Test-Time Augmentation for 3D Point Cloud Classification and Segmentation

Tuan-Anh Vu[1*]   Srinjay Sarkar[2*]   Zhiyuan Zhang[3]   Binh-Son Hua[2,4‡]   Sai-Kit Yeung[1]

[1]The Hong Kong University of Science and Technology   [2]VinAI Research
[3]Singapore Management University   [4]Trinity College Dublin
[*]co-first author   [‡]corresponding author

Abstract

Data augmentation is a powerful technique to enhance the performance of a deep learning task but has received less attention in 3D deep learning. It is well known that when 3D shapes are sparsely represented with low point density, the performance of the downstream tasks drops significantly. In this work, we explore test-time augmentation (TTA) for 3D point clouds. We are inspired by the recent revolution of learning implicit representation and point cloud upsampling, which can produce high-quality 3D surface reconstruction and proximity-to-surface, respectively. Our idea is to leverage the implicit field reconstruction or point cloud upsampling techniques as a systematic way to augment point cloud data. Particularly, we test both strategies by sampling points from the reconstructed results and using the sampled point cloud as test-time augmented data. We show that both strategies are effective in improving accuracy. We observed that point cloud upsampling for test-time augmentation can lead to more significant performance improvement on downstream tasks such as object classification and segmentation on the ModelNet40, ShapeNet, ScanObjectNN, and SemanticKITTI datasets, especially for sparse point clouds.

1. Introduction

Point-based representation is of great importance to computer graphics and computer vision. In the modern era of deep learning, neural networks can be designed to learn features from point clouds, facilitating 3D perception tasks such as object classification, object detection, and semantic segmentation in many downstream applications. Nevertheless, such evolutions still leave 3D perception a challenging and unsolved problem. A typical disadvantage of point-based representation is that surface information is implied by point density and orientation, if any. Due to such ambiguity, techniques for data augmentation on point clouds are relatively scarce and challenging to design.

Recent advances in using neural networks to represent 3D data have opened new opportunities to revise and explore this problem from a new perspective [45, 48, 69]. One type of method is the so-called neural implicit representation based on the idea of training a neural network that can return queries of the 3D space from input coordinates [45, 47, 48, 59]. Particularly, one can train a neural network to encode a 3D point to various attributes such as occupancy, color, or a general feature vector. The power of a neural implicit representation is that the queries can be performed at arbitrary points, and no special mechanism is required for value interpolation. Another type of methods [36, 69, 77] employs upsampling to achieve both distribution uniformity and proximity-to-surface. The advantages of the upsampling-based method lie in self-supervision and more uniformly distributed dense representation without the need of surface ground truth.

In this work, we investigate both types of strategies and leverage them as a systematic way for data augmentation at test time. Particularly, for implicit representation, we leverage the convolutional occupancy network [48] to encode the 3D point clouds to a regular grid representation that allows the interpolation of features at an arbitrary location. For the upsampling-based method, we employ the self-upsampling method [69] to obtain a dense and uniformly distributed proximity-to-surface point cloud. We propose an effective technique to aggregate features of the original and augmented point clouds to generate the final prediction. We select the task of object classification and semantic segmentation as the downstream task to validate our augmentation technique, as they play a key role in many practical applications, including perception in robotics and autonomous driving. We experiment with point cloud data from ModelNet40 [71], ShapeNet [7], ScanObjectNN [63] and SemanticKITTI [4] dataset, which demonstrates significant performance improvement.

In summary, our key contributions are:
• We analyze and compare existing reconstruction ap-

proaches, including surface-based sampling and point cloud upsampling for test-time augmentation.

- We propose a test-time augmentation method for 3D point cloud deep learning, which is suitable for both approaches;
- We identified a self-supervised point cloud upsampling method as a robust method for our test-time augmentation. It uses the proximity-to-surface cues to sample augmented point clouds.
- Extensive experiments and analysis prove the effectiveness of our augmentation method on two downstream tasks, including object classification and semantic segmentation on synthetic and real-world datasets.

2. Related Works

3D Deep Learning. Early methods usually convert the irregular and sparse 3D points into multiple regular 2D views [12, 26, 72] or 3D voxels [13, 24, 62, 68, 71]. Despite the improvements gained by performing CNN on these regular structures, these methods usually suffer information loss and high computational costs.

Point cloud is a universal representation for 3D data. PointNet [50] is the pioneering work that can process 3D points directly by symmetric functions and max-pooling to extract global features. To capture local features, Point-Net++ [51] performs hierarchical PointNets on different scales. In recent years, various convolution operators and networks have been proposed for point clouds, such as PointCNN [37], SpiderCNN [74], DGCNN [66], and Shell-Net [79] with the supreme performance achieved on classification, retrieval, and segmentation tasks. There are also approaches [16, 21, 25, 65, 73] specially designed for semantic segmentation. Huang et al. [25] learn the local structure, particularly for semantic segmentation, by applying learning algorithms from recurrent neural networks. SPG [32] constructs graphs between coarsely segmented super-points for large-scale point cloud semantic segmentation.

To balance the efficiency and accuracy, hybrid works [33, 42, 78] utilize the characteristics of multiple representations. PointGrid [33] assigns fix number of points in each grid cell, making the conventional CNN feasible. While it runs fast, the accuracy is still not high. PVCNN [42] represents the input in points and performs the convolutions in voxels with a superior performance achieved than sole point or voxel representations. To handle large-scale lidar point cloud data, FusionNet [78] divides the input into voxels and extracts features from both voxels and inner points.

Neural 3D Reconstruction. 3D reconstruction works can be classified into four categories in terms of the output representation: voxel-based, point-based, mesh-based, and implicit function-based methods.

Similar to semantic segmentation, voxel is also a popular representation for 3D reconstruction [11, 70, 71]. In the

category, voxel grids are used to store either occupancy that encodes whether the voxel is occupied or not [11, 71] or SDF information that holds signed projective distances from voxels to the closest surfaces [14, 38, 60]. However, as mentioned in the segmentation works, such methods inherit the limitations of high memory costs.

Another line of works output point clouds directly for 3D reconstruction [15, 39, 49, 75]. These methods design generative models to produce dense points for scene representation. Despite the efficiency, the generated points cannot sufficiently represent complicated surfaces as there is no topology between the points.

Mesh is another popular output representation for 3D reconstruction. In this category, some works deform shapes with simple topology to more complicated shapes, which usually constrain to certain fixed templates [27, 53] or topologies [5, 58]. To reconstruct a shape of arbitrary topology, AtlasNet [18] warps multiple 2D planes into 3D shapes. Despite the superior results, this method can result in self-intersecting mesh faces.

To overcome the limitations of the above explicit representations (voxel, point, mesh), more recent works focus on implicit representations that employ occupancy [45, 48] and distance field [6, 47] with a neural network to infer an occupancy probability or distance value for the input 3D points. As implicit representation models shape continuously, more detail is preserved, and more complicated shape topologies can be obtained. In this work, we also employ implicit representation to aim to augment a point cloud for various downstream tasks.

Point Cloud Upsampling. Point cloud upsampling can produce a dense, uniform, and complete point cloud from a sparse and noisy complete with or without missing parts.

Traditional Point Cloud Upsampling: A seminal point cloud upsampling algorithm is to interpolate points as vertices of a Voronoi diagram [1]. [40] later proposed an algorithm by introducing the locally optimal linear projector for surface reconstruction and using it to project a set of points onto the input point cloud. This work was followed by [22], who proposed a weighted locally linear operator in order to make the point cloud distribution more even. [23] introduces an edge-aware resampling method by sampling points on edge and calculating the normals at those points. All of the above-mentioned methods are not data-driven and thus heavily rely on priors like normal estimation.

Deep-Learning Based Point Cloud Upsampling: PU-Net [77] was the first deep learning-based point cloud upsampling method that used a multi-branch feature expansion module to extract multi-scale features and expand a point cloud in the feature space. This was followed by EC-Net [76], which achieves edge-aware point cloud upsampling by learning distance features obtained by the perturbation of
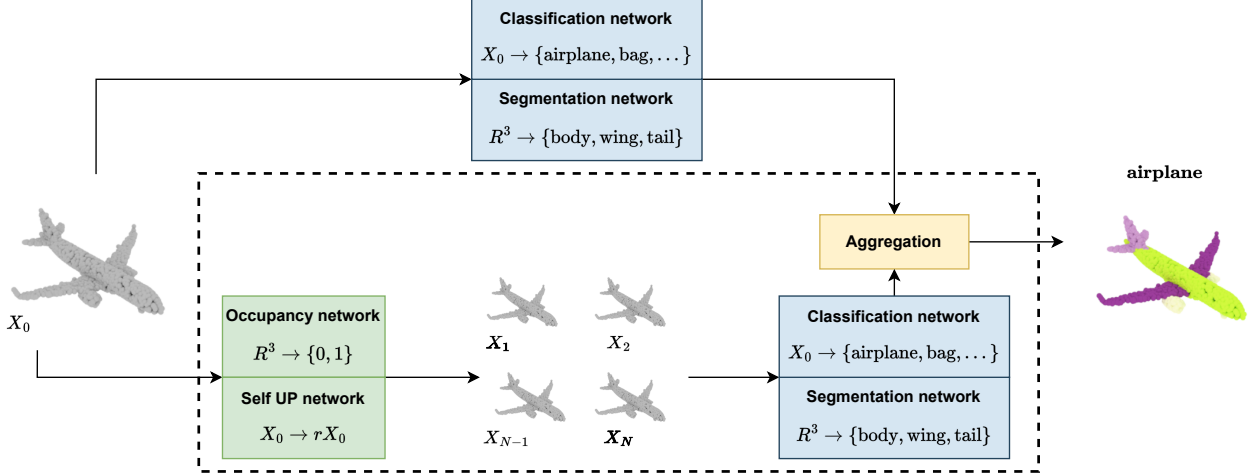
Figure 1. Illustration of our test-time augmentation method for point clouds downstream tasks such as classification and segmentation. We view the input point cloud as a noisy estimate of a latent surface and propose using an implicit field represented by an occupancy network or a point cloud upsampling network to sample augmented point clouds so that the point clouds share the same underlying surfaces. We then perform the downstream task on each point cloud and aggregate the point features to produce the final result.

the generated point cloud relative to the input point cloud. In this work, we propose to use point cloud upsampling as a test-time augmentation technique.

Data Augmentation and Test-Time Augmentation. In modern deep learning, large-scale data is often required for training deep neural networks; however, acquiring a large amount of data is a thorough and prohibitively expensive process. Data augmentation is a common but useful technique to scale up the data artificially. In image classification, popular data augmentation includes simple transformations of the images, including rotations, flipping, cropping, etc. [19, 20, 20, 31, 57, 61]. In 3D deep learning, traditional methods (e.g., PointNet [50] and PointNet++ [51]) utilize similarity transformations such as random rotations, scaling, and jittering for data augmentation during training. Similarity transformations are also used to augment real-world data to build ScanObjectNN [63], a real-world dataset for object classification. Research efforts for more sophisticated augmentation techniques for 3D point clouds are relatively scarce. Recently, PointMixup [8] is proposed to mix two point clouds based on shortest path linear interpolation; PointAugment [35] uses adversarial learning to seek augmented point clouds satisfying a given classifier. PPBA [9] automates the design of augmentation policies for the specific task of 3D object detection. Apart from train-time data augmentation, in this work, we assume that pre-trained models for specific downstream tasks are already given and investigate test-time augmentation techniques [3] that can boost overall performance.

Particularly, Test-time augmentation (TTA) first transforms the input, then perform predictions on the augmented versions of the input, and finally combines the prediction results. This strategy is common to image classification with simple augmentation policies such as flipping, cropping, and scaling [20]. More sophisticated methods involve learning and optimizing augmentation policies [29, 44, 55, 56], or learning to combine predictions [56]. Beyond images, TTA has also been applied to medical image segmentation [46, 64] and text recognition [34]. For point clouds, however, we are unaware of any recent method tailored to test-time augmentation and augmentation policies.

3. Our Method

3.1. Overview

Given a point set $\{\mathbf{p}_i\}_{i=1}^n$ with $\mathbf{p}_i \in \mathbb{R}^3$ represented by a matrix $\mathbf{x}_0 \in \mathbb{R}^{n \times 3}$. Without loss of generality, we assume at inference, $\mathbf{x}_0$ is passed to pre-trained network f for feature extraction, and the features are passed to a network g for final label prediction $f(\mathbf{x}_0)$. Our goal is to achieve performance improvement in the downstream task via test-time augmentation, where the final prediction can be defined as:

$$g(\phi(f(\mathbf{x}_0), f(\mathbf{x}_1), f(\mathbf{x}_2), ...)) \tag{1}$$

where ϕ is an aggregation function to combine multiple features resulting from the original point set $\mathbf{x}_0$ and the augmented point sets $\mathbf{x}_1$, $\mathbf{x}_2$, etc. Note that the network f and g are pre-trained and left untouched in test-time augmentation; only the input is augmented.

Traditionally, a simple method for test-time augmentation is jittering, which adds Gaussian noise to perturb the point
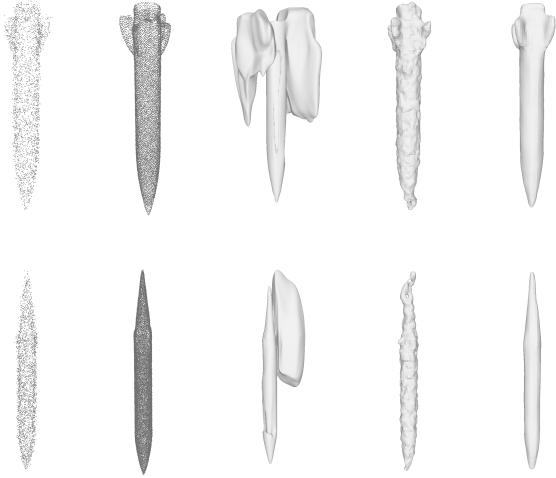
Figure 2. **Visual comparison of different reconstruction and upsampling methods.** From left to right: Input, Up-sampling point clouds, Unsupervised reconstruction, Poisson reconstruction, and Supervised reconstruction. As can be seen, the shape quality of supervised reconstruction [48] using neural implicit representation is finer and smoother compared to unsupervised method [17] and Screened Poisson method [28]. In addition, we can obtain a dense and uniformly distributed proximity-to-surface point cloud using Self-supervised Upsampling [69], which contributes to the success of our method. Best viewed with zoom.

cloud $\mathbf{x}_0$ to generate an augmented point cloud $\mathbf{x}_k$:

$$\mathbf{x}_k = \mathbf{x}_0 + \lambda \mathbf{z}_k \qquad (2)$$

where $\mathbf{z}_k \sim \mathcal{N}(0, I)$ is a random noise vector from a normal distribution, and λ is a scalar value to control the noise magnitude. This simple augmentation has been widely adopted since the seminal PointNet [50]. An issue of such augmentation is that it does not consider the underlying surface or point distribution because the noise $\mathbf{z}$ is independent of $\mathbf{x}_0$, resulting in marginal performance improvement in many cases. In this work, we viewpoint set $\mathbf{x}_0$ as a noisy estimate of a latent surface representation $\mathcal{S}$, and therefore, we define point cloud augmentation as the process of sampling additional point clouds $\mathbf{x}_k$ that explain the same surface. We propose to sample augmented point clouds $\mathbf{x}_k$ ($k \geq 1$) in two ways: surface sampling and point cloud up-sampling. The sampled point clouds can then be leveraged for downstream tasks such as classification and segmentation. Our method is visualized in Figure 1.

In the following sections, we explain the technique for sampling augmented point clouds using an implicit representation network (Section 3.2) and a self-supervised point upsampling network (Section 3.3). We then present downstream tasks that leverage the proposed test-time augmentation and discuss feature aggregation and final label prediction

for point cloud classification and segmentation.

3.2. Augmentation by Implicit Field Reconstruction

We are motivated by the recent advances in geometry reconstruction using neural implicit representation. The basic idea is to learn a mapping $f_\theta : \mathbb{R}^3 \longrightarrow \{0, 1\}$ using a neural network parameterized by θ. This function implicitly encodes the geometry in the 3D space to allow the query of the occupancy at any point in the 3D space. To obtain the geometry explicitly, the Marching Cubes algorithm [43] can generate a triangle mesh containing surfaces at zero crossings in the implicit field. Our neural implicit field is built upon the convolutional occupancy network [45, 48]. The convolutional occupancy network uses a combination of convolutional and linear layers, thus endowing its features with equivariance and scalability. This enables the network to produce implicit representations for both single objects and large-scale scenes. Our implementation uses the network variant that stores features on a 3D regular grid.

Encoder. The encoder is a shallow PointNet [50] but with local pooling layers. By using these input features generated by the local PointNet encoder, we obtain a 32^3 volumetric feature grid that captures the local information in the neighborhood of the points, which is necessary to capture local geometric information about the shape of the input point cloud. Due to memory constraints, the volumetric feature can represent rich 3D information but is restricted to small resolutions and sizes.

Decoder. To endow the encoder features with inductive bias, the occupancy network uses a 3D UNet encoder [81] to process the volumetric feature grid. Since U-Net contains convolutional operations, this also introduces translational equivariance in the encoder features, which makes it able to predict the occupancy of different shapes but from the same categories. These aggregated feature maps from the U-Net [54] are then fed into a decoder for predicting occupancy labels. To predict the occupancy value at any arbitrary position, we use tri-linear interpolation to find the features at that point by using the features of all points belonging to the same voxel in the volumetric grid. This point's location and features are passed through a decoder that outputs an occupancy value for each 3D grid location.

Surface Sampling. We render an output mesh of the given input point cloud from the predicted occupancy of the grid points of the convolutional occupancy network using the MISE algorithm [45]. We then produce an augmented version $\mathbf{x}_k$ of the original point cloud $\mathbf{x}_0$ by randomly sampling a point cloud from the vertices of the rendered mesh, where k indicates the k-th augmentation.

3.3. Augmentation by Point Cloud Upsampling

Inspired by [69], we upsample an input sparse point cloud $\mathbf{x} = \{\mathbf{p}_i\}_{i=1}^n \in \mathbb{R}^{n \times 3}$ to a dense point cloud $\mathbf{y} = \{\mathbf{p}_i\}_{i=1}^N \in \mathbb{R}^{N \times 3}$ including $N = \lfloor r \times n \rfloor$ points, where r is a desirable scaling factor (set default to 4). The high-resolution point cloud $\mathbf{y}$ must be dense, uniform, complete, and noise-tolerant. The self-supervised point cloud upsampling strategy includes four steps: seeds sampling, surface projection, outliers removal, and arbitrary-scale point cloud generation.

Seeds Sampling. To obtain uniformly sampled seed points, given a point cloud, we divide the 3D space into equally spaced voxels and estimate the distance from centers to the surface by computing the distance to the triangles formed by the nearest points. Then we choose the centers in a preset range as the seed points.

Surface Projection. Given a seed point c, we obtain the coordinate of the projection point of the seed point c as: $c_p = c + n \times d$, where $\mathbf{n} \in [-1, 1]^3$ and $d \in \mathbb{R}$ are projection direction and projection distance, respectively. The $\mathbf{n}$ and d can be obtained by two multi-layer fully-connected neural networks f_n, and f_d, which borrows from Occupancy Network [45] and DGCNN [67]. The detail of architectures and training procedures can be found in [69].

Outliers Removal. For a projection point c_p, we determine a point as an outlier if $b_p > 1.5\overline{b}$, where b_p is the average bias between c_p and its nearest points and $\overline{b}$ is the average bias of all projection points.

In practice, outlier removal can be regarded as optional, but we empirically found that outlier removal can yield some minor performance improvement of downstream tasks such as classification and part segmentation, and therefore use this step by default in the augmentation.

Point Cloud Generation. We upsample the input point cloud $\mathbf{x}_0$ to a dense point cloud $\mathbf{y}$ using the upsampling network. Then, we sample a fixed number of points from the upsampled point cloud $\mathbf{y}$ by using the farthest-point sampling algorithm to obtain an augmented point cloud $\mathbf{x}_k$ with the desired number of points, where k indicates the k-th augmented point cloud. The examples are shown in Figure 2.

3.4. Downstream Tasks.

Object Classification. To leverage the augmented point clouds for classification, for both PointNet [50], DGCNN [66], and PointNeXt [52], we extract the global features of each point cloud $\mathbf{x}_k$ including the original point cloud $\mathbf{x}_0$, and then take an average of the features before passing them to the classifier. Without changing of notation, assume that f is the global feature extractor, and g is the

classifier, we can write the label prediction as:

$$g(\text{avgpool}(f(\mathbf{x}_0), f(\mathbf{x}_1), f(\mathbf{x}_2), ...)) \quad (3)$$

Semantic and Part Segmentation For semantic segmentation and part segmentation, the aggregation function is more evolved. The basic idea is first to perform segmentation on each point cloud, and then aggregate the results to produce the final segmentation for the original point cloud $\mathbf{x}_0$, but now the aggregation occurs at a per-point level instead of the global features. Let $f_i(\mathbf{x}_k)$ be the features of point i in point cloud $\mathbf{x}_k$, the label prediction of point i in the original point cloud $\mathbf{x}_0$ can be written as:

$$g(\phi(f_i(\mathbf{x}_0), \{f_{\pi_{1,i}}(\mathbf{x}_1)\}, \{f_{\pi_{2,i}}(\mathbf{x}_2)\}, ...)) \quad (4)$$

where $\pi_{k,i}$ indicates the corresponding points of point i in $\mathbf{x}_0$ to point cloud $\mathbf{x}_k$, and g as the classifier or any post-processing network. Here we propose a simple algorithm to establish such correspondences via nearest neighbors on the logit vectors, which are detailed in Algorithm 1.

Algorithm 1: Pseudo-code for our test-time augmentation for the segmentation task.

```
# get_log(p):   return logit at point p.
# get_feat(p): return 3D coords wo/ or w/ logit
# knn(p, X):    return nearest neighbor of p in X.
# agg(a, b):    combine tensors a and b.
for each point p in X_0:
  logit = get_log(p)
  feat = get_feat(p)
  for i = 1 to N
    neighbors = knn(feat, X_i)
    for each point q in neighbors:
      logit = agg(logit, get_log(q))
  label = argmax(logit)
```

---

4. Experimental Results

4.1. Implementation Details

We implement our method in Pytorch. We use the convolutional occupancy network [48], and self-supervised point upsampling network [69] for test-time augmentation. For downstream tasks, we experiment with pre-trained models for classification and part segmentation such as PointNet [50], DGCNN [67], PointNeXt [52] as well as for large-scale semantic scene segmentation such as RandLANet [21].

Dataset and metric. Our experiments are conducted on different datasets such as ShapeNet [7], ScanObjectNN [63], ModelNet40 [71], and SemanticKITTI [4] datasets, including indoor and outdoor environments with both synthetic and real data. the ShapeNet dataset [7].

We employ several popular metrics for evaluation, such as the overall and mean percentage accuracy are computed for the classification task, the Instance and Category

Table 1. 3D object classification in ModelNet40 [71] and ScanObjectNN [63] using self-supervised upsampling point clouds [69].

Method	ModelNet40		ScanObjectNN (PB_T50_RS)	
	oAcc	mAcc	oAcc	mAcc
PointNet [50]	89.20	86.20	68.20	63.40
Ours	**92.07**	**88.78**	**76.69**	**72.93**
DGCNN [67]	92.90	90.20	78.10	73.60
Ours	**94.23**	**91.79**	**87.71**	**85.84**
PointNeXt [52]	93.96	91.14	88.18	86.83
Ours	**95.48**	**92.96**	**90.38**	**88.99**
PointMixer [10]	91.41	87.89	82.51	80.03
Ours	**92.71**	**90.42**	**84.18**	**81.25**
PointTransformer [80]	90.64	87.84	82.31	80.77
Ours	**92.55**	**89.73**	**83.66**	**81.37**

Intersection-Over-Union (mInsIoU, mCatIoU) are utilized for the part segmentation task, and the mean Accuracy (mACC) and mean IoU (mIoU) are used for semantic segmentation task.

Data processing. For ShapeNet [7] dataset, we use the pre-processed data produced by PointNet [50], which is an early version of ShapeNet (version 0) to train the segmentation network. Nonetheless, the ShapeNet data used to train the convolutional occupancy network [48] is a different version (version 1). Since the number of objects differs in these variants of ShapeNet, we only use the objects that appear in both datasets. For ScanObjectNN [63], and ModelNet40 [71] datasets, we follow the instruction in the official implementation of PointNeXt [52]. For SemanticKITTI [4] dataset, we follow the instruction in the official implementation of RandLANet [21]. We also follow Self-UP [69] to prepare the data for point cloud upsampling.

4.2. Classification Results

The object classification results are shown in Table 1 and are conducted on two challenging datasets (ScanObjectNN [63], and ModelNet40 [71]). ScanObjectNN presents considerable problems to the various point cloud analysis algorithms already in use due to occlusions and noise. Based on PointNeXt [52], we conduct experiments on PB_T50_RS, the most challenging and widely deployed version of ScanObjectNN. Note that the reported performance of PointNet and DGCNN in our paper is higher than the original PointNet and DGCNN paper because we adopt the re-implementation of PointNet and DGCNN from the PointNeXt paper, which includes optimized training strategies.

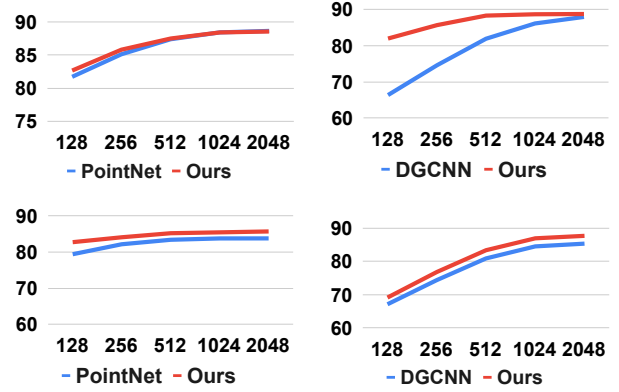As can be seen, the optimized baseline model by PointNet [50] performed very well on both ModelNet40 and

Figure 3. TTA using surface sampling (top row) and self-supervised upsampling (bottom row) on part segmentation on ShapeNet with different numbers of points. We found that applying TTA for sparse point clouds of the surface sampling method yields significant improvement. In contrast, the improvement of TTA on upsampling point clouds is more stable, thanks to dense and uniformly distributed proximity-to-surface point clouds. The horizontal axis is in log scale. Best viewed with zoom.

ScanObjectNN classification. Despite such, applying augmentation with our method leads to a performance boost of $1 - 2\%$, which is a significant gain given the saturating accuracy of this dataset. We also empirically found that augmenting with more than one sampled point cloud does not significantly improve this task.

Note that as convolutional occupancy network [48] requires ground truth signed distance functions to train surface reconstruction, we only perform the classification task using self-supervised point upsampling [69].

4.3. Segmentation Results

Part Segmentation. The part segmentation results are shown in Table 2 and Table 3. It can be seen that by applying our method, the mInsIoU, and mCatIoU are improved compared to the baseline approach. The results also demonstrate the robustness of our method as it works well with different network backbones, e.g., PointNet [50] that involves only per-point and global point cloud features, DGCNN [66] which establishes and learns dynamic graphs in point neighborhoods, and the SOTA PointNeXt [52]. It is worth noting that the improvement is mainly gained from the refinement of the segmentation boundaries (Figure 4).

Semantic Segmentation. To assess the generalizability of our strategy, we also tested on real-world data from SemanticKITTI [4] dataset. As SemanticKITTI data is captured by LiDAR sensors, it is favorable to use point upsampling as the augmentation technique. It can be seen in Table 4, by applying our method, the mAcc and mIoU are improved compared to the baseline approach.

Table 2. Part segmentation on ShapeNet [7] using self-supervised upsampling point clouds as input.

2048 points	mInsIoU	mCatIoU
PointNet [50]	80.74	83.73
Ours	**82.88**	**86.25**
DGCNN [67]	81.08	84.18
Ours	**83.38**	**86.70**
PointNeXt [52]	84.23	86.73
Ours	**85.07**	**87.60**

Table 3. Part segmentation on ShapeNet [7] using surface sampling with different numbers of points.

Method	128 points		256 points	
	mInsIoU	mCatIoU	mInsIoU	mCatIoU
PointNet [50]	79.06	81.72	83.12	85.12
Ours	**79.55**	**82.66**	**83.25**	**85.82**
DGCNN [67]	59.75	66.34	69.88	74.57
Ours	**71.63**	**81.95**	**79.98**	**85.65**

Table 4. Semantic segmentation on SemanticKITTI [4] using self-supervised upsampling point clouds.

Method	mAcc	mIoU
RandLANet [21]	97.23	68.84
Ours	**99.17**	**70.55**

Table 5. Ablation studies of our test-time augmentation on ShapeNet [7] using surface sampling. Performing k-nearest neighbor search on high-dimensional feature space (model A, B) and using the average function (model B) for aggregating predictions result in improved accuracies. The performance can be further boosted by using extra augmented point clouds (model A&C and B&C). The reported metric is mCatIoU.

2048 points	PointNet [50]	DGCNN [67]
xyz (max)	86.45	84.26
A: w/ logit (max)	88.30	85.90
B: w/ logit (avg)	88.43	86.05
C: w/ 10x samples	86.39	84.13
A&C (max)	88.26	85.96
B&C (avg)	**88.58**	**86.16**

4.4. Additional Analysis

We perform additional experiments to validate the performance of our test-time augmentation. We select the segmentation task for these experiments as it produces dense prediction, which can be seen as generalized classifications.

Point density. In Figure 3, we plot the segmentation accuracies (mIoU) across different numbers of input points. Specifically, we randomly sample 128, 256, 512, 1024, and 2048 points as input to perform the segmentation. Compared to PointNet, it can be seen that our augmentation offers significant performance improvement on sparse point clouds (128 and 256 points) and performs similarly to PointNet when the input points get denser.

We also found that by varying the number of input points (Figure 3), DGCNN cannot perform well on sparse point clouds with a very large performance gap between the sparse

and dense point clouds (more than 20% between 128 and 2048 points). This is because for sparse point clouds, the neighbor graphs by DGCNN degenerate [67]. Despite such, our test-time augmentation can still improve the performance and significantly reduce the performance gap to around 6%. This shows that our test-time augmentation is robust to the number of input points.

Ablation study. We conduct an ablation study on the part segmentation task on ShapeNet and provide the results in Table 5. We select the segmentation task as it is a generalized form of classification at per-point level, and also aim to justify the more complex design choices in the aggregation function for this task. We use inputs with 2048 points. Our baseline is an implementation that k-nearest neighbors are performed with just 3D coordinates as features. By adding logits as features, we can have 2% gain in mIoU (model A). We also test different aggregate functions like max pooling and average pooling, and find that average pooling performs better (model B vs. C). Additionally, we repeat the sampling to obtain multiple augmented point clouds. By fusing the segmentation of these augmented point clouds to the original point cloud, further improvement can be achieved (model A&C and B&C). This shows that it is critical to compute accurate correspondences between the augmented point cloud and the original point cloud to achieve higher accuracies. From the above analysis, we can see that multi-sampling and changing aggregate functions can yield further improvement.

Comparison among augmentation techniques. We provide a comparison to study which augmentation technique should be used in practice. We compare the popular Screened Poisson reconstruction [28] to convolutional neural network [48] and point cloud upsampling [69]. The results in Table 6 show that our augmentation techniques are more favorable in performance than Screened Poisson reconstruction. The performance between the convolutional occupancy network and self-supervised upsampling are rather similar, with the convolutional occupancy network is slightly better in the instance IoU metric. We hypothesize that when (ground truth) surface information is available, it could be used to supervise the augmentation, else point cloud upsampling could be an

Table 6. Comparison of different augmentation methods on part segmentation with PointNet [50] as backbone on ShapeNet [7] dataset. TTA is done using Screened Poisson reconstruction [28], Neural Implicit representation [48], and Point Clouds Upsampling [69].

128 points	mCatIoU	mInsIoU
Poisson [28]	81.79	77.86
Implicit [48]	82.66	79.55
Self-UP [69]	82.70	78.99

Table 7. Comparison with traditional augmentation with different values of σ on the classification task using surface sampling on ShapeNet [7]. The backbone of our TTA is PointNet [50].

2048 points	mAcc
$\sigma = 0.05$	98.21
$\sigma = 0.07$	97.69
$\sigma = 0.1$	96.54
Ours	**98.53**

Table 8. Augmentation with normals using surface sampling on ShapeNet [7]: we classify an original point cloud w/o normal vectors by using an implicit field to sample the normals and then classify the augmented point cloud. The backbone is PointNet [50].

2048 points	mAcc
Org. xyz	97.73
Aug. xyz	98.53
Aug. xyz & normals	98.38

effective and robust augmentation in several scenarios. We also explore a recent unsupervised reconstruction [17] but find that the shape quality is poor compared to Screened Poisson reconstruction, and thus unsuitable for augmentation. Exploring more robust reconstruction could lead to interesting augmentation techniques for future work.

Comparison to traditional augmentation. Adding Gaussian noise is a commonly used traditional data augmentation scheme that perturbs the points by sampling from a Gaussian distribution. Combining the results from this perturbation in test-time augmentation is known as voting [30, 41]. In our implementation, we sample offsets from a zero-mean Gaussian with different standard deviation σ and add the offsets back to the original point clouds to form augmented point clouds. For our TTA, we sample one more point cloud and then average the global features of the additional point cloud and the original point cloud before passing them to the classifier. As can be seen in Table 7, our method outperforms the traditional augmentation scheme due to the implicit representation that allows more effective point sampling. Comparison with train-time augmentation is left for future work, as it requires model retraining, which is both more expensive and less robust especially when only pretrained models are provided.

Augmentation without normals. We experiment with implicit surface representation for data augmentation in a practical setting where normals are not available. In this case, existing methods have to rely on the pure 3D coordinates (xyz), causing a performance decrease, while our method can easily solve this problem by sampling the 3D points as well as the normal vectors directly from the implicit surface. In this way, our method can maintain the performance to the same level regardless of the normal existence. This is verified in Table 8 that our augmentation outperforms the baseline when only the 3D coordinates (xyz) are available.

Computation overhead. While having better performance, modern TTA, including our method, relies on a neural network to predict augmented samples from each
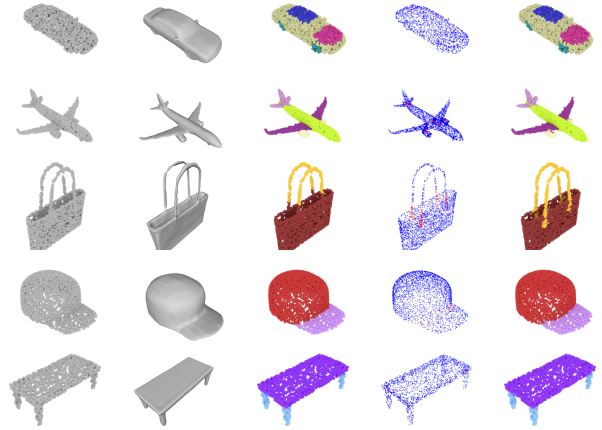

Figure 4. **Visualization of part segmentation results.** From left to right: Input, Reconstruction, Ours, Difference map, and GT. In the difference map, where blue and red points indicate correct and wrong labels, respectively, our test-time augmentation mainly deals with the labels along the boundaries, improving accuracies through aggregating predictions from augmented point clouds.

input and thus has more overhead compared to traditional methods. For example, if we use M augmented point clouds, the overhead is approximate M times the original time cost. To circumvent this problem, we propose to exploit parallelism and use batched inference instead. More discussions can be found in the supplementary material.

5. Discussion and Conclusions

We presented a new method for augmenting point clouds at test time by leveraging a neural implicit network and a point upsampling network to sample augmented point clouds and showed that such augmentation works effectively for the classification and semantic segmentation task. Our results are encouraging since this is one of the first attempts to design a test-time augmentation technique for 3D point cloud deep learning.

A main difference between our TTA and traditional meth-

ods is that traditional methods only use simple transformations and are thus lightweight, but not input-aware and less robust. While our TTA requires more resources, the extra computation remains affordable and our method shows good results across tasks and datasets. We believe further explorations to reduce such performance trade-offs would be valuable contributions to this less-explored area of test-time augmentation for 3D point clouds.

References

[1] Marc Alexa, Johannes Behr, Daniel Cohen-Or, Shachar Fleishman, David Levin, and Cláudio T. Silva. Computing and rendering point set surfaces. *IEEE Trans. Vis. Comput. Graph.*, 9:3–15, 2003. 2

[2] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1534–1543, 2016. 2

[3] Murat Seckin Ayhan and Philipp Berens. Test-time data augmentation for estimation of heteroscedastic aleatoric uncertainty in deep neural networks. In *International conference on Medical Imaging with Deep Learning*, 2018. 3

[4] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, Cyrill Stachniss, and Jurgen Gall. Semantickitti: A dataset for semantic scene understanding of lidar sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9297–9307, 2019. 1, 5, 6, 7, 2

[5] Heli Ben-Hamu, Haggai Maron, Itay Kezurer, Gal Avineri, and Yaron Lipman. Multi-chart generative surface modeling. *ACM Transactions on Graphics (TOG)*, 37(6):1–15, 2018. 2

[6] Rohan Chabra, Jan E Lenssen, Eddy Ilg, Tanner Schmidt, Julian Straub, Steven Lovegrove, and Richard Newcombe. Deep local shapes: Learning local sdf priors for detailed 3d reconstruction. In *European Conference on Computer Vision*, pages 608–625, 2020. 2

[7] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 1, 5, 6, 7, 8

[8] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees G. M. Snoek. Pointmixup: Augmentation for point clouds. In *ECCV*, 2020. 3

[9] Shuyang Cheng, Zhaoqi Leng, Ekin Dogus Cubuk, Barret Zoph, Chunyan Bai, Jiquan Ngiam, Yang Song, Benjamin Caine, Vijay Vasudevan, Congcong Li, Quoc V. Le, Jonathon Shlens, and Dragomir Anguelov. Improving 3d object detection through progressive population based augmentation. In *ECCV*, 2020. 3

[10] Jaesung Choe, Chunghyun Park, Francois Rameau, Jaesik Park, and In So Kweon. Pointmixer: Mlp-mixer for point cloud understanding. In *Computer Vision – ECCV 2022*, pages 620–640, 2022. 6

[11] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644, 2016. 2

[12] Angela Dai and Matthias Nießner. 3dmv: Joint 3d-multi-view prediction for 3d semantic scene segmentation. In *Proceedings of the European Conference on Computer Vision*, pages 452–468, 2018. 2

[13] 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. 2

[14] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5868–5877, 2017. 2

[15] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 2

[16] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9224–9232, 2018. 2

[17] Amos Gropp, Lior Yariv, Niv Haim, Matan Atzmon, and Yaron Lipman. Implicit geometric regularization for learning shapes. In *Proceedings of Machine Learning and Systems 2020*, pages 3569–3579, 2020. 4, 8

[18] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 216–224, 2018. 2

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[20] Andrew G. Howard. Some improvements on deep convolutional neural network based image classification. *arXiv 1312.5402*, 2013. 3

[21] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 2, 5, 6, 7

[22] Hui Huang, Dan Li, Hao Zhang, U. Ascher, and Daniel Cohen-Or. Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph.*, 28, 2009. 2

[23] Hui Huang, Shihao Wu, Minglun Gong, Daniel Cohen-Or, and U. Ascher. Edge-aware point set resampling. *ACM Transactions on Graphics*, 32, 2013. 2

[24] Jing Huang and Suya You. Point cloud labeling using 3d convolutional neural network. In *2016 23rd International*

Conference on Pattern Recognition (ICPR), pages 2670–2675, 2016. 2

[25] Qiangui Huang, Weiyue Wang, and Ulrich Neumann. Recurrent slice networks for 3d segmentation of point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2626–2635, 2018. 2

[26] Evangelos Kalogerakis, Melinos Averkiou, Subhransu Maji, and Siddhartha Chaudhuri. 3d shape segmentation with projective convolutional networks. In *proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3779–3788, 2017. 2

[27] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7122–7131, 2018. 2

[28] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the Fourth Eurographics Symposium on Geometry Processing*. Eurographics Association, 2006. 4, 7, 8

[29] Ildoo Kim, Younghoon Kim, and Sungwoong Kim. Learning loss for test-time augmentation. In *Advances in Neural Information Processing Systems*, 2020. 3

[30] Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 863–872, 2017. 8

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, 2012. 3

[32] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4558–4567, 2018. 2

[33] Truc Le and Ye Duan. Pointgrid: A deep network for 3d shape understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9204–9214, 2018. 2

[34] Hui Li, Peng Wang, Chunhua Shen, and Guyu Zhang. Show, attend and read: A simple and strong baseline for irregular text recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 8610–8617, 2019. 3

[35] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. PointAugment: An auto-augmentation framework for point cloud classification. In *CVPR*, 2020. 3

[36] Ruihui Li, Xianzhi Li, Pheng-Ann Heng, and Chi-Wing Fu. Point cloud upsampling via disentangled refinement. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 344–353, 2021. 1

[37] 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. 2

[38] Yiyi Liao, Simon Donne, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018. 2

[39] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[40] Yaron Lipman, Daniel Cohen-Or, David Levin, and Hillel Tal-Ezer. Parameterization-free projection for geometry reconstruction. *ACM Trans. Graph.*, 26:22, 2007. 2

[41] 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. 8

[42] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. In *Advances in Neural Information Processing Systems*, pages 965–975, 2019. 2

[43] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph.*, 21(4):163–169, 1987. 4

[44] Alexander Lyzhov, Yuliya Molchanova, Arsenii Ashukha, Dmitry Molchanov, and Dmitry Vetrov. Greedy policy search: A simple baseline for learnable test-time augmentation. In *Proceedings of the 36th Conference on Uncertainty in Artificial Intelligence (UAI)*, 2020. 3

[45] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4460–4470, 2019. 1, 2, 4, 5

[46] Nikita Moshkov, Botond Mathe, Attila Kertesz-Farkas, Reka Hollandi, and Peter Horvath. Test-time augmentation for deep learning-based cell segmentation on microscopy images. *Scientific reports*, 10(1):1–7, 2020. 3

[47] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 165–174, 2019. 1, 2

[48] Songyou Peng, Michael Niemeyer, Lars Mescheder, Marc Pollefeys, and Andreas Geiger. Convolutional occupancy networks. In *ECCV*, 2020. 1, 2, 4, 5, 6, 7, 8

[49] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4332–4341, 2019. 2

[50] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 2, 3, 4, 5, 6, 7, 8

[51] 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 5105–5114, 2017. 2, 3

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

[53] Anurag Ranjan, Timo Bolkart, Soubhik Sanyal, and Michael J Black. Generating 3d faces using convolutional mesh autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 704–720, 2018. 2

[54] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241, 2015. 4

[55] Ikuro Sato, Hiroki Nishimura, and Kensuke Yokoi. Apac: Augmented pattern classification with neural networks. *arXiv 1505.03229*, 2015. 3

[56] Divya Shanmugam, Davis Blalock, Guha Balakrishnan, and John Guttag. When and why test-time augmentation works. *arXiv 2011.11156*, 2020. 3

[57] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. 3

[58] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3d shape surfaces using geometry images. In *European conference on computer vision*, pages 223–240, 2016. 2

[59] Vincent Sitzmann, Michael Zollhöfer, and Gordon Wetzstein. Scene representation networks: Continuous 3d-structure-aware neural scene representations. In *Advances in Neural Information Processing Systems*, 2019. 1

[60] David Stutz and Andreas Geiger. Learning 3d shape completion from laser scan data with weak supervision. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1955–1964, 2018. 2

[61] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 3

[62] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. Segcloud: Semantic segmentation of 3d point clouds. In *2017 international conference on 3D vision (3DV)*, pages 537–547. IEEE, 2017. 2

[63] 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 *International Conference on Computer Vision*, 2019. 1, 3, 5, 6

[64] Guotai Wang, Wenqi Li, Michael Aertsen, Jan Deprest, Sébastien Ourselin, and Tom Vercauteren. Aleatoric uncertainty estimation with test-time augmentation for medical image segmentation with convolutional neural networks. *Neurocomputing*, 338:34–45, 2019. 3

[65] Shenlong Wang, Simon Suo, Wei-Chiu Ma, Andrei Pokrovsky, and Raquel Urtasun. Deep parametric continuous convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2589–2597, 2018. 2

[66] 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*, 2019. 2, 5, 6

[67] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (TOG)*, 2019. 5, 6, 7

[68] Zongji Wang and Feng Lu. Voxsegnet: Volumetric cnns for semantic part segmentation of 3d shapes. *IEEE transactions on visualization and computer graphics*, 2019. 2

[69] Zhao Wenbo, Liu Xianming, Zhong Zhiwei, Jian Junjun, Gao Wei, Li Ge, and Ji Xiangyang. Self-supervised arbitrary-scale point clouds upsampling via implicit neural representation. In *CVPR*, 2022. 1, 4, 5, 6, 7, 8, 2

[70] Jiajun Wu, Chengkai Zhang, Tianfan Xue, William T Freeman, and Joshua B Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, 2016. 2

[71] 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, 6

[72] Zhige Xie, Kai Xu, Wen Shan, Ligang Liu, Yueshan Xiong, and Hui Huang. Projective feature learning for 3d shapes with multi-view depth images. *Computer Graphics Forum*, 34(7): 1–11, 2015. 2

[73] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *European Conference on Computer Vision*, pages 1–19, 2020. 2

[74] Yifan Xu, Tianqi Fan, Mingye Xu, Long Zeng, and Yu Qiao. Spidercnn: Deep learning on point sets with parameterized convolutional filters. In *Proceedings of the European Conference on Computer Vision*, pages 87–102, 2018. 2

[75] Guandao Yang, Xun Huang, Zekun Hao, Ming-Yu Liu, Serge Belongie, and Bharath Hariharan. Pointflow: 3d point cloud generation with continuous normalizing flows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4541–4550, 2019. 2

[76] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Ec-net: an edge-aware point set consolidation network. *CoRR*, abs/1807.06010, 2018. 2

[77] Lequan Yu, Xianzhi Li, Chi-Wing Fu, Daniel Cohen-Or, and Pheng-Ann Heng. Pu-net: Point cloud upsampling network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2790–2799, 2018. 1, 2

[78] Feihu Zhang, Jin Fang, Benjamin W Wah, and Philip HS Torr. Deep fusionnet for point cloud semantic segmentation. In *ECCV (24)*, pages 644–663, 2020. 2

[79] Zhiyuan Zhang, Binh-Son Hua, and Sai-Kit Yeung. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1607–1616, 2019. 2

[80] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16259–16268, 2021. 6

[81] Özgün Çiçek, A. Abdulkadir, S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, 2016. 4

Test-Time Augmentation for 3D Point Cloud Classification and Segmentation

Supplementary Material

6. Experiments

6.1. Computation overhead

While having better performance, modern TTA, including our method, relies on a neural network to predict augmented samples from each input and thus has more overhead compared to traditional methods. For example, if we use M augmented point clouds, the overhead is approximately M times the original time cost. To circumvent this problem, we propose to exploit parallelism and use batched inference instead. First, our computation overhead is sub-linear, which means that even if we have 10 times as many augmented samples (the same number of augmented samples that we used for all of our experiments), the overhead will only increase by a factor of two (see Table 9). This overhead is manageable and can be reduced even further through the utilization of batched prediction, as demonstrated in Table 9 below. Additionally, additional engineering like deploying the network to an inference-only framework (TensorFlow Lite) would further optimize inference. Second, we can reduce the number of augmented samples ($M \in \{2, 4, 8, 10\}$), which would result in milder improvement in comparison to the baseline but would incur significantly less overhead (see Table 9). Finally, the overhead of TTA can be offset by its ease of use compared to other methods for performance improvement, e.g., when only pre-trained models are given or when retraining the entire model is not possible.

Table 9. Running time breakdown of different stages of our TTA with different augmentation samples M for the classification task on ModelNet40.

	M=10	M=8	M=4	M=2	PointNeXt
B. forward 10x	0.6573	0.1025	0.1021	0.1016	0.2147
B. GPU FPS 10x	0.1036	0.5258	0.2892	0.1446	0.0
Aggregation	0.0968	0.0959	0.0948	0.0934	0.0
Others	0.4382	0.4334	0.4295	0.4252	0.4382
Total time	1.2959	1.1576	0.9156	0.7648	0.6529
oAcc	95.84	95.28	94.81	94.29	93.96
mAcc	92.96	92.51	91.87	91.38	91.14

6.2. Semantic Segmentation on S3DIS dataset

We provide quantitative and qualitative results of semantic segmentation on S3DIS dataset in Table 10 and Figure 5. As can be seen, our TTA is effective and improves upon the baseline PointNeXt.

6.3. Semantic Segmentation on SemanticKITTI

We provide qualitative results of semantic segmentation on SemanticKITTI dataset in Figure 6.

6.4. Part Segmentation on ShapeNet dataset

We provide more qualitative results of part segmentation on ShapeNet dataset in Figure 7 and Figure 8.

Table 10. TTA with point cloud upsampling [69] on semantic segmentation on S3DIS [2].

Method	mIoU (%)	mAcc (%)	Category IoU												
			ceiling	floor	wall	beam	column	window	door	table	chair	sofa	bookcase	board	clutter
PointNeXt	64.26	70.69	94.07	98.26	80.89	0	23.84	48.66	66.58	80.99	90.06	67.94	72.08	58.06	54.04
Ours	**65.23**	**71.75**	95.48	99.73	82.11	0	24.21	49.39	67.58	82.21	91.41	68.96	73.16	58.93	54.85

PointNeXt   Ours   Ground Truth

Figure 5. Semantic segmentation results of PointNeXt [52], ours and ground truth on S3DIS dataset [2].
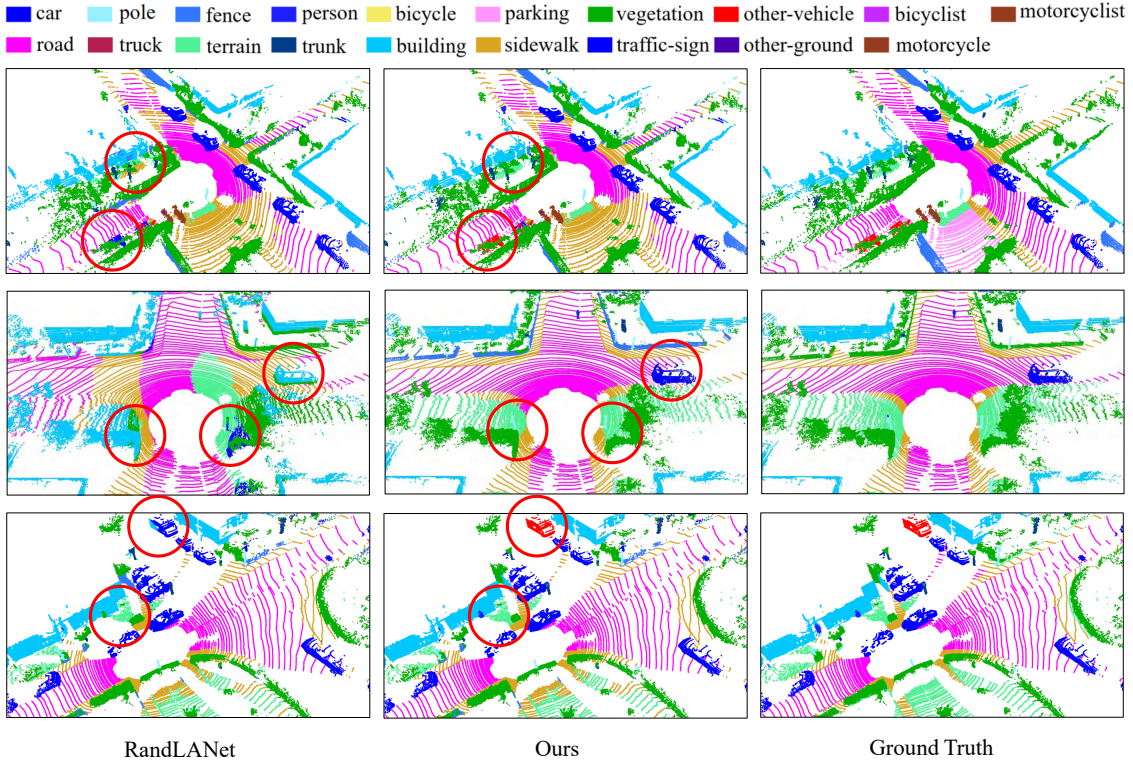

RandLANet   Ours   Ground Truth

Figure 6. Semantic segmentation results of RandLANet [21], ours and ground truth on SemanticKITTI dataset [4].

(a) Input   (b) Reconstruction   (c) Our result   (d) Difference map   (e) Ground truth
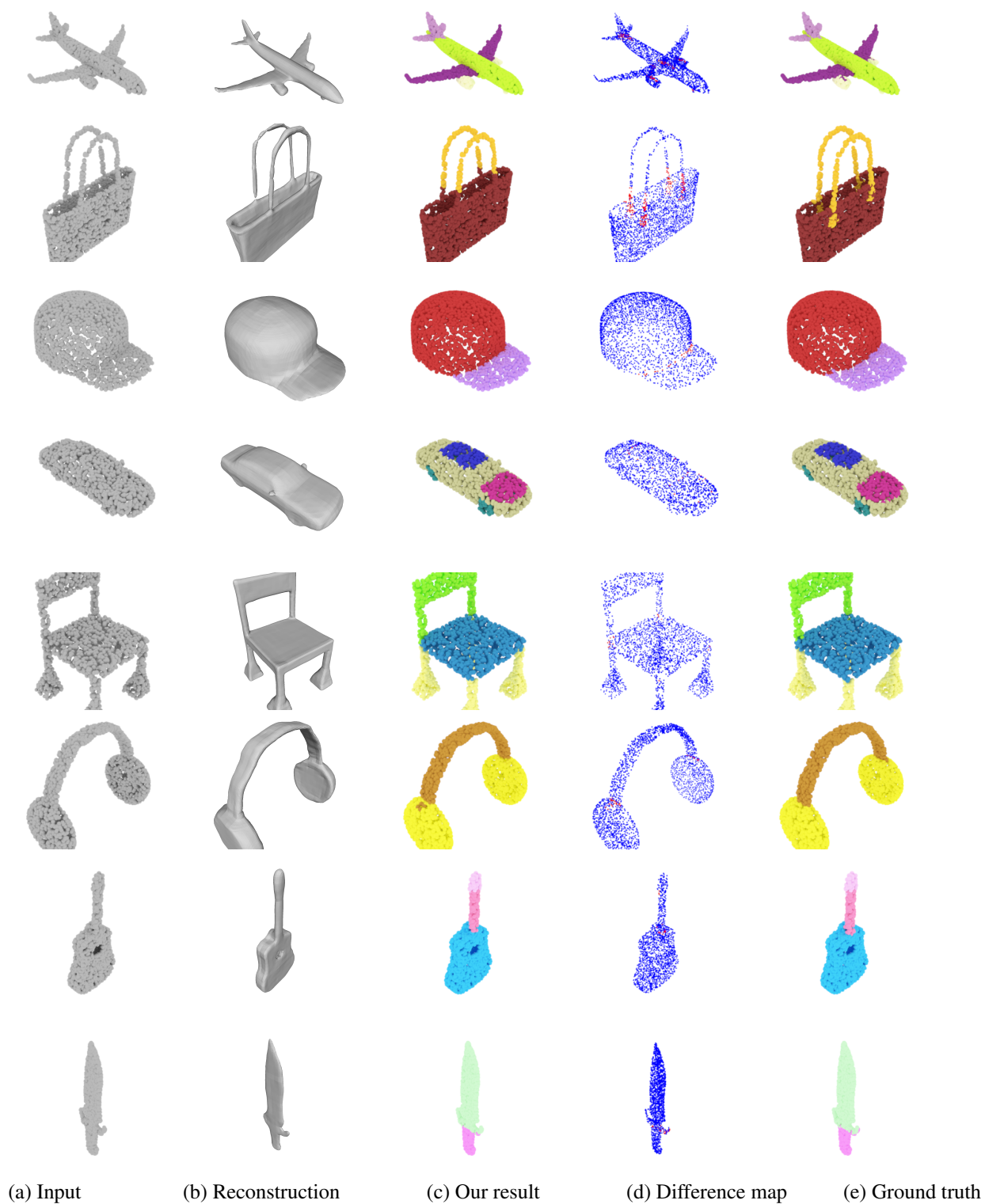
Figure 7. Visualization of part segmentation results. As can be seen in the difference map, where blue and red points indicate correct and wrong labels, respectively, our test-time augmentation mainly deals with the labels along the boundaries, improving their accuracies through aggregating predictions from augmented point clouds. Best viewed with zoom.

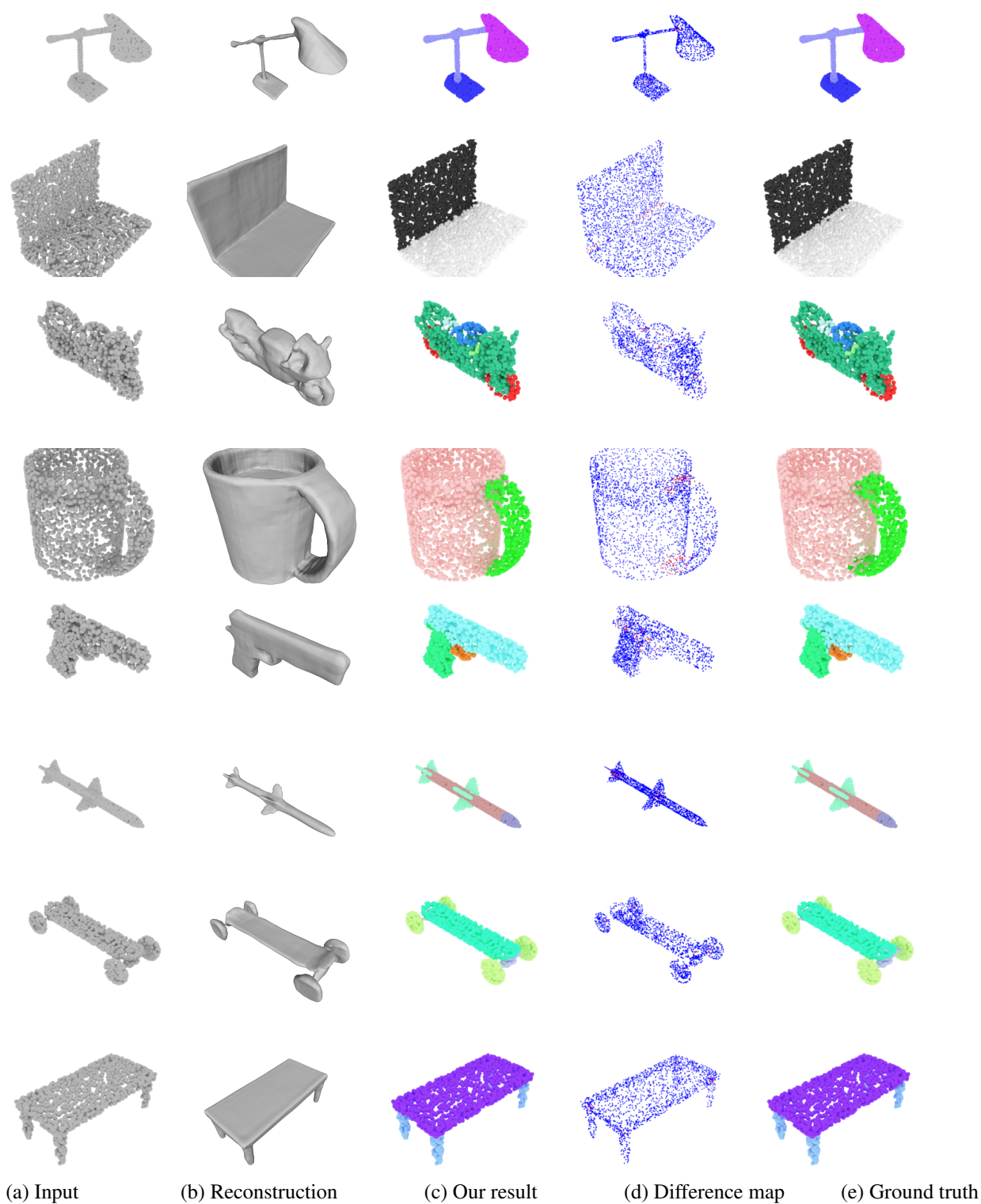| (a) Input | (b) Reconstruction | (c) Our result | (d) Difference map | (e) Ground truth |

Figure 8. Visualization of part segmentation results. As can be seen in the difference map, where blue and red points indicate correct and wrong labels, respectively, our test-time augmentation mainly deals with the labels along the boundaries, improving their accuracies through aggregating predictions from augmented point clouds. Best viewed with zoom.