A TINY MACHINE LEARNING MODEL FOR POINT CLOUD OBJECT CLASSIFICATION

A PREPRINT

Min Zhang *
Media Communications Lab
University of Southern California
Los Angeles, CA, USA

Jintang Xue *
Media Communications Lab
University of Southern California
Los Angeles, CA, USA

Pranav Kadam
Media Communications Lab
University of Southern California
Los Angeles, CA, USA

Hardik Prajapati
Media Communications Lab
University of Southern California
Los Angeles, CA, USA

Shan Liu
Tencent Media Lab
Tencent America
Palo Alto, CA, USA

C.-C. Jay Kuo
Media Communications Lab
University of Southern California
Los Angeles, CA, USA

March 21, 2023

ABSTRACT

The design of a tiny machine learning model, which can be deployed in mobile and edge devices, for point cloud object classification is investigated in this work. To achieve this objective, we replace the multi-scale representation of a point cloud object with a single-scale representation for complexity reduction, and exploit rich 3D geometric information of a point cloud object for performance improvement. The proposed solution is named Green-PointHop due to its low computational complexity. We evaluate the performance of Green-PointHop on ModelNet40 and ScanObjectNN two datasets. Green-PointHop has a model size of 64K parameters. It demands 2.3M floating-point operations (FLOPs) to classify a ModelNet40 object of 1024 down-sampled points. Its classification performance gaps against the state-of-the-art DGCNN method are 3% and 7% for ModelNet40 and ScanObjectNN, respectively. On the other hand, the model size and inference complexity of DGCNN are 42X and 1203X of those of Green-PointHop, respectively.

***K*eywords** Point clouds · object classification · tiny learning model · green learning · PointHop

1 Introduction

Given a point cloud object scan, the goal of point cloud classification is to assign it a category label. Point cloud object classification is a fundamental task in point cloud analysis and processing. It lays the foundation for other advanced 3D vision tasks such as semantic/instance segmentation, object detection, registration and generation. Different from semantic segmentation or object detection, which needs to deal with large-scale and noisy point clouds, point cloud object classification often targets at classifying small-scale objects that are clean and well aligned. The point cloud object classification task relies on the understanding of the local and global structures of an object. Moreover, efficiency of a classifier matters in practical 3D real-time systems that often contain a few advanced tasks and applications. The design of a tiny machine learning model to be deployed in mobile and edge devices is our focus in this work.

Two families of machine learning models have been proposed for 3D point cloud object classification. The first one relies on the end-to-end optimized deep learning (DL) technique. A representative method is PointNet [1]. The second one adopts the green learning (GL) methodology [2] aiming at smaller model sizes and lower computational complexity. An illustrative example is PointHop [3]. To design a tiny point cloud classifier, we have observations on methods of these two families below.

*The first two authors contributed equally to this work. Corresponding author: Min Zhang, zhan980@alumni.usc.edu.

As a pioneering work, PointNet has influenced DL-based neural networks on point cloud classification significantly. Its follow-ups include [4–8]. All of them attempt to learn a richer context. Recently, more complex models [9, 10] are proposed to push the performance furthermore. Their computational complexity and memory costs increase accordingly. On the other hand, we observe that the performance of point cloud classifiers is not much affected by their layer depth and complexity. For instance, PointNet achieves 47.6% mIoU (the mean intersection over union) in semantic segmentation for the S3DIS dataset [11] while Point Transformer [10] increases the mIoU value to 73.5%. In contrast, PointNet and Point Transformer achieve 89.2% and 93.7% classification accuracy on the ModelNet40 dataset [12], respectively. As compared with the 25.9% performance gain in semantic segmentation, the improvement of 4.5% classification accuracy of Point Transformer over PointNet is not impressive.

Solutions to point cloud classification, segmentation and registration can be designed using the GL methodology to achieve lower inference complexity and smaller model sizes. Since no backpropagation is used to determine model parameters, their training is also very fast. All of these advantages are enabled by the unsupervised representation learning module in GL. PointHop is the first GL-based method on point cloud classification. It learns local-to-global attributes hierarchically. The framework has been successfully extended to other tasks such as point cloud segmentation [13] and point cloud registration [14]. Nevertheless, the limited performance gain in classification accuracy (i.e., 2%) from PointHop to its enhanced solution, PointHop++ [15], is also observed.

Based on the above observations, there appears to be little gain in building a complex point cloud classifier. Although GL-based point cloud classifiers are more efficient than their DL-based counterparts, we wonder whether it is possible to obtain an even more economical model. Models of both DL-based and GL-based families share one common principle – all of them represent point cloud objects in multiple scales. To reduce the complexity of existing solutions, we replace the multi-scale representation of a point cloud object with a single-scale representation for complexity reduction, and exploit rich 3D geometric information of a point cloud object for performance preservation. The proposed solution is named Green-PointHop due to its low computational complexity.

Since Green-PointHop has only one hop, its computation time and model size can be reduced. It compensates the performance loss of a shallow model by concatenating 3D complementary geometric information using the global, cone and inverted cone aggregations. As compared to PointHop and PointHop++, which need four hops to learn hierarchical representations of point clouds, Green-PointHop has a much simpler architecture. We evaluate the performance of Green-PointHop on ModelNet40 and ScanObjectNN two datasets. Green-PointHop has a model size of 64K parameters. It demands 2.3M floating-point operations (FLOPs) to classify a ModelNet40 object of 1024 down-sampled points. Its classification performance gaps against the state-of-the-art DGCNN method are 3% and 7% for ModelNet40 and ScanObjectNN, respectively. On the other hand, the model size and inference complexity of DGCNN are 42X and 1203X of those of Green-PointHop, respectively.

The rest of the paper is organized as follows. Related work is reviewed in Sec. 2. The Green-PointHop method is proposed in Sec. 3. Experimental results are shown in Sec. 4. Finally, concluding remarks are given in Sec. 5.

2 Related Work

2.1 Methods for Point Cloud Analysis

For traditional point cloud analysis methods, it is common to use local geometric properties as point cloud representations. Representative 3D local geometric descriptors include PFH [16], FPFH [16], ISS [17], and SHOT [18], etc. Traditional point cloud analysis methods are mathematically interpretable. They offer good and fast results in simple scenarios. Yet, they are not effective in handling complex objects.

With the success of DL in image processing and computer vision, researchers tried to apply DL to point clouds but encountered difficulty due to the unordered and unstructured nature of 3D points initially. As a pioneering DL solution, PointNet solves this problem using multi-layer perceptrons (MLPs) and max aggregation. Since the max aggregation function is a symmetric one, PointNet is invariant to the order of points. PointNet only learns the global structure of point clouds without considering local structures of points.

Its follow-ups, e.g., PointNet++ [7], PointCNN [6], PointSIFT [5], DGCNN [8], RandLANet [4], learn richer local contexts. PointNet++ applies PointNet to the local neighborhood of each point and aggregates local features hierarchically. PointCNN uses a χ-Conv to aggregate features in a local region with a latent and potentially canonical order. Inspired by the SIFT descriptor [19], PointSIFT uses an orientation-encoding unit to encode eight orientations. DGCNN [8] updates local regions at each layer using point features (instead of 3D coordinates). This idea works better in capturing the semantic information. RandLANet [4] has an attention mechanism and aggregates features with attention scores.

DL-based methods achieve good performance in complex tasks but require a large number of training data, expensive annotation, and huge computational resources. Moreover, they are not as transparent as traditional methods. GL-based point cloud analysis methods strike a balance between simple traditional methods and costly DL-based methods.

PointHop [3] and PointHop++ [15] are two pioneering GL-based point cloud processing systems. PointHop++ is an improved version of PointHop method. They exploit successive subspace learning (SSL) [20] to learn rich representations of point clouds. Specifically, they adopt an unsupervised feature extraction module without any class labels. They have an extremely low training complexity since no backpropagation is used. They only demand one feedforward pass to learn model parameters in the feature extraction module.

PointHop and PointHop++ lay the foundation for other advanced GL-based point cloud methods. The representation learning technique proposed in PointHop is commonly adopted by others. Examples include point cloud classification [3, 15, 21], small object part segmentation [13], large-scale semantic segmentation [22], registration [14, 23], odometry [24], joint object retrieval and pose estimation [25], rotational-invariant object classification [21], and scene flow estimation [26].

2.2 GL Methodology and Applications

GL was originally developed for image classification tasks [20, 27, 28] based on a sequence of efforts in analyzing the operations of convolutional neural networks (CNNs) [2, 29–32]. The GL paradigm has been successfully applied to other 2D vision tasks such as face biometrics [33], deepfake detection [34], anomaly detection [35], image generation [36–38], and object tracking [39–42]. The GL paradigm has also been exteneded to 3D vision problems, where a wide range of point cloud processing algorithms have been developed in recent years [43].

3 Green-PointHop Method

3.1 Motivation and System Overview

It is common to use an hierarchical processing pipeline for object recognition in 2D images, say, the multi-layer network architecture in the context of CNNs. The computational neurons at shallower (or deeper) layers have smaller (or larger) receptive fields. More neurons are needed at deeper layers due to higher content diversity in a larger receptive field. Although the same principle holds for 3D point clouds, there is a major difference in pooling operations between images and point cloud data.

A (2x2)-to-(1x1) local pooling is applied to filter responses in hierarchical image processing. This is proper due to the structured order of image pixels. All relative 2D spatial information is still preserved in pooled responses. Yet, the situation changes in 3D point clouds. To deal with unstructured 3D points, we need to define two operations, pooling and aggregation, separately. The pooling operation is applied to input 3D points (rather than filter responses) while the aggregation function, which has to be a symmetric one, is applied to filter responses. The underlying 3D spatial information is lost after aggregation. The hierarchical architecture computes spectral representations at different scales. Its cost becomes higher as the layer goes deeper.

To reduce the computational and memory costs, we consider a single-hop model called Green-PointHop, where no pooling is used to reduce the point number. It computes the spectral representation of a local neighborhood centered at each point using the Saab transform [32], and conducts aggregations across points of an object and its partial views (or sub-objects). This new yet simple idea has a clear geometric interpretation. Furthermore, its model size and computation complexity can be significantly reduced since it has filtering operations at a single scale.

An overview of the proposed Green-PointHop system is shown in Fig. 1. The input point cloud scan consists of N points. Each point, p_i, has three Cartesian coordinates (x_i, y_i, z_i), representing its location in the 3D space. To obtain a discriminant point cloud representation for classification, we propose a three-stage design. In the first stage, we compute local representations for each point as detailed in Sec. 3.2. In the second stage, we perform geometric aggregations of point-wise representations in Sec. 3.3. In the third stage, we rank the discriminant power of each representation dimension from the highest to the lowest, select discriminant ones based on the elbow point of the curve, and feed them into a classifier to obtain an object label. The detail is given in Sec. 3.4.

3.2 Stage 1: Point-wise Representation Learning

Raw Point-wise Representation Vector. For each point, we find its K nearest neighbors using the K-NN search. As shown in the first stage of Fig. 1, the red point is the target point while the yellow points are its K nearest neighbors. We compute local coordinates of yellow points with the red point as the origin, partition them using eight octants,

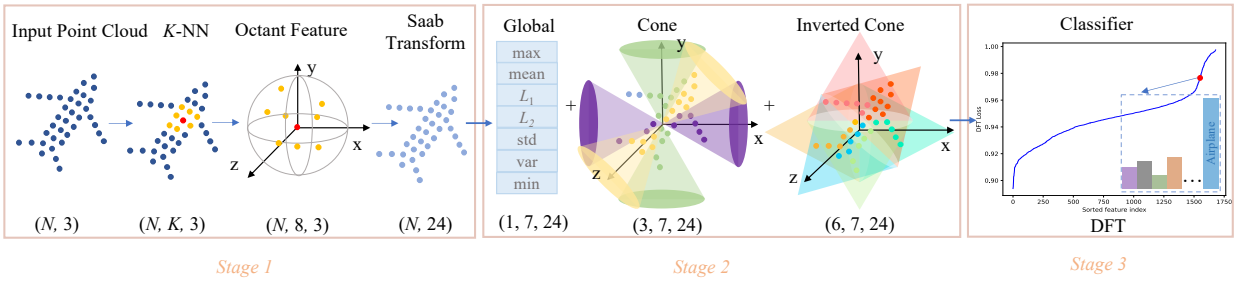

Figure 1: An overview of the proposed Green-PointHop method. It consists of three stages. In the first stage, it computes a point-wise representation for every point in the input point cloud. For example, the neighborhood of the red point is marked in yellow and determined by the KNN search. Then, the point-wise representation vector is derived using the eight-octant partitioning and averaging, which are followed by Saab filtering. In the second stage, seven schemes (i.e., max, mean, L_1, L_2, std, var, and min) are used to aggregate point-wise representation vectors into global and regional representation vectors. Regions are defined by points inside cones or inverted cones. In the third stage, discriminant features are selected using the discriminant feature test (DFT) and object classification is performed using the linear-least-squares-regression (LLSR) classifier. Unsupervised learning is adopted in the first and the second stages, where no object labels are used. Supervised learning is used in the third stage.

and calculate the centroid of points inside each octant. If an octant does not have any point, its centroid is set to $(0, 0, 0)^T$. Since there are 8 octants and each centroid has 3 spatial coordinates, we have a raw representation vector of 24 dimensions.

Saab Filtering. There are correlations between dimensions of raw point-wise representation vector. It is advantageous to decorrelate them using the Saab transform based on our prior experiences in PointHop and PointHop++. The Saab transform [32] contains one DC kernel (or filter) and 23 AC kernels (or filters). The DC kernel is a constant-element vector. Since DC-removed vectors can be treated as zero-mean vectors, we conduct the principal component analysis (PCA) on them and obtain 23 AC kernels. Then, by multiplying the Saab transform matrix of dimension 24×24 with the raw representation vector of dimension 24, we obtain 24 filter responses, which define the ultimate point-wise representation vector.

The point-wise representation vector was first proposed in [3]. It has three nice properties. First, it is invariant under index permutation of points. Second, it is a spectral representation via the Saab transform, which is more effective than the raw representation in the spatial domain. Third, it is a representation derived by unsupervised learning. No object labels are needed.

3.3 Stage 2: Global and Regional Representation Learning

Point-wise representation vectors alone are too local to be discriminant. Besides, the number of these vectors is proportional to the number of points, which is too many to feed to the classifier directly. It is essential to aggregrate point-wise representation vectors across a set of points to offer robust and discriminant representations of manageable sizes. The aggregation of point-wise octant representation vectors is proven to be effective in prior GL-based point cloud analysis work such as PointHop, PointHop++ and R-PointHop [23]. Some adjustments are made to tailor to Green-PointHop as described below.

Aggregation Functions. The aggregation function is applied to individual elements of the representation vectors of selected points. It should be a symmetric function so that the aggregation output is invariant to the permutation of point indices. The maximum (or max) aggregation function was used in PointNet. Three more aggregation functions are added in PointHop and PointHop++; namely, the mean, the L_1-norm, and the L_2-norm. They aggregate point-wise representation vectors in a complementary manner and offer better performance. Here, we add three more aggregation functions to Green-PointHop: the minimum (min), the standard deviation (std) and the variance (var). Thus, Green-PointHop has seven aggregation functions in total as shown in the second stage of Fig. 1. The effectiveness of these aggregation functions is analyzed in Sec. 4.2.3.

Aggregation Regions. The aggregation function has to work on a set of points. One default choice is to include all points of an input point cloud. It is called the global aggregation. Furthermore, we can consider a subset of points. We use the centroid of all points as the new origin while keeping the same pointing directions of the x, y, and z three axes in the following discussion. We define a symmetric cone-shaped region by including all points whose angles with

respect to the positive or negative x-axis are less than a predefined parameter, θ_1. This cone region is shown in purple in the second stage of Fig. 1. Similarly, we define two more symmetric cone-shaped regions with respect to the y- and z-axes. They are colored in green and yellow, respectively. We can further manipulate the cone that covers points in the positive x-axis as follows. We move the vertex of the cone to $(\Delta, 0, 0)^T$, where $\Delta > 0$ is a parameter selected by users and invert the cone orientation by 180 degrees. Then, points with the new vertex as the origin with angles against the negative x-axis less than θ_2 define an inverted cone region that has its base on the y-z plane (i.e., $x = 0$). With the same manipulations, we can obtain 6 inverted cone-shape regions. They are marked by 6 different colors in the right of the second module in Fig. 1. In the experiments, we set $\theta_1 = 75°$ for three symmetric cone-shaped regions and $\theta_2 = 45°$ for six inverted cone-shaped regions. The former three are used to capture symmetrical shape properties while the latter six are used to capture non-symmetric shape characteristics.

To summarize, there are 10 regions in our design (i.e., one global, three symmetric cones and six inverted cones). We apply the 7 aggregation functions to the 24D point-wise representation vector of all points in each region, and concatenate all of them to form a joint representation vector of dimension $7 \times 24 \times 10 = 1680$, which serves as the input to the last stage.

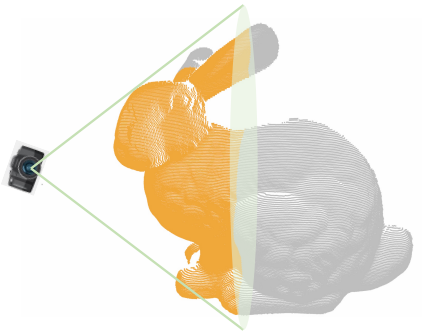

Figure 2: An illustration for the intuition behind the regional cone-shaped aggregation. A pinhole camera is set at a certain distance to a Stanford bunny and views a specific part of it.

There is an intuitive way to explain the regional cone-shaped design. Each cone can be treated as a pinhole camera at different locations viewing a specific part of the object. For example, the pinhole camera is at the center of the object and views parts of the object along each axis. The viewing angle has a certain limitation bounded by θ_1. As an alternative, the camera goes to different sides of the object, turns around, and examines the object from each side to the center, where the viewing angle is bounded by θ_2. In Fig. 2, the Stanford bunny [44] is used to illustrate our intuition.

3.4 Stage 3: Feature Selection and Decision Learing

From Representation Vectors to Features. As stated above, the total dimension number from Stage 2 is 1680. We can select a subset of them to serve as the final feature set. Here, we adopt a supervised feature selection method called the discriminant feature test (DFT) [45]. The DFT is a tool used to measure the discriminant power of each dimension independently. It consists of two steps: 1) The weighted entropy loss for each 1D dimension is calculated at a partition, which is chosen from a set of uniformly partitioned points. 2) The lowest weighted entropy loss is set to the DFT loss of the dimension. Then, we sort dimensions from the lowest to the highest DFT loss values as shown in the third stage of Fig. 1, where the x-axis is the sorted dimension index and the y-axis is the DFT loss value. The lower the DFT loss, the higher the discriminant power. Then, we can use the elbow point of the curve as indicated by the red point to determine a subset of discriminant features. Features with their loss values less than that of the red point are selected.

Classifier Selection. We try a couple of classifiers and find that the linear-least-squares-regression (LLSR) classifier gives the best tradeoff between the classification performance, computational complexity, and the number of model parameters. Thus, we choose it as the classifier for Green-PointHop.

4 Experiments

4.1 Experimental Setup

We first evaluate Green-PointHop on ModelNet40 [12]. It has 40 object categories. The dataset contains 9,843 training samples and 2,468 test samples. We randomly down-sample each point cloud scan from 2,048 points to 1,024 points

for further processing. In experiments, we set $K = 32$ in the KNN search, select 1569 features using DFT, and feed them into the LLSR classifier. Exhaustive experimental results such as classification accuracy, model sizes, and time complexity in inference, and ablation study are reported in Sec. 4.2.

To show the generalizability of Green-PointHop, we also evaluate its performance on the ScanObjectNN dataset [46]. It consists of objects extracted from real world scenes. The dataset has several scenarios such as the "object only", "object with background", and "object with perturbation". To align with ModelNet40, we evaluate the "object only" scenario only. For this scenario, we have 2,309 training samples and 581 test samples from 15 object classes. Since real world models have background noise and incomplete scans, we adjust some hyper-parameters in Green-PointHop. In experiments, we set $K = 48$ in the KNN search, 65° for all cones, and select 1108 features based on DFT. The experimental results are given in Sec. 4.3.

4.2 ModelNet40 Dataset

4.2.1 Classification Performance

We compare the classification accuracy results of several methods on ModelNet40 in Table 1. All benchmarking methods are evaluated in two metrics [1]: the average of per-class accuracy (class-avg) and the overall accuracy of all scans. Among DL-based methods, we see 4.5% overall accuracy improvement from PointNet in 2017 to Point Transformer in 2021 [10]. Nevertheless, the improvement comes at the price of a substantially larger model size and a significantly higher complexity. Among GL-based methods, Green-PointHop achieves 86% class-avg and 90.6% overall accuracy, which are second (but close) to PointHop++. Since Green-PointHop is a single-scale method with only one hop, we report the classification accuracy of PointHop and PointHop++ using only one hop in the same table for comparison. Clearly, both of them have a large performance gap with respect to Green-PointHop. It demonstrates the effectiveness of aggregating complementary geometric information through multiple oriented cones. By comparing DL- and GL-based methods, the performance of Green-PointHop is comparable with those of PointNet and PointNet++.

Table 1: Classification performance comparison of representative DL- and GL-based point cloud object classification methods on ModelNet40.

Learning Scheme	Method		Accuracy (%)	
			class-avg	overall
Deep Learning	PointNet [1]		86.2	89.2
	PointNet++ [7]		-	90.7
	PointCNN [6]		88.1	92.5
	PointConv [47]		-	92.5
	DGCNN [48]		90.2	92.9
	KPConv [49]		-	92.9
	PCT [9]		-	93.2
	Point Transformer [10]		90.6	93.7
Green Learning	4 hops	PointHop [3]	84.4	89.1
		PointHop++ [15]	87	91.1
	1 hop	PointHop	44.8	60.7
		PointHop++	49.7	64.5
		Green-PointHop (Ours)	86	90.6

4.2.2 Model and Time Complexities

We compare the model and time complexities of representative DL-based and GL-based methods in Table 2. We adopt different hardware platforms for DL- and GL-based methods due to their different algorithmic nature. As reported in the second column, DL-based methods are trained on a single GeForce GTX TITAN X GPU while GL-based methods are trained on a 24-thread Intel(R) Xeon(R) CPU. It takes several hours to train DL models. In contrast, it takes around 20 minutes and 25 minutes to train PointHop and PointHop++, respectively. The new Green-PointHop model completes its training within 5 minutes. Although we conduct Green-PointHop's inference on CPU, its inference time (23 ms) is close to that of PointNet and PointNet++ using GPU (i.e., 10 ms and 14 ms, respectively).

One platform-independent computational complexity measure is the number of floating-point operations (FLOPs). The number of FLOPs is proportional to power consumption as well as carbon footprint. In the 3rd column, we report the numbers of FLOPs in the unit of millions and show their relative ratios with respect to that of Green-PointHop inside

Table 2: Complexity comparison of representative DL- and GL-based methods in four measures: 1) hardware configuation, 2) FLOPs in inference, 3) training and inference time, and 4) the number of model parameters (i.e., model size), where 'M' stands for the unit of a million.

Method	Hardware Configuration	FLOPs (M)	Time (hr, ms)		Parameter No. (M)		
			Training	Inference	Filter	Classifier	Total
PointNet [1]	GPU	957 (416X)	7	10	-	-	3.48 (56.13X)
PointNet++ [7]		3136 (1363X)	7	14	-	-	1.48 (23.87X)
DGCNN [48]		2768 (1203X)	21	154	-	-	2.63 (42.42X)
PointHop [3]	CPU	7.7 (3X)	0.33	108	0.037	-	-
PointHop++ [15]		4.0 (2X)	0.42	97	0.009	0.15	0.159 (2.56X)
Green-PointHop (Ours)		2.3 (1X)	0.08	23	0.002	0.06	0.062 (1X)

the parentheses. Green-PointHop has around 2.3 million FLOPs, which is about one half and one third of PointHop and PointHop++, respectively. For the FLOP number of DL-based methods, we cite the data in [50]. As shown in Table 2, the FLOP numbers of DL-based methods are about three orders of magnitude of that of Green-PointHop, which shows power efficiency of Green-PointHop.

Finally, for model size comparison, we list the numbers of model parameters in the last column of the table. For GL-based methods, besides the total number of model parameters, we report the parameter numbers in two modules separately: 1) the filters used in unsupervised feature learning, and 2) the LLSR classifier. Green-PointHop has significantly fewer parameters in these two modules than PointHop and PointHop++ because of its single-hop architecture. The total number of parameters of Green-PointHop is only 62K, which can easily fit into a 256k-byte cache if each parameter is stored in four bytes.

4.2.3 Ablation Study

We conduct extensive experiments with various settings of Green-PointHop on ModelNet40 to shed light on several design choices.

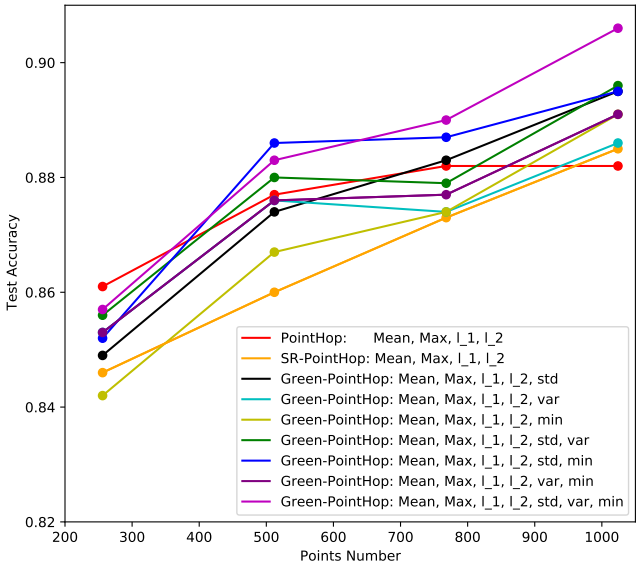

Figure 3: The plot of Green-PointHop's overall classification accuracy as a function of the aggregating point number under eight different settings those adopt a subset of seven aggregation quantities (i.e., max, min, mean, L_1, L_2, std and variance values) of selected points.

Global Aggregation Schemes. We first study the effect of using different global aggregation schemes on the overall classification accuracy in Fig. 3. The x-axis of the figure indicates the aggregation across four sets of points (i.e., 256, 512, 768, and 1024 points). The first three sets are randomly sampled from a total of 1024 points. The general trend

is that the test accuracy improves as the number of sampled points increases, being attributed to a better coverage of underlying point cloud scans.

As to the number of aggregation quantities, we begin with the best setting of PointHop, which is an ensemble of max, mean, L_1 and L_2 four quantities. Under this setting, Green-PointHop has worse performance than PointHop except for 1024 points. To improve the performance, we calculate three more quantities as the new aggregation schemes. They are the min, the standard deviation and the variance across a selected set of spatial points. They offer new ways to summarize the global information obtained at the first stage in Fig. 1. We compare the test accuracy of eight aggregation cases of Green-PointHop in Fig. 3. As shown in the figure, the ensemble of all seven aggregation schemes gives the best performance. The robustness of Green-PointHop is demonstrated by good test performance with fewer points in point cloud object scans. Although Green-PointHop has one scale only, it offers the best performance for cases of 768 and 1024 points and the second best performance for cases of 256 and 512 points.

Sizes of Nearest Neighborhood. Next, we investigate the effect of the number of nearest neighbors (or the number of K in the KNN search) adopted in the first stage of Fig. 1. We examine $K = 16$, 32 and 64 three cases in the first three columns of Table 3. By focusing on the last three rows, we see $K = 32$ gives the best result when other conditions are equal. It outperforms the cases of $K = 16$ and $K = 64$ by 1.5% and 0.85%, respectively. Thus, we successfully reduce K from 64 in PointHop to 32 in Green-PointHop. This helps reduce the memory size and time complexity.

Effects of Global and Local Cone and Inverted Cone Aggregations. When $K = 32$, we see from Table 3 that Green-PointHop has the highest accuracy (90.64%) by concatenating features obtained from the global aggregation, the local cone aggregation, and the local inverted cone aggregation. It validates that the proposed cone and inverted cone aggregations can capture more shape information of objects than the simple global aggregation. This study shows the power of geometric aggregations of local representations.

Table 3: Ablation Study on the number of nearest neighbors and different combinations of global and local aggregations. For the number of nearest neighbors, only the results of three commonly used K are given for comparison. For the column of aggregation, cones and inverted cones are tested separately.

Nearest Neighbor # (K)			Aggregation			Overall
16	32	64	Global	Cone	Inverted Cone	Accuracy (%)
	✓		✓			72.37
	✓			✓		83.75
	✓				✓	87.1
	✓		✓	✓		85.94
	✓		✓		✓	88.49
	✓			✓	✓	89.75
	✓		✓	✓	✓	**90.64**
✓			✓	✓	✓	89.14
		✓	✓	✓	✓	89.79

4.3 ScanObjectNN Dataset

We conduct experiments on a real world dataset, ScanObjectNN with object only, to show the generizability and effectiveness of Green-PointHop in this subsection. For fair comparison, we perform the same data augmentation, including random rotation and per-point jitter, as done in [46]. We compare the classification accuracy results of several DL-based methods and Green-PointHop in Table 4. Green-PointHop achieves 77.5% per-class accuracy and 79.3% overall accuracy. These numbers are comparable with PointNet and SpiderCNN. Green-PointHop has a performance gap against more advanced DL-based methods, ranging from 5% to 6.9% in the overall classification accuracy. Since the discussion on model/time complexities in Sec. 4.2.2 still holds here, Green-PointHop offers a different tradeoff between complexities and effectiveness.

We have an interesting observation by comparing the performance of PointNet and Green-PointHop. The per-class accuracy of Green-PointHop is 3.1% higher than that of PointNet while their overall accuracy results are about the same. It means that Green-PointHop has more balanced performance across different classes. To check this, we compare the per-class accuracy of Green-PointHop and PointNet in Table 5. Green-PointHop achieves higher accuracy in 11 classes and lower accuracy in the remaining 4 classes (cabinet, desk, door, and table).

To understand it further, we show some objects in the ScanObjectNN dataset with ground truth (GT) labels and predictions made by PointNet and Green-PointHop in Fig. 4, where labels and predictions are annotated for error

Table 4: Comparison of classification results on ScanObjectNN.

Method	Accuracy (%)	
	class-avg	overall
3DmFV [51]	68.9	73.8
PointNet [1]	74.4	79.2
SpiderCNN [52]	77.4	79.5
PointNet++ [7]	82.1	84.3
PointCNN [6]	83.3	85.5
DGCNN [48]	84.0	86.2
Green-PointHop (Ours)	77.5	79.3

Table 5: Comparison of per-class classification accuracy of PointNet and Green-PointHop on the ScanObjectNN dataset.

Method	Bag	Bin	Box	Cabinet	Chair
PointNet	47.1	80.0	35.7	**80.0**	93.6
Green-PointHop (Ours)	**70.6**	**90.0**	**46.4**	65.3	**96.2**
	Desk	Display	Door	Shelf	Table
PointNet	**86.7**	73.8	**97.6**	81.6	**88.9**
Green-PointHop (Ours)	56.7	**83.3**	92.9	**83.7**	79.6
	Bed	Pillow	Sink	Sofa	Toilet
PointNet	59.1	76.2	54.2	85.7	76.5
Green-PointHop (Ours)	**81.8**	**81.0**	**58.3**	**88.1**	**88.2**

analysis. We show six examples for each of the following four cases: 1) both predict correctly (the top group); 2) only PointNet predicts correctly (the second group); 3) only Green-PointHop predicts correctly (the third group); 4) neither predicts correctly (the bottom group).

Visualization of these objects helps us understand the challenges of the real world dataset. One main obstacle is that the shape of many objects is incomplete (with partial surfaces only). These incomplete objects are difficult to classify as shown in Fig. 4. For example, the door located in the bottom left corner of the figure has only some borders. Even humans have difficulty in recognizing it. Also, we observe that door, cabinet and display form a confusing group. They share a similar geometric shape – a flat surface. Only some details on surfaces or edges can make them distinctive. However, these discriminant parts could be missing.

5 Conclusion and Future Work

An extremely lightweight machine learning model, called Green-PointHop, was proposed for point cloud object classification in this work. Green-PointHop simplifies the PointHop model by reducing the hop number from four to one. To enrich the representations of a point cloud object, Green-PointHop exploited aggregated features obtained from points in the whole object and its partial views. This new idea has a clear geometric interpretation. Furthermore, its model size and computation complexity can be significantly reduced since it has filtering operations at a single scale. Experiments on ModelNet40 and ScanObjectNN datasets showed that the classification performance of Green-PointHop is comparable with PointNet and PointNet++.

There are several research topics worth future exploration. First, it is interesting to extend Green-PointHop to solve the point cloud segmentation and registration problems. Second, it is important to have a better understanding on the pooling operation among spatial points and aggregations of filter responses across points in spatial regions. The understanding will help guide the selection of optimal hyper-parameters of Green-PointHop.

Acknowledgment

This work was supported by a research grant from Tencent. The authors acknowledge the Center for Advanced Research Computing (CARC) at the University of Southern California for providing computing resources that have contributed to the research results reported in this publication.

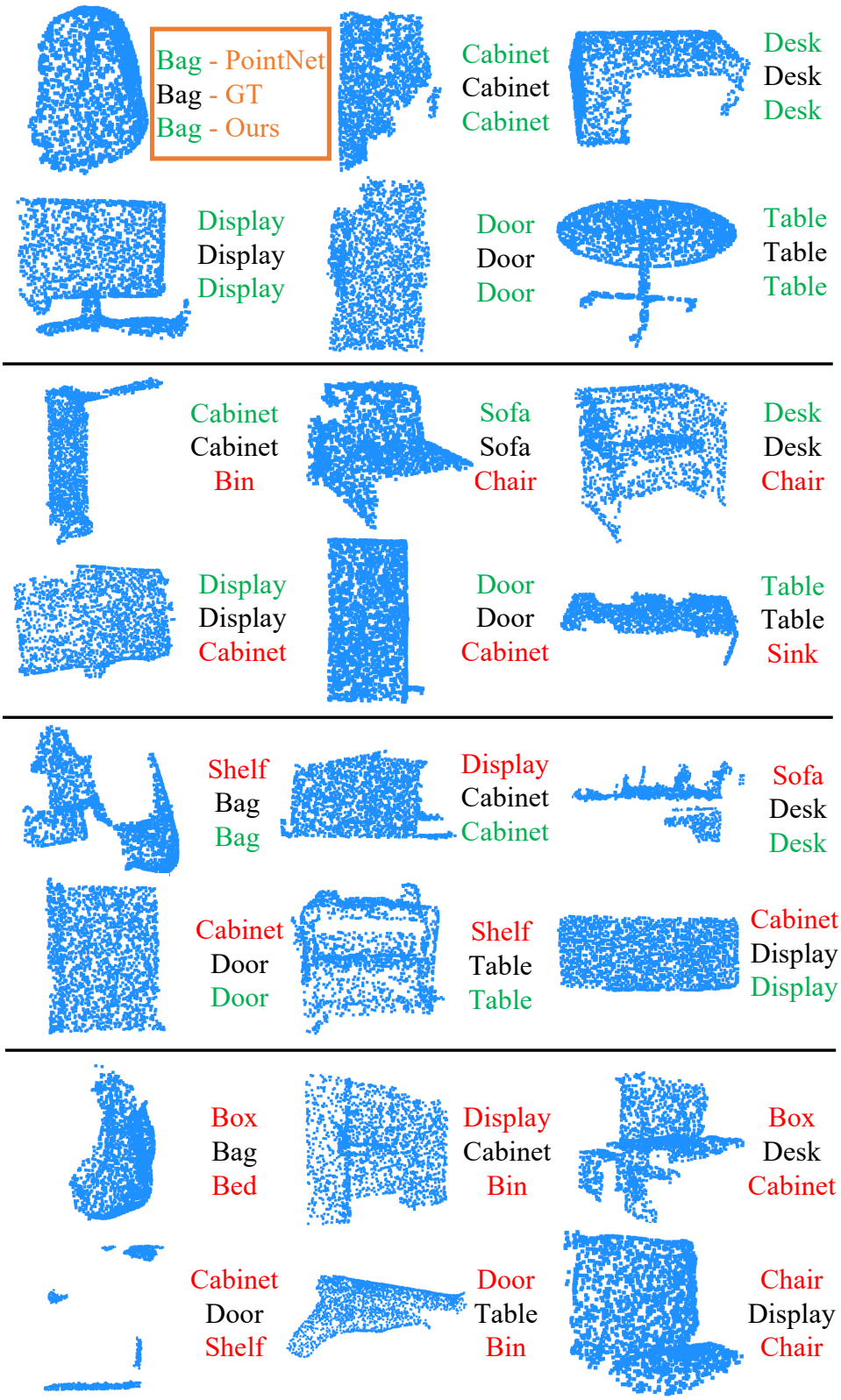

Figure 4: Prediction error analysis for ScanObjectNN. Each object is visualized with its GT label in the middle while the labels predicted by PointNet and Green-PointHop are, respectively, shown above and below for comparison. Also, wrong and correct predictions are shown in red and green, respectively.

References

[1] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 652–660, 2017.

[2] C-C Jay Kuo and Azad M Madni. Green learning: Introduction, examples and outlook. In *Journal of Visual Communication and Image Representation*, page 103685. Elsevier, 2022.

[3] Min Zhang, Haoxuan You, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Pointhop: An explainable machine learning method for point cloud classification. *IEEE Transactions on Multimedia*, 22(7):1744–1755, 2020.

[4] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[5] Mingyang Jiang, Yiran Wu, Tianqi Zhao, Zelin Zhao, and Cewu Lu. Pointsift: A sift-like network module for 3d point cloud semantic segmentation. *arXiv preprint arXiv:1807.00652*, 2018.

[6] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[7] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[8] 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)*, 38(5):1–12, 2019.

[9] 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*, 7(2):187–199, 2021.

[10] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021.

[11] 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 *Proceedings of the IEEE International Conference on Computer Vision and Pattern Recognition*, 2016.

[12] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

[13] Min Zhang, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Unsupervised feedforward feature (uff) learning for point cloud classification and segmentation. In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 144–147. IEEE, 2020.

[14] Pranav Kadam, Min Zhang, Shan Liu, and C-C Jay Kuo. Unsupervised point cloud registration via salient points analysis (spa). In *2020 IEEE International Conference on Visual Communications and Image Processing (VCIP)*, pages 5–8. IEEE, 2020.

[15] Min Zhang, Yifan Wang, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Pointhop++: A lightweight learning model on point sets for 3d classification. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3319–3323. IEEE, 2020.

[16] Radu Bogdan Rusu, Nico Blodow, and Michael Beetz. Fast point feature histograms (fpfh) for 3d registration. In *2009 IEEE international conference on robotics and automation*, pages 3212–3217. IEEE, 2009.

[17] Yu Zhong. Intrinsic shape signatures: A shape descriptor for 3d object recognition. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, pages 689–696. IEEE, 2009.

[18] Federico Tombari, Samuele Salti, and Luigi Di Stefano. Unique signatures of histograms for local surface description. In *European conference on computer vision*, pages 356–369. Springer, 2010.

[19] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[20] Yueru Chen and C-C Jay Kuo. Pixelhop: A successive subspace learning (ssl) method for object recognition. *Journal of Visual Communication and Image Representation*, 70:102749, 2020.

[21] Pranav Kadam, Hardik Prajapati, Min Zhang, Jintang Xue, Shan Liu, and C-C Jay Kuo. S3I-PointHop: SO(3)-Invariant pointhop for 3d point cloud classification. *arXiv preprint arXiv:2302.11506*, 2023.

[22] Min Zhang, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Gsip: Green semantic segmentation of large-scale indoor point clouds. *Pattern Recognition Letters*, 164:9–15, 2022.

[23] Pranav Kadam, Min Zhang, Shan Liu, and C-C Jay Kuo. R-pointhop: A green, accurate, and unsupervised point cloud registration method. In *IEEE Transactions on Image Processing*. IEEE, 2022.

[24] Pranav Kadam, Min Zhang, Jiahao Gu, Shan Liu, and C-C Jay Kuo. Greenpco: An unsupervised lightweight point cloud odometry method. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 01–06. IEEE, 2022.

[25] Pranav Kadam, Qingyang Zhou, Shan Liu, and C-C Jay Kuo. Pcrp: Unsupervised point cloud object retrieval and pose estimation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 1596–1600. IEEE, 2022.

[26] Pranav Kadam, Jiahao Gu, Shan Liu, and C-C Jay Kuo. Pointflowhop: Green and interpretable scene flow estimation from consecutive point clouds. *arXiv preprint arXiv:2302.14193*, 2023.

[27] Yueru Chen, Mozhdeh Rouhsedaghat, Suya You, Raghuveer Rao, and C-C Jay Kuo. Pixelhop++: A small successive-subspace-learning-based (ssl-based) model for image classification. In *2020 IEEE International Conference on Image Processing (ICIP)*, pages 3294–3298. IEEE, 2020.

[28] Yijing Yang, Vasileios Magoulianitis, and C-C Jay Kuo. E-pixelhop: An enhanced pixelhop method for object classification. In *2021 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1475–1482. IEEE, 2021.

[29] Yueru Chen, Zhuwei Xu, Shanshan Cai, Yujian Lang, and C-C Jay Kuo. A saak transform approach to efficient, scalable and robust handwritten digits recognition. In *2018 Picture Coding Symposium (PCS)*, pages 174–178. IEEE, 2018.

[30] C-C Jay Kuo. Understanding convolutional neural networks with a mathematical model. *Journal of Visual Communication and Image Representation*, 41:406–413, 2016.

[31] C-C Jay Kuo and Yueru Chen. On data-driven saak transform. *Journal of Visual Communication and Image Representation*, 50:237–246, 2018.

[32] C-C Jay Kuo, Min Zhang, Siyang Li, Jiali Duan, and Yueru Chen. Interpretable convolutional neural networks via feedforward design. *Journal of Visual Communication and Image Representation*, 2019.

[33] Mozhdeh Rouhsedaghat, Yifan Wang, Xiou Ge, Shuowen Hu, Suya You, and C C Jay Kuo. Facehop: A light-weight low-resolution face gender classification method. In *Pattern Recognition. ICPR International Workshops and Challenges: Virtual Event, January 10-15, 2021, Proceedings, Part VIII*, pages 169–183. Springer, 2021.

[34] Hong-Shuo Chen, Mozhdeh Rouhsedaghat, Hamza Ghani, Shuowen Hu, Suya You, and C-C Jay Kuo. Defakehop: A light-weight high-performance deepfake detector. In *2021 IEEE International Conference on Multimedia and Expo (ICME)*, pages 1–6. IEEE, 2021.

[35] Kaitai Zhang, Bin Wang, Wei Wang, Fahad Sohrab, Moncef Gabbouj, and C-C Jay Kuo. Anomalyhop: an ssl-based image anomaly localization method. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–5. IEEE, 2021.

[36] Xuejing Lei, Wei Wang, and C-C Jay Kuo. Genhop: An image generation method based on successive subspace learning. In *2022 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 3314–3318. IEEE, 2022.

[37] Xuejing Lei, Ganning Zhao, and C-C Jay Kuo. Nites: A non-parametric interpretable texture synthesis method. In *2020 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)*, pages 1698–1706. IEEE, 2020.

[38] Xuejing Lei, Ganning Zhao, Kaitai Zhang, and C-C Jay Kuo. Tghop: an explainable, efficient, and lightweight method for texture generation. *APSIPA Transactions on Signal and Information Processing*, 10:e17, 2021.

[39] Zhiruo Zhou, Hongyu Fu, Suya You, Christoph C Borel-Donohue, and C-C Jay Kuo. Uhp-sot: An unsupervised high-performance single object tracker. In *2021 International Conference on Visual Communications and Image Processing (VCIP)*, pages 1–5. IEEE, 2021.

[40] Zhiruo Zhou, Hongyu Fu, Suya You, and C-C Jay Kuo. Gusot: Green and unsupervised single object tracking for long video sequences. In *2022 IEEE 24th International Workshop on Multimedia Signal Processing (MMSP)*, pages 1–6. IEEE, 2022.

[41] Zhiruo Zhou, Hongyu Fu, Suya You, C-C Jay Kuo, et al. Uhp-sot++: An unsupervised lightweight single object tracker. *APSIPA Transactions on Signal and Information Processing*, 11(1), 2022.

[42] Zhiruo Zhou, Hongyu Fu, Suya You, and C-C Jay Kuo. Unsupervised lightweight single object tracking with uhp-sot++. *arXiv preprint arXiv:2111.07548*, 2021.

[43] Shan Liu, Min Zhang, Pranav Kadam, and Chung-Chieh Jay Kuo. *3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods*. Springer, 2021.

[44] Greg Turk and Marc Levoy. Zippered polygon meshes from range images. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 311–318, 1994.

[45] Yijing Yang, Wei Wang, Hongyu Fu, C-C Jay Kuo, et al. On supervised feature selection from high dimensional feature spaces. In *APSIPA Transactions on Signal and Information Processing*, volume 11. Now Publishers, Inc., 2022.

[46] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

[47] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9621–9630, 2019.

[48] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *arXiv preprint arXiv:1801.07829*, 2018.

[49] 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 *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6411–6420, 2019.

[50] Can Chen, Luca Zanotti Fragonara, and Antonios Tsourdos. Go wider: An efficient neural network for point cloud analysis via group convolutions. *Applied Sciences*, 10(7):2391, 2020.

[51] Yizhak Ben-Shabat, Michael Lindenbaum, and Anath Fischer. 3dmfv: Three-dimensional point cloud classification in real-time using convolutional neural networks. *IEEE Robotics and Automation Letters*, 3(4):3145–3152, 2018.

[52] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European conference on computer vision (ECCV)*, pages 87–102, 2018.