7. Ablation Studies

We conducted comprehensive ablative studies in order to better understand the impact of each hyper-parameter. The hyper-parameters are: Number of nearest neighbors for curve extraction, M; Number of points in Curve sampling, N_c; Number of points in Patch sampling, N_p; Number of points in Random sampling, N_r and number of partial point clouds for each sampling method, $\tilde{K}$ (we retained the restriction that $\tilde{K}$ is the same for all sampling methods). Our parameter search was performed in the following manner: The base parameters were chosen to be $M = 40$, $N_c = 576$, $N_p = 512$, $N_r = 128$ and $\tilde{K} = 32$. At each stage we scanned the values of a specific parameter, while fixing all other parameters (the order is as listed above). We fixed the parameter to the optimal value after each stage. The optimized value was chosen manually by considering both overall accuracy and robustness to corruptions. Mean is used as an aggregating scheme. The chosen values of hyper-parameters are: $M = 40$, $N_c = 512$, $N_p = 512$, $N_r = 128$ and $\tilde{K} = 4$. We observe that local sub-samples should be large enough to allow both accuracy and robustness. A global sub-sampling, such as random, requires far less points. Surprisingly, the ensemble size may be very small. Good results are obtained for $\tilde{K} = 4$, allowing very reasonable inference time. See Figs 7, 8, 9, 10 and 11.

Extended Ablation Studies We demonstrate the influence of each hyper-parameter under any of the corruptions in Figs 12 and 13. The most meaningful hyper-parameter is $\tilde{K}$. Ensemble size of 12 yielding very good trade-off between running time, overall accuracy and robustness to OOD degradations. All corruptions are improved when enlarging the ensemble size, where Add-L is affected the most.

8. Training procedure

We train all three partial point-cloud networks simultaneously and derivate each sub-sampling network's loss with regard to the entire point-cloud label as detailed in Algorithm 2

9. Sub-Samples analysis of more models

GDANet. For training, We lower the batch size to 50 and changed the GDM Module to accept point clouds with less than 256 elements by lower M value to 128 in Geometry-Disentangle Module (GDM).

All other networks did not require any extra tweaks, the analysis results of all networks is detailed in Tables 7, 8, 9, 10, 11, 12, 13 and 14. Additionally, for a fair comparison, we analyze the Un-Augmented DGCNN model (Tab. 9) with each of the sub-sampling strategies applied using 12 members, instead of 4. The results confirm the main paper's assertion that robustness is achieved through the diversity of the ensemble.

10. Network size (shallowness) analysis

DGCNN is constructed from 4 levels of depth. Features of current level are recalculated based on features of previous level. In order to examine the required depth for our case we used different versions of DGCNN. Each version is constructed using a different number of graph reconstructions. For example, DGCNN-v1 is built only from one graph based on the spatial distance, therefore features are directly derived from spatial information. As depth increases (DGCNN-v2 through DGCNN-v5) the network acquires additional, more complex, semantic contextual information. DGCNN-v1 has up to 64 embedding dimensions, DGCNN-v2 up to 128, and so on. Each version doubles the embedding dimensions (which grows exponentially with depth). Comparing to the classical DGCNN (v4), we observe that EPiC with DGCNN-v2 yields a highly robust network (MCE=0.773 compare to 1.000) with much less learnable parameters (636K compare to 1.8M). This comes at the cost of a slight accuracy drop (92.2% compared to 92.6%). Thus, the EPiC framework can be very lean and economic from a production perspective.

11. Corruptions and sub-sampling

Here we give some of our insights on how the different sub-sampling methods react to different corruptions. Examples are shown in Figs. 14, 15, 16 and 17.

Add global. Curves are highly insensitive to relatively far points (which often appear in Add-G). Therefore, they perform best under this corruption. Patches and Random are not influenced by the conductance, thus the added points may be considered as well, corrupting the sample.

Add local. Here, the corruption is smoother in space and may be interpreted as part of the shape. Patches and Curves are more likely to be affected by it.

Drop Global. Drop Global can be viewed as globally changing the density of the points of the shape. Therefore, it makes sense that there is a direct proportion between sub-sampling performance and typical density (Patches are dense while Random is sparse, Curves are somewhere in between). Moreover, we take into account a variable number of points, e.g. for Patches $min(N, N_p)$. Thus, when the number of points is lower, as in the case of Drop-Global, Patches cover more of the shape.

Drop Local. This corruption may cause separation between different shape parts (creating disconnected regions). Curves may thus be "stuck" in a disconnected region. Patches are less affected by this corruption and perform better.

11.1. Networks and augmentation evaluation

Evaluation of most augmented and un-augmented networks was reproduced by the code supplied by Ren et al. Additionally, in the ensemble section we show results on PointGuard. These experiments are detailed below.

Point Guard. This method obtains highly sparse random sub-samples (they suggest $N = 16$), using a large ensemble of size $K = 1,000$. Its theoretical analysis and experimental setting do not assume a specific architecture for the classifier. Thus, DGCNN is used as the basic classifier. As suggested in PointGuard we trained our classifier for 7500 epochs to work on the sub-samples, obtaining an overall accuracy of 80.5% (estimated on randomly 4 sub-samples per each sample in the test set). Predictions are aggregated using majority voting. Note that to obtain provable robustness they use very small N and try to compensate by extremely large K. However this tradeoff turns out to be significantly inferior in terms of overall accuracy and less competitive in mCE.

WolfMix. We follow ModelNet-C evaluation metrics. we use the default hyper-parameters in PointWOLF (Kim et al., 2021). We set the number of anchors to 4, sampling method to farthest point sampling, kernel bandwidth to 0.5, maximum local rotation range to 10 degrees, maximum local scaling to 3, and maximum local translation to 0.25. AugTune proposed along with PointWOLF is not used in training. For the mixing step, we use the default hyper-parameters in RSMix (Lee et al., 2021). We set RSMix probability to 0.5, β to 1.0, and the maximum number of point modifications to 512. For training, the number of neighbors in k-NN is reduced to 20, the number of epochs is increased to 500.

12. On Batch Normalization and OOD Robustness

Using fixed learnt batch normalization parameters at test time can be problematic, since it assumes that the samples of the train and test are both approximately of similar distributions. However, different types of corruptions can yield different statistics. Obviously, learning the statistics at test time may violate the OOD principle. Nevertheless, for some "offline" applications, which can work in batches, we wanted to examine whether test time batch normalization is helpful and increases robustness. To demonstrate this idea, normalization parameters were computed during evaluation (with a batch size of 256). We observe that mCE significantly improves, as can be seen in Table 15.

13. Mixture of Corruptions.

We went a step further to create a more realistic data setting by analyzing our proposed method on a blend of corruptions from ModelNet-C. Comparing our results with those of existing networks supports our strategy in challenging scenarios, which are closer to real-world corruptions (Tab. 6).

Mixture	EPiC (DGCNN)	EPiC (RPC)	RPC (cur SOTA)
Add-G+Drop-G	0.589	**0.493**	0.912
Add-G+Drop-L	0.580	**0.472**	1.106
Add-L+Drop-G	**0.882**	0.902	0.911
Add-L+Drop-L	**0.815**	0.847	0.929

Table 6: **mCE of challenging mixtures.** Large improvement from RPC (SOTA) with substantial deviation from training.

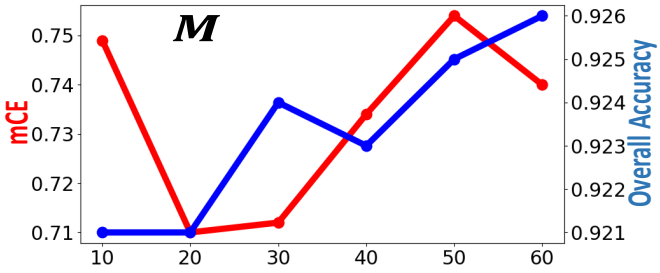

Figure 7: **Neighbors in random walk, curve extraction**.

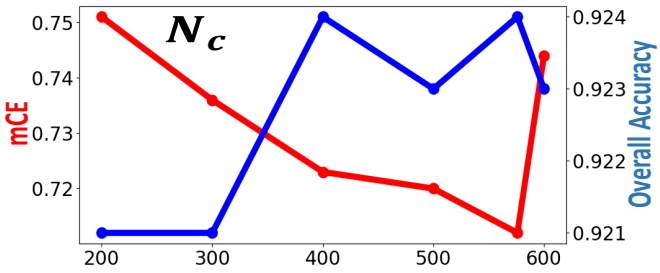

Figure 8: **Curve sub-sample size**.

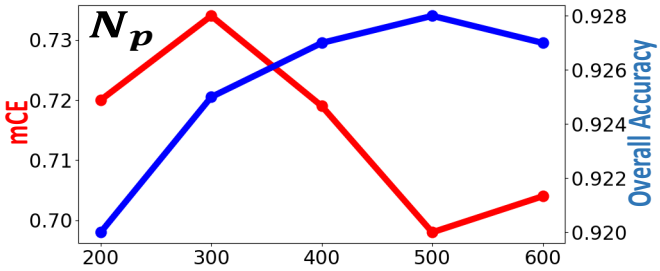

Figure 9: **Patch sub-sample size**.

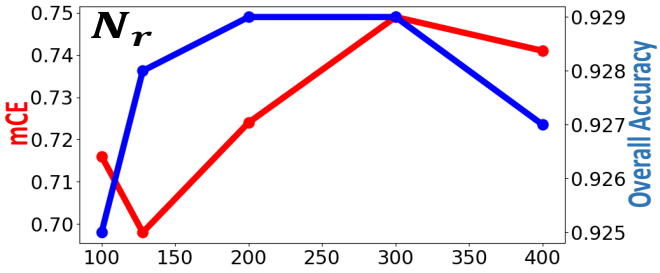

Figure 10: **Random sub-sample size**.

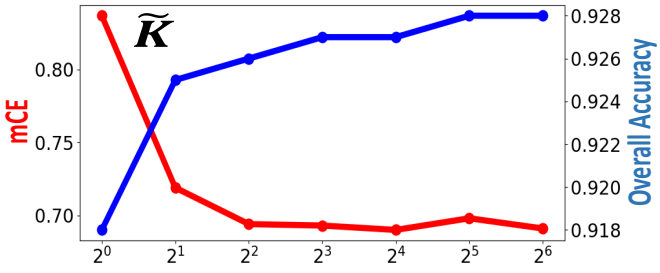

Figure 11: **Ensemble size per sub-sample.**

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
GDANet-Curves (#4)	91.3%	1.047	1.298	1.551	0.423	0.599	0.336	1.549	1.572
GDANet-Patches (#4)	93.2%	0.861	0.957	1.358	0.504	0.551	0.481	1.044	**1.130**
GDANet-Random (#4)	90.9%	0.819	1.287	**0.462**	0.359	0.836	0.512	0.898	1.381
GDANet-Mean (#12)	**93.6%**	**0.704**	**0.936**	0.864	**0.315**	**0.478**	**0.295**	**0.862**	1.177
GDANet-Maj. Vot.(#12)	93.2%	0.749	0.968	1.028	0.343	0.498	0.325	0.876	1.205

Table 7: **Un-Augmented GDANet sub-samples analysis. Bold** best. Underline second best.

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
GDANet-Curves (#4)	90.8%	0.740	1.266	0.984	0.411	0.614	0.332	0.818	0.758
GDANet-Patches (#4)	92.1%	0.667	0.979	1.275	0.464	0.493	0.353	0.531	**0.572**
GDANet-Random (#4)	90.8%	0.646	1.234	**0.462**	0.383	0.758	0.359	0.571	0.758
GDANet-Mean (#12)	**92.5%**	**0.530**	**0.968**	0.639	**0.343**	**0.473**	**0.275**	**0.433**	0.577
GDANet-Maj. Vot.(#12)	92.2%	0.558	1.000	0.725	0.355	0.478	0.292	0.462	0.591

Table 8: **WolfMix Augmented GDANet sub-samples analysis. Bold** best. Underline second best.

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	0.363	1.618	1.414
DGCNN-Curves (#12)	91.3%	1.011	–	–	–	–	–	–	–
DGCNN-Patches (#4)	92.8%	0.793	**0.989**	1.165	0.577	0.536	0.505	0.851	**0.930**
DGCNN-Patches (#12)	92.6%	0.795	–	–	–	–	–	–	–
DGCNN-Random (#4)	91.5%	0.766	1.234	**0.399**	0.351	0.812	0.580	**0.793**	1.195
DGCNN-Random (#12)	91.5%	0.742	–	–	–	—	–	–	–
DGCNN-Mean(#12)	**93.0%**	**0.669**	1.000	0.680	**0.331**	**0.498**	**0.349**	0.807	1.019
DGCNN-Maj. Vot.(#12)	92.6%	0.706	1.043	0.794	0.359	0.517	0.380	0.800	1.051

Table 9: **Un-Augmented DGCNN sub-samples analysis. Bold** best. Underline second best.

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
DGCNN-Curves (#4)	89.5%	0.802	1.426	0.896	0.468	0.676	0.386	0.873	0.888
DGCNN-Patches (#4)	91.7%	0.618	1.053	0.823	0.536	0.512	0.380	0.433	**0.591**
DGCNN-Random (#4)	91.2%	0.639	1.298	**0.405**	0.379	0.768	0.369	0.527	0.730
DGCNN-Mean (#12)	92.1%	**0.529**	**1.021**	0.541	**0.355**	**0.488**	**0.288**	**0.407**	0.600
DGCNN-Maj. Vot.(#12)	**92.3%**	0.552	1.053	0.592	0.379	0.498	0.308	0.418	0.614

Table 10: **WolfMix Augmented DGCNN sub-samples analysis. Bold** best. Underline second best.

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
PCT-Curves (#4)	91.1%	0.934	1.234	1.320	0.391	0.551	0.322	1.393	1.330
PCT-Patches (#4)	92.5%	0.915	0.989	1.557	0.556	0.541	0.549	1.080	1.135
PCT-Random (#4)	91.9%	0.741	1.245	**0.427**	0.335	0.744	0.502	0.785	1.149
PCT-Mean (#12)	**93.4%**	**0.646**	**0.894**	0.851	**0.306**	**0.435**	**0.285**	**0.735**	**1.019**
PCT-Maj. Vot.(#12)	93.1%	0.693	0.957	0.994	0.331	0.454	0.315	0.760	1.042

Table 11: **Un-Augmented PCT sub-samples analysis. Bold** best. Underline second best.

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
PCT-Curves (#4)	90.4%	0.699	1.255	0.927	0.427	0.570	0.332	0.698	0.684
PCT-Patches (#4)	**92.7%**	0.633	**0.904**	1.339	0.391	**0.420**	0.312	0.505	0.558
PCT-Random (#4)	91.0%	0.636	1.245	**0.554**	0.375	0.700	0.342	0.567	0.670
PCT-Mean (#12)	**92.7%**	**0.510**	0.915	0.699	**0.323**	0.425	**0.268**	**0.404**	**0.535**
PCT-Maj. Vot.(#12)	92.6%	0.532	0.947	0.756	0.343	0.430	0.271	0.422	0.558

Table 12: **WolfMix Augmented PCT sub-samples analysis. Bold** best. Underline second best.

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
RPC-Curves (#4)	91.6%	1.068	1.287	1.680	0.399	0.594	0.322	1.495	1.702
RPC-Patches (#4)	92.4%	0.934	0.979	1.532	0.488	0.536	0.498	1.233	**1.274**
RPC-Random (#4)	91.5%	0.804	1.181	**0.491**	0.355	0.739	0.498	**0.855**	1.512
RPC-Mean (#12)	**93.6%**	**0.750**	**0.915**	1.057	**0.323**	**0.440**	**0.281**	0.902	1.330
RPC-Maj. Vot.(#12)	93.0%	0.791	0.957	1.168	0.343	0.473	0.319	0.913	1.367

Table 13: **Un-Augmented RPC sub-samples analysis. Bold** best. Underline second best.

Sub-samples	OA ↑	mCE ↓	Scale	Jitter	Drop-G	Drop-L	Add-G	Add-L	Rotate
RPC-Curves (#4)	91.7%	0.686	1.106	1.038	0.375	0.551	0.315	0.698	0.716
RPC-Patches (#4)	92.2%	0.603	0.957	1.190	0.399	0.430	0.298	0.415	**0.535**
RPC-Random (#4)	91.2%	0.609	1.170	**0.459**	0.359	0.662	0.342	0.542	0.730
RPC-Mean (#12)	**92.7%**	**0.501**	**0.915**	0.680	**0.315**	**0.420**	**0.251**	**0.382**	0.544
RPC-Maj. Vot.(#12)	**92.7%**	0.526	0.947	0.766	0.335	**0.420**	0.268	0.400	0.549

Table 14: **WolfMix Augmented RPC sub-samples analysis. Bold** best. Underline second best.

Algorithm 2 Classification using EPiC (Training)

for $epoch \in epochs$ **do**
  for $X, label \in TrainingDataSet$ **do**
    $anchors \leftarrow RandomlySelect([0:1023])$
    for $anchor \in anchors$ **do**
      ▷ Fetch partial point clouds
      $Patch \leftarrow FetchPatch(X, anchors(k))$
      $Curve \leftarrow FetchCurve(X, anchors(k))$
      $Random \leftarrow FetchRandom(X)$
      ▷ Apply models
      $P_{Patch} \leftarrow model_{Patches}(Patch)$
      $P_{Curve} \leftarrow model_{Curves}(Curve)$
      $P_{Random} \leftarrow model_{Random}(Random)$
      ▷ Derivate with regard to the entire point-cloud label
      $Loss_{Patch} \leftarrow backward(P_{Patch}, label)$
      $Loss_{Curve} \leftarrow backward(P_{Curve}, label)$
      $Loss_{Random} \leftarrow backward(P_{Random}, label)$
    end for
  end for
end for

Method	OA ↑	mCE
DGCNN	93.0%	0.669
DGCNN + BatchNorm	92.7%	0.527
DGCNN (W.M)	93.2%	0.590
DGCNN (W.M) + BatchNorm	92.0%	0.512

Table 15: **BatchNorm at test time.** Whereas overall accuracy is slightly degraded, OOD robustness is significantly increased, yielding lower mCE. This violates standard OOD assumptions but may be useful in some scenarios.

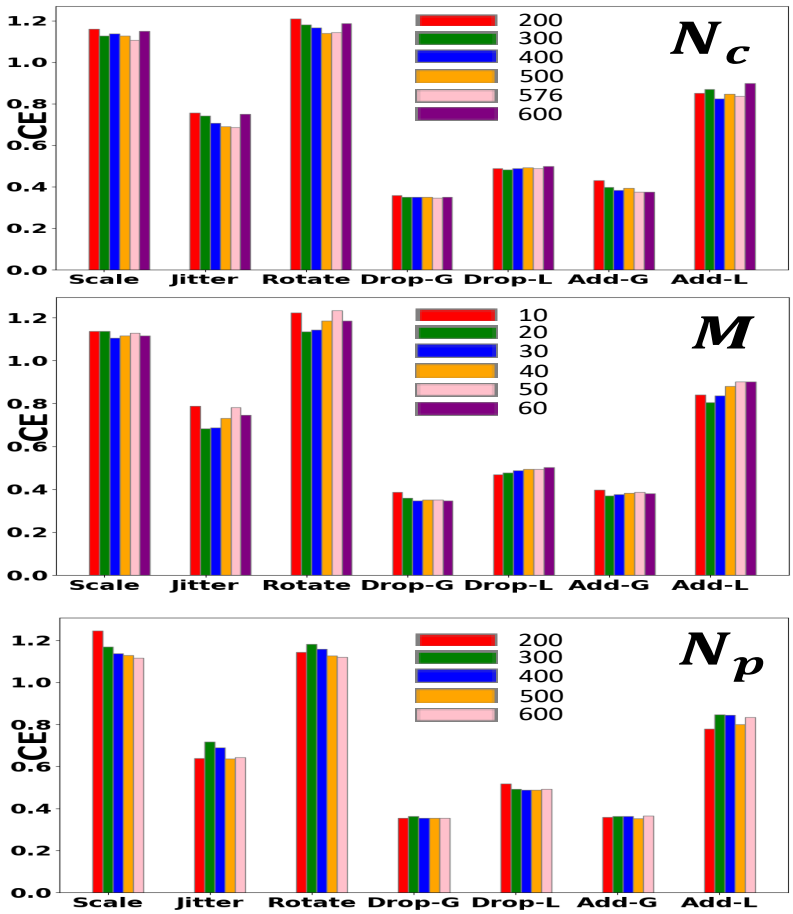

Figure 12: N_c, M **and** N_p

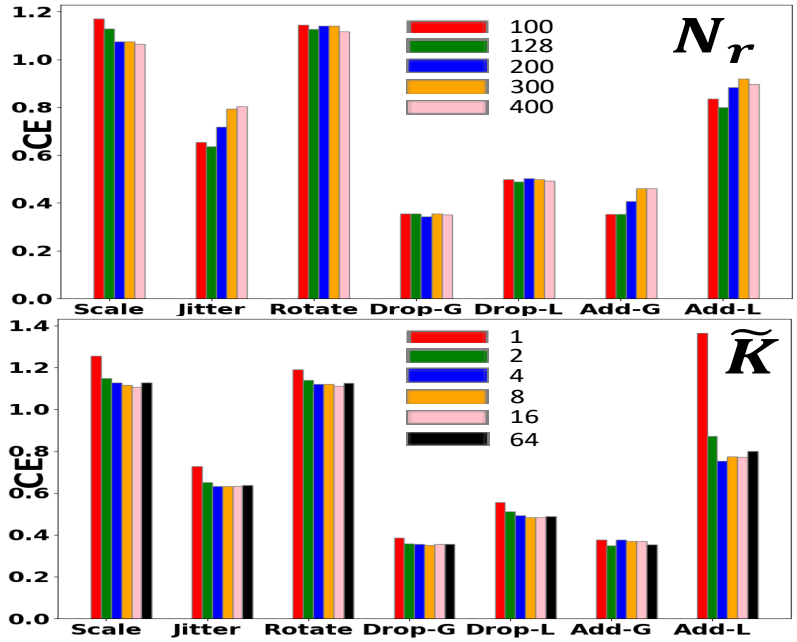

Figure 13: N_r **and** $\tilde{K}$**.**

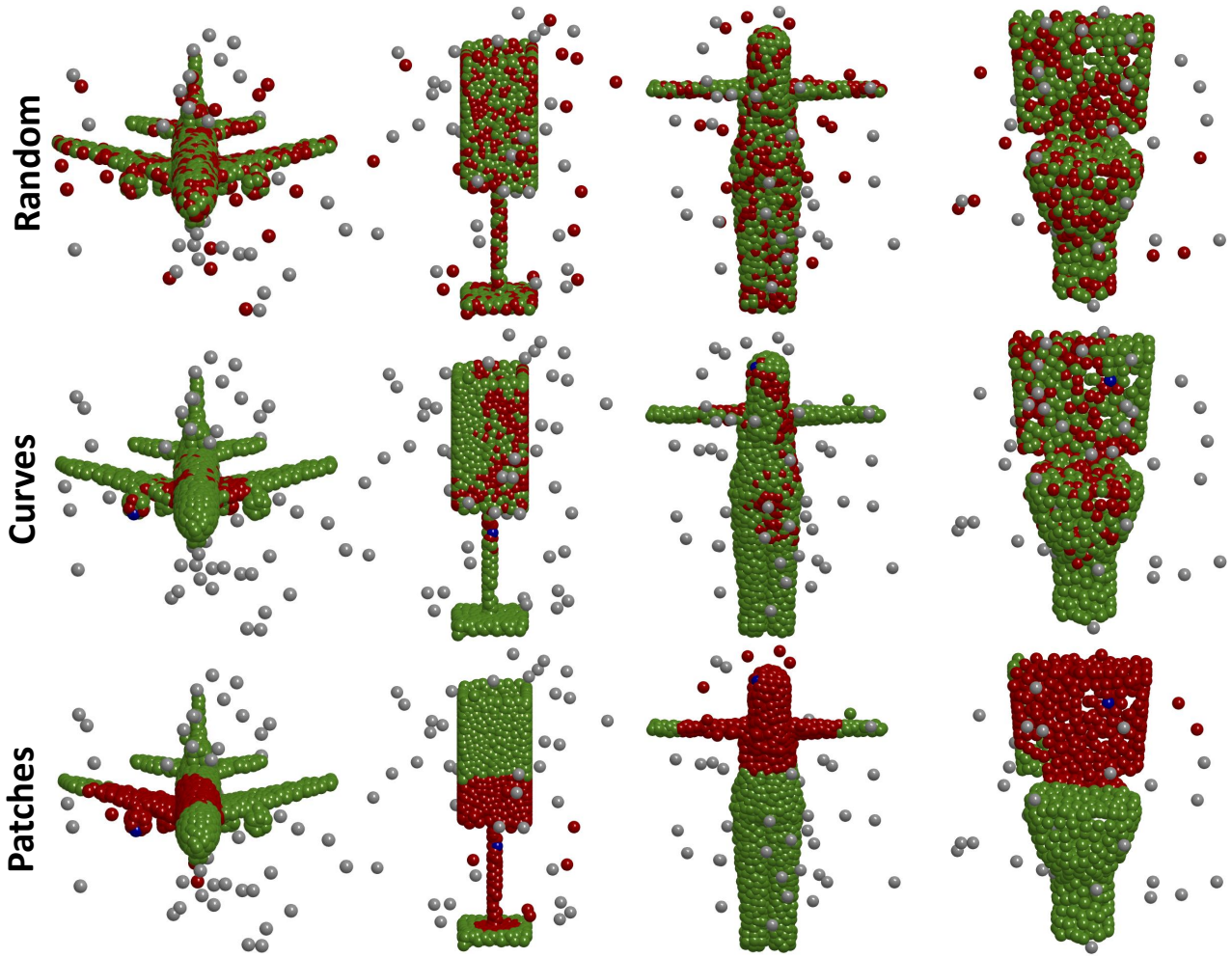

Figure 14: **Add Global**

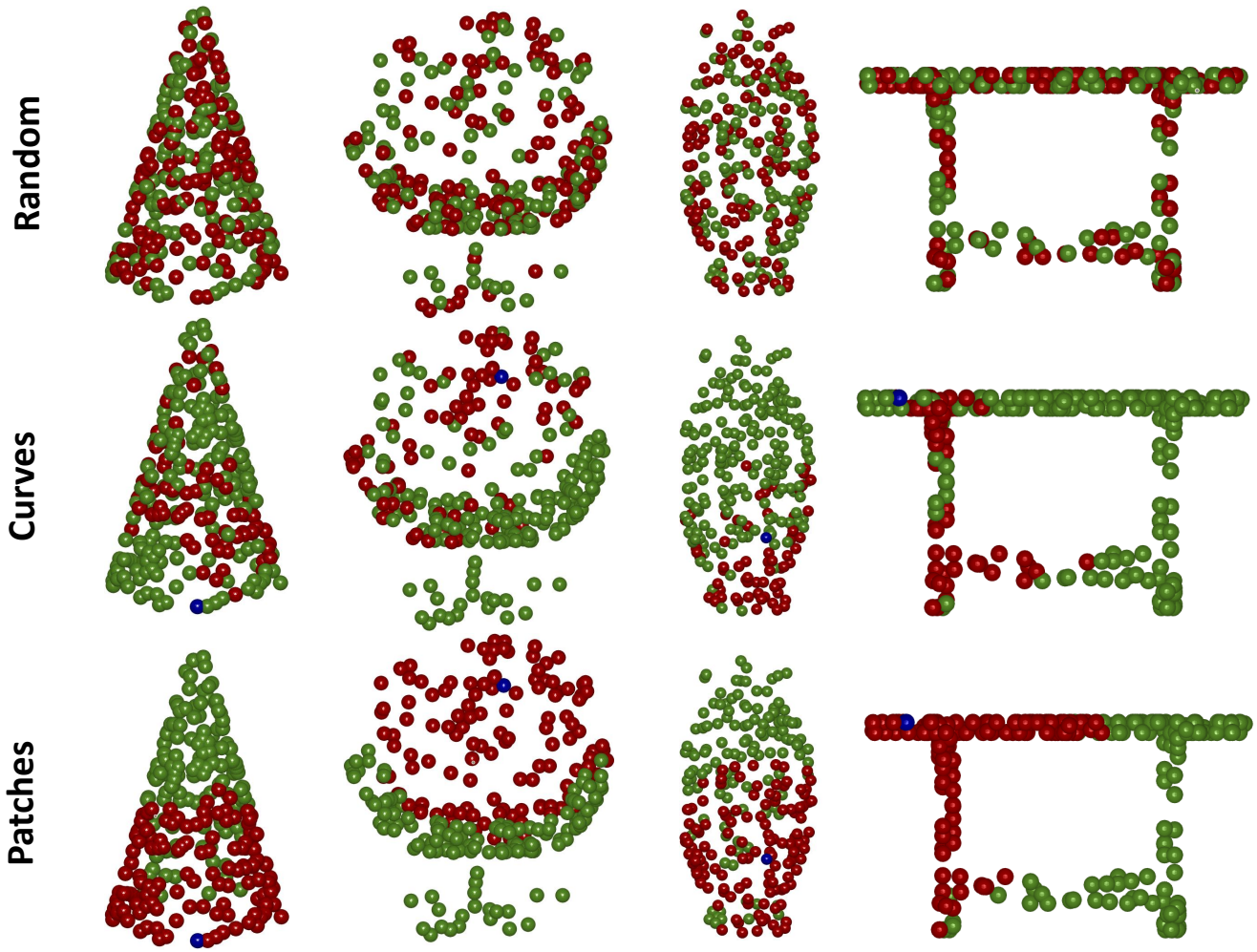

Figure 15: **Drop Global**

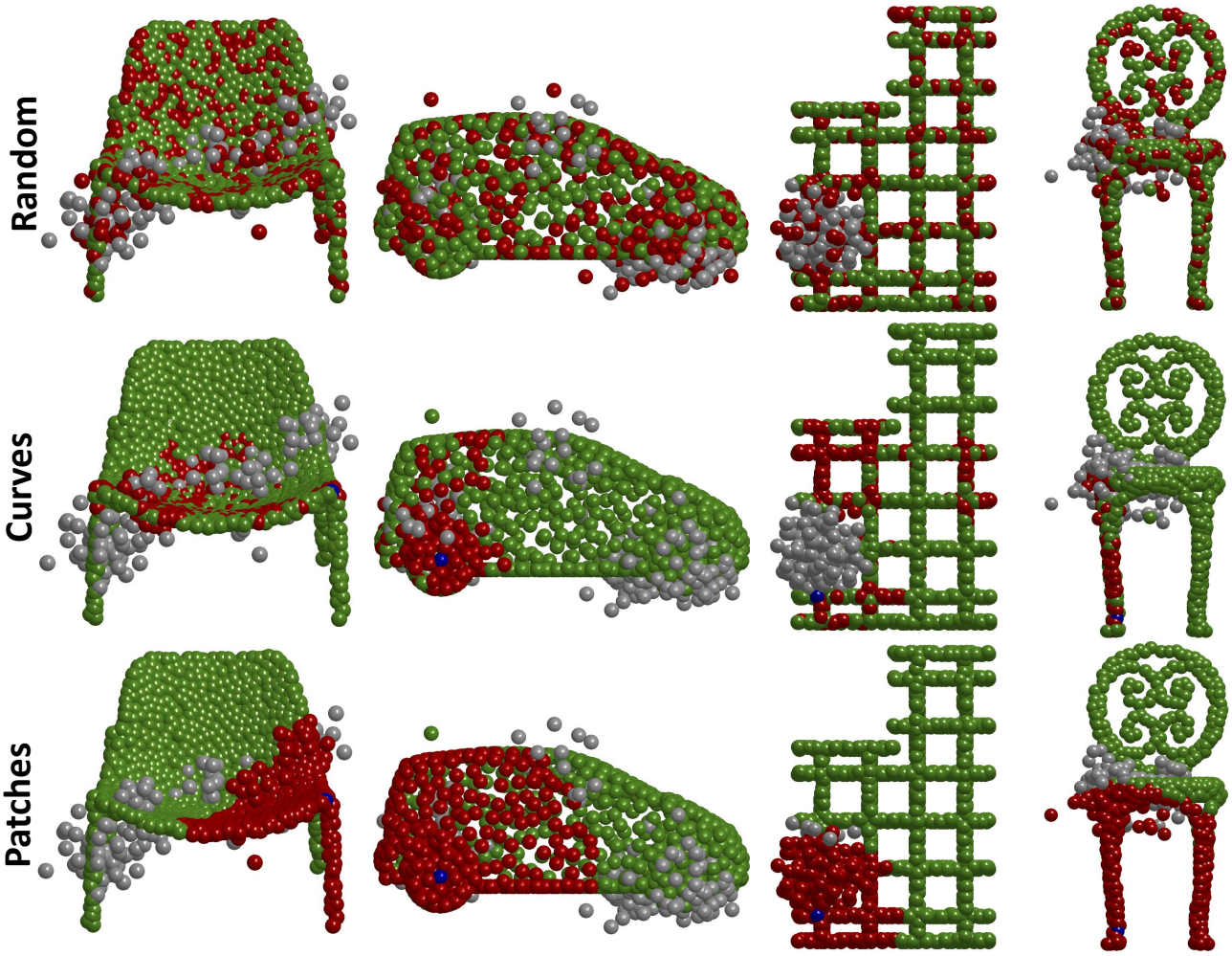

Figure 16: **Add Local.**

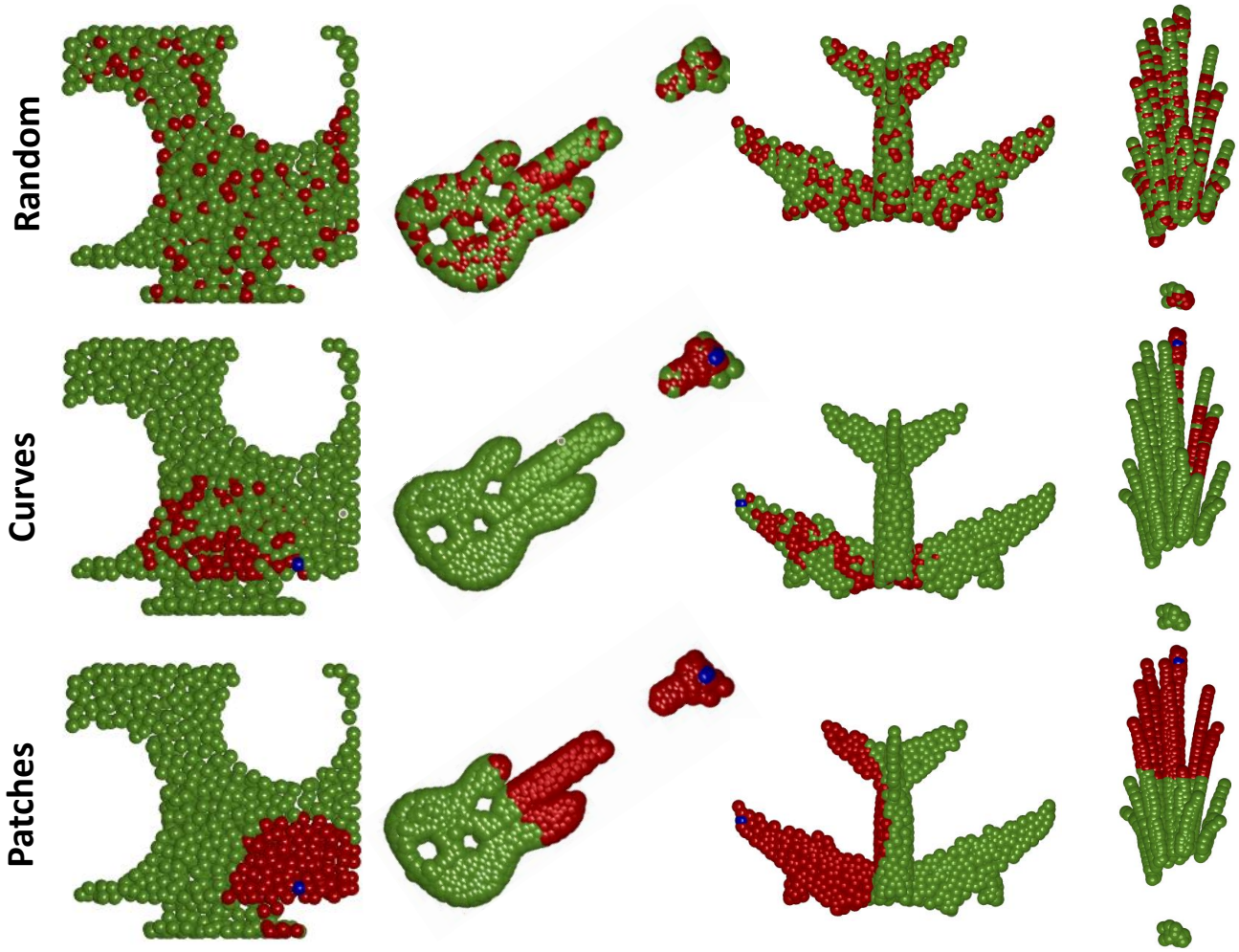

Figure 17: **Drop Local.**