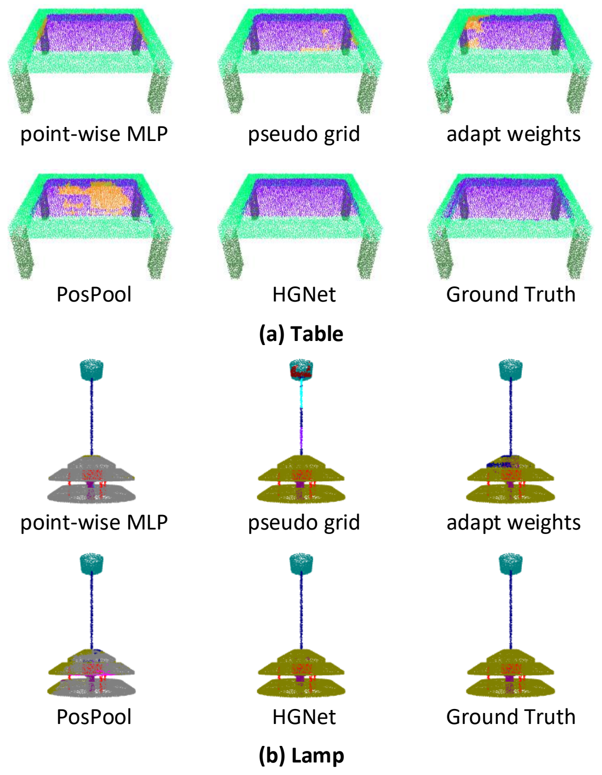

Figure 3. Examples of part segmentation results on PartNet.

4.4. Semantic Segmentation on S3DIS

Settings. We further evaluate our HGNet on S3DIS benchmark for semantic segmentation task. S3DIS is an indoor scene segmentation dataset that contains 271 scenes from 6 large-scale real indoor areas. The dataset consists of 273M points in total, and each point is annotated over 13 classes. As in [34], We take Area-5 as the testing scene and the remaining scenes as training set. Considering that each scene contains a large number of points, we downsample the point clouds with a voxel size of 0.04m, and use 24,000 points per batch as inputs following [33, 34, 44, 59] during training. The data pre-processing pipeline includes color auto-contrast, random scaling, random rotation, random jittering, and color drop data augmentations. We optimize the

Table 4. Comparison results of HGNet with other state-of-the-art methods on S3DIS for semantic segmentation.

Method	OA	mAcc	mIoU	Params.	Throughput	ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNet [30]	-	49.0	41.1	3.6M	162	88.8	97.3	69.8	**0.1**	3.9	46.3	10.8	59	52.6	5.9	40.3	26.4	33.2
PointCNN [13]	85.9	63.9	57.3	0.6M	-	92.3	98.2	79.4	0.0	17.6	22.8	62.1	74.4	80.6	31.7	66.7	62.1	56.7
DGCNN [46]	83.6	-	47.9	1.3M	8	-	-	-	-	-	-	-	-	-	-	-	-	-
DeepGCN [12]	-	-	52.5	3.6M	3	-	-	-	-	-	-	-	-	-	-	-	-	-
PVCNN [20]	87.1	-	59.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
KPConv [44]	-	72.8	67.1	15.0M	30	92.8	97.3	82.4	0.0	23.9	**58.0**	69.0	81.5	91.0	75.4	75.3	66.7	58.9
ASSANet-L [33]	-	-	66.8	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
PatchFormer [57]	-	-	68.1	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
RepSurf-U [39]	90.2	76.0	68.9	0.99M	69	-	-	-	-	-	-	-	-	-	-	-	-	-
PointNeXt-l [34]	90.0	75.3	69.0	7.1M	115	94.0	98.5	83.5	0.0	30.3	57.3	74.2	**82.1**	91.2	74.5	75.5	76.7	58.9
HGNet	**90.7**	**76.9**	**70.8**	7.8M	92	**95.4**	**98.7**	84.1	0.0	**39.7**	56.4	**77.3**	81.6	**91.7**	**77.9**	**76.3**	**77.3**	**63.3**

Table 5. Ablation study by comparing different variants of HG block in HGNet on ScanObjectNN for 3D classification task. MAX and SA represents that the reduction function is implemented as the max pooling or self-attention for aggregation.

#	Point	Edge	Surface	Reduction	OA	mAcc
1	✓			MAX	87.7	85.8
2		✓		MAX	87.6	86.0
3			✓	MAX	87.8	86.5
4	✓	✓		MAX	88.5	87.1
5	✓		✓	MAX	88.6	87.3
6	✓	✓	✓	MAX	88.9	87.3
7	✓	✓	✓	SA	89.2	87.5

whole architecture with AdamW optimizer [24] with label smoothing on a single V100 GPU. The whole optimization includes 100 epochs with cosine decay learning rate scheduler [23]. The batch size, learning rate, and weight decay are set as 8, 0.005, and 0.0001. We report the mean per-class IoU (mIoU), mean per-class accuracy (mAcc) and overall point accuracy (OA) for evaluation.

Performance Comparison. Table 4 lists the performance comparisons on S3DIS. Similarly, our HGNet clearly surpasses the point-based, edge/relation-based, and surface-based approaches, leading to the best performances across all the three metrics. Specifically, in comparison to PointNeXt-l, HGNet boosts up the performances by 0.7%, 1.6%, and 1.8% in OA, mAcc, and mIoU, respectively. The results clearly show the effectiveness of HGNet on semantic segmentation task.

4.5. Ablation study

In this section, we perform ablation study to validate the effectiveness of each design in our HG block. Table 5 details the performance comparisons among different ablated runs of HGNet on ScanObjectNN for 3D classification.

Ablation on Geometry Features in Different Levels. We first examine how performance is affected when capitalizing on different geometry features in HG block. As shown in Table 5 (Row 1-3), the use of each kind of feature from single level (point-level/edge-level/surface-level) in general achieves a good performance. In between, by exploring finer geometric structures, the surface-level feature achieves the best performances. Next, the linear fusion of point-level and edge-level/surface-level features (Row 4&5) constantly outperforms the individual representation. The linear integration of all features from three different levels (Row 6) further boosts up the performance. The results basically demonstrate the complementarity among the point-level, edge-level, and surface-level features.

Ablation on Different Reduction Functions. We then compare the performances by using different reduction functions for aggregation. One is the simple max pooling (Row 6), and the other is the use of self-attention mechanism via Transformer structure in our HG block (Row 7). Compared to max pooling, HG block dynamically exploits contextual information among point-level, edge-level, and surface-level features, manifesting better performances.

5. Conclusion

In this paper, we present a new architecture for point cloud analysis, namely Hierarchical Geometry Network (HGNet), that integrates hierarchical geometry structures within point sets to facilitate point cloud representation learning. Particularly, we study the problem from the viewpoint of interpreting the bottom-up hierarchical geometry from the leaf level of irregular points, to the intermediary levels of edges & surfaces, and the root level of super-surfaces. A four-level hierarchical topology is accordingly constructed, which in turn enhances surface-level, edge-level, and point-level features via top-down aggregation. HGNet further capitalizes on Transformer structure to holistically unify all the surface-level, edge-level, and point-level features. We empirically validate the superiority of HGNet over the state-of-the-art approaches in multiple 3D vision tasks (e.g., 3D object classification, part segmentation, and semantic segmentation tasks).

References

[1] Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105*, 2017. 5

[2] Silin Cheng, Xiwu Chen, Xinwei He, Zhe Liu, and Xiang Bai. Pra-net: Point relation-aware network for 3d point cloud analysis. *IEEE TIP*, 2021. 6

[3] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *CVPR*, 2018. 2

[4] A Foorginejad and K Khalili. Umbrella curvature: a new curvature estimation method for point clouds. *Procedia Technology*, 2014. 2

[5] Matheus Gadelha, Rui Wang, and Subhransu Maji. Multiresolution tree networks for 3d point cloud processing. In *ECCV*, 2018. 2

[6] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[7] Haiyun Guo, Jinqiao Wang, Yue Gao, Jianqiang Li, and Hanqing Lu. Multi-view 3d object retrieval with deep embedding network. *IEEE Transactions on Image Processing*, 2016. 2

[8] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 2021. 2, 6

[9] Abdullah Hamdi, Silvio Giancola, and Bernard Ghanem. Mvtn: Multi-view transformation network for 3d shape recognition. In *ICCV*, 2021. 5

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 2

[12] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, 2019. 6, 7, 8

[13] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 6, 7, 8

[14] Yehao Li, Yingwei Pan, Ting Yao, Jingwen Chen, and Tao Mei. Scheduled sampling in vision-language pretraining with decoupled encoder-decoder network. In *AAAI*, 2021. 2

[15] Yehao Li, Ting Yao, Yingwei Pan, and Tao Mei. Contextual transformer networks for visual recognition. *IEEE TPAMI*, 2022. 2

[16] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *ICCV*, 2019. 6

[17] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *CVPR*, 2019. 2, 6

[18] Ze Liu, Han Hu, Yue Cao, Zheng Zhang, and Xin Tong. A closer look at local aggregation operators in point cloud analysis. In *ECCV*, 2020. 2, 3, 6, 7

[19] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 5

[20] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *NeurIPS*, 2019. 8

[21] Fuchen Long, Zhaofan Qiu, Yingwei Pan, Ting Yao, Jiebo Luo, and Tao Mei. Stand-alone inter-frame attention in video models. In *CVPR*, 2022. 2

[22] Fuchen Long, Zhaofan Qiu, Yingwei Pan, Ting Yao, Chong-Wah Ngo, and Tao Mei. Dynamic temporal filtering in video models. In *ECCV*, 2022. 2

[23] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6, 8

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6, 7, 8

[25] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. In *ICLR*, 2021. 2, 5, 6

[26] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, 2015. 2

[27] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*, 2019. 5

[28] Yingwei Pan, Yehao Li, Jianjie Luo, Jun Xu, Ting Yao, and Tao Mei. Auto-captions on gif: A large-scale video-sentence dataset for vision-language pre-training. In *ACM Multimedia*, 2022. 2

[29] Yingwei Pan, Ting Yao, Yehao Li, and Tao Mei. X-linear attention networks for image captioning. In *CVPR*, 2020. 2

[30] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 1, 2, 6, 7, 8

[31] Charles R 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*, 2016. 2

[32] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. 2017. 1, 2, 3, 6, 7

[33] Guocheng Qian, Hasan Hammoud, Guohao Li, Ali Thabet, and Bernard Ghanem. Assanet: An anisotropic separable set abstraction for efficient point cloud representation learning. In *NeurIPS*, 2021. 7, 8

[34] Guocheng Qian, Yuchen Li, Houwen Peng, Jinjie Mai, Hasan Abed Al Kader Hammoud, Mohamed Elhoseiny, and Bernard Ghanem. Pointnext: Revisiting pointnet++ with improved training and scaling strategies. In *NeurIPS*, 2022. 2, 5, 6, 7, 8

[35] Guocheng Qian, Xingdi Zhang, Abdullah Hamdi, and Bernard Ghanem. Pix4point: Image pretrained transformers for 3d point cloud understanding. *arXiv preprint arXiv:2208.12259*, 2022. 6

[36] Shi Qiu, Saeed Anwar, and Nick Barnes. Dense-resolution network for point cloud classification and segmentation. In *WACV*, 2021. 6

[37] Shi Qiu, Saeed Anwar, and Nick Barnes. Geometric back-projection network for point cloud classification. *IEEE TMM*, 2021. 6

[38] Zhaofan Qiu, Yehao Li, Yu Wang, Yingwei Pan, Ting Yao, and Tao Mei. Spe-net: Boosting point cloud analysis via rotation robustness enhancement. In *ECCV*, 2022. 1

[39] Haoxi Ran, Jun Liu, and Chengjie Wang. Surface representation for point clouds. In *CVPR*, 2022. 1, 2, 6, 8

[40] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 2

[41] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 2

[42] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 2

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 2

[44] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *ICCV*, 2019. 6, 7, 8

[45] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 5, 6

[46] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 2019. 1, 2, 6, 8

[47] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *CVPR*, 2019. 6

[48] 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*, 2015. 2

[49] 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*, 2015. 5

[50] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. In *ICCV*, 2021. 2

[51] Mutian Xu, Runyu Ding, Hengshuang Zhao, and Xiaojuan Qi. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. In *CVPR*, 2021. 6

[52] Mutian Xu, Junhao Zhang, Zhipeng Zhou, Mingye Xu, Xiaojuan Qi, and Yu Qiao. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *AAAI*, 2021. 6

[53] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *CVPR*, 2020. 2, 6

[54] Ting Yao, Yehao Li, Yingwei Pan, Yu Wang, Xiao-Ping Zhang, and Tao Mei. Dual vision transformer. *arXiv preprint arXiv:2207.04976*, 2022. 2

[55] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *ECCV*, 2018. 2

[56] Ting Yao, Yingwei Pan, Yehao Li, Chong-Wah Ngo, and Tao Mei. Wave-vit: Unifying wavelet and transformers for visual representation learning. In *ECCV*, 2022. 2

[57] Cheng Zhang, Haocheng Wan, Xinyi Shen, and Zizhao Wu. Patchformer: An efficient point transformer with patch attention. In *CVPR*, 2022. 8

[58] Renrui Zhang, Ziyu Guo, Peng Gao, Rongyao Fang, Bin Zhao, Dong Wang, Yu Qiao, and Hongsheng Li. Point-m2ae: Multi-scale masked autoencoders for hierarchical point cloud pre-training. In *NeurIPS*, 2022. 6

[59] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *ICCV*, 2021. 2, 6, 7

[60] Shen Zheng, Jinqian Pan, Changjie Lu, and Gaurav Gupta. Pointnorm: Dual normalization is all you need for point cloud analysis. *arXiv preprint arXiv:2207.06324*, 2022. 2, 6