Point Cloud Augmentation with Weighted Local Transformations (Supplementary Materials)

In this supplement, we provide detailed discussions and experimental results. This includes **1)** Implementation details of PointWOLF, **2)** Proof of Proposition 1: smoothness of our transformation, **3)** Pseudocode of PointWOLF from a perspective of the kernel regression for transformations, **4)** Additional analyses on global transformation robustness and anchor points/kernel bandwidth, **5)** Qualitative analysis on shape identity, **6)** Estimation of α^*, and **7)** Applying AugTune to CDA.

1. Implementation Details

Our PointWOLF is a local data augmentation method. Other augmentation techniques can be applied before or after our method. In our experiments, we applied several conventional data augmentations (CDAs) after the PointWOLF augmentation such as normalization and global transformations. Table 1 shows the default augmentation range of PointWOLF with ModelNet40 (MN40) [1], ScanObjectNN (SONN) [2], and ShapeNetPart [3] respectively, where M is the number of anchor points, h is a kernel bandwidth, ρ_r is a rotation range, ρ_s is a scaling range, and ρ_t is a translation range. For synthetic data such as MN40 and ShapeNetPart, we adopt a relatively *small* local transformation range based on the assumption that a relatively less strong augmentation will be beneficial since the point clouds are pre-aligned and less noisy. On the other hand, in the case of real-world data, we adopt a larger range since we observe that the dataset has a larger variability. Furthermore, we use the projection matrix $\Pi = \text{diag}(\pi_x, \pi_y, \pi_z)$ for the binary axis selection follows $\pi_x, \pi_y, \pi_z \sim \text{Bernoulli}(0.5)$. On top of this, we do an additional projection of the axes for local scaling and translation for more shape diversity.

Table 1. **Default Augmentation Range.**

Dataset	(M, h, ρ_r, ρ_s, ρ_t)
ModelNet40	(4,0.5,10,3,0.25)
ShapeNetPart	(4,0.3,15,3,1)
ScanObjectNN	(4,0.3,30,3,1)

2. Proof of Smooth Transformation

Proposition 1 *If a kernel function $K_h(\cdot,\cdot)$ and all local transformations $\{T_j\}_{j=1}^M$ are smooth, then the locally weighted transformation $\hat{T}(\cdot)$ in (1) is a smooth transformation.*

Proof: Let the kernel function $K_h : \mathbb{R}^3 \to \mathbb{R}$ and all local transformations $\{T_j\}_{j=1}^M : \mathbb{R}^3 \to \mathbb{R}^3$ are smooth functions. We will show that the locally weighted transformation $\hat{T}(\mathbf{p})$ is a smooth transformation, where $\hat{T}(\mathbf{p})$ is written as

$$\hat{T}(\mathbf{p}) = \frac{\sum_{j=1}^M K_h(\mathbf{p}, \mathbf{p}_j^{\mathcal{A}}) T_j(\mathbf{p})}{\sum_{k=1}^M K_h(\mathbf{p}, \mathbf{p}_k^{\mathcal{A}})} = \sum_{j=1}^M \tilde{K}_j(\mathbf{p}) T_j(\mathbf{p}). \tag{1}$$

We simply rewrite $\frac{K_h(\mathbf{p}, \mathbf{p}_j^{\mathcal{A}})}{\sum_{k=1}^M K_h(\mathbf{p}, \mathbf{p}_k^{\mathcal{A}})}$ as $\tilde{K}_j(\mathbf{p})$. The smoothness of $\tilde{K}_j(\mathbf{p})$ where the denominator is non-zero is trivial. This follows from the sum rule and the quotient rule for derivatives [4], where it holds under the mild conditions (*i.e.*, $K_h(\mathbf{p}_i, \mathbf{p}_j) > 0$ for $\forall \mathbf{p}_i, \forall \mathbf{p}_j$).

If $\tilde{K}_j(\mathbf{p}) T_j(\mathbf{p})$ is smooth for each j, $\hat{T}(\mathbf{p})$ is smooth by the sum rule [4]. Thus proving the smoothness of $\tilde{K}_j(\mathbf{p}) T_j(\mathbf{p})$ for an arbitrary j is sufficient to prove the smoothness of $\hat{T}(\mathbf{p})$. For notational simplicity, let us rewrite $T_j(\mathbf{p})$ as $T(\mathbf{p})$ and $\tilde{K}_j(\mathbf{p})$ as $\tilde{K}(\mathbf{p})$. In addition, $\mathbf{p}_{(1)}$, $\mathbf{p}_{(2)}$, $\mathbf{p}_{(3)}$ denote the values of x, y, z coordinate of $\mathbf{p}$ respectively, *i.e.*, $\mathbf{p} = \begin{bmatrix}\mathbf{p}_{(1)} & \mathbf{p}_{(2)} & \mathbf{p}_{(3)}\end{bmatrix}^\top$. This notation is also used for $T(\mathbf{p})$ and $F(\mathbf{p})$, where $F(\mathbf{p}) = \tilde{K}(\mathbf{p}) T(\mathbf{p})$.

In order to prove that F is smooth, we should show that for each $n \in \mathbb{N}$, all partial derivatives $\frac{\partial^n F_{(i)}}{\partial \mathbf{p}_{(1)}^{n_1} \partial \mathbf{p}_{(2)}^{n_2} \partial \mathbf{p}_{(3)}^{n_3}}$ exist and are continuous for every non-negative n_1, n_2, n_3, such that $n_1 + n_2 + n_3 = n$.

First, consider the case $n = 1$. We can obtain a partial derivative of F by using the product rule [4].

$$\frac{\partial F_{(i)}}{\partial \mathbf{p}_{(j)}} = T_{(i)} \frac{\partial \tilde{K}}{\partial \mathbf{p}_{(j)}} + \tilde{K} \frac{\partial T_{(i)}}{\partial \mathbf{p}_{(j)}}, \quad i, j = 1, 2, 3. \tag{2}$$

Similar to Eq. (2), for arbitrary $n \geq 1$, the partial derivatives of F is obtained by simply using the product rule [4]. $\frac{\partial^n F_{(i)}}{\partial \mathbf{p}_{(1)}^{n_1} \partial \mathbf{p}_{(2)}^{n_2} \partial \mathbf{p}_{(3)}^{n_3}}$ can be written as

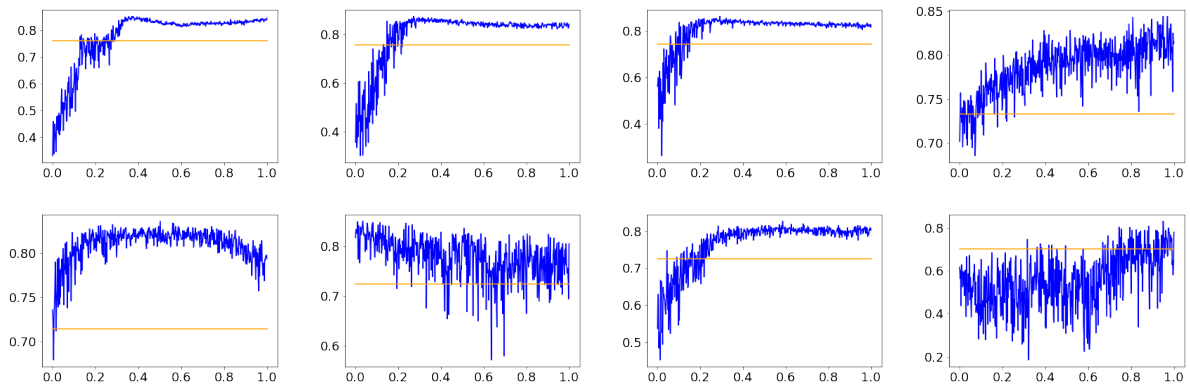

Figure 1. **Confidence Score Curves to α with Toy Examples.** Confidence scores of $\mathcal{P}^*$ (blue) are based on different $\alpha \in [0, 1]$. Desired difficulty (orange line '-') is $c = \lambda c_{\mathcal{P}}$. The results suggest that the optimization by gradient-based algorithms is impractical since fluctuating curves have too many local minima.

$$\sum_{\substack{k_1\in\{0,\dots,n_1\}\\ k_2\in\{0,\dots,n_2\}\\ k_3\in\{0,\dots,n_3\}}} c_{k_1,k_2,k_3} \frac{\partial^{k_1+k_2+k_3}\tilde{K}}{\partial \mathbf{p}_{(1)}^{k_1}\partial \mathbf{p}_{(2)}^{k_2}\partial \mathbf{p}_{(3)}^{k_3}} \frac{\partial^{n-k_1-k_2-k_3}T_{(i)}}{\partial \mathbf{p}_{(1)}^{n_1-k_1}\partial \mathbf{p}_{(2)}^{n_2-k_2}\partial \mathbf{p}_{(3)}^{n_3-k_3}}, \tag{3}$$

where c_{k_1,k_2,k_3} is a coefficient. Also, it is well known that the product of two continuous functions is continuous [4], thus F is smooth. Hence, the locally weighted transformation $\hat{T}(\mathbf{p})$ is a smooth transformation. □

3. Kernel Regression for Transformations

As mentioned in the main paper, we derive the PointWOLF algorithm from the perspective of kernel regression for transformations. Before the derivation, we briefly revisit the local transformation and augmentation process. The point $\mathbf{p}_i$ after local transformation according to anchor point $\mathbf{p}_j^{\mathcal{A}}$ as follows:

$$\mathbf{p}_i^j = \mathbf{S}_j\mathbf{R}_j(\mathbf{p}_i - \mathbf{p}_j^{\mathcal{A}}) + \mathbf{b}_j + \mathbf{p}_j^{\mathcal{A}}. \tag{4}$$

When the kernel function is equal to

$$w_i^j = \exp\left(\frac{-\|\Pi_j(\mathbf{p}_i - \mathbf{p}_j^{\mathcal{A}})\|_2^2}{2h^2}\right), \tag{5}$$

the augmented point becomes

$$\mathbf{p}_i' = \frac{\sum_{j=1}^M w_i^j \mathbf{p}_i^j}{\sum_{k=1}^M w_i^k}. \tag{6}$$

The above method transforms each point M times by the local transformations and interpolates them by the weights using a kernel. Alternatively, we can create and apply a smoothly varying transformation for each point once by weighting local transformations. We can decompose the weighted local transformation into a transformation matrix $\mathbf{A}_i'$ and a translation vector $\mathbf{b}_i'$ as following:
Transform matrix $\mathbf{A}_i'$ is denoted as

$$\mathbf{A}_i' = \frac{\sum_{j=1}^M w_i^j \mathbf{S}_j\mathbf{R}_j}{\sum_{k=1}^M w_i^k}, \tag{7}$$

and transform vector $\mathbf{b}_i'$ is denoted as

$$\mathbf{b}_i' = \frac{\sum_{j=1}^M w_i^j(-\mathbf{S}_j\mathbf{R}_j\mathbf{p}_j^{\mathcal{A}} + \mathbf{b}_j + \mathbf{p}_j^{\mathcal{A}})}{\sum_{k=1}^M w_i^k}. \tag{8}$$

Algorithm 1 PointWOLF as Kernel Regression

Input: original point cloud $\mathcal{P} \in \mathbb{R}^{3\times N}$, kernel bandwidth h
Input: # points N, # anchor points M
Input: range for scaling ρ_s, range for rotation ρ_r, range for translation ρ_t, axis dropout probability β
Output: augmented point cloud $\mathcal{P}' \in \mathbb{R}^{3\times N}$

$\mathcal{P}^{\mathcal{A}} \leftarrow$ FPS$(\mathcal{P}, M)$ ▷ $\mathcal{P}^{\mathcal{A}} \in \mathbb{R}^{3\times M}$
for $j = 1$ to M **do**
  $\mathbf{S}_j \leftarrow \text{diag}(s_x, s_y, s_z)$ ▷ $s \sim \text{U}_{[1,\rho_s]}$
  $\mathbf{R}_j \leftarrow \text{RotationMatrix}(\theta_x, \theta_y, \theta_z)$ ▷ $\theta \sim \text{U}_{[-\rho_r,\rho_r]}$
  $\mathbf{b}_j \leftarrow (b_x, b_y, b_z)$ ▷ $b \sim \text{U}_{[-\rho_t,\rho_t]}$
  $\mathbf{\Pi}_j \leftarrow (\pi_x, \pi_y, \pi_z)$ ▷ $\pi \sim \text{Bernoulli}(\beta)$
end for
for $i = 1$ to N **do**
  for $j = 1$ to M **do**
    $w_i^j \leftarrow K_h(\mathbf{p}_i, \mathbf{p}_j^{\mathcal{A}}; \Pi_j)$ ▷ Eq. (5)
  end for
  $\mathbf{A}_i' \leftarrow \frac{\sum_{j=1}^M w_i^j \mathbf{S}_j\mathbf{R}_j}{\sum_{k=1}^M w_i^k}$ ▷ Eq. (7)
  $\mathbf{b}_i' \leftarrow \frac{\sum_{j=1}^M w_i^j(-\mathbf{S}_j\mathbf{R}_j\mathbf{p}_j^{\mathcal{A}} + \mathbf{b}_j + \mathbf{p}_j^{\mathcal{A}})}{\sum_{k=1}^M w_i^k}$ ▷ Eq. (8)
  $\mathbf{p}_i' \leftarrow \mathbf{A}_i'\mathbf{p}_i + \mathbf{b}_i'$ ▷ Eq. (9)
end for
$\mathcal{P}' \leftarrow \{\mathbf{p}_i'\}_{i=1}^N$

Once the smoothly varying transformation $\mathbf{A}'$ and $\mathbf{b}'$ are obtained, we apply them to $\mathbf{p}_i$ as

$$\hat{T}(\mathbf{p}_i) = \mathbf{A}'_i \mathbf{p}_i + \mathbf{b}'_i. \tag{9}$$

The pseudocode derived from this interpretation is provided in Algorithm 1.

4. Additional Analyses of PointWOLF

For further analyses, we use PointNet++ as the base model.

4.1. Robustness against Global Transformations

We evaluate the robustness to global transformations by adding different rotations and scalings to MN40. As shown in Table 2, we observe PointWOLF highly improves the robustness to global transformations (*e.g.*, 4.7% for z-axis rotation and 9.7% for scaling by $2\times$) except y-axis rotation.

4.2. # Anchor points M and Kernel Bandwidth h

According to Table 3, PointWOLF consistently outperforms the base model (PointNet++ with 86.6%), suggesting it is fairly robust across varying # anchor points M and kernel bandwidth h with SONN. Both M and h are both related to kernel effects on PointWOLF algorithm. To explore the kernel effects on the PointWOLF alone, we did **not** use AugTune for this analysis. Especially, we assume that if M is small, large local influence h at each anchor point is preferred as it can produce a good augmentation maintaining its original shape (*e.g.*, for $M = 2$, worst at $h = 0.1$, best at $h = 0.5, 0.7$). Conversely, small local influence is preferred with large M due to diversity (*e.g.*, for $M = 256$, worst at $h = 0.7$, best at $h = 0.1$). The only worse result from $(M, h) = (2, 0.1)$ can be explained as generation of some heavily augmented objects due to extreme contrast between two local influence. Visualizations are provided in Section 5.

5. Qualitative Analysis on Shape Identity

As mentioned in the main paper, if the shape identity is lost by strong augmentation (*e.g.*, PointWOLF with small number of anchor points and small local influence), AugTune adaptively adjusts the magnitude of augmentation and preserve the shape identity. Moreover, we rarely observed such failure samples with reasonable parameters. Herein in Figure 2 we provide additional visualization of unrealistic samples for analysis. Given the original object, PointWOLF with extreme parameters $(M, h) = (2, 0.1)$ sometimes generates unrealistic samples (top row). Then, AugTune works as a safeguard to preserve the shape identity (bottom row) by interpolating between the original and the augmented samples.

Table 2. **Robustness to global Transformations.** R:Rot, S:Scale.

Method	R:X-axis	R:Y-axis	R:Z-axis	S:0.6	S:2.0
CDA	74.6	**89.4**	74.2	83.3	62.4
PointWOLF	**78.6**	88.3	**78.9**	**89.1**	**72.1**

Table 3. **Anchor points and kernel bandwidth analysis.**

h	$M = 2$	$M = 4$	$M = 16$	$M = 32$	$M = 256$
0.1	76.9	89.0	87.6	88.3	88.8
0.3	88.3	**89.7**	88.0	88.3	88.5
0.5	88.5	88.5	88.5	88.1	88.5
0.7	88.5	87.3	88.1	87.8	87.3

Algorithm 2 **Estimation of α^***

Input: original point cloud $\mathcal{P} \in \mathbb{R}^{3\times N}$, ground truth $\mathbf{y}$
Input: classifier $f(\cdot; \mathbf{w})$, difficulty coefficient λ
Output: α

$\mathcal{P}' \leftarrow$ PointWOLF$(\mathcal{P})$ ▷ Algorithm 1 in Section 3.2
$\hat{\mathbf{y}}_{\mathcal{P}} \leftarrow f(\mathcal{P}; \mathbf{w})$, $\hat{\mathbf{y}}_{\mathcal{P}'} \leftarrow f(\mathcal{P}'; \mathbf{w})$
$c_{\mathcal{P}} \leftarrow \hat{\mathbf{y}}_{\mathcal{P}}^{\top}\mathbf{y}$, $c_{\mathcal{P}'} \leftarrow \hat{\mathbf{y}}_{\mathcal{P}'}^{\top}\mathbf{y}$ ▷ confidence scores
$c \leftarrow \max(c_{\mathcal{P}'}, (1-\lambda)c_{\mathcal{P}})$ ▷ target confidence scores
Estimating α^* with one of method below

Method 1 Gradient Descent

Input: learning rate γ, max iteration k

Initialize α
for $i = 1$ to k **do**
  $\mathcal{L} \leftarrow \| c - f(\alpha\mathcal{P} + (1-\alpha)\mathcal{P}')^{\top}\mathbf{y} \|^2$
  $\alpha \leftarrow \alpha - \gamma\nabla_{\alpha}\mathcal{L}$ ▷ Gradient descent for optimizing Eq. (10)
end for
$\alpha^* \leftarrow \alpha$

Method 2 Grid Search

$\mathbf{H} \leftarrow [0, 0.1, \cdots, 0.9, 1]$
$\alpha^* \leftarrow \underset{\alpha\in\mathbf{H}}{\operatorname{argmin}} \| c - f(\alpha\mathcal{P} + (1-\alpha)\mathcal{P}')^{\top}\mathbf{y} \|^2$ ▷ Eq. (10)

Method 3 Linear Approximation

$\alpha^* \leftarrow \dfrac{c - c_{\mathcal{P}'}}{c_{\mathcal{P}} - c_{\mathcal{P}'}}$ ▷ approximate α by Eq. (11)

6. Estimating α^* for AugTune

The goal of AugTune is to find the optimal solution α^* to the following optimization problem.

$$\alpha^* = \underset{\alpha}{\operatorname{argmin}} \| c - f(\alpha\mathcal{P} + (1-\alpha)\mathcal{P}') \|^2 . \tag{10}$$

Ideally, AugTune with α^* generates the augmented sample with the exact target difficulty c. In this section, we describe three approaches we have tried to estimate α^*, which

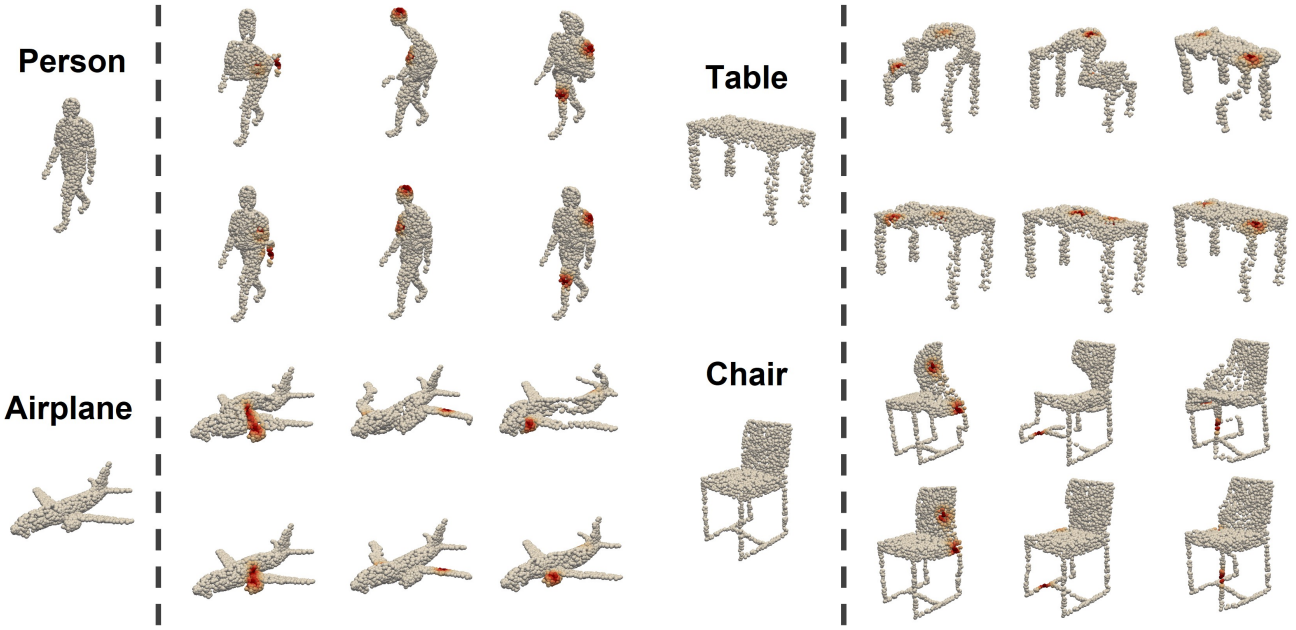

Figure 2. **Unrealistic samples of PointWOLF.** Given an original object, PointWOLF with extreme parameters sometimes generates unrealistic samples via strong augmentation (Top). However, AugTune linearly interpolates the augmented sample with the original sample to alleviate the difficulty, resulting in the shape identity preservation (Bottom).

is specified in Algorithm 2. The first approach is the gradient descent method. But the direct optimization seems to be *infeasible*. In Figure 1, the confidence score curves with respect to α largely fluctuate. It suggests that the optimization by gradient-based methods is not just computationally expensive but also almost impossible to reach the target difficulty from a random initial point.

Next, we consider a grid search on α, which is less time-consuming when the grid is coarse. We select α with the smallest error to the target confidence score among a predefined list of values between 0 and 1 with 0.1 intervals. The grid search leads to a reasonably good α with similar difficulty as targeted but still, it takes a substantial amount of time. The computational time linearly increases as the number of the predefined values increases although the small interval for α does not guarantee better performance.

Lastly, we approximate α^* by the convex combination in confidence space.

$$\alpha c_{\mathcal{P}} + (1-\alpha) c_{\mathcal{P}'} = c. \tag{11}$$

Table 4 illustrates the experimental comparison between the grid search and the linear approximation. Higher performance from the linear approximation (89.7%) shows that it is the simplest yet effective approach. Hence, we choose this approach as the final estimation method for AugTune.

Table 4. **Various approximation of α.**

Method	CDA	Grid Search	Approx
Acc	86.6	89.0	89.7

Table 5. **Applying AugTune to CDA.**

Search Space	w/o AugTune	w/ AugTune
S	87.8	88.7
$2S$	87.7	88.1
$3S$	86.7	87.0
$4S$	85.1	86.9

7. Applying AugTune to CDA.

We demonstrate the extensibility of AugTune by applying to classical data augmentations in point clouds, *i.e.* CDAs. We employ ReducedMN40 with PointNet++ for ablation. Except for rotation, scaling, translation, and point-wise jittering are default augmentations for ReducedMN40 as it has already been known to be pre-aligned. We set the augmentation range S same as [5], *e.g.*, (ρ_s=$[0.8, 1.2]$, ρ_t=$[-0.1, 0.1]$, σ=0.01) and use the multiples of the augmentation ranges: kS=($k\rho_s$, $k\rho_t$, $k\sigma$). σ denotes the standard deviation of Gaussian noise. For the difficulty coefficient λ, we perform a grid search with a range of [0.7, 0.9] with 0.05 intervals. Table 5 shows that CDA **w/ AugTune** outperforms CDA **w/o AugTune** by 0.3 % $\sim$ 1.8 %. Notice that additional tuning to basic CDA in default space

S shows prominent improvement, where it achieves comparable performance to other complex augmentations. This observation follows our previous assumption in Section 1 of less strong augmentation being preferred by synthetic dataset. Therefore, we believe that AugTune benefits not just PointWOLF, but also other classical data augmentation methods.

References

[1] Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*, 2015. 1

[2] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Duc Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*, 2019. 1

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

[4] H Anton, Irl Bivens, and S Davis. *Calculus Early Transcendentals, 10th Edition E-Text*. 01 2011. 1, 2

[5] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 4