PatchAugment: Local Neighborhood Augmentation in Point Cloud Classification

Shivanand Venkanna Sheshappanavar, Vinit Veerendraveer Singh and Chandra Kambhamettu
Video/Image Modeling and Synthesis (VIMS) Lab., Dept. of Computer and Information Sciences
University of Delaware, Newark, DE, USA 19716
{ssheshap, vinitvs, chandrak}@udel.edu

Abstract

Recent deep neural network models trained on smaller and less diverse datasets use data augmentation to alleviate limitations such as overfitting, reduced robustness, and lower generalization. Methods using 3D datasets are among the most common to use data augmentation techniques such as random point drop, scaling, translation, rotations, and jittering. However, these data augmentation techniques are fixed and are often applied to the entire object, ignoring the object's local geometry. Different local neighborhoods on the object surface hold a different amount of geometric complexity. Applying the same data augmentation techniques at the object level is less effective in augmenting local neighborhoods with complex structures. This paper presents PatchAugment, a data augmentation framework to apply different augmentation techniques to the local neighborhoods. Our experimental studies on PointNet++ and DGCNN models demonstrate the effectiveness of PatchAugment on the task of 3D Point Cloud Classification. We evaluated our technique against these models using four benchmark datasets, ModelNet40 (synthetic), ModelNet10 (synthetic), SHREC'16 (synthetic) and ScanObjectNN (real-world).

1. Introduction

3D Computer Vision is an active research area with broad applications in augmented/virtual reality, robotic perception, 3D shape designs, and autonomous driving vehicles. Over the past few years, deep neural networks' application to 3D Computer Vision has evolved tremendously, especially for 3D point cloud processing. These deep neural network models have achieved state-of-the-art results on several computer vision tasks such as 3D Semantic Segmentation, 3D Scene Understanding, 3D Shape Retrieval, and 3D Object Classification [1, 2, 3, 4, 5, 6]. One important reason behind these promising results is the availability of 3D datasets to train deep neural networks. Most 3D datasets are available as CAD models requiring some preprocessing.

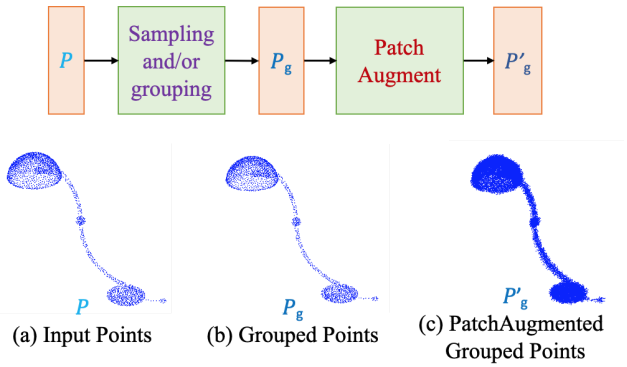

Figure 1: **Where does PatchAugment fit in?** Deep neural networks whose layers contain neighborhood querying step(s) can plugin PatchAugment. After sampling of points from the input, neighborhood points are queried and grouped. Note: (c) has 12288 points which is equivalent to 12 whole objects with 1024 points in each. The sample object is a lamp from ModelNet40 dataset.

Although several 3D datasets [7, 8, 9, 10, 11, 12, 13, 14, 15] as listed in Table 1 are available to train deep learning models, the datasets are not close enough in scale when compared to 2D Datasets (e.g. ImageNet [16]). Table 1 also lists the official split of the datasets into training and testing sets. Existing 3D datasets are limited by the number of samples (ranging from 500 to 50,000) and have a small number of classes (10 to 55). Along with these limitations existing datasets are predominantly synthetic datasets [7, 10, 12, 13, 14, 15]. In recent years, a few datasets from the real-world scenarios [8, 9, 11] have been released. When trained on synthetic datasets and tested using more complex real-world datasets, deep learning networks show a significant drop in accuracy compared to the reported results. As mentioned earlier, the limitations lead to major drawbacks such as overfitting, lesser generalizability, and lack of robustness while evaluating neural networks for point cloud analysis.

Table 1: Details of datasets for 3D object classification along with the year of publication. PCs = Point Clouds, S = Synthetic, RW = Real-World, GCN = Graph Convolutional Networks, CNN = Convolutional Neural Networks, FDG = Faster Dynamic Graph, SVM = Support Vector Machine, and DI = Dual Input.

Dataset	Year	Classes	Samples	Train	Test	Type	Format	Acc.	Model
McGill Benchmark [13]	2008	19	458	190	268	S	Mesh	83.7	SVM [17]
Syndney Urban [9]	2013	14	588	-	-	RW	PCs	-	-
ModelNet10 [10]	2015	10	4899	3991	605	S	Mesh	97.5	Grid-GCN [18]
ModelNet40 [10]	2015	40	12311	9843	2468	S	Mesh	94.2	CurveNet [19]
ShapeNet [7]	2015	55	51190	-	-	S	Mesh	-	-
SHREC16 [14]	2017	55	51162	35764	10265	S	PCs	90.9	View-GCN [20]
ScanNet [8]	2017	17	12283	9677	2606	RW	PCs	85.2	Sim2real [21]
ScanObjectNN [11]	2019	15	2902	2321	581	RW	PCs	81.3	DI-PointCNN [22]
3D-Future [12]	2020	34	9992	6699	3293	S	Mesh	73.7	AttWalk [23]
RobustPointSet [15]	2020	40	12308	9840	2468	S	PCs	92.5	DGCNN [5]

When faced with the limited availability of samples in datasets, Data Augmentation (DA) is one of the most common techniques researchers use to tackle the associated problems. DA avoids overfitting, alleviates class imbalance problems, makes deep neural networks more robust, and achieves better generalization. DA also diversifies and enlarges the datasets, enabling the deep learning models to learn better from the dataset. However, existing models apply fixed data augmentations to entire objects and make this dataset enlargement less effective. For example, for an object with fewer or no local complex curvatures, a random scaling of the object would have the same effect on all local structures. However, at an object level, random fixed scaling would not be enough to enlarge its complex local neighborhoods in effectively capturing local geometry. Hence, Conventional DA techniques used in training existing state-of-the-art deep learning network models are not sufficient for effective training.

This paper focuses on augmenting each sample's different local neighborhoods. We enhance the overall augmentation of the sample object, thereby improving the learning of local geometry by the deep neural network to achieve better 3D Classification. Figure 1 illustrates the set abstraction level of a single scale grouping PointNet++ [2] model. The number of unique points increases by several folds after applying PatchAugment. Check for the points from the grouping layer in Figure 1(b) and the points after patch augmentation in Figure 1(c). Figure 1(c) shows the visual robustness of PatchAugment.

Our PatchAugment augments the neighborhood points as follows; first, it randomly drops a small fraction of the queried neighborhood points. Second, it randomly scales these neighborhood points, followed by a random perturb rotation by small angles and a random rotation along the up-axis direction. Then, it randomly translates each augmented neighborhood group. Note that this random translation is common to all points within a patch neighborhood but different for points belonging to different neighboring patches/grouped points. Finally, it adds random jitter to the points at the individual point level. Jittering makes points to move randomly within a neighborhood space. Although these six types of augmentations are the same as conventional DA, they are not fixed and are applied to each grouped neighborhood.

Our key contributions in this paper are:

- **PatchAugment:** We augment each local neighborhood with random points drop, random scale, random rotations, random translation, and random jitter.
- **Impact of PatchAugment:** We show the effectiveness of PatchAugment on PointNet++ [2] and DGCNN [5].
- **Evaluation:** We evaluate our method of patch augmentation using synthetic (ModelNet40 [10], ModelNet10 [10], and SHREC'16 [14]) and real-world (ScanObjectNN [11]) datasets.

2. Related Works

Applying data augmentation techniques to datasets (2D and 3D) is a rich and extensive area of research.

2.1. Data Augmentation on 2D Datasets

Random transformation [24, 25] of input samples is a well-known DA technique for 2D data. Another popular DA technique is interpolation [26, 27, 28] of images within the image feature space using simple transformations. Generative Adversarial Networks (GANs) based methods [29, 30] generate augmented samples using the input data. However, the methods mentioned above produce data different from the original data resulting in unreliable samples. Other

interpolation-based techniques [24, 25, 31] involve pixel-wise interpolation for images. Particularly, Smart Augmentation [31] generates augmented data by merging samples from the same class. Unlike their application on 2D images with the regular grid structure, these methods cannot be applied to point clouds because of their unordered property and irregular structure.

MixMatch [32] guesses low-entropy labels for data-augmented unlabeled examples. It mixes labeled and unlabeled data using MixUp [25]. Between-Class Learning (BCL) [33] generates between-class images by mixing two images belonging to different classes with a random ratio. The primary argument in BCL is that CNN's treat input data as waveforms; since BCL works on sound datasets, they infer that BCL must also work on images. AutoAugment [34] generates symbolic transformation operations by using reinforcement learning to learn DA policies from the data. However, for large-scale problems, AutoAugment is not computationally practical. Based on density matching, Fast AutoAugment [35] finds effective augmentation policies via a more efficient search strategy. For augmentation, a few hyper-parameter optimization methods [35, 36] though see best transformations, they are limited to finding a fixed augmentation strategy for all training samples.

2.2. Data Augmentation on 3D Datasets

3D datasets or representations are four types; voxels, multi-view camera projections, meshes, and point clouds. While voxels suffer from insufficient resolution and memory costs, the meshes are not directly acquired from 3D sensors. Multi-view camera projections are 2D images of the 3D object from multiple views. Both PointNet [1] and PointNet++ [2] use the same DA techniques of random rotation about the up-axis scaling, random rotation with perturbations, random shifting, and random jittering of points on the input object sample. Several state-of-the-art models such as RS-CNN [6], DensePoint [4] followed similar DA strategies with minor variations.

Recent, PointMixUp [37] a new model-agnostic data augmentation method uses interpolation on 3D dataset for point cloud augmentation. PointAugment [38] a new auto-augmentation framework for 3D point clouds jointly optimizes augmentation along with the classification network for training. PointAugment [38] produces transformation functions based on the properties of individual training samples and the network capability during the training process.

Patch-based DA methods for 2D have boosted performance. Part-aware [39], first extends 2D image patch to 3D partitions and then extend 3D partition to 3D point clouds. Although, Part-aware [39] applies five different types of DAs to different partitions, makes the network robust, improves performance, its application is unexplored beyond 3D object detection.

2.3. Deep learning on point cloud

PointNet [1] applied data augmentation at the object level while processing raw point sets to extract features. However, PointNet ignores local patterns losing semantic information. PointNet++ [2] follows steps of PointNet in augmenting the input, but partitions point sets producing common hierarchical structures and applies pointnet recursively to learn contextual representation. The neighborhood's contextual information though exploited in case of PointNet++, it was limited by the max-pooling operations. PointCNN [40] along with random scaling and point shuffling uses $\mathcal{X}$-operator to transform the input points while weighing and permuting the input features associated with the points generalizing CNNs. With non-uniform sampling PointConv [41] performs convolution on 3D pointsets and applies inverse density scale on learned weights to overcome the effect of non-uniform sampling. Unlike the above works, we employ data augmentation at neighborhood levels by augmenting the points captured from kNN querying before feeding them into MLPs. These augmentations are not fixed and differ from neighborhood to neighborhood. Our method results in multiple combinations of the original object instead of a single augmented object achieving significant augmentation at the object level.

3. Method

We use PointNet++ [2] model with single scale grouping as the baseline model. PatchAugment is straightforward and can be plugged into models with the grouping of points at the neighborhood level. The placement of PatchAugment in such models can be made right after the grouping step as depicted in Figure 1. In patch augmentation a sample object as shown in Figure 2(a) undergoes a series of DA techniques at patch level, i.e., random points dropout ($\boldsymbol{\lambda}$), random scale ($\boldsymbol{S}$), random rotations (both perturbed $\boldsymbol{R}$ and up-axis rotations $\boldsymbol{R_\theta}$), random translation ($\boldsymbol{T}$), and random jittering ($\boldsymbol{J}$) as shown in part (b) to (g) of Figure 2 respectively.

Our novel PatchAugment applies the DA techniques mentioned above to each neighborhood obtained after grouping points in a set abstraction level setting (particularly in models that borrow the sampling and grouping steps into their model architectures). In this paper we do not propose a new deep neural network model but introduce the PatchAugment technique (algorithm 1) into models after neighborhood querying to boost 3D classification accuracy. PatchAugment is straightforward to plug it in the models similar to PointNet++ or the models that group neighborhood points in the first or subsequent layers.

Figure 2 shows application of PatchAugment technique to queried neighborhood points obtained at the grouping layer in a set abstraction level of PointNet++. Neighborhood points can be queried by k-NN or ball querying as

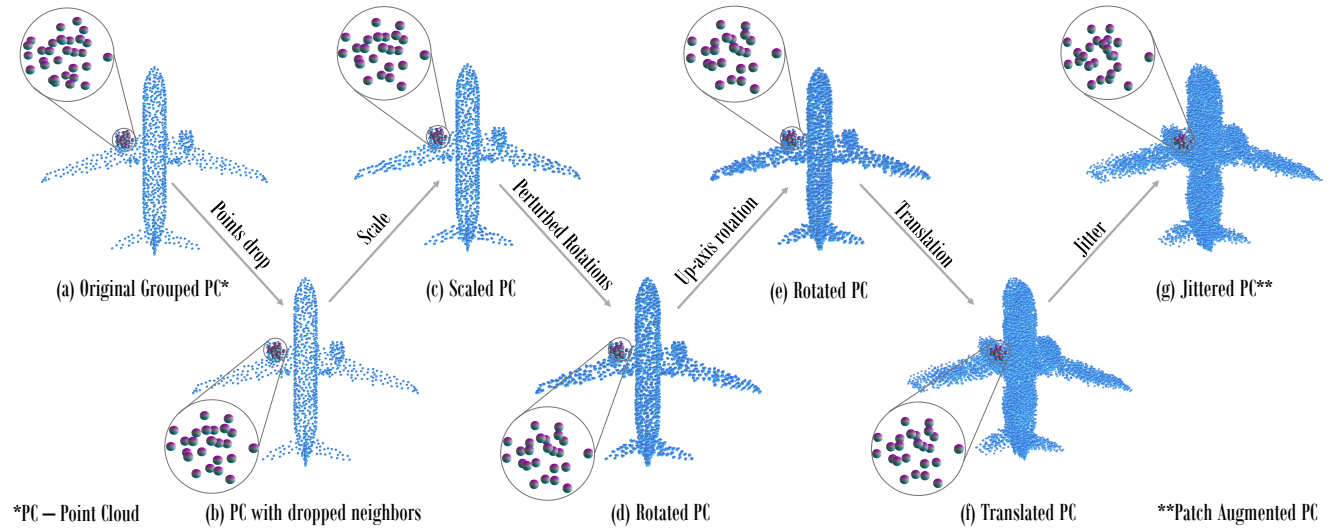

Figure 2: PatchAugment module takes (a) original grouped points as input and applies (b) random point drop, (c) random scale, (d & e) random rotations (perturbed and up-axis), (f) random translation, and (e) random jitter augmentations to each of the neighborhood/group differently to generate augmented grouped points. Note: The points are not queried at each step. They are the same points queried before (a) but are augmented at each step. [Zoom to view better.]

mentioned in [2] or by ellipsoid querying as mentioned in [42]. We denote the output of the grouping layer by $\boldsymbol{P_g} \in \mathbb{R}^{m \times k \times 3}$. Here m denotes the number of farthest sampled points from the input in the sampling layer. k represents the number of points queried around each of the farthest sampled points. Hence, there are m groups with k points in each group. For each of the queried neighborhood group points of m farthest sampled points, the PatchAugment technique applies random point drop. Then, it performs random scaling, rotations, translation, and jitter on the remaining group points, augmenting the object at the neighborhood level. The resulting patch augmented neighboring points are denoted by $\boldsymbol{P'_g}$.

For the sake of reference, we denote the grouped point cloud in each step of Figure 2 (a-g) by notations as described next. Observe changes to both the object and the patch in Figure 2. The object changes significantly due to neighborhood augmentations, while the changes to the patch appear very small to negligible. $\boldsymbol{P_g}$ represents original grouped points from the grouping layer. They visually appear as just 1024 unique input points as shown in Figure 2(a). $\boldsymbol{P_{gd}}$ represents grouped points after points drop at the neighborhood level as shown in Figure 2(b)(different to points drop at the object level, visually still appear as 1024 points due to overlapping neighborhoods). $\boldsymbol{P_{gds}}$ represents scaled group points. The points in each neighborhood are scaled with different small scale factors altering the positions of points slightly. Each point, due to its presence in multiple neighborhoods, is scaled separately within that neighborhood, resulting in smaller lumps of points around each point as shown in Figure 2(c). $\boldsymbol{P_{gdsr}}$ represents points after perturbed rotations along all axes as shown in Figure 2(d). $\boldsymbol{P_{gdsr\theta}}$ represents points after up-axis rotations as shown in Figure 2(e). A significant change in the points' positions is visible at the object level due to patch augmentation. The rotations further move the scaled points around their original position. $\boldsymbol{P_{gdsr\theta t}}$ represents translated grouped points as shown in Figure 2(f). $\boldsymbol{P'_{gd}}$ represents jittered grouped points (also, the final Patch Augmented points) as shown in Figure 2(g). Both translation and jitter displace the points randomly, resulting in a denser point cloud with augmented neighborhoods.

We use a random drop factor of $\boldsymbol{\lambda}$ to drop the grouped points $\boldsymbol{P_g}$ to get grouped points after dropping $\boldsymbol{P_{gd}}$ (Eq. 1).

$$\boldsymbol{P_{gd}} = Drop(\boldsymbol{P_g}, \boldsymbol{\lambda}), \tag{1}$$

These points $\boldsymbol{P_{gd}}$ are scaled using different scale factors $\boldsymbol{S} \in \mathbb{R}^{m \times 1 \times 1}$ to get $\boldsymbol{P_{gds}}$ (Eq. 2).

$$\boldsymbol{P_{gds}} = \boldsymbol{P_{gd}} \times \boldsymbol{S} \tag{2}$$

The perturbed rotation matrix $\boldsymbol{R}$ [43](Eq. 6) is formed by the multiplication of rotation matrices for rotation along x, y, z axes represented by Eq. 3, Eq. 4 and Eq. 5 respectively. The $\boldsymbol{P_{gds}}$ points are rotated using the rotation matrix $\boldsymbol{R}$ to get perturbed rotated grouped points $\boldsymbol{P_{gdsr}}$ (Eq. 7) . Rotation angles α, β, γ are used for rotations along x, y, z axes

respectively.

$$\boldsymbol{R_x}= \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\alpha & -sin\alpha \\ 0 & sin\alpha & cos\alpha \end{bmatrix} \tag{3}$$

$$\boldsymbol{R_y}= \begin{bmatrix} cos\beta & 0 & sin\beta \\ 0 & 1 & 0 \\ -sin\beta & 0 & cos\beta \end{bmatrix} \tag{4}$$

$$\boldsymbol{R_z}= \begin{bmatrix} cos\gamma & -sin\gamma & 0 \\ sin\gamma & cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{5}$$

$$\boldsymbol{R} = (\boldsymbol{R_z} * \boldsymbol{R_y} * \boldsymbol{R_x}) \tag{6}$$

$$\boldsymbol{P_{gdsr}} = (\boldsymbol{R} * \boldsymbol{P_{gds}^T})^T \tag{7}$$

These $\boldsymbol{P_{gdsr}}$ points are rotated along the up-axis by an angle θ using up-axis rotation matrix $\boldsymbol{R_\theta}$ (Eq. 8) to get $\boldsymbol{P_{gdsr\theta}}$ (Eq. 9). Here, $\boldsymbol{R}$, $\boldsymbol{R_x}$, $\boldsymbol{R_y}$, $\boldsymbol{R_z}$, $\boldsymbol{R_\theta} \in \mathbb{R}^{m\times3\times3}$. Rotations are performed as per the right-hand rule.

$$\boldsymbol{R_\theta}= \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{8}$$

$$\boldsymbol{P_{gdsr\theta}} = (\boldsymbol{R_\theta} * \boldsymbol{P_{gdsr}^T})^T \tag{9}$$

The $\boldsymbol{P_{gdsr\theta}}$ points with m groups are translated using $\boldsymbol{T_r} \in \mathbb{R}^{m\times1\times1}$ (Eq. 10) to get $\boldsymbol{P_{gdsr\theta t}}$ points.

$$\boldsymbol{P_{gdsr\theta t}} = \boldsymbol{P_{gdsr\theta}} + \boldsymbol{T_r}, \tag{10}$$

The jitter factor $\boldsymbol{J} \in \mathbb{R}^{m\times1\times3}$ is added to $\boldsymbol{P_{gdsr\theta t}}$ (Eq. 11) to get final patch augmented group points represented by $\boldsymbol{P'}$.

$$\boldsymbol{P'} = \boldsymbol{P_{gdsr\theta t}} + \boldsymbol{J}, \tag{11}$$

Algorithm 1 PatchAugment

function PATCHAUGMENT($\boldsymbol{P_g}$)
▷ Input: grouped points $\boldsymbol{P_g}$
▷ Output: augmented grouped points $\boldsymbol{P'_g}$
▷ Size of $\boldsymbol{P_g} \rightarrow B, m, k, 3$
▷ Size of $\boldsymbol{P'_g} \rightarrow B, m, k', 3$ ▷ $k' = (1 - \boldsymbol{\lambda}) * k$
▷ batch size, FPS points, sampled points, 3D co-ordinates
$\boldsymbol{S}, \alpha, \beta, \gamma, \theta, \boldsymbol{T}, \boldsymbol{J} \leftarrow uniformsampler()$
$\boldsymbol{P_{gd}} \leftarrow Drop(\boldsymbol{P_g}, \boldsymbol{\lambda})$ ▷ $\boldsymbol{\lambda}$ constant dropout ratio
$\boldsymbol{P_{gds}} \leftarrow \boldsymbol{P_{gd}} \times \boldsymbol{S}$ ▷ $\times$ is scalar mul
$\boldsymbol{R} \leftarrow \boldsymbol{R_z}(\gamma)\boldsymbol{R_y}(\beta)\boldsymbol{R_x}(\alpha)$
$\boldsymbol{P_{gdsr}} \leftarrow (\boldsymbol{R} * \boldsymbol{P_{gds}^T})^T$ ▷ * is matmul, T is transpose
$\boldsymbol{P_{gdsr\theta}} \leftarrow (\boldsymbol{R_\theta} * \boldsymbol{P_{gdsr}^T})^T$
$\boldsymbol{P_{gdsr\theta t}} \leftarrow \boldsymbol{P_{gdsr\theta}} + \boldsymbol{T_r}$ ▷ $\boldsymbol{T_r}$ is translation factor
$\boldsymbol{P'_g} \leftarrow \boldsymbol{P_{gdsr\theta t}} + \boldsymbol{J}$
return $\boldsymbol{P'_g}$
end function

4. Experiments

In this section, we discuss the use of k-NN querying in our experiments instead of ball querying, neighborhood augmentation parameters, the benchmark datasets used in our experiments, and other training-testing details.

4.1. Ball query vs. k-NN query for PatchAugment

Ball query restricts querying of points within the ball volume of radius r, and in many cases, ball querying does not fetch the specified k number of points from the neighborhood. In such cases, the centroid point used for querying the neighborhood is repeated to fulfill the k points count to keep tensor shapes suitable for operations. However, these repeated centroid points upon patch augmentation result in lumps of closer points as shown in Figure 3(c) affecting the geometry of the local neighborhood. Unlike the ball, k-NN fetches k distinct neighborhood points, and patch augmentation on these points results in meaningful points within the neighborhood as shown in Figure 3(d).

4.2. Neighborhood Augmentation Parameters

To explore local neighborhood-level augmentation in our experiments, we used $\boldsymbol{\lambda}$ = 0.25 to drop 25% of the randomly selected points from the grouped neighborhood points. Small scale factor values are randomly sampled to scale the grouped neighborhood points, i.e., $\boldsymbol{S} \in$ [0.95, 1.05]. The rotation angles are in radians sampled from small ($\alpha, \beta, \gamma, \theta \in$ [-0.1, 0.1]) ranges to avoid larger distortions to neighborhood geometry. The translation factor values are sampled randomly from the range $\boldsymbol{T} \in$ [-0.05, 0.05]. Jitter factor values are sampled randomly from the range such that $\boldsymbol{J} \in$ [-0.01, 0.01]. The translation and jitter values are small to keep the grouped points within the neighborhoods. Also, from our ablation studies (shown later in Table 9) we found smaller ranges for these parameters were effective. These augmentation parameters are sampled randomly from the uniform distribution within the above-defined ranges.

4.3. Datasets

In our experiments, we have employed four 3D benchmark datasets, i.e., ModelNet40 [10], ModelNet10 [10], SHREC'16 [14], and ScanObjectNN [11] which we denote as MN40, MN10, SR16, and SONN respectively.. Among them ModelNet40 dataset [10], ModelNet10 [10], and SHREC'16 [14] datasets are synthetic datasets and the ScanObjectNN [11] dataset is a real-world dataset. ModelNet40 dataset has 40 categories of objects comprising a total of 12,311 shapes. The official split of ModelNet40 is 9843 and 2468 objects for training and testing respectively.

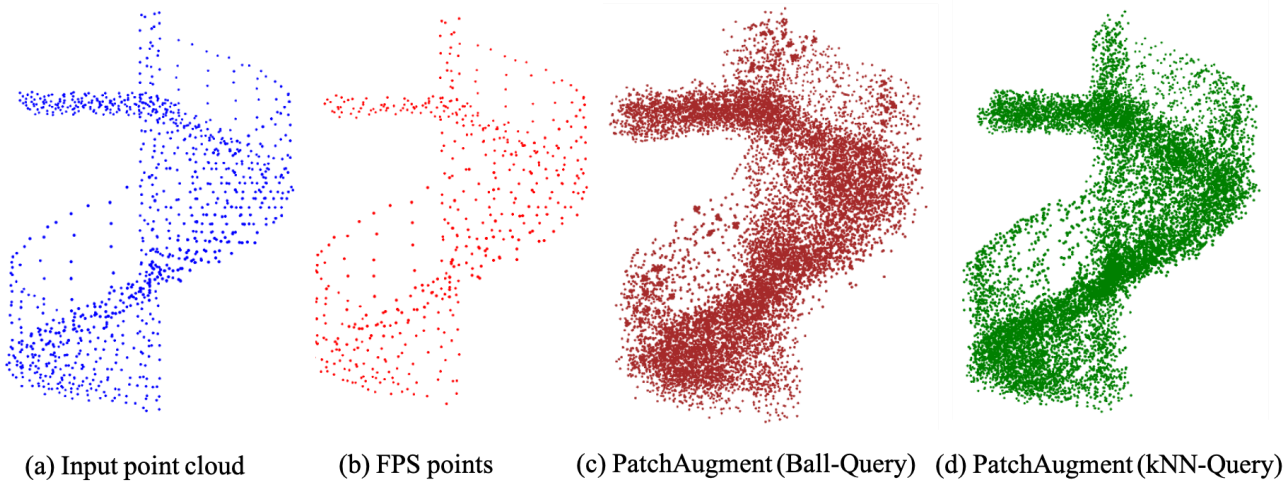

Figure 3: Difference between Ball-query-based Patch Augmentation and k-NN query-based Patch Augmentation. (a) Input point cloud (stairs), (b) Farthest Point Sampled (FPS) Points, (c) ball-queried and patch augmented points (lumps of points occur due to repetition of centroid point when ball-query falls short of the required number of points within the ball volume) and (d) k-NN queried, and patch augmented points (no lumps because k-NN always returns k points from the neighborhood).

We have retained this official split as in the training and testing of the PointNet++ model. ModelNet10 dataset has 10 categories (a subset of ModelNet40) of objects comprising a total of 4,899 shapes. The official split of the benchmark ModelNet10 dataset is 3991 objects for training and 908 objects for testing.

ScanObjectNN is a more complex dataset consisting of 2902 real-world objects spread across 15 classes. ScanObjectNN has 700 unique scenes developed from two popular scene object datasets i.e., SceneNN [44] and ScanNet [8] with 100 and 1513 objects respectively. From the several variants of ScanObjectNN dataset we have considered the six prominently used variants for our experiments i.e., OBJ_ONLY, OBJ_BG, PB_T25, PB_T25_R, PB_T50_R and PB_T50_RS. These variants represent different levels of difficulty due to background points and perturbations introduced into them. OBJ_ONLY variant is similar to the ModelNet40 dataset that is extracted from CAD models. OBJ_BG variant contains background elements and/or parts of objects from the object neighborhood. The remaining four variants prefixed with 'PB_' represent perturbations of objects with translation ($\boldsymbol{T}$), rotation ($\boldsymbol{R}$) and scaling ($\boldsymbol{S}$). PB_T25 variant has objects with 25% translation. PB_T25_R variant contains objects with 25% translation and involve rotations. PB_T50_R variant refers to objects with 50% translation and rotation. PB_T50_RS variant has objects with 50% translation along with rotations and scaling. Each perturbed variant consists of five randomly sampled objects from the ground truth objects enlarging ScanObjectNN dataset into 14,510 perturbed objects.

4.4. Training and Testing details

We have uniformly sampled 1024 points as input to the models in our experiments for each of the datasets mentioned earlier. We accommodate PatchAugment in [45] PointNet++ and single set abstraction PointNet++ models and trained for 200 epochs. DGCNN is trained for 400 epochs. For PointNet++ based models, we use a learning rate of 0.001, a batch size of 24, a dropout rate of 40%, and Adam optimizer during training, and we have evaluated these models with a majority voting of 12 votes. For DGCNN experiments, we follow the experimental setup as described in [5], i.e, a learning rate of 0.1 reduced by cosine annealing till 0.001, SGD optimizer, momentum 0.9 and a batch size of 32. For fair comparison with DGCNN results majority voting is not used in DGCNN with PatchAugment experiments. Our code is available at https://github.com/VimsLab/PatchAugment.git.

5. Experimental Results

We conducted extensive experiments with PatchAugment using a single scale grouping(SSG) PointNet++ [2], DGCNN [5] and single Set Abstraction-based SSG PointNet++ (demonstrated in [46] to improve 3D point cloud classification accuracy) models. Our experimental results constitute 3D classification accuracies of PointNet++ and DGCNN with PatchAugment on four benchmark datasets. i.e., ModelNet40 (synthetic), ModelNet10 (synthetic), SHREC'16 (synthetic), and ScanObjectNN (real-world).

Table 2: Improvement in 3D object classification accuracy by using PatchAugment technique in Pointnet++, DGCNN and PointNet++ with a single Set Abstraction layer models on CAD models based datasets i.e., SHREC16, ModelNet40, ModelNet10 and OBJ_ONLY variant of ScanObjectNN. Accuracies reported here are obtained by training the models from scratch.

Method	SHREC16	ModelNet40	ModelNet10	ScanObjectNN
# Training Samples	35764	9843	3991	2309
# Testing Samples	10265	2468	908	581
# Classes	55	40	10	15
PointNet++	85.1	90.7	94.1	84.3
DGCNN	87.0	92.2	94.7	86.2
PointNet++(with single SA)	84.5	91.3	95.0	82.4
with PatchAugment				
PointNet++	86.9(↑ 1.8)	92.4(↑ 1.7)	95.1(↑ 1.0)	87.1(↑2.8)
DGCNN	87.2(↑ 0.2)	93.1(↑ 0.9)	95.6(↑ 0.9)	86.9 (↑ 0.7)
PointNet++(with single SA)	87.4(↑ 2.9)	93.0(↑ 1.7)	95.6(↑ 0.6)	85.7 (↑ 3.3)

Table 3: Comparison of Various augmentation techniques by evaluation accuracy with single scale grouping for PointNet++ [2] on ModelNet40 (M40) and ModelNet10 (M10). '-' not available.

Method	M40 (Acc. %)	M10 (Acc. %)
Conventional DA [2]	90.7	-
RSMix [47]	92.1	94.4
PointMixUp [37]	91.7	-
PointAugment [38]	92.9	95.8
PatchAugment	92.4	95.1

5.1. Experiments Using Synthetic Datasets

Table 2 shows boost in the 3D classification accuracy by using PatchAugment technique in PointNet++ by 1.7%, DGCNN by 0.9%, single set abstraction PointNet++ by 1.7% on ModelNet40 dataset. Similarly, the improvement of accuracies for these three models on SHREC16, ModelNet10 and the CAD-based variant of ScanObjectNN (referred as OBJ_ONLY) are shown in Table 2. Table 3 shows the comparable performance of PatchAugment as a DA technique with other recent DA techniques accommodated in single scale grouping based PointNet++ model.

5.2. Experiments Using Real-World Dataset

In the case of a more complex real-world dataset ScanObjectNN, our PatchAugment technique boosted the classification accuracy of PointNet++ on all the six variants. i.e., OBJ_ONLY (2.8% ↑), OBJ_BG (3.1% ↑), PB_T25 (1.9% ↑), PB_T25_R (3.3% ↑), PB_T50_R (0.5% ↑), and PB_T50_RS (3.1 % ↑) as shown in Table 4 along with improvement in performance of DGCNN model due to PatchAugment. Table 5 shows class-wise accuracies of models with PatchAugment for the most challenging perturbed variant of the ScanObjectNN dataset, i.e., PB_T50_RS, in comparison with the same models with conventional DA techniques.

5.3. Ablation Studies

Table 6 shows a brief ablation study on few augmentations at patch level combined with patch level random points drop with a drop ratio of $\boldsymbol{\lambda}$. Further we follow [47] to show the ablation studies using ModelNet40 and ModelNet10 datasets as shown in Table 7 and Table 8 on single scale grouping PointNet++ [2] and DGCNN [5] models respectively. Table 9 lists an ablation study with different ranges of values for scale factors and rotation factors (both perturbed and up-axis rotations) while keeping random point drop, random translation and jitter at 25%, [-0.05, 0.05] and [-0.01, 0.01] respectively.

6. Conclusion

In this work, we introduced PatchAugment, a novel neighborhood-level data augmentation framework. The framework samples out data augmentation parameters for each of the neighborhoods. We use these parameters to drop points randomly, scale, rotate, translate, and jitter different point patches or neighborhoods differently. We demonstrate that PatchAugment is straightforward to include in deep network models similar to the PointNet++ network model, i.e., models that involve neighborhood querying. We have evaluated the impact of our augmentation technique on PointNet++ and DGCNN using various synthetic and real-world datasets, i.e., ModelNet40, ModelNet10, SHREC'16, and ScanObjectNN. Our experimental results encourage the use of PatchAugment for neighborhood augmentation in models that involve neighborhood querying.

Table 4: Improvements in 3D Object Classification accuracy (%) by using PatchAugment technique in the single scale grouping PointNet++ model, trained and tested on six variants(row 1) of the real-world ScanObjectNN dataset.

Models/Variants	OBJ_ONLY	OBJ_BG	PB_T25	PB_T25_R	PB_T50_R	PB_T50_RS
PointNet++[2]	84.3	82.3	82.7	81.4	79.1	77.9
PointNet++ w/ PatchAugment	**87.1**	**85.4**	**84.6**	**84.7**	79.6	**81.0**
% ↑ from PointNet++	2.8	3.1	1.9	3.3	0.5	3.1
DGCNN[5]	86.2	82.8	83.3	81.5	80.0	78.1
DGCNN w/ PatchAugment	86.9	84.2	84.3	82.3	**80.7**	79.7
% ↑ from DGCNN	0.7	1.4	1.0	0.8	0.7	1.6

Table 5: Improvements in 3D Object Classification accuracy (%) by using PatchAugment technique in PointNet++ and DGCNN models on the hardest variant, i.e., PB_T50_RS of ScanObjectNN benchmark dataset. Abbreviations: OA (Overall Accuracy), ACA (Average Class Accuracy), '-' means not available.

Methods	OA	ACA	bag	bin	box	cabinet	chair	desk	display	door	shelf	table	bed	pillow	sink	sofa	toilet
#shapes	-	-	298	794	406	1344	1585	592	678	892	1084	922	564	405	469	1058	325
PointNet++ [2]	77.9	75.4	49.4	84.4	31.6	77.4	91.3	**74.0**	79.4	85.2	72.6	72.6	75.5	**81.0**	**80.8**	90.5	**85.9**
DGCNN [5]	78.1	73.6	49.4	82.4	33.1	**83.9**	91.8	63.3	77.0	89.0	79.3	**77.4**	64.5	77.1	75.0	91.4	69.4
with PatchAugment																	
PointNet++	**81.0**	**79.7**	**66.3**	81.1	**63.5**	80.3	**91.9**	69.4	**91.2**	93.8	**80.6**	65.8	**84.2**	76.6	71.4	**93.4**	85.4
DGCNN	79.7	76.4	56.5	**83.0**	57.3	82.1	91.0	61.1	90.3	**95.0**	79.9	72.8	79.8	69.3	75.1	89.1	64.1

Table 6: An ablation study on different patch augmentations with random drop ($\boldsymbol{\lambda}$) in PointNet++ [2] on ModelNet40. $\boldsymbol{R}$ and $\boldsymbol{R_\theta}$ represent perturbed and up-axis rotations respectively.

	Scaling	$\boldsymbol{R}$	$\boldsymbol{R_\theta}$	Translation	Jitter
Acc.(%)	91.6	92.0	92.0	91.9	91.8

Table 7: An ablation study for evaluation accuracy on single scale grouping PointNet++ [2] model on ModelNet40.

Jitter+Shift	Rotation	Scaling	RandDrop	PatchAug	Acc.(%)
					91.5
				✓	**92.4** (↑ 0.9)
✓					91.7
✓				✓	91.8 (↑ 0.1)
✓			✓		91.9
✓			✓	✓	92.0 (↑ 0.1)
	✓	✓			90.5
	✓	✓		✓	90.8 (↑ 0.3)
	✓	✓	✓		91.0
	✓	✓	✓	✓	90.9 (↓0.1)
✓	✓	✓			90.2
✓	✓	✓		✓	90.7 (↑ 0.5)
✓	✓	✓	✓		90.6
✓	✓	✓	✓	✓	90.8 (↑ 0.3)

Table 8: An ablation study for DGCNN [5] on ModelNet40 (MN40) and ModelNet10 (MN10). Random scaling augmentation was applied as ConvDA.

ConvDA	RandDrop	PatchAug	Acc(%)	Dataset
			92.5	MN40
		✓	**93.1** (↑ 0.6)	MN40
✓			92.6	MN40
✓		✓	92.8 (↑ 0.2)	MN40
✓	✓		92.2	MN40
✓	✓	✓	93.0 (↑ 0.8)	MN40
			94.6	MN10
		✓	**95.6** (↑ 1.0)	MN10
✓			94.8	MN10
✓		✓	95.2 (↑ 0.4)	MN10
✓	✓		94.7	MN10
✓	✓	✓	95.4 (↑ 0.7)	MN10

Table 9: An ablation study on different ranges of scale $\boldsymbol{S}$ and different ranges of rotation angles represented by $\boldsymbol{R'} = \{\alpha, \beta, \gamma, \theta\}$. PointNet++ [2] trained on ModelNet40.

$\boldsymbol{S} \downarrow \boldsymbol{R'} \rightarrow$	[-0.1, 0.1]	[-0.2, 0.2]	[-0.3, 0.3]	[-0.4, 0.4]	[-0.5, 0.5]
[0.95, 1.05]	**92.4**	91.5	91.7	92.1	92.2
[0.90, 1.10]	91.7	92.0	91.8	91.2	91.6

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. 1, 3

[2] 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. 1, 2, 3, 4, 6, 7, 8

[3] 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. 1

[4] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. Densepoint: Learning densely contextual representation for efficient point cloud processing. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5239–5248, 2019. 1, 3

[5] 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. 1, 2, 6, 7, 8

[6] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019. 1, 3

[7] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 2

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5828–5839, 2017. 1, 2, 6

[9] Mark De Deuge, Alastair Quadros, Calvin Hung, and Bertrand Douillard. Unsupervised feature learning for classification of outdoor 3d scans. In *Australasian Conference on Robitics and Automation*, volume 2, page 1, 2013. 1, 2

[10] 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. 1, 2, 5

[11] 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 International Conference on Computer Vision*, pages 1588–1597, 2019. 1, 2, 5

[12] Huan Fu, Rongfei Jia, Lin Gao, Mingming Gong, Binqiang Zhao, Steve Maybank, and Dacheng Tao. 3d-future: 3d furniture shape with texture. *arXiv preprint arXiv:2009.09633*, 2020. 1, 2

[13] Kaleem Siddiqi, Juan Zhang, Diego Macrini, Ali Shokoufandeh, Sylvain Bouix, and Sven Dickinson. Retrieving articulated 3-d models using medial surfaces. *Machine vision and applications*, 19(4):261–275, 2008. 1, 2

[14] Manolis Savva, Fisher Yu, Hao Su, M Aono, B Chen, D Cohen-Or, W Deng, Hang Su, Song Bai, Xiang Bai, et al. Shrec16 track: largescale 3d shape retrieval from shapenet core55. In *Proceedings of the eurographics workshop on 3D object retrieval*, volume 10, 2016. 1, 2, 5

[15] Saeid Asgari Taghanaki, Jieliang Luo, Ran Zhang, Ye Wang, Pradeep Kumar Jayaraman, and Krishna Murthy Jatavallabhula. Robustpointset: A dataset for benchmarking robustness of point cloud classifiers. *arXiv preprint arXiv:2011.11572*, 2020. 1, 2

[16] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 1

[17] Hussein Kanaan and Alireza Behrad. Three-dimensional shape recognition and classification using local features of model views and sparse representation of shape descriptors. *Journal of Information Processing Systems*, 16(2):343–359, 2020. 2

[18] Qiangeng Xu, Xudong Sun, Cho-Ying Wu, Panqu Wang, and Ulrich Neumann. Grid-gcn for fast and scalable point cloud learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5661–5670, 2020. 2

[19] Tiange Xiang, Chaoyi Zhang, Yang Song, Jianhui Yu, and Weidong Cai. Walk in the cloud: Learning curves for point clouds shape analysis. *arXiv preprint arXiv:2105.01288*, 2021. 2

[20] Xin Wei, Ruixuan Yu, and Jian Sun. View-gcn: View-based graph convolutional network for 3d shape analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1850–1859, 2020. 2

[21] Jean-Baptiste Weibel, Timothy Patten, and Markus Vincze. Addressing the sim2real gap in robotic 3-d object classification. *IEEE Robotics and Automation Letters*, 5(2):407–413, 2019. 2

[22] Ruifeng Zhai, Xueyan Li, Zhenxin Wang, Shuxu Guo, Shuzhao Hou, Yu Hou, Fengli Gao, and Junfeng Song. Point cloud classification model based on a dual-input deep network framework. *IEEE Access*, 8:55991–55999, 2020. 2

[23] Ran Ben Izhak, Alon Lahav, and Ayellet Tal. Attwalk: Attentive cross-walks for deep mesh analysis. *arXiv preprint arXiv:2104.11571*, 2021. 2

[24] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6023–6032, 2019. 2, 3

[25] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 2, 3

[26] Zitian Chen, Yanwei Fu, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-level semantic feature augmentation for one-shot learning. *IEEE Transactions on Image Processing*, 28(9):4594–4605, 2019. 2

[27] Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017. 2

[28] Bo Liu, Xudong Wang, Mandar Dixit, Roland Kwitt, and Nuno Vasconcelos. Feature space transfer for data augmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9090–9098, 2018. 2

[29] Alexander J Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Advances in neural information processing systems*, pages 3236–3246, 2017. 2

[30] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2107–2116, 2017. 2

[31] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *Ieee Access*, 5:5858–5869, 2017. 3

[32] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. In *Advances in Neural Information Processing Systems*, pages 5049–5059, 2019. 3

[33] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Between-class learning for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5486–5494, 2018. 3

[34] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 3

[35] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast autoaugment. In *Advances in Neural Information Processing Systems*, pages 6665–6675, 2019. 3

[36] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019. 3

[37] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. *arXiv preprint arXiv:2008.06374*, 2020. 3, 7

[38] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Pointaugment: an auto-augmentation framework for point cloud classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6378–6387, 2020. 3, 7

[39] Jaeseok Choi, Yeji Song, and Nojun Kwak. Part-aware data augmentation for 3d object detection in point cloud. *arXiv preprint arXiv:2007.13373*, 2020. 3

[40] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in neural information processing systems*, pages 820–830, 2018. 3

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

[42] Shivanand Venkanna Sheshappanavar and Chandra Kambhamettu. A novel local geometry capture in pointnet++ for 3d classification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 262–263, 2020. 4

[43] Eric W Weisstein. Rotation matrix. *https://mathworld. wolfram. com/*, 2003. 4

[44] Binh-Son Hua, Quang-Hieu Pham, Duc Thanh Nguyen, Minh-Khoi Tran, Lap-Fai Yu, and Sai-Kit Yeung. Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101. IEEE, 2016. 6

[45] Xu Yan. Pointnet/pointnet++ pytorch. `https://github.com/yanx27/Pointnet_Pointnet2_pytorch`, 2019. 6

[46] Shivanand Venkanna Sheshappanavar and Chandra Kambhamettu. Dynamic local geometry capture in 3d point cloud classification. In *2021 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. IEEE Computer Society, 2021. 6

[47] Dogyoon Lee, Jaeha Lee, Junhyeop Lee, Hyeongmin Lee, Minhyeok Lee, Sungmin Woo, and Sangyoun Lee. Regularization strategy for point cloud via rigidly mixed sample. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15900–15909, 2021. 7