S1. Supplementary

S1.1. Explanability vs. Interpretability

In this paper, we alternatively utilize both the terms "explainability" and "Interpretability" for illustration. To avoid confusion, we follow the definition of [30]: Interpretability indicates that the model is inherently interpretable (priori), while explainability implies that the model is explained with post-hoc approaches (posterio). For instance, among the two methods involving in this paper for trustworthiness, *Gini Importance* based on the impurity is a property of the model itself and is therefore known as "interpretability", while *Grouped feature ablation* belongs to "explainability", as we calculate it by posterior approaches such as feature ablation and retraining. However, for simplicity, we describe the results of the two methods jointly as "explanations".

S1.2. Hausdorff dimension

The Hausdorff dimension can be formulated as:

$$dim_H(X) := \inf\left\{d \geq 0 : \mathcal{H}^d(X) = 0\right\} \tag{S1}$$

where $\mathcal{H}^d(X)$ is *d-dimensional Hausdorff Measure* and formulated as:

$$H_\delta^d(S) = \inf\left\{\sum_{i=1}^{\infty}(diam\, U_i)^d : \bigcup_{i=1}^{\infty} U_i \supseteq S, diam\, U_i < \delta\right\} \tag{S2}$$

S1.3. Equidistance vs. Exponential window sizes

Fig. S1 illustrates the difference in coverage between equidistant and exponential window sizes. If the window size decreases equidistantly, the coverage explodes when the window size is small, which results in a large number of repetitive statistical features (the top right plot). The window shrinks progressively slower in the exponential window size, leading to a smoother increase in the coverage (the bottom left plot).

S1.4. Visualization of Gaussian features

As a supplement to Section 3.2, Figure S2 illustrates the visualization of Gaussian features.

S1.5. Hyperparameter settings

For FPF, there are several important hyperparameters that affect the classification performance: 1) the number of trees in the random forest $n\_estimators$, 2) the maximum depth of the trees max_depth, 3) the number of rotation augmentations R, 4) the type of rotation augmentation "$feature$" or "$quantity$" and 5) the feature filtering threshold TH_p.

n_estimators	20	40	60	80	100	120
OA(%)	83.7	85.6	85.4	85.5	**86.3**	85.9

Table S1. Hyperparameter setting of $n\_estimators$.

max_depth	10	20	30	40
OA(%)	79.3	86.0	**85.7**	85.6

Table S2. Hyperparameter setting of max_depth.

	1	3	5	7
OA(%)	85.4	**86.3**	86.1	85.5

Table S3. Hyperparameter setting of R.

Feat.	No RA.	Quan.	Feat.
OA(%)	85.4	84.7	**86.3**

Table S4. Hyperparameter setting of rotation augmentation types, where No RA. denotes no rotation augmentation and Quan. and Feat. denote augmentation type "quantity" and "feature", respectively.

TH_p	1e-5	3e-5	5e-5	7e-5	9e-5
OA(%)	85.3	85.8	85.4	**86.3**	85.4

Table S5. Hyperparameter setting of TH_p.

Feat.	NWD.	PWS.	Both
OA(%)	78.8	81.5	**82.5**

Table S6. The performance of other features. where NWD. and PWS. stand for *Non-zero window distribution* and *Pure window statistics*, respectively, and both indicate combining both as input features.

We exhibit the results in Table S1, S2, S3, S4 and S5, respectively. The optimal configurations are : $n\_estimator = 100$, $R = 3$, $TH_p = 7e-5$ and *rotation augmentation type* is "*feature*". Although $max\_depth = 30$ performed better than other settings, we find that configuring it manually fails to achieve the best performance. Therefore, in our experiments, we do not limit the maximum depth of the trees. Empirically, the average depth of the trees generated in the random forest is approximately 30.

Additional features: We try additional hand-crafted features and augmentations that, though slightly underperform on ModelNet40, may have potential on other datasets. 1) *Non-zero window distribution*: The distribution (mean and variance) of all the windows that contain at least one point. 2) *Pure window statistics*: Pure statistical information for all windows, including the windows containing the most and least points and their indexes, as well as the mean and variance of the number of points contained in all window. We report the results in table S6.

S1.6. Results for rotation vore

We complement the quantitative results for the conclusions made in the rotation vote of Section 4.3, which is shown in Table S7.

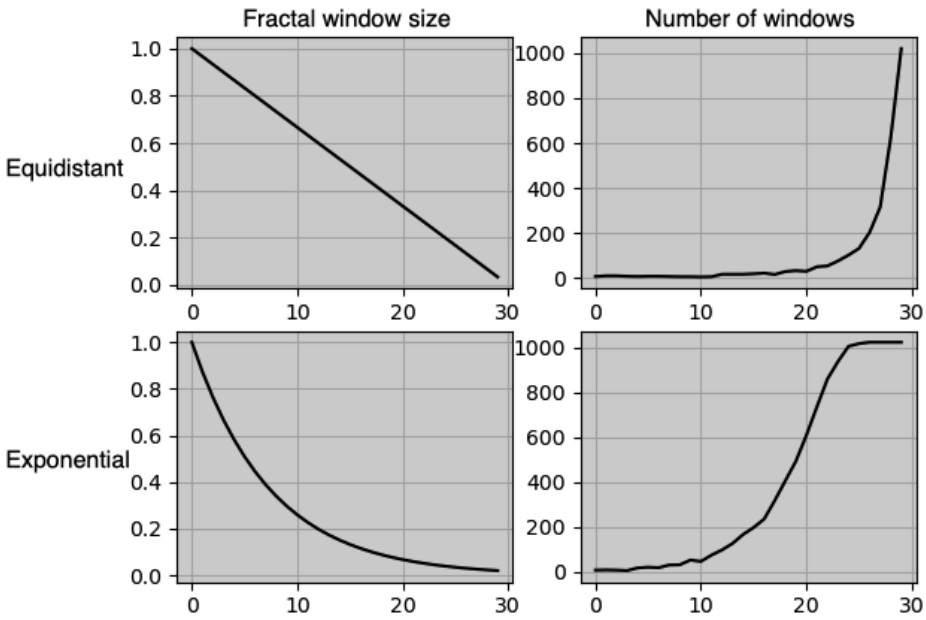

Figure S1. The correlation between window size (left) and window coverage (right), The top and bottom are the equidistant and exponential window sizes, respectively. The horizontal coordinates represent the 30 fractal windows, the vertical coordinates of the left two plots represent the window size, and the vertical coordinates of the right two plots represent the coverage of the windows (i.e., the number of windows containing at least one point).

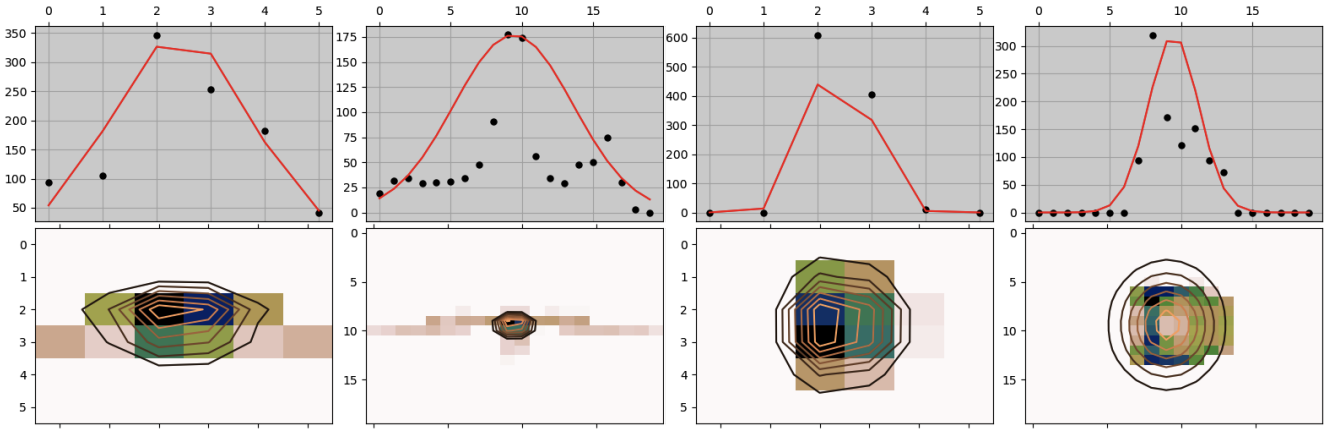

Figure S2. Examples of estimated Gaussian distribution of the projections. The upper and lower rows are the estimates of the 1D and 2D projections, respectively. The first and last two columns show examples of "airplane" and "car", respectively, and the fractal size of the first and third columns is 0.34, while that of the second and fourth columns is 0.10.

	raw	w. R.	w/o. R.
OA(%)	**85.4**	84.3	44.9

Table S7. Performance comparison of rotation vote, where raw denotes no rotation vote, w. R. and wo. R. denote rotation vote with and without rotation augmentation, respectively.

S1.7. Experiment configuration

All experiments in section 4.1 on processing time are conducted on an Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz. To conduct the speed tests, for PointNet, PointNet++ and DGCNN, we reproduce the results with third-party open source codes (https://github.com/yanx27/Pointnet_Pointnet2_pytorch and https://github.com/AnTao97/dgcnn.pytorch), which optimize the codes so that the results perform slightly better than claimed. We report our real experimental results, rather than copying directly from their papers. For PointMLP and PointHop, we utilize the official code. For the non-DL approach, we only compare the model training and inference times since the data is preprocessable. For the non-DL methods PointHop and

FPF, we only record the duration of training and inference to compare the complexity of the generated features.

We design two simple neural networks as baselines for fractual features. FC is a network consisting entirely of fully connected layers, and CNN contains both convolutional and fully connected layers. The structures of the two models are illustrated in Figure S3.

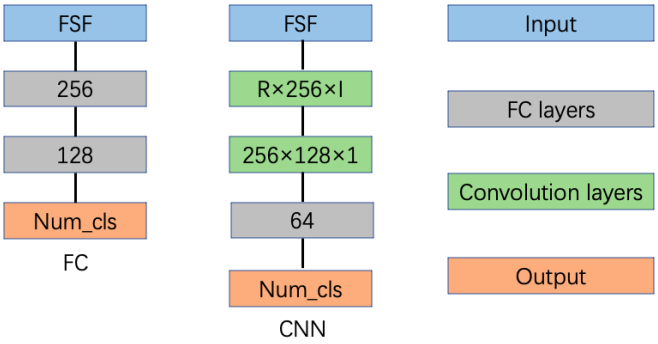

Figure S3. Structures of NN baselines. FSF denotes Fractal Statistical Features, which is extracted by proposed method. Num_cls denotes the total number of classes, for ModelNet40, $Num\_cls = 40$. R and I represent the number of rotational augmentations and the number of fractal windows, respectively. The left and middle columns show the architecture of FC and CNN, respectively. The right column is the legend.