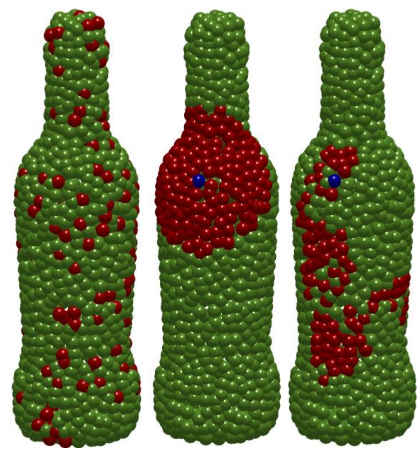
EPiC: Ensemble of Partial Point Clouds for Robust Classification

Meir Yossef Levi, Guy Gilboa
Viterbi Faculty of Electrical and Computer Engineering
Technion - Israel Institute of Technology, Haifa, Israel
me.levi@campus.technion.ac.il ; guy.gilboa@ee.technion.ac.il

Abstract

Robust point cloud classification is crucial for real-world applications, as consumer-type 3D sensors often yield partial and noisy data, degraded by various artifacts. In this work we propose a general ensemble framework, based on partial point cloud sampling. Each ensemble member is exposed to only partial input data. Three sampling strategies are used jointly, two local ones, based on patches and curves, and a global one of random sampling.

We demonstrate the robustness of our method to various local and global degradations. We show that our framework significantly improves the robustness of top classification netowrks by a large margin. Our experimental setting uses the recently introduced ModelNet-C database by Ren et al.[24], where we reach SOTA both on unaugmented and on augmented data. Our unaugmented mean Corruption Error (mCE) is 0.64 (current SOTA is 0.86) and 0.50 for augmented data (current SOTA is 0.57). We analyze and explain these remarkable results through diversity analysis. Our code is availabe at: https://github.com/yossilevii100/EPiC



(a) Random (b) Patch (c) Curve

Figure 1: **EPiC concept.** Three sampling mechanism are used in our ensemble: **Random** captures global information, **Patch** holds full local resolution, and **Curve** is more exploratory in nature. **Blue** - anchor point, **Red** - sampled points. We show and explain why such ensembles are highly robust to various corruptions.

1. Introduction

A major obstacle in data-driven algorithms is the strong relation between performance and precise knowledge of input statistics. A way to test network robustness is to create corrupted test sets, where training is unaware of the specific corruptions. In recent years this has been investigated for images, with the creation of corrupted benchmarks e.g.: ImageNet-C, CIFAR10-C, CIFAR100-C [11] and MNIST-C [17]. In [24] the idea is extended to point cloud classification, with the introduction of ModelNet-C. In our work we present a generic framework, based on sampling, which is light, flexible and robust to Out-Of-Distribution (OOD) samples.

Ensemble learning is a long-standing concept in robust machine learning [8, 10, 6]. We would like to obtain an ensemble of learners, which are loosely correlated, that generalize well also to OOD input. There are two main questions: 1) How to form these learners? 2) How to combine their results?

One classical way to form ensembles in deep learning (see for instance [2]), is to rely on the partial-randomness of the training process. In this setting, the ensemble consists of networks of the same architecture trained on the same training-set. Variations of networks' output stem from the different random parameter initialization and the stochastic gradient descent process. This induces *Stochastic diversity*. Another major approach to generate ensembles (both in classical machine learning and in deep learning [8, 2, 39]) is to change the sampling distribution of the training set for each learner. A mixture-of-experts approach is to train different types of classifiers, in neural networks this is accomplished by different network architectures. This induces

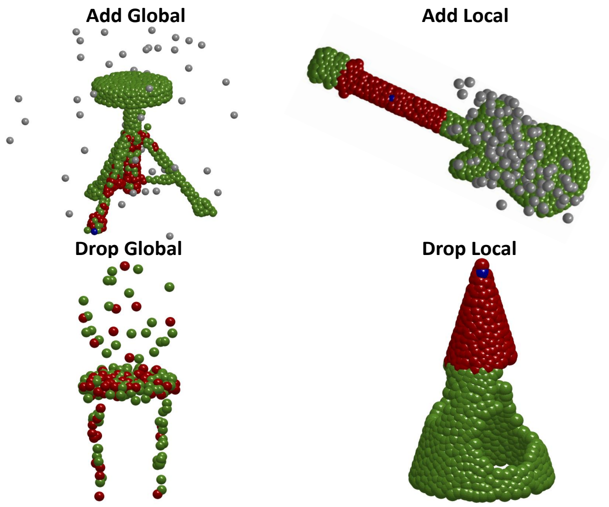


Figure 2: **Overcoming corruptions by sampling**. Sampled partial point clouds (red points) have instances in which they are mostly not exposed to corruptions. The ensemble becomes highly robust to various unknown degradations. This is mostly apparent for non-uniform corruptions.

Architecture diversity. These approaches, however, may not yield sufficient OOD robustness (our experiments show their diversity is limited). Deep ensembles are being investigated with new ways to calibrate and to estimate uncertainty through increasing the ensemble diversity [38, 30].

Our approach to form ensembles is by exposing each ensemble member to limited input data. It is performed by generating different samples of each point cloud at both training and testing, see Fig. 1. The ensemble becomes highly diverse, since each classifier has access to different parts of the data. Moreover, each sampling method has unique characteristics. We observe that many types of corruption do not corrupt the data uniformly (see examples in Fig. 2). Some partial point clouds may thus be less corrupted, achieving accuracy closer to the case of clean data. Highly corrupted partial point clouds yield almost random network response, and can be modeled as adding noise to the classification decision. In this case, applying mean over the outputs of the ensemble significantly diminishes noise and improves accuracy . Four major advantages are gained by our proposed scheme: 1) The framework is completely generic and can essentially work with any point cloud classification network (we experimented with five, gaining improvement in all); 2) Many diverse classifiers can be generated; 3) Partial data is robust to local corruptions and to outliers, performing well on OOD samples; 4) The required networks to be trained is the number of sampling methods (in our case three), and not the ensemble size.

We reach robustness to various local and global degradations, yielding state-of-the-art results on corrupted ModelNet-C [24] ($mCE = 0.646$ using PCT[9] and $mCE = 0.501$ using augmented with WolfMix[25, 4] version of RPC [24]), even with a small ensemble of size 12.

2. Related Work

Point Cloud classification. It is customary to categorize point cloud classification networks to three mechanisms: multi-view, voxelizing, and point-wise networks. One of the challenges in point cloud processing is that, unlike images, 3D points are irregular in space. Multi-view methods project the point cloud into different viewpoints, generate 2D images, and apply CNN based networks [22, 1, 27]. The projection forces the data to be regular. These approaches are slower because of the rendering phase, and might lose useful geometric information. In the voxelizing mechanism, the solution for the irregularity problem is to partition the 3D space into a voxel grid [22, 32, 16]. This approach suffers heavily from sensitivity to the choice of grid regularity. In the context of point-wise networks, PointNet [20] presented a pioneering method of applying MLP on the raw 3D points. DGCNN[31], Dynamic Graph CNN, dynamically constructs a graph through the network. The graph is initially based on the raw 3D points, and progresses to more evolved feature spaces with semantic connectivity. GDANet [13], Geometry-Disentangled Attention Network, dynamically disentangles point clouds into contour and flat components. Respective features are fused to provide distinct and complementary geometric information from each representation. Recently, PCT [9], Point Cloud Transformer, adopted transformer architecture [29] to create per-patch embedding, leveraging self attention mechanisms. CurveNet[26] generates features for each point by guided-walk over the cloud followed by a curve grouping operator. Ren et al.[24] studied the impact of several architectural choices on the robustness of the network and combined the most robust components to create RPC, which is the SOTA network on ModelNet-C. RPC takes advantage of 3D representations, using KNN, frequency grouping and self-attention mechanism that turned out to be the most robust combination. Another related work is Point-BERT[36] which splits the point cloud into patches as well. It then tokenizes the patches and uses pre-trained transformers with Mask Point Modeling. Point-MLP[15] is based on residual MLPs, achieving impressive results on the clean data.

Robustness to shifts from the training set. Traditionally, most of the attention in terms of robustness focused on basic deformations: jitter, scale and rotation. There are different approaches in the literature focusing on some specific deformations, degradations or corruptions. For example, a long standing research topic is rotation invariance, thoroughly addressed, e.g. by ClusterNet[3] and LGR-Net[7]. PointCleanNet[23] and PointASNL[35] addressed specifically robustness to outliers. Studies investigating robust-

Network (#Ensemble size)	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN [31]	92.6%	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
EPiC(#12) (Ours)	93.0%	<u>0.669</u>	1.000	0.680	0.331	0.498	0.349	<u>0.807</u>	1.019
GDANet[13]	93.4%	0.892	**0.830**	0.839	0.794	0.894	0.871	1.036	0.981
EPiC(#12) (Ours)	93.6%	0.704	0.936	0.864	<u>0.315</u>	0.478	0.295	0.862	1.177
CurveNet[26]	<u>93.8%</u>	0.927	0.872	0.725	0.710	1.024	1.346	1.000	**0.809**
EPiC(#12) (Ours)	92.1%	0.742	1.245	**0.617**	0.363	0.585	0.495	1.029	<u>0.860</u>
PCT[9]	93.0%	0.925	0.872	0.870	0.528	1.000	0.780	1.385	1.042
EPiC(#12) (Ours)	93.4%	**0.646**	0.894	0.851	**0.306**	**0.435**	<u>0.285</u>	**0.735**	1.019
RPC[24]	93.0%	0.863	<u>0.840</u>	0.892	0.492	0.797	0.929	1.011	1.079
EPiC(#12) (Ours)	93.6%	0.750	0.915	1.057	0.323	<u>0.440</u>	**0.281**	0.902	1.330
PointNet[20]	90.7%	1.422	1.266	<u>0.642</u>	0.500	1.072	2.980	1.593	1.902
PointNet++[21]	93.0%	1.072	0.872	1.177	0.641	1.802	0.614	0.993	1.405
RSCNN[34]	92.3%	1.130	1.074	1.171	0.806	1.517	0.712	1.153	1.479
SimpleView[1]	**93.9%**	1.047	0.872	0.715	1.242	1.357	0.983	0.844	1.316
PAConv[14]	93.6%	1.104	0.904	1.465	1.000	1.005	1.085	1.298	0.967

Table 1: **Main Experimental Result. ModelNet-C Unaugmented classification comparison. Bold** best, <u>underline</u> second best. Our framework dramatically improves robustness of all examined networks, as indicated by the mCE measure. Experiments using EPiC were conducted on the five most robust networks ($mCE \leq 1$).

ness to occlusions (common in 3D-sensors) use transformers [37] and voting mechanisms [40]. Another aspect of robustness is related to 3D adversarial attacks, adressed by [33, 5]. PointGuard[12] is also focused on guarding against adversarial attacks. K random sampling procedures are performed for each point cloud, where the overall prediction is obtained by majority voting. The theoretical analysis of [12] advocates the use of sampling very few points and using a huge ensemble ($K = 10000$). Our experiments indicate that such ensembles do not perform as well on the clean data and are competitive only for certain types of OOD samples.

A major approach to increase robustness is by augmentation techniques which enrich the training set. RSmix[4] suggests mixing two samples from the database in a smooth manner, along with smoothing the labels of both samples, creating new virtual samples. This method inherently inserts jittering, scaling, translation and rotation. PointWolf[25] applies non-rigid manipulations on selected anchor points. These include scaling, 3D rotation and translation with local jittering. Such complex manipulations are able, for example, to change a person's posture or the style of a chair. We note that augmentation introduces expected corruptions within the training set, thus violating, to some degree, the testing principles of OOD robustness. To make our work complete, we examine the proposed framework also on augmented data (WolfMix[24], a combination of PointWolf and RSmix). We achieve SOTA in this case as well.

2.1. Benchmarks

ModelNet-40 [32] is a widely used dataset consisting of CAD meshes from 40 classes, such as airplane, chair and sofa. Each mesh was uniformly sampled to form a 3D point cloud consisting of 1024 points. The dataset has 12,311 samples, divided to 9843 for training and 2468 for test. The dataset contains closely semantic classes like vase, plant and flower pot, or chair and stool, which makes this dataset highly challenging. Recently, in [24] Ren et al proposed a corrupted point cloud benchmark to assess OOD robustness. It is based on the well studied ModelNet-40 and is referred to as ModelNet-C. We evaluate the robustness for OOD corruptions by applying it on ModelNet-C [24]. This dataset contains seven types of corruptions: jitter, scale, rotate, add-global, add-local, drop-global and drop-local. Each of these corruptions is rendered in five levels of difficulty. In [24] a unified calculation mechanism was defined to measure robustness. This measure is termed mCE (mean Corruption Error). It basically evaluates the results in comparison to DGCNN, which serves as a reference algorithm (and hence, by definition, has $mCE \equiv 1$). Each specific corruption has a similar score, relative to DGCNN, averaged over all five levels of difficulty. Since the measure is of error, a lower score is better. Please refer to [24] for more details.

3. Our Proposed Method

3.1. Notations

We denote by N the number of cloud points at the input of a network (which, due to sampling, may vary). The

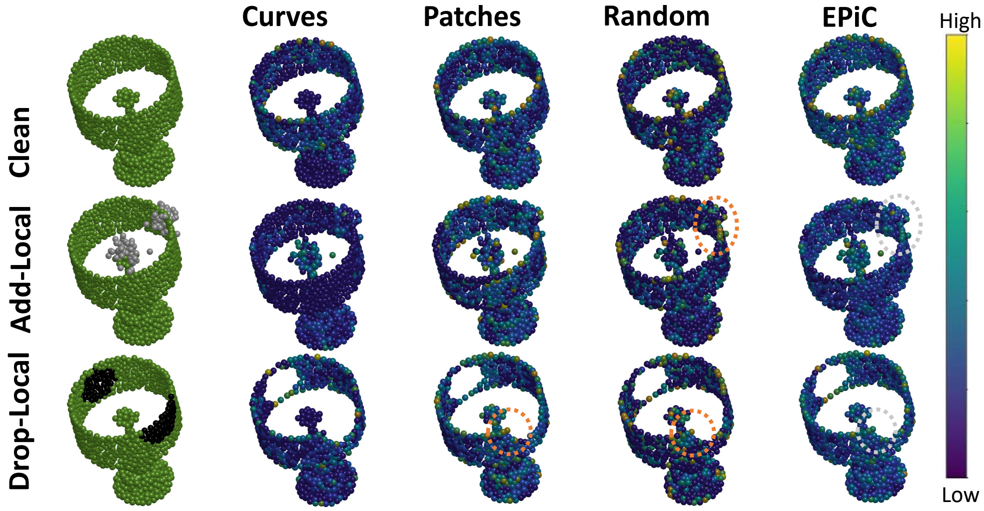


Figure 3: **Pointwise Importance**. Left column - corruptions (grey - added points, black - removed points). The rest of the columns show color-coded pointwise importance. Although Random is more influenced by Add-local, and Patches+Random by Drop-local (dashed orange circles), the aggregated result is robust to both (dashed grey circles).

number of features for each point in the cloud is F. We refer to $\tilde{K}$ as the ensemble size of each sub-sample mechanism and to K as the combined ensemble size. In our case we suggest three sampling mechanisms of the same size, therefore $K = 3\tilde{K}$. N_p, N_c and N_r are the number of points in patches, curves and random sub-samples, respectively. In curve sampling there may be occasional repeating points, $N \leq N_c$. For curve extraction we have a hyper-parameter M controlling the number of neighbors to choose from in a random-walk iteration. C is the number of classes ($C = 40$ in ModelNet-40), and $p \in \mathbb{R}^C$ is a single prediction vector. $P \in \mathbb{R}^{K \times C}$ is an ensemble of predictions.

3.2. Motivation of our approach

The ability to classify in a robust manner is closely coupled with the ability to classify based on partial information. We would like a classification network to perform reasonably well also when some information is missing, noise is added or when outliers are introduced. Thus, it is desired to obtain a diverse set of features which are based on different locations in the shape. This could be demonstrated well in the experiment illustrated in Fig. 3.

We visualize the internal classification importance of points for networks specializing in curves, patches and random. For the demonstration we focused on two degradations of adding and removing points locally. Commonly, a network gets a point cloud $X \in \mathbb{R}^{N \times 3}$ and encodes it into features $X_f \in \mathbb{R}^{N \times F}$ by a variety of sophisticated layers. The standard method to aggregate the points axis is by applying a symmetric function which is permutation invariant, such as max or mean. In this example, in order to obtain features X_f, we use DGCNN[31]. We calculate the importance of each point j, denoted by $Imp(j)$, in the following manner,

$$Imp(j) = \sum_{k=1}^{F} I(j == \arg\max_n(X_f(n,k))), \quad (1)$$

where I is an indicator function (1 when true and 0 otherwise). The importance attempts to quantify the number of features of a specific point which are part of the global feature vector. This usually means that the feature's magnitude at that point is maximal, dominating the respective feature vector entries among all other points. In this example, to cover the entire point cloud, we calculate the importance for all partial point clouds and present the average result in Fig. 3.

When parts of the cloud are missing, or outliers are introduced, potentially corrupting features, we want the classification to be based on a variety of features, even though they may be less prominent in the clean set. Therefore, our aim is to "spread" the importance as evenly as possible. This insight motivates our approach for using partial point cloud ensembles to impose feature diversity. Additional analysis and insights appear hereafter.

3.3. Proposed approach

We use three types of sub-samples: two local ones, *Curves* and *Patches* and a global one consisting of *Random-sampling*. For the local methods, we first use farthest point sampling (FPS) algorithm [21] to choose $\tilde{K}$ anchors. From each anchor we extract a patch and a curve.

Patch extraction is done by finding N_p nearest neighbors. This sub-sample mechanism is more conservative, hence it preserves well the local information.

Curve extraction is done by a random-walk process, beginning from the anchor point. N_c random-walk iterations are performed, at each iteration one of M nearest neighbors is chosen randomly. The choice is with replacement (hence the sampled partial point cloud may be smaller than N_c). This mechanism is more exploratory in nature and less structured.

Random extraction is done by simply sub-sampling N_r random points from the entire point cloud (without replacement, $N = N_r$).

The values of these parameters were determined once and were used in the same manner in all our experiments for all classification networks (see details in Supp).

Generic framework. Our approach is generic and can be applied in conjunction with any point cloud classification network. The experiments (detailed in the experimental section) are conducted with five different architectures. These architectures are the most OOD-robust. Our method considerably improves the robustness of every one of them, as indicated by the mCE measure. We use three instances of the same architecture. Each instance is trained to classify point clouds obtained by a certain sub-sampling mechanism. A recap of our approach (inference) is given in Algorithm 1, where the training procedure is detailed in the Supp.

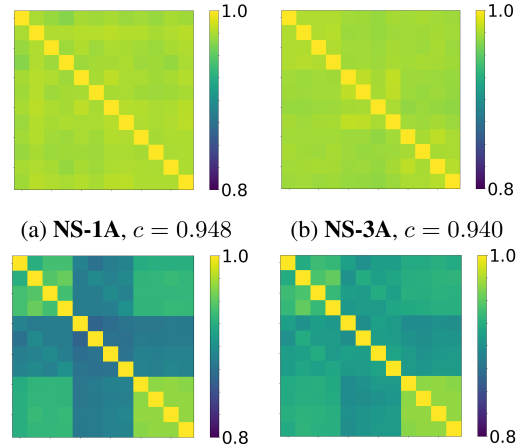
Algorithm 1 Classification using EPiC (inference)

Require: $X, params_{Patches}, params_{Curves}, params_{Random}$
$model_{Patches} \leftarrow params_{Patches}$
$model_{Curves} \leftarrow params_{Curves}$
$model_{Random} \leftarrow params_{Random}$
$anchors \leftarrow FarthestPointSampling(X, \tilde{K})$
for $k \in \tilde{K}$ **do**
  $Patch \leftarrow FetchPatch(X, anchors(k))$ ▷ Local
  $Curve \leftarrow FetchCurve(X, anchors(k))$ ▷ Local
  $Random \leftarrow FetchRandom(X)$ ▷ Global
  $P^k_{Patch} \leftarrow model_{Patches}(Patch)$
  $P^k_{Curve} \leftarrow model_{Curves}(Curve)$
  $P^k_{Random} \leftarrow model_{Random}(Random)$
end for
$P_{ensemble} \leftarrow Concatenate(P^{1:\tilde{K}}_{Patch}, P^{1:\tilde{K}}_{Curve}, P^{1:\tilde{K}}_{Random})$
$P \leftarrow Mean(P_{ensemble}); Class = \arg\max(P).$

4. Diversity Analysis

In order to leverage the advantage afforded by ensembles, a high level of diversity of ensemble members is required. This motivates us to quantitatively study the diversity of several types of ensembles. We investigate the following sources of diversity:

1. **Stochastic.** Diversity stemming from the stochasticity of the training process (initialization and SGD).



(a) **NS-1A**, $c = 0.948$ (b) **NS-3A**, $c = 0.940$

(c) **S-1A**, $c = 0.825$ (d) **S-3A**, $c = 0.847$

Figure 4: **Correlation between ensemble members.** Correlation output of ensemble members on full non-sampled clean test-set (NS), top, with a single and three architectures, compared to sampling (S) by our approach, bottom, see setting details in Section 4. c is defined in Eq. (2) (lower means higher diversity). Sampling affords higher diversity, compared to stochastic and architecture sources of diversity.

2. **Architecture.** Diversity caused by using different network architectures.

3. **Sampling.** Diversity due to different sampling methods and randomness of the sampling process.

Four types of ensembles were examined (each consisting of 12 members, correlation results are shown in Fig. 4):

1. **No sampling, single architecture (NS-1A).** *Stochastic diversity.* The ensemble consists of 12 instances of DGCNN[31]. The entire point-cloud is used as input.

2. **No sampling, three architectures (NS-3A).** *Architecture + stochastic diversity.* The ensemble consists of three architectures (each of 4 instances, in this order): PCT[9], GDANet[13] and DGCNN[31]. The entire point-cloud is used as input.

3. **Sampling (ours), single architecture (S-1A).** *Sampling diversity.* An ensemble consisting of three instances of DGCNN[31]. Each instance is trained to specialize in a different sampling mechanism. The sampling methods (in this order) are: Patches, Curves and Random. Each instance uses 4 different subsample inputs.

4. **Sampling (ours), three architecture (S-3A).** *Sampling + architecture diversity.* Similar setting to the sampling diversity experiment. Here the following architectures, GDANet[13], PCT[9] and DGCNN[31],

Networks (#Ensemble size)	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN[31]+W.M	93.2%	0.590	0.989	0.715	0.698	0.575	0.285	0.415	0.451
EPiC(#12)+W.M (Ours)	92.1%	0.529	1.021	**0.541**	0.355	0.488	0.288	0.407	0.600
GDANet[13]+W.M	**93.4%**	0.571	**0.904**	0.883	0.532	0.551	0.305	0.415	**0.409**
EPiC(#12)+W.M (Ours)	92.5%	0.530	0.968	0.639	0.343	0.473	0.275	0.433	0.577
PCT[9]+W.M	**93.4%**	0.574	1.000	0.854	0.379	0.493	0.298	0.505	0.488
EPiC(#12)+W.M (Ours)	92.7%	0.510	0.915	0.699	0.323	0.425	0.268	0.404	0.535
RPC[24]+W.M	93.3%	0.601	1.011	0.968	0.423	0.512	0.332	0.480	0.479
EPiC(#12)+W.M (Ours)	92.7%	**0.501**	0.915	0.680	**0.315**	**0.420**	**0.251**	**0.382**	0.544

Table 2: **ModelNet-C Augmented (WolfMix) classification Comparison.** **Bold** best, underline second best. 1) dramatically robustification gained by using EPiC on conventional methods. 2) Our method improves mCE and almost all OOD corruptions.

Ensembles	OA↑	mCE↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
NS-1A [31]	93.5%	1.006	**0.862**	0.709	0.762	0.889	2.064	1.025	**0.730**
NS-3A [31, 9, 13]	93.4%	0.856	**0.862**	0.642	0.657	0.797	1.275	0.982	0.777
S-1A [31], Ours	93.0%	**0.669**	1.000	0.680	0.331	0.498	0.349	0.807	1.019
S-3A [31, 9, 13], Ours	**93.8%**	0.671	0.915	0.813	**0.315**	**0.469**	**0.302**	0.811	1.074
Point-Guard (#1,000) [12]	89.6%	0.949	1.947	**0.617**	0.427	0.599	0.376	**0.702**	1.977

Table 3: **ModelNet-C Ensemble methods classification Comparison.** **Bold** best, underline second best. Robustness improved by our suggested method.

were used on sampled inputs of Patches, Curves and Random, respectively.

In addition, PointGuard [12] was examined, as a different ensemble reference, with an ensemble of size 1000.

Measuring diversity through correlation. We introduce a measure which quantifies (inverse) diversification by

$$c = \frac{1}{S}\sum_{i=1}^{S}\frac{||C_i - I||^2}{K^2 - K}, \tag{2}$$

where S is the dataset size, K is the ensemble size, $I_{K\times K}$ is a unit matrix, $||\cdot||$ is the Frobenius norm and C is the Pearson correlation matrix of the ensemble predictions. The final measure $c \in [0, 1]$ is a scalar quantifying the diversity of the ensemble (in an inverse manner) for a given dataset. For $c = 0$ the members' response is completely uncorrelated (most diverse). For $c = 1$ the members are fully correlated (zero diversity) and an ensemble is not necessary (would produce identical results as a single member). Note that since each member is quite accurate (a "strong learner"), with an accuracy of around 90%, on binary problems we expect a diverse ensemble to be with $c \approx 0.9^2$ (in the multiclass case analysis requires additional assumptions, but should be in similar ranges). As can be seen in Fig. 4, ensembles based on sampling are considerably more diverse than those based on stochastic or architecture diversity. Moreover, curves and patches are lowly correlated, although the same anchor points are used. Diversity stems mostly from the sampling scheme, in addition to locality. In Table 3 the improved robustness gained by the four ensemble methods is shown. Our partial-point-cloud strategy improves robustness in most criteria, excelling in mCE, with the three architecture configuration (S-3A) having superior results also in overall accuracy (clean set).

5. Experiments

We present our results for point cloud classification on ModelNet-C dataset, training on the clean dataset only and measuring performance on both clean and corrupted sets. **Implementation details.** We train the basic models independently on partial point clouds. The predictions are aggregated using mean. For the unaugmented version we use only basic, standard augmentation procedures (detailed below) in order not to violate the OOD principle. For WolfMix augmented version we first augment the entire sample, then we generate the different sub-samples. We eliminate the randomness with a fixed seed. All three models are trained simultaneously 300 epochs with learning rate of $5e^{-4}$, with cosine annealing scheduler [18] to zero. We use a batch size of 256. For the unaugmented version we followed DGCNN [31] protocol for augmentation: 1) random anisotropic scaling in the range $[2/3, 3/2]$; 2) random translation in the range $[-0.2, +0.2]$. The implementation uses Pytorch library [19]. Cross-Entropy loss is minimized. In the training phase we split each sample to 4 farthest point samples, and independently predict each partial point cloud class. The inference procedure is detailed in Algorithm 1.

5.1. Main evaluation results

The EPiC framework is implemented using several widely used point cloud classification networks. Table 1 shows the Unaugmented results and Table 2 shows results following augmentation by WolfMix [25, 4].

Unaugmented. A major advantage is achieved in terms of mCE for all networks. Using PCT[9] with EPiC reaches mCE=0.646, which outperform current SOTA of RPC[24] (mCE=0.863) by a large margin. With respect to accuracy on the clean data set (OA), we see improvement in four out of five cases, with only CurveNet[26] degrading, where RPC[24] and GDANet[13] reach 93.6%.

Augmented (WolfMix). We further examined wheather our method can improve augmentation procedures (which to some extent violate some OOD assumptions). EPiC on augmented data consistently improves robustness. Using RPC [24] EPiC achieves mCE=0.501 surpassing current augmented SOTA (mCE=0.571). OOD robustness comes at a cost of minor accuracy drops, consistently for all networks. The trade-off between accuracy and robustness is well demonstrated in [24] for point-cloud classification, and in [28] for general classification settings.

5.2. When our method performs best?

Let $X \in \mathbb{R}^{N\times 3}$ be the set of coordinates of a clean point cloud. Following a corruption transformation T_c acting on X we get a corrupted point cloud $X_c \in \mathbb{R}^{M\times 3}$, where $X_c = T_c(X)$. Let the intersection of these sets be define by $X_\cap := X \cap X_c$ of size $|X_\cap|$ points. We define the *uniformity* of the corruption transformation by

$$u(X, T_c) := 1 - \frac{|X_\cap|}{max(N, M)}. \quad (3)$$

For a fully uniform corruption T_c, all points of the original point cloud change, hence $X_\cap = \emptyset$ and $|X_\cap| = 0$, yielding $u = 1$. When the transformation is highly selective, affecting only a few points, $X_\cap \approx X$ and $u \to 0$. We refer to the latter as a highly *nonuniform corruption*. This measure is very general and can quantify various diverse corruptions.

We can roughly divide the corruptions of ModelNet-C to

1. *Uniform:* Scale, Rotation.
2. *Nonuniform:* Drop-Global, Drop-Local, Add-Global, Add-Local.

In the case of Jitter one can slightly extend the definition of uniformity by a ball of size ϵ around each point, when calculating $X_\cap$. Then uniformity grows with the standard deviation of the Jitter.

Our method performs best on nonuniform corruptions. In these cases some partial point clouds in the ensemble are mostly not exposed to the corruption, as shown in Fig. 2. In these cases the classification is with high accuracy. Partial point clouds which are highly exposed to the corruption have high chances of yielding missclassification. However, our experiments indicate these missclassifications are quite random and can be approximately modeled as noise. As we perform a *mean* operation over the ensemble outputs, this noise is mostly averaged out, where the correct (mostly unexposed) members of the ensemble dominate the decision. This phenomenon can be seen in the experiment shown and explained in Fig. 6. A plot of mCE as a function of u is shown in Fig. 5, illustrating the above rationale.

The case of Jitter. Let us assume the points in the cloud are sampled approximately evenly, with a mean distance between each point of ℓ. Let the Jitter corruption be of standard deviation σ. If $\sigma \ll \ell$ essentially the point cloud is similar to the original clean one and most classifiers would perform well. In terms of the relaxed definition of uniformity, we can set $\epsilon = \ell$ and for $\sigma \ll \epsilon$ we get $u \to 0$ (the corruption is "invisible" to the classification network). Random sampling copes very well with Jitter. We can view random sampling as reducing the point cloud resolution. Basically, we now have a larger distance $L > \ell$ between the points. Since the classifier is trained on the low resolution input, as long as $\sigma \ll L$ we get $u \to 0$. Hence, we are more robust to Jitter with a larger standard deviation. See [12] for additional perspectives and insights on this topic.

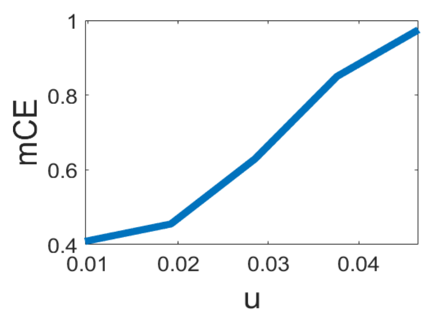


Figure 5: **mCE versus uniformity.** mCE is monotonically increasing with u (average result on test set of ModelNet-C, Add-Global corruption, 5 degrees of severity).

5.3. Sampling, aggregation and network size

In Table 4 we show experimental results for each subsampling method with DGCNN[31] as the basic model (additional models are shown in the Supp.). Ensembles of size four are tested, aggregated using mean. In addition, the results of all three methods (forming an ensemble of 12 members) are aggregated using either mean or majority-voting. Each sampling method has its strengths and weaknesses: Patches are much better in terms of accuracy on clean point clouds. In this case features are well preserved and patches

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN-Curves (#4)	90.7%	1.069	1.628	1.297	0.431	0.729	<u>0.363</u>	1.618	1.414
DGCNN-Patches (#4)	92.8%	0.793	**0.989**	1.165	0.577	0.536	0.505	0.851	**0.930**
DGCNN-Random (#4)	91.5%	0.766	1.234	**0.399**	<u>0.351</u>	0.812	0.580	**0.793**	1.195
DGCNN-Mean (#12)	**93.0%**	**0.669**	<u>1.000</u>	<u>0.680</u>	**0.331**	**0.498**	**0.349**	0.807	<u>1.019</u>
DGCNN-Maj. Voting (#12)	<u>92.6%</u>	<u>0.706</u>	1.043	0.794	0.359	<u>0.517</u>	0.380	<u>0.800</u>	1.051

Table 4: **Sub-Samples vs. Aggregated. Bold** best among aggregations, <u>underline</u> best among sub-samples. Aggregations are almost always superior for any corruption. Thus, it can be inferred that the partial classifiers are low-correlated.

have access to the full resolution within a region. Since patches have compact support, they perform better also on Drop-L and are somewhat immune to global corruptions (but less so, compared to the two other sampling methods). Curves perform best on Add-G. Intuitively, curves are most likely to sample regions with high conductance, thus global outliers are less likely to be selected. Random is excellent in the case of Jitter and generally has the best mCE among sampling methods. The aggregated ensemble results, either using mean or majority-voting, are both better than any specific sampling method. Mean yields the best mCE, while majority voting, a popular ensemble aggregation method, turns out to be slightly worse (this trend is consistent when using other networks as well).

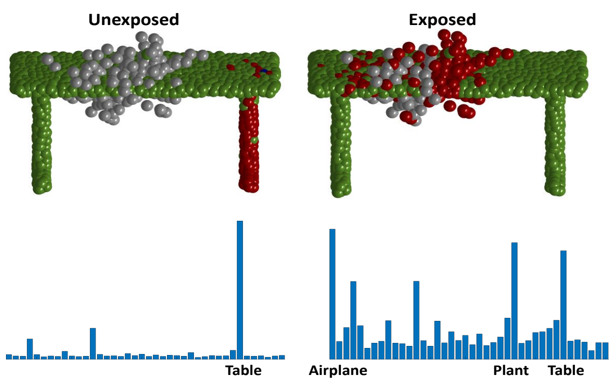


Figure 6: **Exposure to corruption modeled as noise**. This illustration is based on the following experiment: All instances of class Table were corrupted by Add-Local. Curve is used for sampling. The sampled instances were divided into two groups: **Unexposed** (left) are curves containing less than 10 corrupted points. **Exposed** (right) are curves containing more than 50 corrupted points. Soft-Max predictions of each group were averaged (bottom row). The prediction of exposed curves is almost random (highly noisy) and can be well handled by averaging over the ensemble outputs. Unexposed instances are classified well.

Affect of network size. Since each ensemble member has access only to partial data , we check weather a full size model is required. Five versions of DGCNN are examined, the most shallow is $v1$ and the deepest is $v5$, where $v4$ is the original network (architectures details appear in the Supp.). The results are shown in Table 5. The number of parameters required for the entire ensemble (3 instances) is shown on the right column. Note that $v2$ yields a robust version with about third of the parameters of the classical network (top row), with just a slight degradation in overall accuracy.

Network	OA ↑	mCE ↓	#parameters
Classic DGCNN	**92.6%**	**1.000**	**1.8M**
EPiC based on			
DGCNN-v1	88.7%	1.041	159K
DGCNN-v2	**92.2%**	**0.773**	**636K**
DGCNN-v3	92.3%	0.720	2.47M
DGCNN-v4	93.0%	0.669	5.4M
DGCNN-v5	92.7%	0.684	39M

Table 5: **Network size vs Performance.** EPiC with DGCNN-v2 is an excellent compromise of a lean architecture, with a small number of parameters, which is much more robust (and almost as accurate), compared to the original full DGCNN network.

6. Conclusion

In this work, we demonstrated the OOD robustness of ensembles based on partial information input for the task of point cloud classification. The approach relies on obtaining lowly-correlated input samples for each member of the ensemble. We integrate three types of sampling schemes: Curves, Patches and Random. In terms of training - only a single network for each sampling scheme is trained, which saves training time. The ensemble is created in real time. This naturally increases inference time, but is reasonable for small ensembles (we show a size of $K = 12$). For highly demanding real time applications, the networks can be duplicated and processed in parallel. For highly demanding memory consumption applications, significantly smaller networks can be used, which are still highly robust. Note that ensembles can be distilled back to a single network, for better hardware and time efficiency, as suggested for instance in [41]. Since our proposed approach is purely abstract it can be extended to additional problems. We plan to further investigate how such mechanisms can improve robustness in other fields as well.

Acknowledgements

We acknowledge support by the Israel Science Foundation (Grant No. 534/19), by the Ministry of Science and Technology (Grant No. 5074/22) and by the Ollendorff Minerva Center.

References

[1] Goyal A., Law H., Liu B., Newell A., and Deng J. Revisiting point cloud shape classification with a simple and effective baseline. *International Conference on Machine Learning*, 2021. 2, 3

[2] Lakshminarayanan Balaji, Pritzel Alexander, and Blundell Charles. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in neural information processing systems*, 30, 2017. 1

[3] Chao Chen, Guanbin Li, Ruijia Xu, Tianshu Chen, Meng Wang, and Liang Lin. Clusternet: Deep hierarchical cluster network with rigorously rotation-invariant representation for point cloud analysis. *In Conference on Computer Vision and Pattern Recognition, pages 4994–5002*, 2019. 2

[4] Lee D., Lee J., , Lee H., Lee M., Woo S., and Lee S. Regularization strategy for point cloud via rigidly mixed sample. *In CVPR, pp. 15900–15909. Computer Vision Foundation / IEEE*, 2021. 2, 3, 7

[5] Liu D and Yu R. Su. Extending adversarial attacks and defenses to deep 3d point cloud classifiers. *arXiv:1901.03006*, 2019. 3

[6] T. G. Dietterich. Ensemble methods in machine learning. *In J. Kittler and F. Roli, editors, Multiple Classifier Systems, pages 1–15. LNCS Vol. 1857, Springer*, 2001. 1

[7] Zhao Chen et al. Rotation invariant point cloud classification: Where local geometry meets global topology. *arXiv preprint arXiv:1911.00195*, 2019. 2

[8] Yoav Freund and Robert E. Schapire. Experiments with a new boosting algorithm. *In ICML, volume 96, pages 148–156*, 1996. 1

[9] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R. Martin, and Shi-Min Hu. Pct: Point cloud transformer. *arXiv preprint arXiv:2012.09688*, 2020. 2, 3, 5, 6, 7

[10] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 12, no. 10, pp. 993–1001*, 1990. 1

[11] Dan Hendrycks and Thomas G Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *In International Conference on Learning Representations*, 2019. 1

[12] Hongbin Liu, Jinyuan Jia, and Neil Zhenqiang Gong. Pointguard: Provably robust 3d point cloud classification. *In IEEE Conference on Computer Vision and Pattern Recognition, CVPR*, 2021. 3, 6, 7

[13] Xu M., Zhang J., Zhou Z., Xu M., Qi X., and Qiao Y. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. *In AAAI, pp. 3056–3064. AAAI Press*, 2021. 2, 3, 5, 6, 7

[14] Xu M., Ding R., Zhao H., and Qi X. Paconv: Position adaptive convolution with dynamic kernel assembling on point clouds. *In CVPR*, 2021. 3

[15] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *in International Conference on Learning Representations (ICLR)*, 2022. 2

[16] Daniel Maturana and Sebastian Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. *In 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pages 922–928*, 2015. 2

[17] N. Mu and J. Gilmer. Mnist-c: A robustness benchmark for computer vision. *http://arxiv.org/abs/1906.02337*, 2019. 1

[18] Smith Leslie N. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*, 2018. 6

[19] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, and Luca Antiga et al. Pytorch: An imperative style, high-performance deep learning library. *In Advances in Neural Information Processing Systems (NeurIPS), pages 8024–8035*, 2019. 6

[20] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *IEEE Conference on Computer Vision and Pattern Recognition*, pages 77–85, 2017. 2, 3

[21] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *In Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 3, 4

[22] Qi C. R., Su H., Nießner M., Dai A., Yan M., and Guibas L. J. Volumetric and multi-view cnns for object classification on 3d data. *In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 5648–5656*, 2016. 2

[23] 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. *In Computer Graphics Forum. Wiley Online Library*, 2019. 2

[24] Jiawei Ren, Liang Pan, and Ziwei Liu. Benchmarking and analyzing point cloud classification under corruptions. *International Conference on Machine Learning (ICML)*, 2022. 1, 2, 3, 6, 7

[25] Kim S., Lee S., Hwang D., Lee J., Hwang S. J., and Kim H. J. Point cloud augmentation with weighted local transformations. *In Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 548–557*, 2021. 2, 3, 7

[26] Xiang T., Zhang C., Song Y., Yu J., and Cai W. Walk in the cloud: Learning curves for point clouds shape analysis. *In*

Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 915–924, 2021. 2, 3, 7

[27] C. R. Thomas H. andQi, Deschaud J. E., Marcotegui B., Goulette F., and Guibas L. J. Kpconv: Flexible and deformable convolution for point clouds. *In Proceedings of the IEEE/CVF international conference on computer vision (pp. 6411-6420).*, 2019. 2

[28] Tsipras and Dimitris et al. Robustness may be at odds with accuracy. *arXiv preprint arXiv:1805.12152*, 2018. 7

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *In Advances in Neural Information Processing Systems, pages 6000–6010*, 2017. 2

[30] Xianyu Wang, Ke Zhang, Haoyu Li, Hua Luo, and Jingyu Wang. Probnet: Bayesian deep neural network for point cloud analysis. *Computers and Graphics 104: 106-115*, 2022. 2

[31] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph cnn for learning on point clouds, 2019. ACM Transactions on Graphics, 38(5):146:1–146:12. 2, 3, 4, 5, 6, 7

[32] 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 Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015, pages 1912–1920*, 2015. 2, 3

[33] Chong Xiang, Charles R Qi, and Bo Li. Generating 3d adversarial point clouds. *arXiv preprint arXiv:1809.07016*, 2018. 3

[34] Liu Y., Fan B., Xiang S., and Pan C. Relation-shape convolutional neural network for point cloud analysis. *In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 8895–8904*, 2019. 3

[35] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui. Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling. *in Proceedings IEEE Conference Computer Vision Pattern Recognition*, 2020. 2

[36] Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. *ArXiv, abs/2111.14819*, 2021. 2

[37] W. Yuan, D. Held, C. Mertz, and M. Hebert. Iterative transformer network for 3d point cloud. *arXiv preprint arXiv:1811.11209*, 2018. 3

[38] Zaidi, Sheheryar, Zela Arber, Elsken Thomas, Holmes Chris C., Hutter Frank, and Teh Yee. Neural ensemble search for uncertainty estimation and dataset shift. *Advances in Neural Information Processing Systems*, 34:7898–7911, 2021. 2

[39] Zefrehi, Hossein Ghaderi, and Altınçay Hakan. Imbalance learning using heterogeneous ensembles. *Expert Systems with Applications*, 142:113005, 2020. 1

[40] J. Zhang, W. Chen, Y. Wang, R. Vasudevan, and M. Johnson-Roberson. Point set voting for partial point cloud analysis. *arXiv preprint arXiv:2007.04537*, 2020. 3

[41] Shen Zhiqiang, He Zhankui, and Xue Xiangyang. Meal: Multi-model ensemble via adversarial learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4886–4893, 2019. 9