PointAugment: an Auto-Augmentation Framework for Point Cloud Classification

Ruihui Li Xianzhi Li Pheng-Ann Heng Chi-Wing Fu

The Chinese University of Hong Kong

{lirh,xzli,pheng,cwfu}@cse.cuhk.edu.hk

Abstract

We present PointAugment[1], a new auto-augmentation framework that automatically optimizes and augments point cloud samples to enrich the data diversity when we train a classification network. Different from existing auto-augmentation methods for 2D images, PointAugment is sample-aware and takes an adversarial learning strategy to jointly optimize an augmentor network and a classifier network, such that the augmentor can learn to produce augmented samples that best fit the classifier. Moreover, we formulate a learnable point augmentation function with a shape-wise transformation and a point-wise displacement, and carefully design loss functions to adopt the augmented samples based on the learning progress of the classifier. Extensive experiments also confirm PointAugment's effectiveness and robustness to improve the performance of various networks on shape classification and retrieval.

1. Introduction

In recent years, there has been a growing interest in developing deep neural networks [23, 24, 37, 19, 18] for 3D point cloud processing. To robustly train a network often relies on the availability and diversity of the data. However, unlike 2D image benchmarks such as ImageNet [10] and MS COCO dataset [15], which have over millions of training samples, 3D datasets are typically much smaller in quantity, with relatively small amount of labels and limited diversity. For instance, ModelNet40 [38], one of the most commonly-used benchmark for 3D point cloud classification, only has 12,311 models of 40 categories. The limited data quantity and diversity may cause overfitting problem and further affect the generalization ability of the network.

Nowadays, data augmentation (DA) is a very common strategy to avoid overfitting and improve the network generalization ability by artificially enlarging the quantity and diversity of the training samples. For 3D point clouds, due to the limited amount of training samples and an immense augmentation space in 3D, conventional DA strategies [23, 24] often simply perturb the input point cloud randomly in a small and fixed pre-defined augmentation range to maintain the class label. Despite its effectiveness for the existing classification networks, this conventional DA approach may lead to insufficient training, as summarized below.

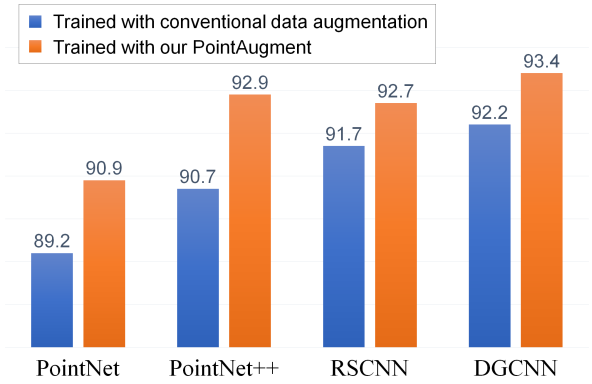

Figure 1: Classification accuracy (%) on ModelNet40 with or without training the networks with our PointAugment. We can see clear improvements on four representative networks. More comparison results are presented in Section 5.

First, existing methods for deep 3D point cloud processing regard the network training and DA as two independent phases without *jointly optimizing* them, *e.g.*, feedback the training results to enhance the DA. Hence, the trained network could be suboptimal. Second, existing methods apply the *same fixed augmentation* process with rotation, scaling, and/or jittering, to all input point cloud samples. The shape complexity of the samples is ignored in the augmentation, *e.g.*, a sphere remains the same no matter how we rotate it, but a complex shape may need larger rotations. Hence, conventional DA may be redundant or insufficient for augmenting the training samples [6].

To improve the augmentation of point cloud samples, we formulate *PointAugment*, a new auto-augmentation framework for 3D point clouds, and demonstrate its effective-

[1]Code: https://github.com/liruihui/PointAugment

ness to enhance shape classification; see Figure 1. Different from the previous works for 2D images, PointAugment *learns to produce augmentation functions specific to individual samples*. Further, the learnable augmentation function considers both *shape-wise transformation* and *point-wise displacement*, which relate to the characteristics of 3D point cloud samples. Also, PointAugment *jointly optimizes the augmentation process with the network training*, via an adversarial learning strategy to train the augmentation network (augmentor) together with the classification network (classifier) in an end-to-end manner. By taking the classifier losses as *feedbacks*, the augmentor can learn to enrich the input samples by enlarging the intra-class data variations, while the classifier can learn to combat this by extracting insensitive features. Benefited by such adversarial learning, the augmentor can then learn to generate augmented samples that best fit the classifier in different stages of the training, thus maximizing the capability of the classifier.

As the first attempt to explore auto-augmentation for 3D point clouds, we show by replacing conventional DA with PointAugment, clear improvements in shape classification on ModelNet40 [38] (see Figure 1) and SHREC16 [28] (see Section 5) datasets can be achieved on four representative networks, including PointNet [23], PointNet++ [24], RSCNN [18], and DGCNN [37]. Also, we demonstrate the effectiveness of PointAugment on shape retrieval and evaluate its robustness, loss configuration, and modularization design. More results are presented in Section 5.

2. Related Work

Data augmentation on images. Training data plays a very important role for deep neural networks to learn to perform tasks. However, training data usually has limited quantity, compared with the complexity of our real world, so data augmentation is often needed as a means to enlarge the training set and maximize the knowledge that a network can learn from the training data. Instead of randomly transforming the training data samples [42, 41], some works attempted to generate augmented samples from the original data by using image combination [12], generative adversarial network (GAN) [31, 27], Bayesian optimization [35], and image interpolation in the latent space [4, 16, 2]. However, these methods may produce unreliable samples that are different from those in the original data. On the other hand, some image DA techniques [12, 42, 41] apply pixel-by-pixel interpolation for images with regular structures; however, they cannot handle order-invariant point clouds.

Another approach aims to find an optimal combination of predefined transformation functions to augment the training samples, instead of applying the transformation functions based on a manual design or by complete randomness. AutoAugment [3] suggests a reinforcement learning strategy to find the best set of augmentation functions by alternatively training a proxy task and a policy controller, then applying the learned augmentation function to the input data. Soon after, two other works, FastAugment [14] and PBA [8], explore advanced hyper-parameter optimization methods to more efficiently find the best transformations for the augmentation. Different from these methods, which learn to find a fixed augmentation strategy for all the training samples, PointAugment is sample-aware, meaning that we dynamically produce the transformation functions based on the properties of individual training samples and the network capability during the training process.

Very recently, Tang *et al.* [33] and Zhang *et al.* [43] suggested to learn augmentation policies on target tasks using an adversarial strategy. They tend to directly maximize the loss of augmented samples to improve the generalization of image classification networks. Differently, PointAugment enlarges the loss between the augmented point clouds and their original ones by an explicitly-designed boundary (see Section 4.2 for details); it dynamically adjusts the difficulty of the augmented samples, so that the augmented samples can better fit the classifier for different training stages.

Data augmentation on point cloud. In existing points processing networks, data augmentation mainly include random rotation about the gravity axis, random scaling, and random jittering [23, 24]. These handcrafted rules are fixed throughout the training process, so we may not obtain the best samples to effectively train the network. So far, we are not aware of any work that explores auto-augmentation to maximize the network learning with 3D point clouds.

Deep learning on point cloud. Improving on the PointNet architecture [23], several works [24, 17, 18] explored local structures to enhance the feature learning. Some others explored the graph convolutional networks by creating a local graph [36, 37, 29, 45] or geometric elements [11, 22]. Another stream of works [32, 34, 19] projected irregular points into a regular space to allow traditional convolutional neural networks to work on. Different from the above works, our goal is not on designing a new network but on boosting the classification performance of existing networks by effectively optimizing the augmentation of point cloud samples. To this end, we design an augmentor to learn a sample-specific augmentation function and adjust the augmentation based also on the learning progress of the classifier.

3. Overview

The main contribution of this work is the PointAugment framework that automatically optimizes the augmentation of the input point cloud samples for more effectively training the classification network. Figure 2 illustrates the design of our framework, which has two deep neural network components: (i) an augmentor $\mathcal{A}$ and (ii) a classifier $\mathcal{C}$. Given

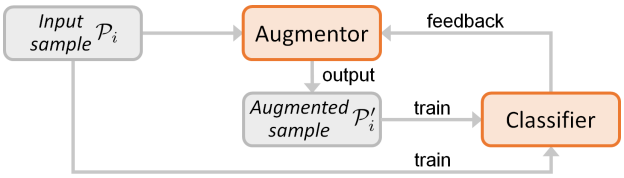

Figure 2: An overview of our PointAugment framework. We jointly optimize the augmentor and classifier in an end-to-end manner with an adversarial learning strategy.

an input training dataset $\{\mathcal{P}_i\}_{i=1}^{M}$ of M samples, where each sample has N points, before we train classifier $\mathcal{C}$ with sample $\mathcal{P}_i$, we feed $\mathcal{P}_i$ first to our augmentor $\mathcal{A}$ to generate an augmented sample $\mathcal{P}'_i$. Then, we feed $\mathcal{P}_i$ and $\mathcal{P}'_i$ separately to classifier $\mathcal{C}$ for training, and further take $\mathcal{C}$'s results as feedback to guide the training of augmentor $\mathcal{A}$.

Before elaborating the PointAugment framework, we first discuss our key ideas behind the framework. These are new ideas (not present in previous works [3, 14, 8]) that enable us to efficiently augment the training samples, which are now 3D point clouds instead of 2D images.

- *Sample-aware*. Rather than finding a universal set of augmentation policy or procedure for processing every input data sample, we aim to regress a specific augmentation function for each input sample by considering the underlying geometric structure of the sample. We call this a sample-aware auto-augmentation.
- *2D vs. 3D augmentation*. Unlike 2D augmentations for images, 3D augmentation involves a more immense and different spatial domain. Accounting for the nature of 3D point clouds, we consider two kinds of transformations on point cloud samples: shape-wise transformation (including rotation, scaling, and their combinations), and point-wise displacement (jittering of point locations), where our augmentor should learn to produce them to enhance the network training.
- *Joint optimization*. During the network training, the classifier will gradually learn and become more powerful, so we need more challenging augmented samples to better train the classifier, as the classifier becomes stronger. Hence, we design and train the PointAugment framework in an end-to-end manner, such that we can jointly optimize both the augmentor and classifier. To achieve so, we have to carefully design the loss functions and dynamically adjust the difficulty of the augmented samples, while considering both the input sample and the capacity of the classifier.

4. Method

In this section, we first present the network architecture details of the augmentor and classifier (Section 4.1). Then, we present our loss functions formulated for the augmentor (Section 4.2) and classifier (Section 4.3), and introduce

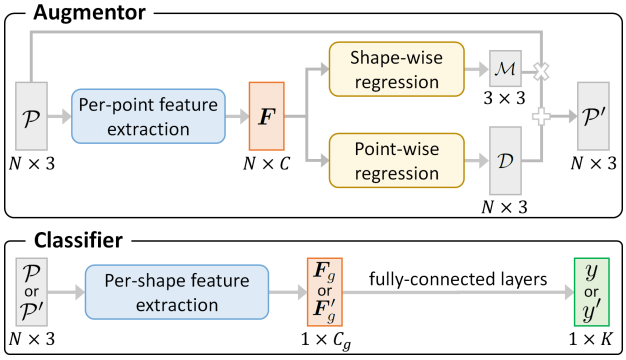

Figure 3: Illustrations of the augmentor and classifier. The augmentor generates augmented sample $\mathcal{P}'$ from $\mathcal{P}$, and the classifier predicts the class label given $\mathcal{P}'$ or $\mathcal{P}$ as inputs.

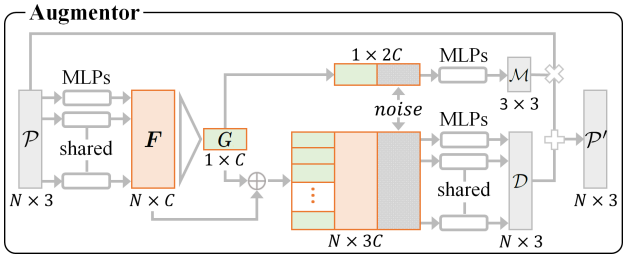

Figure 4: Our implementation of the augmentor.

our end-to-end training strategy (Section 4.4). Lastly, we present the implementation details (Section 4.5).

4.1. Network Architecture

Augmentor. Different from existing works [3, 14, 8], our augmentor is sample-aware, and it learns to generate a specific function for augmenting each input sample. From now on, we drop subscript i for ease of reading, and denote $\mathcal{P}$ as the training sample input to augmentor $\mathcal{A}$ and $\mathcal{P}'$ as the corresponding augmented sample output from $\mathcal{A}$.

The overall architecture of our augmentor is illustrated in Figure 3 (top). First, we use a per-point feature extraction unit to embed point features $\boldsymbol{F} \in \mathbb{R}^{N \times C}$ for all N points in $\mathcal{P}$, where C is the number of feature channels. From $\boldsymbol{F}$, we then regress the augmentation function specific to input sample $\mathcal{P}$ using two separate components in the architecture: (i) shape-wise regression to produce transformation $\mathcal{M} \in \mathbb{R}^{3 \times 3}$ and (ii) point-wise regression to produce displacement $\mathcal{D} \in \mathbb{R}^{N \times 3}$. Note that, the learned $\mathcal{M}$ is a linear matrix in 3D space, combining mainly rotation and scaling, whereas the learned $\mathcal{D}$ gives point-wise translation and jittering. Using $\mathcal{M}$ and $\mathcal{D}$, we can then generate the augmented sample $\mathcal{P}'$ as $\mathcal{P} \cdot \mathcal{M} + \mathcal{D}$.

The design of our proposed framework for the augmentor is generic, meaning that we may use different models to build its components. Figure 4 shows our current im-

plementation, for reference. Specifically, similar to PointNet [23], we first employ a series of shared multi-layer perceptron (MLPs) to extract per-point features $\boldsymbol{F} \in \mathbb{R}^{N\times C}$. We then employ max pooling to obtain the per-shape feature vector $\boldsymbol{G} \in \mathbb{R}^{1\times C}$. To regress $\mathcal{M}$, we generate a C-dimension noise vector based on a Gaussian distribution and concatenate it with $\boldsymbol{G}$, and then employ MLPs to obtain $\mathcal{M}$. Note that the noise vector enables the augmentor to explore more diverse choices in regressing the transformation matrix through the randomness introduced into the regression process. To regress $\mathcal{D}$, we concatenate N copies of $\boldsymbol{G}$ with $\boldsymbol{F}$, together with an $N \times C$ noise matrix, whose values are randomly and independently generated based on a Gaussian distribution. Lastly, we employ MLPs to obtain $\mathcal{D}$.

Classifier. Figure 3 (bottom) shows the general architecture of classifier $\mathcal{C}$. It takes $\mathcal{P}$ and $\mathcal{P}'$ as inputs in two separate rounds and predicts corresponding class labels y and y'. Both y and $y' \in \mathbb{R}^{1\times K}$, where K is the total number of classes in the classification problem. In general, $\mathcal{C}$ first extracts per-shape global features $\boldsymbol{F}_g$ or $\boldsymbol{F}'_g \in \mathbb{R}^{1\times C_g}$ (from $\mathcal{P}$ or $\mathcal{P}'$), and then employ fully-connected layers to regress a class label. Also, the choice of implementing $\mathcal{C}$ is flexible. We may employ different classification networks as $\mathcal{C}$. In Section 5, we shall show that the performance of several conventional classification networks can be further boosted when equipped with our augmentor in the training.

4.2. Augmentor loss

To maximize the network learning, augmented sample $\mathcal{P}'$ generated by the augmentor should satisfy two requirements: (i) $\mathcal{P}'$ should be more challenging than $\mathcal{P}$, *i.e.*, we aim for $L(\mathcal{P}') \geq L(\mathcal{P})$; and (ii) $\mathcal{P}'$ should not lose its shape distinctiveness, meaning that it should describe a shape that is not too far (or different) from $\mathcal{P}$.

To achieve requirement (i), a simple way to formulate the loss function for the augmentor (denoted as $\mathcal{L}_\mathcal{A}$) is to maximize the difference between the cross entropy losses on $\mathcal{P}$ and $\mathcal{P}'$, or equivalently, to minimize

$$\mathcal{L}_\mathcal{A} = \exp[-(L(\mathcal{P}') - L(\mathcal{P}))], \tag{1}$$

where $L(\mathcal{P})=-\sum_{c=1}^{K} \hat{y}_c \log(y_c))$ is $\mathcal{P}$'s cross entropy loss; $\hat{y}_c \in \{0, 1\}$ denotes the one-hot ground-truth label when $\mathcal{P}$ belongs to the c-th class; and $y_c \in [0, 1]$ is the probability of predicting $\mathcal{P}$ as c-th class. Note also that, for $\mathcal{P}'$ to be more challenging than $\mathcal{P}$, we assume that $L(\mathcal{P}')\geq L(\mathcal{P})$ and a larger $L(\mathcal{P}')$ indicates a larger magnitude of augmentation, which can be defined as $\xi = L(\mathcal{P}')-L(\mathcal{P})$.

However, if we naively minimize Eq. (1) for $\mathcal{L}_\mathcal{A}\to 0$, we encourage $L(\mathcal{P}')-L(\mathcal{P})\to\infty$. So, a simple solution for $\mathcal{P}'$ is an arbitrary sample regardless of $\mathcal{P}$. Such $\mathcal{P}'$ clearly violates requirement (ii). Hence, we further restrict the augmentation magnitude ξ. Inspired by LS-GAN [25], we first

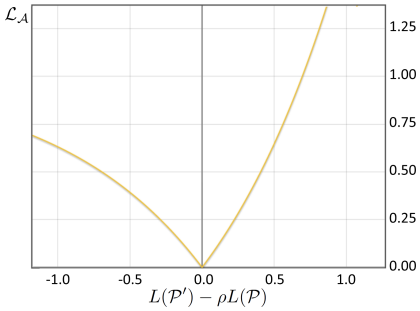

Figure 5: Graph plot of Eq. (2).

introduce a dynamic parameter ρ and re-formulate $\mathcal{L}_\mathcal{A}$ as

$$\mathcal{L}_\mathcal{A} = |1.0 - \exp[L(\mathcal{P}') - \rho L(\mathcal{P})]|. \tag{2}$$

See Figure 5 for the graph plot of Eq. (2). In this formulation, we want $L(\mathcal{P}')$ to be large (for requirement (i)) but it should not be too large (for requirement (ii)), so we upper-bound $L(\mathcal{P}')$ by $\rho L(\mathcal{P})$. Hence, we can obtain

$$\xi \;=\; L(\mathcal{P}') - L(\mathcal{P}) \;\leq\; (\rho - 1)L(\mathcal{P}), \tag{3}$$

where we denote $\xi_o = (\rho - 1)L(\mathcal{P})$ as ξ's upper bound.

Note that, when we train the augmentor, the classifier is fixed (to be presented in Section 4.4), so $L(\mathcal{P})$ is fixed. Hence, ξ_o depends only on ρ. Since it should be non-negative, we thus ensure $\rho\geq 1$. Moreover, considering that the classifier is very fragile at the beginning of the training, we pay more attention to training the classifier rather than generating a challenging $\mathcal{P}'$. Hence, ξ_o should not be too large, meaning that $\mathcal{P}'$ should not be too challenging. Later, when the classifier becomes more powerful, we can gradually enlarge ξ_o to allow the augmentor to generate a more challenging $\mathcal{P}'$. Therefore, we design a dynamic ρ to control ξ_o with the following formulation:

$$\rho = \max\Big(1, \exp(\sum_{c=1}^{K} \hat{y}_c \cdot y_c)\Big), \tag{4}$$

where $\max(1, *)$ ensures $\rho\geq 1$. At the beginning of the network training, the classifier predictions may not be accurate. Hence, the prediction probability y_c is generally small, resulting in a small ρ, and ξ_o will also be small according to Eq. (3). When the classifier becomes more powerful, y_c will increase, and we will have larger ρ and ξ_o accordingly.

Lastly, to further ensure the augmented sample $\mathcal{P}'$ to be shape distinctive (for requirement (ii)), we add $L(\mathcal{P}')$, as a fidelity term, to Eq. (2) to construct the final loss $\mathcal{L}_\mathcal{A}$:

$$\mathcal{L}_\mathcal{A} = L(\mathcal{P}') + \lambda|1.0 - \exp(L(\mathcal{P}') - \rho L(\mathcal{P}))|, \tag{5}$$

where λ is a fixed hyper-parameter to control the relative importance of each term. A small λ encourages the augmentor to focus more on the classification with less augmentation on $\mathcal{P}$, and vice versa. In our implementation (all experiments), we set $\lambda = 1$ to treat the two terms equally.

4.3. Classifier loss

The goal of the classifier $\mathcal{C}$ is to correctly predict both $\mathcal{P}$ and $\mathcal{P}'$. Additionally, $\mathcal{C}$ should also have the ability to learn stable per-shape global features, no matter given $\mathcal{P}$ or $\mathcal{P}'$ as input. We thus formulate the classifier loss $\mathcal{L}_{\mathcal{C}}$ as

$$\mathcal{L}_{\mathcal{C}} = L(\mathcal{P}') + L(\mathcal{P}) + \gamma \|\boldsymbol{F}_g - \boldsymbol{F}_{g'}\|_2, \tag{6}$$

where γ is to balance the importance of the terms (we empirically set γ as 10.0), and $\|\boldsymbol{F}_g - \boldsymbol{F}_{g'}\|_2$ helps explicitly penalize the feature difference between the augmented sample and the original one, and stabilize the network training.

4.4. End-to-end training strategy

Algorithm 1 summarizes our end-to-end training strategy. Overall, the procedure alternatively optimizes and updates the learnable parameters in augmentor $\mathcal{A}$ and classifier $\mathcal{C}$, while fixing the other one, during the training. Given input sample $\mathcal{P}_i$, we first employ $\mathcal{A}$ to generate its augmented sample $\mathcal{P}'_i$. We then update the learnable parameters in $\mathcal{A}$ by calculating the augmentor loss using Eq. (5). In this step, we keep $\mathcal{C}$ unchanged. After updating $\mathcal{A}$, we keep $\mathcal{A}$ unchanged, and generate the updated $\mathcal{P}'_i$. We then feed $\mathcal{P}_i$ and $\mathcal{P}'_i$ to $\mathcal{C}$ one by one to obtain $L(\mathcal{P})$ and $L(\mathcal{P}')$, respectively, and update the learnable parameters in $\mathcal{C}$ by calculating the classifier loss using Eq. (6). In this way, we can optimize and train $\mathcal{A}$ and $\mathcal{C}$ in an end-to-end manner.

4.5. Implementation details

We implement PointAugment using PyTorch [21]. In detail, we set the number of training epochs $S = 250$ with a batch size $B = 24$. To train the augmentor, we adopt the Adam optimizer with a learning rate of 0.001. To train the classifier, we follow the respective original configuration from the released code and paper. Specifically, for PointNet [23], PointNet++ [24], and RSCNN [18], we use the Adam optimizer with an initial learning rate of 0.001, which is gradually reduced with a decay rate of 0.5 every 20 epochs. For DGCNN [37], we use the SGD solver with a momentum of 0.9 and a base learning rate of 0.1, which decays using a cosine annealing strategy [9].

Note also that, to reduce model oscillation [5], we follow [31] to train PointAugment by using mixed training samples, which contain the original training samples as one half and our previously-augmented samples as the other half, rather than using only the original training samples. Please refer to [31] for more details. Moreover, to avoid overfitting, we set a dropout probability of 0.5 to randomly drop or keep the regressed shape-wise transformation and point-wise displacement. In the testing phase, we follow previous networks [23, 24] to feed the input test samples to the trained classifier to obtain the predicted labels, without any additional computational cost.

Algorithm 1: Training Strategy in PointAugment

Input: training point sets $\{\mathcal{P}_i\}_{i=1}^{M}$, corresponding ground-truth class labels $\{\hat{y}_i\}_{i=1}^{M}$, and the number of training epochs S.
Output: $\mathcal{C}$ and $\mathcal{A}$.
for $s = 1, \cdots, S$ **do**
  for $i = 1, \cdots, M$ **do**
    // Update augmentor $\mathcal{A}$
    Generate augmented sample $\mathcal{P}'_i$ from $\mathcal{P}_i$
    Calculate the augmentor loss using Eq. (5)
    Update the learnable parameters in $\mathcal{A}$
    // Update classifier $\mathcal{C}$
    Calculate the classifier loss using Eq. (6) by feeding $\mathcal{P}_i$ and $\mathcal{P}'_i$ alternatively to $\mathcal{C}$
    Update the learnable parameters in $\mathcal{C}$
  end
end

Table 1: Statistics of the ModelNet10 (MN10) [38], ModelNet40 (MN40) [38], and SHREC16 (SR16) [28] datasets, including the number of categories (classes), number of training and testing samples, average number of samples per class, and the corresponding standard deviation value.

Dataset	#Class	#Training	#Testing	Average	Std.
MN10	10	3991	908	399.10	233.36
MN40	40	9843	2468	246.07	188.64
SR16	55	36148	5165	657.22	1111.49

5. Experiments

We conducted extensive experiments on PointAugment. First, we introduce the benchmark datasets and classifiers employed in our experiments (Section 5.1). We then evaluate PointAugment on shape classification and shape retrieval (Section 5.2). Next, we perform detailed analysis on PointAugment's robustness, loss configuration, and modularization design (Section 5.3). Lastly, we present further discussion and potential future extensions (Section 5.4).

5.1. Datasets and Classifiers

Datasets. We employed three 3D benchmark datasets in our evaluations, *i.e.*, ModelNet10 [38], ModelNet40 [38], and SHREC16 [28], for which we denote as MN10, MN40, and SR16, respectively. Table 1 presents statistics about the datasets, showing that, MN10 is a very small dataset with only 10 classes. Though most networks [23, 17] can achieve a high classification accuracy on MN10, they may easily overfit. SR16 is the largest data with over 36,000 training samples. However, the high standard deviation (std.)

Table 2: Comparing the overall shape classification accuracy (%) on MN40, MN10, and SR16, for various classifiers equipped with conventional DA (first four rows) and with our PA (last four rows); PA denotes PointAugment. We can observe improvements for all datasets and all classifiers.

Method	MN40	MN10	SR16
PointNet [23]	89.2	91.9	84.4
PointNet++ [24]	90.7	93.3	85.1
RSCNN [18]	91.7	94.2	86.6
DGCNN [37]	92.2	94.8	87.0
PointNet (+PA)	90.9 (1.7↑)	94.1 (2.2↑)	88.4 (4.0↑)
PointNet++ (+PA)	92.9 (2.2↑)	95.8 (2.5↑)	89.5 (4.4↑)
RSCNN (+PA)	92.7 (1.0↑)	96.0 (1.8↑)	90.1 (3.5↑)
DGCNN (+PA)	93.4 (1.2↑)	96.7 (1.9↑)	90.6 (3.6↑)

value, *i.e.*, 1111, shows the uneven distribution of training samples among the classes. For example, in SR16, the *Table* class has 5,905 training samples, while the *Cap* class has only 39 training samples. For MN40, we directly adopt the data kindly provided by PointNet [23] and follow the same train-test split. For MN10 and SR16, we uniformly sample 1,024 points on each mesh surface and normalize the point sets to fit a unit ball centered at the origin.

Classifiers. As explained in Section 4.1, our overall framework is generic, and we can employ different classification networks as classifier $\mathcal{C}$. To show that the performance of conventional classification networks can be further boosted when equipped with our augmentor, in the following experiments, we employ several representative classification networks as classifier $\mathcal{C}$, including (i) PointNet [23], a pioneer network that processes points individually; (ii) PointNet++ [24], a hierarchical feature extraction network; (iii) RSCNN[1] [18], a recently-released enhanced version of PointNet++ with a relation weight inside each local region; and (iv) DGCNN [37], a graph-based feature extraction network. Note that, most existing networks [44, 34, 17] are built and extended from the above networks with various means of adaptation.

5.2. PointAugment Evaluation

Shape classification. First, we evaluate our PointAugment on the shape classification task using the classifiers listed in Section 5.1. For comparison, when we train the classifiers without PointAugment, we follow [24] to augment the training samples by random rotation, scaling, and jittering, which are considered as conventional DA.

Table 2 summarizes the quantitative evaluation results for comparison. We report the overall classification accuracy (%) of each classifier on all the three benchmark datasets, with conventional DA and with our PointAugment. From the results we can clearly see that, by employing PointAugment, the shape classification accuracies of all classifier networks can improve for all the three benchmark datasets. Particularly, on MN40, the classification accuracy achieved by DGCNN+PointAugment is 93.4%, which is a very high accuracy value comparable with the very recent works [44, 34, 17]. Moreover, our PointAugment is shown to be more effective on the imbalanced SR16 dataset; see the right-most column in Table 2, showing that PointAugment can alleviate the class size imbalance problem through our sample-aware auto-augmentation strategy to introduce more intra-class variation to the augmented samples.

[1]Only the single-scale RSCNN [18] is released so far.

Table 3: Comparing the shape retrieval results (mAP, %) on MN40, for various methods equipped with conventional DA or with our PointAugment. Again, we can observe clear improvements in retrieval accuracy for all the four methods.

Method	Conventional DA	PointAugment	Change
PointNet [23]	70.5	75.8	5.2↑
PointNet++ [24]	81.3	86.7	5.4↑
RSCNN [18]	83.2	86.6	3.4↑
DGCNN [37]	85.3	89.0	3.7↑

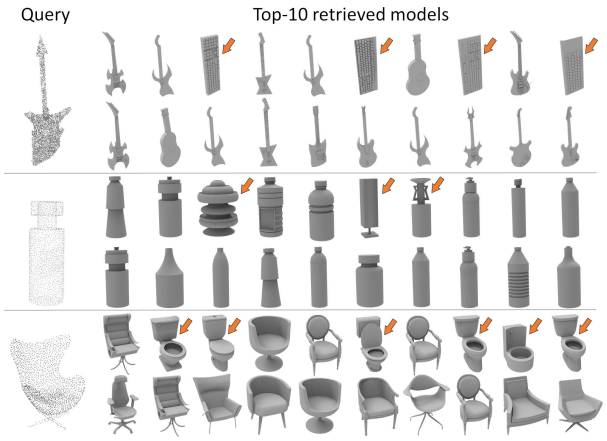

Figure 6: Shape retrieval results on MN40. For each query shape on the left, we present two rows of Top-10 retrieval results: the top row uses PointNet [23] and the bottom row uses PointNet+PointAugment. Note that the obviously-wrong retrieval results are marked with red arrows.

Shape retrieval. To validate whether PointAugment facilitates the classifiers to learn a better shape signature, we compare the shape retrieval performance on MN40. Specifically, we regard each sample in the testing split as a query, and aim to retrieve the best similar shapes from the testing split by comparing the cosine similarity between their global features $\boldsymbol{F}_g$. In this experiment, we employ the mean Average Precision (mAP) as the evaluation metric.

Table 3 presents the evaluation results, which clearly show that PointAugment improves the shape retrieval per-

Table 4: Robustness test to compare our PointAugment with conventional DA. Here, we corrupt each input test sample by random jittering (Jitt.) with Gaussian noise in [-1.0, 1.0], by scaling with a ratio of 0.9 or 1.1, or by a rotation of 90° or 180° along the gravity axis. Also, we show the original accuracy (Ori.) without using corrupted samples.

Method	Ori.	Jitt.	0.9	1.1	90°	180°
Without DA	89.1	88.2	88.2	88.2	48.2	40.1
Conventional DA	90.7	90.3	90.3	90.3	89.9	89.7
PointAugment	**92.9**	**92.8**	**92.8**	**92.8**	**92.7**	**92.6**

Table 5: Ablation study of PointAugment. $\mathcal{D}$: point-wise displacement, $\mathcal{M}$: shape-wise transformation, DP: dropout, and Mix: mixed training samples (see Section 4.4).

Model	$\mathcal{D}$	$\mathcal{M}$	DP	Mix	Acc.	Inc.↑
A					90.7	-
B	✓				91.7	1.0
C		✓			91.9	1.2
D	✓	✓			92.5	1.8
E	✓	✓	✓		92.8	2.1
F	✓	✓	✓	✓	**92.9**	**2.2**

formance for all the four classifier networks. Especially, for PointNet [23] and PointNet++ [24], the percentage of improvement is over 5%. Besides, we show visual results on shape retrieval for three different query models in Figure 6. Compared with the original PointNet [23], which is equipped with conventional DA, the augmented version with PointAugment produces more accurate retrievals.

5.3. PointAugment Analysis

Further, we conducted more experiments to evaluate various aspects of PointAugment, including a robustness test (Section 5.3.1), an ablation study (Section 5.3.2), and a detailed analysis on its Augmentor network (Section 5.3.3). Note that, in these experiments, we employ PointNet++ [24] as the classifier and perform experiments on MN40.

5.3.1 Robustness Test

We conducted the robustness test by corrupting test samples using the following five settings: (i) adding random jittering with Gaussian noise ranged $[-1.0, 1.0]$; (ii,iii) adding uniform scaling with a ratio of 0.9 or 1.1; and (iv,v) adding rotation with 90° or 180° along the gravity axis. For each setting, we use three different DA strategies: without DA, conventional DA, and our PointAugment.

Table 4 reports the results, where we show also the original test accuracy (Ori.) without using corrupted test samples as a reference. The results in the first two rows show that

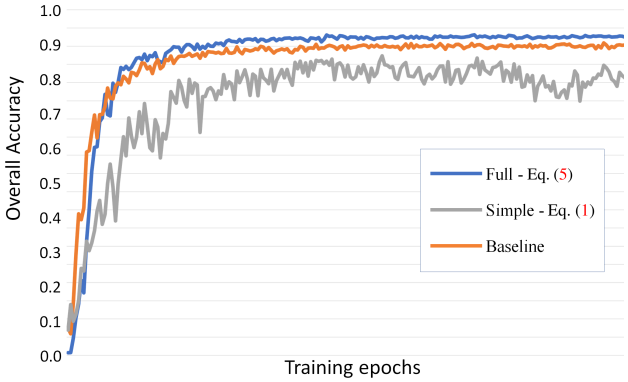

Figure 7: Evaluation curves: shape classification accuracy using different versions of $\mathcal{L}_{\mathcal{A}}$ over training epochs.

DA is an efficient way to improve the classification performance. Further, comparing the last two rows in the table, we can see that for all settings, our PointAugment consistently outperforms the conventional DA, which is random-based and may not yield good augmented samples all the time. Particularly, by comparing the results with the original test accuracy, PointAugment is less sensitive to corruption, where the achieved accuracy reduces only slightly. Such a result shows that PointAugment improves the robustness of a network with better shape recognition.

5.3.2 Ablation Study

Table 5 summarizes the results of the ablation study. Model A denotes PointNet++ [24] without our augmentor, which gives a baseline classification accuracy of 90.7%. On top of Model A, we employ our augmentor with point-wise displacement $\mathcal{D}$ alone (Model B), with shape-wise transformation $\mathcal{M}$ alone (Model C), or with both (Model D). From the results shown in the first four rows in Table 5, we can see that, each of the augmentation functions contributes to produce more effective augmented samples.

Besides, we also ablate the dropout strategy (DP) for training, and the use of mixed training samples (Mix), as presented in Section 4.5, where we create Models E & F for comparison; see Table 5. By comparing the classification accuracies achieved by Models D, E, and F, we can see that both DP and Mix help to slightly improve the overall results. Note, these strategies are typically for stabilizing the model training and exploring more transformations.

5.3.3 Augmentor Analysis

Analysis on $\mathcal{L}_{\mathcal{A}}$. As described in Section 4.2, we employ $\mathcal{L}_{\mathcal{A}}$ (see Eq. (5)) to guide the training of our augmentor. To demonstrate its superiority, we compare it with (i) a simple version (see Eq. (1)) and (ii) a baseline, *i.e.*, the conventional DA employed in PointNet++ [24]. Figure 7 plots the evaluation accuracy curves in terms of the training epochs.

Table 6: Classification accuracy (%) for diff. λ in Eq. (5).

$\lambda = 0.5$	$\lambda = 1.0$	$\lambda = 2.0$
92.1	**92.9**	92.3

Table 7: Accuracy (%) with diff. feature extraction units.

DenseConv	EdgeConv	our
92.5	92.7	**92.9**

Clearly, the training state achieved by using the simple version is very unstable; see the gray plot.This indicates that simply enlarging the difference between $\mathcal{P}$ and $\mathcal{P}'$ without restriction will create turbulence in the training process, resulting in a worse classification performance, when compared even with the baseline; see the orange plot. Comparing the blue and orange plots in Figure 7, we can see that, at the beginning of the training, since the augmentor is initialized randomly, the accuracy of employing PointAugment is slightly lower than the baseline. However, when the training continues, PointAugment rapidly surpasses the baseline and shows a clear improvement over the baseline, showing the effectiveness of our designed augmentor loss.

Hyper-parameter λ in Eq. (5). Table 6 shows the classification accuracy for different choices of λ. As we mentioned in Section 4.2, a small λ encourages the augmentor to focus more on the classification. However, if it is too small, *e.g.*, 0.5, the augmentor tends to take no actions in the augmentation function, thus leading to a worse classification performance; see the comparisons in the left two columns in Table 6. On the other hand, if we set a larger λ, *e.g.*, 2.0, the augmented samples may be too difficult for the classifier. Such a result also hinders the network training; see the right-most column. Hence, in PointAugment, we adopt $\lambda = 1.0$ to equally treat the two components.

Analysis on the augmentor architecture design. As we mentioned in Section 4.1, we employ a per-point feature extraction unit to embed per-point features $\boldsymbol{F}$ given the training sample $\mathcal{P}$. In our implementation, we use shared MLPs to extract $\boldsymbol{F}$. In this part, we further explore two other choices of feature extraction units for replacing the MLPs, including DenseConv [17] and EdgeConv [37]. Please refer to their original papers for the detailed methods. Table 7 shows the accuracy comparisons for the three implementations. From the results, we can see that, although using MLPs is a relative simple implementation compared with DenseConv and EdgeConv, it can lead to the best classification performance. We think of the following reasons. The aim of our augmentor is to regress a shape-wise transformation $\mathcal{M} \in \mathbb{R}^{3\times3}$ and a point-wise displacement $\mathcal{D} \in \mathbb{R}^{N\times3}$ from the per-point features $\boldsymbol{F}$, which is not a very tough task. If we apply a complex unit to extract $\boldsymbol{F}$, it may easily have overfitting problem. The results shown in Table 7 demonstrate that the MLPs is already enough for our augmentor to regress the augmentation functions.

5.4. Discussion and Future work

Overall, the augmentor network learns the sample-level augmentation function in a self-supervised manner, by taking the feedback from the classifier to update its parameters. As a result, the advantage on the classifier network is that by exploring those well-tuned augmented samples, the classifier can enhance its capability and better learn to uncover intrinsic variations among the different classes and discover the intra-class insensitive features.

In the future, we plan to adapt PointAugment for more tasks, such as part segmentation [23, 18], semantic segmentation [20, 45, 1], object detection [39, 30], upsampling [40, 13], denoising [7, 26], etc. However, it is worth to note particularly that different tasks require different considerations. For example, for parts segmentation, it is better for the augmentor to be part-aware and produce distortions on parts without changing the parts semantic; for object detection, the augmentor should be able to generate a richer variety of 3D transformations for various kinds of object instances in 3D scenes, etc. Therefore, one future direction is to explore part-aware or instance-aware auto-augmentation to extend PointAugment for other tasks.

6. Conclusion

We presented PointAugment, the first auto-augmentation framework that we are aware of for 3D point clouds, considering both the capability of the classification network and the complexity of the training samples. First, PointAugment is an end-to-end framework that jointly optimizes the augmentor and classifier networks, such that the augmentor can learn to improve based on feedback from the classifier and the classifier can learn to process wider variety of training samples. Second, PointAugment is sample-aware with its augmentor learns to produce augmentation functions specific to the input samples, with a shape-wise transformation and a point-wise displacement for handling point cloud samples. Third, we formulate a novel loss function to enable the augmentor to dynamically adjust the augmentation magnitude based on the learning state of the classifier, so that it can generate augmented samples that best fit the classifier in different training stages. In the end, we conducted extensive experiments and demonstrated how PointAugment contributes to improve the performance of four representative networks on the MN40 and SR16 datasets.

Acknowledgments. We thank anonymous reviewers for the valuable comments. The work is supported by the Research Grants Council of the Hong Kong Special Administrative Region (CUHK 14201717 & 14201918), and CUHK Research Committee Direct Grant for Research 2018/19.

References

[1] Zhutian Chen, Wei Zeng, Zhiguang Yang, Lingyun Yu, Chi-Wing Fu, and Huamin Qu. LassoNet: Deep Lasso-selection of 3D point clouds. *IEEE Trans. Vis. & Comp. Graphics (IEEE Visualization)*, 26(1):195–204, 2020. 8

[2] Zitian Cheny, Yanwei Fuy, Yinda Zhang, Yu-Gang Jiang, Xiangyang Xue, and Leonid Sigal. Multi-level semantic feature augmentation for one-shot learning. *IEEE Trans. Image Proc (TIP)*, 28(9):4594–4605, 2019. 2

[3] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. AutoAugment: Learning augmentation strategies from data. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019. 2, 3

[4] Terrance DeVries and Graham W. Taylor. Dataset augmentation in feature space. In *Int. Conf. on Learning Representations (ICLR)*, 2017. 2

[5] Ian Goodfellow. NeurIPS 2016 tutorial: Generative adversarial networks. *arXiv preprint arXiv:1701.00160*, 2016. 5

[6] Benjamin Graham. Fractional max-pooling. *arXiv preprint arXiv:1412.6071*, 2014. 1

[7] Pedro Hermosilla, Tobias Ritschel, and Timo Ropinski. Total Denoising: Unsupervised learning of 3D point cloud cleaning. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 52–60, 2019. 8

[8] Daniel Ho, Eric Liang, Ion Stoica, Pieter Abbeel, and Xi Chen. Population Based Augmentation: Efficient learning of augmentation policy schedules. In *Int. Conf. on Machine Learning. (ICML)*, 2019. 2, 3

[9] Gao Huang, Yixuan Li, Geoff Pleiss, Zhuang Liu, John E. Hopcroft, and Kilian Q. Weinberger. Snapshot ensembles: Train 1, get M for free. In *Int. Conf. on Learning Representations (ICLR)*, 2017. 5

[10] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, pages 1097–1105, 2012. 1

[11] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018. 2

[12] Joseph Lemley, Shabab Bazrafkan, and Peter Corcoran. Smart augmentation learning an optimal data augmentation strategy. *IEEE Access*, 5:5858–5869, 2017. 2

[13] Ruihui Li, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-GAN: a point cloud upsampling adversarial network. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 7203–7212, 2019. 8

[14] Sungbin Lim, Ildoo Kim, Taesup Kim, Chiheon Kim, and Sungwoong Kim. Fast AutoAugment. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, 2019. 2, 3

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In *European Conf. on Computer Vision (ECCV)*, pages 740–755, 2014. 1

[16] Bo Liu, Xudong Wang, Mandar Dixit, Roland Kwitt, and Nuno Vasconcelos. Feature space transfer for data augmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9090–9098, 2018. 2

[17] Yongcheng Liu, Bin Fan, Gaofeng Meng, Jiwen Lu, Shiming Xiang, and Chunhong Pan. DensePoint: Learning densely contextual representation for efficient point cloud processing. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 5239–5248, 2019. 2, 5, 6, 8

[18] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8895–8904, 2019. 1, 2, 5, 6, 8

[19] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3D point cloud understanding. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1578–1587, 2019. 1, 2

[20] Hsien-Yu Meng, Lin Gao, Yu-Kun Lai, and Dinesh Manocha. VV-Net: Voxel VAE net with group convolutions for point cloud segmentation. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 8500–8508, 2019. 8

[21] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NeurIPS Workshop*, 2017. 5

[22] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 4332–4341, 2019. 2

[23] Charles R. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 652–660, 2017. 1, 2, 4, 5, 6, 7, 8

[24] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, pages 5099–5108, 2017. 1, 2, 5, 6, 7

[25] Guo-Jun Qi. Loss-sensitive generative adversarial networks on Lipschitz densities. *arXiv preprint arXiv:1701.06264*, 2017. 4

[26] Marie-Julie Rakotosaona, Vittorio La Barbera, Paul Guerrero, Niloy J. Mitra, and Maks Ovsjanikov. PointCleanNet: Learning to denoise and remove outliers from dense point clouds. *Computer Graphics Forum*, 2019. To appear. 8

[27] Alexander J. Ratner, Henry Ehrenberg, Zeshan Hussain, Jared Dunnmon, and Christopher Ré. Learning to compose domain-specific transformations for data augmentation. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, pages 3236–3246, 2017. 2

[28] 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*, pages 89–98, 2016. 2, 5

[29] Yiru Shen, Chen Feng, Yaoqing Yang, and Dong Tian. Mining point cloud local structures by kernel correlation and graph pooling. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4548–4557, 2018. 2

[30] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. PointRCNN: 3D object proposal generation and detection from point cloud. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–779, 2019. 8

[31] Ashish Shrivastava, Tomas Pfister, Oncel Tuzel, Joshua Susskind, Wenda Wang, and Russell Webb. Learning from simulated and unsupervised images through adversarial training. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2107–2116, 2017. 2, 5

[32] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2530–2539, 2018. 2

[33] Zhiqiang Tang, Xi Peng, Tingfeng Li, Yizhe Zhu, and Dimitris N. Metaxas. AdaTransform: Adaptive data transformation. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 2998–3006, 2019. 2

[34] Hugues Thomas, Charles R. Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, Francois Goulette, and Leonidas J. Guibas. KPConv: Flexible and deformable convolution for point clouds. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 6411–6420, 2019. 2, 6

[35] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A Bayesian data augmentation approach for learning deep models. In *Conference and Workshop on Neural Information Processing Systems (NeurIPS)*, pages 2797–2806, 2017. 2

[36] Chu Wang, Babak Samari, and Kaleem Siddiqi. Local spectral graph convolution for point set feature learning. In *European Conf. on Computer Vision (ECCV)*, pages 52–66, 2018. 2

[37] 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 Trans. on Graphics*, 38(5):146:1–12, 2019. 1, 2, 5, 6, 8

[38] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3D ShapeNets: A deep representation for volumetric shapes. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1912–1920, 2015. 1, 2, 5

[39] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. STD: Sparse-to-dense 3D object detector for point cloud. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1951–1960, 2019. 8

[40] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. PU-Net: Point cloud upsampling network. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2790–2799, 2018. 8

[41] 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 *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 6023–6032, 2019. 2

[42] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *Int. Conf. on Learning Representations (ICLR)*, 2018. 2

[43] Xinyu Zhang, Qiang Wang, Jian Zhang, and Zhao Zhong. Adversarial AutoAugment. In *Int. Conf. on Learning Representations (ICLR)*, 2020. 2

[44] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. ShellNet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *IEEE Int. Conf. on Computer Vision (ICCV)*, pages 1607–1616, 2019. 6

[45] Hengshuang Zhao, Li Jiang, Chi-Wing Fu, and Jiaya Jia. PointWeb: Enhancing local neighborhood features for point cloud processing. In *IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5565–5573, 2019. 2, 8