Robustifying Point Cloud Networks by Refocusing

Supplementary Material

6. Filtering in feature space

The algorithm outlined in the main paper involves a dual forward pass. The first pass is necessary for querying the importance score, while the second pass is utilized to predict the point cloud filtered based on the previously calculated importance score. Employing filtering in Euclidean space serves as a safeguard against the propagation of outliers from the primal space to the feature space. An alternative approach we propose is to perform filtering directly in the feature space instead of the input space. This alternative method requires only a single forward pass of the base network, incurring no additional latency. This approach can be likened to a dropout layer that emphasizes the filtration of dominant feature elements instead of random ones. While feature space filtering yields lower robustness compared to filtering in the primal space, as outliers in the Euclidean space can still contaminate features, it does provide a notable boost in robustness, compared to the vanilla base network. Thus, if speed is essential, one can consider also this approach in point cloud classification networks. To illustrate this improvement, we integrated feature filtering into the DGCNN and RPC models and evaluated their performance on the ModelNet-C dataset. The results, as presented in Tab. 4, highlight the significant enhancement in robustness achieved through the implementation of our proposed layer. This improvement is demonstrated in comparison to the vanilla version of the network, with no associated latency cost.

7. Benchmarks

ModelNet40. Synthetic dataset, constitutes a widely employed collection of CAD meshes spanning 40 distinct classes, encompassing objects such as monitors, beds and persons. These meshes have undergone uniform sampling procedures, resulting in the formation of 3D point clouds, each comprising 1024 points. The dataset encompasses a total of 12,311 samples, partitioned into 9,843 samples designated for training and 2,468 samples allocated for testing purposes.

ModelNet-C. Ren et al., introduced a corrupted point cloud benchmark denoted as ModelNet-C, based on ModelNet40, to facilitate the OOD robustness. ModelNet-C encompasses a spectrum of seven distinct corruption types, namely jitter, scale, rotation, add-global, add-local, drop-global, and drop-local, each exhibiting five levels of difficulty. To quantitatively gauge robustness, introduced a comprehensive metric termed mean Corruption Error (mCE), a relative robustness measure with DGCNN algorithm serving as a pivot network (and thus by definition mCE value of 1). Given the error-based nature of this metric, a lower mCE score is indicative of superior performance.

ScanObjectNN. Contains 2902 point clouds spanning 15 distinct categories, constituting a more intricate collection that arises from real-world scans featuring background elements and instances of occlusion. There exist classes which overlapp with ModelNet40 classes e.g. chairs, desks and sofas.

8. Implementation details

The procedure of the training phase is provided in Alg. 2.

Algorithm 2 $Refocusing$ (Training)

Require: $DataSet, Influence$
  for $X, label \in DataSet$ **do**
    $X_f = model(X)$
    $\hat{I}_F(X) = \frac{Influence(X)}{\Sigma_{i=1}^{N}(Influence(X_i))}$
    $H_n = NormalizedEntropy(\hat{I}_F)$
    $K = rand(256, 1024)$
    $X_{sampled} = SelectLowest(X, \hat{I}_F, K)$
    $P = model(X_{sampled})$
    $L = CrossEntropyLoss(P, label)$
    $L.backward()$
  end for
  $params_{Refocusing} \leftarrow BestModelParams$

8.1. Adversarial defense

Defense parameters. Following the Shape-Invariant method, SRS was applied with 30% of filtered points, as well as with 50%, while SOR was applied with 2 KNN and $\sigma = 1.1$. Regarding LPF-Defense, we utilized $l_{max} = 16$, and in line with the ablation study conducted in LPF-Defense paper, we executed it with $\sigma = 20$. The paper recommends training the network on the clean dataset and employing these parameters during inference. Consequently, for each network, training was performed with these parameters and testing was conducted using the same parameters. The training process was over 300 epochs, employing the ADAM optimizer with a learning rate of 5^{-3}, momentum of 0.9, and weight decay of 1^{-4}.

Attack setting. To underscore our defense capabilities, and recognizing that the Shape-Invariant attack leverages a sensitivity map characterized by transferability across diverse networks, we decided to examine the most challenging scenario. This entails investigating the situation where the

Model	Approach	OA%	mCE	#Forward Pass
DGCNN	Vanilla	92.6	1.000	1
	Refocusing feature space.**(Ours)**	92.7	0.818	1
	Refocusing Euclidean space.**(Ours)**	91.6	0.688	2
RPC	Vanilla	93.0	0.863	1
	Refocusing feature space**(Ours)**	92.0	0.797	1
	Refocusing Euclidean space **(Ours)**	91.6	0.728	2

Table 4. **Filtering in feature space vs. Euclidean space using Refocusing** Employing *Refocusing* for filtering in the feature space, as opposed to the primary Euclidean space, enhances robustness compared to the standard network, all without incurring any latency costs.

Model	Corruption	Severity-0 (%)	Severity-1 (%)	Severity-2 (%)	Severity-3 (%)	Severity-4 (%)
DGCNN [44]	Scale	89.9	**90.1**	88.9	89.5	89.3
	Jitter	**91.3**	**89.9**	**88.4**	**84.5**	**79.5**
	Rotate	91.0	**88.8**	**82.7**	**70.8**	**56.1**
	Dropout Global	91.0	91.2	90.7	89.1	83.3
	Dropout Local	90.3	88.1	83.6	79.1	68.6
	Add Global	**91.7**	**91.6**	**91.4**	89.3	86.5
	Add Local	**88.6**	**85.2**	**82.4**	**80.5**	**77.7**
GDANet [52]	Scale	**90.6**	89.8	90.3	**89.9**	**89.4**
	Jitter	91.1	89.3	87.1	82.2	73.9
	Rotate	91.0	88.3	79.4	68.4	54.2
	Dropout Global	91.0	90.5	90.4	88.1	81.5
	Dropout Local	90.9	87.3	83.3	77.8	66.2
	Add Global	91.6	91.4	91.3	**90.4**	**89.1**
	Add Local	88.3	85.1	79.9	76.9	74.7
RPC [35]	Scale	90.4	**90.1**	**90.5**	89.8	**89.4**
	Jitter	90.7	87.6	83.7	76.8	64.9
	Rotate	**91.5**	86.8	77.7	64.1	51.4
	Dropout Global	**91.8**	**92.0**	**91.5**	**91.1**	**89.3**
	Dropout Local	**91.4**	**89.4**	**87.4**	**81.2**	**71.5**
	Add Global	88.6	87.3	86.3	84.6	84.0
	Add Local	88.4	84.6	81.6	78.5	76.3

Table 5. **Comprehensive Performance Table on ModelNet-C by Corruption Severity.** The best performance for each corruption at a given severity level is marked in **bold**. RPC excels in drop corruptions, while DGCNN performs best in add corruptions. Note that GDANet's performance is almost unaffected by the severity of global additions, similar to RPC's performance in global dropping.

surrogate model, responsible for generating the sensitivity map, aligns with the victim model. DGCNN serves as both the surrogate model and the victim. Similarly, PointNet fulfills dual roles as the surrogate and victim, as does GDANet.

8.2. Robust classification

In the unaugmented setting, we initially apply conventional augmentation methodologies to adhere to the OOD principle. In contrast, for the augmented iteration using WolfMix, the augmentation process is first applied to the sample, followed by our importance-based subsampling technique. To mitigate the effects of randomness, a predetermined seed is utilized. All models undergo training for 300 epochs in the unaugmented scenario and 500 epochs in the augmented case, utilizing a learning rate of 5e-4. A cosine annealing scheduler is employed to drive the learning rate to converge to zero. A batch size of 64 is adopted. In the unaugmented version, the augmentation protocol outlined by DGCNN is followed, encompassing two steps: 1) stochastic anisotropic scaling spanning the range of [2/3, 3/2]; and 2) random translation within the interval of [-0.2, +0.2]. The implementation makes use of the PyTorch library. The primary training objective involves minimizing the Cross-Entropy loss. During training, we computed the importance score and cropped the sample to a random size within the range of [256, 1024] contains points with the lowest importance.

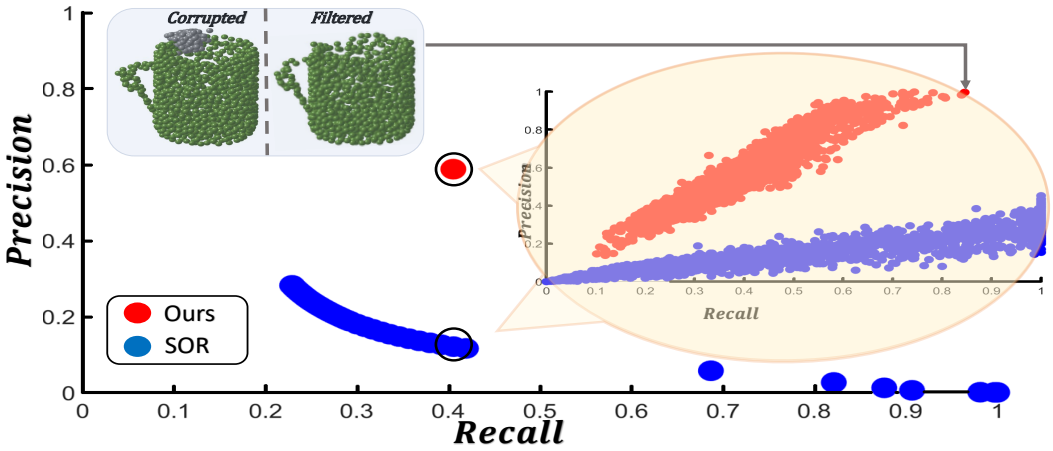

Figure 8. **Outlier removal on add-local.** Refocusing outperforms SOR investigated on a grid of standard deviations. **Zoom-in on SOR with corresponding recall.** Our approach achieves superior results on each sample, precise removal is exemplified by the cup shapes.

This prepares the model to accommodate a wide range of sample sizes during inference, where we crop the sample to an unknown size based on the adaptive threshold. Due to variations in the number of points in each cloud, inference is performed using a batch size of 1, a common practice in real-world applications. Timing estimates are obtained by averaging the time taken for 100 iterations of batches with a size of 1.

The combination of our sampling approach with EPiC is achieved by applying our sub-sampling technique once for each sample, resulting in a fixed size of 600 points. The prediction is then duplicated four times and concatenated with other sub-samples. This duplication is carried out to ensure an equal impact for each of the sampling schemes. Consequently, the ensemble created using our sampling approach comprises 16 members.

8.3. Full results

We present comprehensive and detailed results encompassing both the adversarial attack experiment (refer to Tab. 6) and the robust classification analysis (refer to Tab. 7). **Adversarial attack experiment.** The comprehensive table includes three additional distance measures between adversarial and benign shapes: Chamfer, Hausdorff, and MSE distances. **Robust classification**. The detailed table contains additional results for mCE under each specified corruption. Notably, certain defenses display a correlation with specific corruptions across various networks. For instance, Drop-Local is most effectively mitigated by EPiC, while scale is best addressed by PointGuard. However, when considering the mCE metric, which aggregates the overall robustness scores across all corruptions, EPiC & Refocusing (our method combined with EPiC) consistently achieve the highest scores among the networks examined. Moreover, we elaborate on the accuracy performance of each network across all five degrees of severity. For a detailed explanation of how the corruptions were constructed, please refer to the ModelNet-C paper [35]. The results are summarized in Tab. 5.

9. Discussion and future directions

Outliers removal. In outlier detection and removal, two main categories of methods are prevalent. Learnable methods, which require clean and corrupted pairs as ground truth, thus unsuitable for the OOD regime, which is our primary focus. Classical methods are a valid choice, with statistical outlier removal (SOR) being a common example. We note that SOR requires prior knowledge on the outlier characteristics. Moreover, its performance deteriorates for subtle, smooth corruptions, found in real-world scenarios. For outliers removal task we propose to use a another variant of influence, defined as: $I_F(i) = \sum_{k=1}^{G} |X_G(i,k)|$. And average as adaptive threshold as follows: $S_X := \{X_i : I_F(i) \leq \frac{\Sigma_{i=1}^{N}(I_F(i))}{N}\}$. A preliminary evaluation of our approach appears promising. We attempt to detect Add-local corruption of ModelNet-C. For SOR, we experiment with various σ values, calculating mean recall and precision for each. We focus on the settings that yield the same recall as our method and present recall-precision values per sample (See Fig. 8). Our findings conclusively demonstrate the superior performance of our approach in effective outlier detection.

Statistical acquisition analysis. One can analyze certain statistical aspects of different acquisition scenarios. For instance, acquisition of objects containing background and ones which do not. Samples with background exhibit relatively higher focus, as the background may contain prominent features that divert the network's attention. See Fig. 9 for analysis of the real-world ScanObjectNN dataset.

Surrogate	Defense	ASR(%)	A.Q(times)	C.D(10^{-4})	H.D(10^{-2})	MSE
DGCNN	–	99.3	106.7	3.18	4.54	1.22
	SOR	75.6	795.6	2.59	3.48	1.65
	SRS (50%)	78.4	566.3	1.21	3.59	0.69
	SRS (30%)	68.6	790.3	1.64	4.11	0.85
	LPF-Defense	47.8	1148.0	1.84	4.09	0.93
	Refocusing (Ours) - Fixed (600)	43.5	1265.0	2.30	4.16	1.11
	Refocusing (Ours) - Adaptive	**37.5**	**1376.1**	2.28	4.06	1.17
PointNet	-	99.8	18.9	2.09	4.46	0.80
	SOR	78.4	592.9	9.26	4.33	2.74
	SRS (50%)	94.0	190.9	1.67	4.42	0.71
	SRS (30%)	97.6	93.5	1.76	4.43	0.72
	LPF-Defense	98.2	123.1	5.36	4.56	1.41
	Refocusing (Ours) - Fixed (600)	74.1	693.6	11.33	4.70	2.66
	Refocusing (Ours) - Adaptive	**72.0**	**730.4**	11.13	4.68	2.72
GDANet	-	99.4	95.6	4.16	4.67	1.27
	SOR	69.9	913.2	3.67	3.98	1.78
	SRS (50%)	78.1	595.4	1.70	4.16	0.78
	SRS (30%)	72.4	714.0	2.19	4.42	0.92
	LPF-Defense	52.6	1071.48	2.18	4.26	0.94
	Refocusing (Ours) - Fixed (600)	**32.9**	**1447.3**	2.18	4.32	1.03
	Refocusing (Ours) - Adaptive	34.6	1425.5	2.13	4.21	1.03

Table 6. **Comprehensive table of adversarial defenses against Shape-Invariant attack on ModelNet40.** Incorporating our approach with a fixed threshold (600 points). Our adaptive approach outperforms the fixed threshold in two out of the three evaluated networks.

Model	Approach	OA%	mCE	Scale	Jitter	Drop-Global	Drop-Local	Add-Global	Add-Local	Rotate
DGCNN	Vanilla	92.6	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	EPiC	93.0	0.669	1.000	0.680	0.331	0.498	0.349	0.807	1.019
	PointGuard	83.8	1.165	0.743	0.809	0.840	0.812	0.838	0.785	0.584
	Refocusing	91.6	0.688	0.896	0.868	0.891	0.820	0.902	0.829	0.779
	EPiC & Refocusing	93.4	0.557	0.957	0.494	0.335	0.522	0.258	0.382	0.949
RPC	Vanilla	93.0	0.863	0.840	0.892	0.492	0.797	0.929	1.011	1.079
	EPiC	93.6	0.750	0.915	1.057	0.323	0.440	0.281	0.902	1.330
	PointGuard	86.9	1.051	0.773	0.817	0.868	0.843	0.867	0.804	0.590
	Refocusing	91.6	0.728	0.901	0.808	0.912	0.842	0.862	0.819	0.744
	EPiC & Refocusing	93.2	0.616	0.957	0.690	0.319	0.464	0.258	0.444	1.177
GDANet	Vanilla	93.4	0.892	0.830	0.839	0.794	0.894	0.871	1.036	0.981
	EPiC	93.6	0.704	0.936	0.864	0.315	0.478	0.295	0.862	1.177
	PointGuard	84.8	1.132	0.755	0.804	0.847	0.819	0.846	0.787	0.589
	Refocusing	91.4	0.718	0.900	0.848	0.884	0.812	0.908	0.810	0.763
	EPiC & Refocusing	93.4	0.587	0.926	0.617	0.323	0.512	0.258	0.393	1.079

Table 7. **Comprehensive comparison table for augmented free ModelNet-C.**

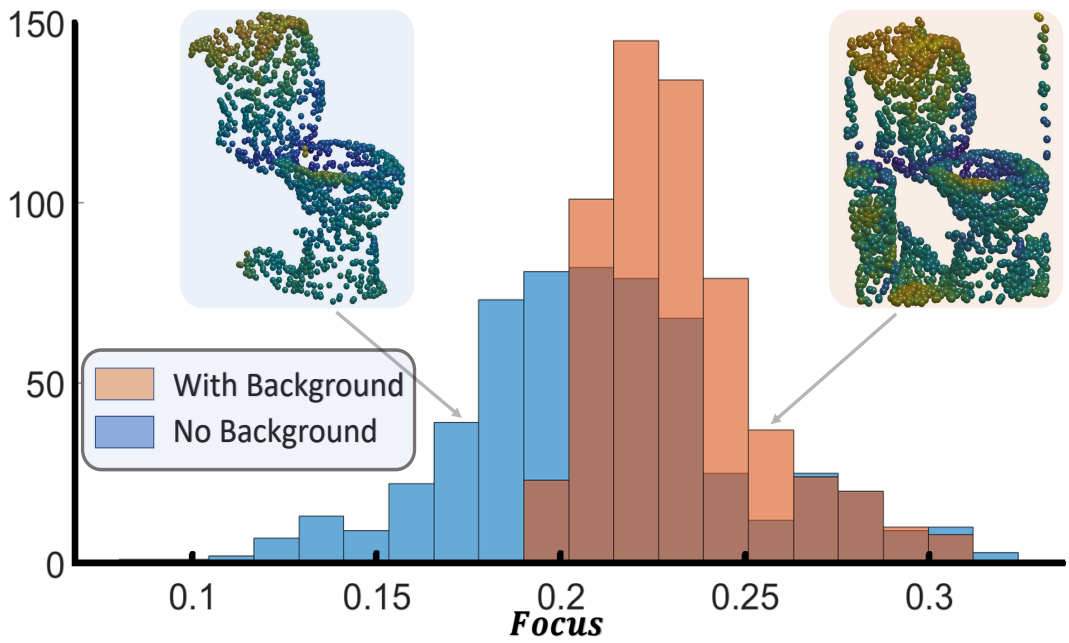

Figure 9. **Focus histogram on ScanObjectNN.** Samples contains background yield higher focus than those without background.