This WACV 2023 paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Fractual Projection Forest: Fast and Explainable Point Cloud Classifier

Hanxiao Tan
AI Group, TU Dortmund
hanxiao.tan@tu-dortmund.de

Abstract

Point clouds are playing an increasingly important roll in autonomous driving and robotics. Although current point cloud classification models have achieved satisfactory accuracies, most of them trade slight performance gains by stacking complex modules on the grouping-local-global framework, which leads to prolonged processing time and deteriorating interpretability. In this work, we propose a new pipeline named Fractual Projection Forest that exploits fractal features to enable traditional machine learning models to achieve competitive performance with DNNs on classification tasks. Though compromises by few percentages in accuracy, FPF is faster, more interpretable, and easily extendable. We hope that FPF may provide the community with a novel view of point cloud classification. Our code is available on `https://github.com/Explain3D/FracProjForest`.

1. Introduction

In recent years, point clouds have attracted great attention due to their potential in fields such as autonomous driving [3, 18] and robotics [27, 14]. In Deep Learning (DL), due to irregularity, traditional convolutional kernels cannot be applied directly on point clouds and early solutions mainly include voxelization [23] and multi-view approaches [35]. With the proposal of PointNet [28], 3D recognition has started a new period, i.e., learning features and predicting directly with raw point clouds. Upon this foundation, a large number of deep architectures have been proposed [29, 40, 17, 22, 43], which have achieved higher accuracies.

However, most of the latest point cloud models are unable to get rid of the general term [22], i.e.:

$$g_i = \mathcal{A}(\Phi(f_{i,j})|j = 1, \cdots, K) \tag{1}$$

where $\mathcal{A}$, Φ and f are global symmetric function (various pooling layers), local feature extractor (MLP [28, 29], CNN [17], Residual [22], attention [43] etc.) and local grouping function (MSG [29], Graphs [40], etc.) respectively. Recent models frequently incorporate more complex tricks to these components in exchange for minor accuracy gains. Although the "arms races" have won slight boosts in accuracy, the side effects cannot be ignored. First, excessive module burden leads to prolonged runtime. This can be derived from the quantitative comparison of PointMLP [22], even the latest and cleanest MLP architecture has double the processing time compared to PointNet. More importantly, these models with sophisticated modules suffer from a lack of interpretability [9]. Humans cannot trust models that are not interpretable, especially in the field of autonomous driving or medicines, where the predicted results are crucial for human lives [10]. Although several recent researches have addressed the explainability of point cloud models by demonstrating the decision basis through post-hoc methods [7, 37], explainability is still in its infancy as an emerging field and numerous existing studies have indicated that the currently widely utilized explainability methods may not be reliable [38, 15, 11]. Besides, the lack of ground truth makes post-hoc explainability methods difficult to be evaluated [47].

In addition to post hoc approaches, the employment of interpretable models also facilitate plausibility [4]. One study suggests that if appropriate features are chosen, even with simpler and interpretable models, there will be no significant drop in accuracy compared to black-box ones [30]. Fortunately, recent post-hoc attribution researches reveal that the features of point clouds are sparse [7, 37]. Comparable accuracy can be achieved using simple interpretable models if these features are filtered out from point cloud input in advance.

In this work, we propose a new pipeline of point cloud classification by extracting the input projections through multi-size fractal windows. This approach jumps out from the race of adding tricks to Deep Neural Network (DNN)s, and performs classification through simpler models with only few percent accuracy compromised. Moreover, our method enables the training of the entire ModelNet40 in tens of seconds (several minutes if pre-processing is included) with a CPU, which is barely possible for DNNs. Besides, our model provides two explanations, i.e., intrinsic

Gini impurity and perturbation-based attributions. For the latter, our model allows the ablation of entire grouped features, thus avoiding the concern of feature correlation and out-of-distribution issues [11] as in raw point clouds. Additionally, our approach is more extensible as it can adapt to different classification scenarios by manually adding appropriate features or switching models. In summary, our contribution is primarily summarized as follows:

- We propose a non-DL pipeline Fractual Projection Forest (FPF), that converts point cloud classification, which is previously only solvable with DNNs, into traditional machine learning tasks. Compared with DNNs, FPF is faster, more interpretable and extensible, while compromising only few percent of accuracy.
- We demonstrate two explanations of FPF, both of which are challenging to accomplish on DNNs.

The overall structure of this paper is as follows: In Section 2, we introduce popular point clouds models and corresponding explainability methods. Section 3 elaborates the ideas and technical details of FPF. In Section 4, we show the performance of the proposed method and the corresponding explanations. In Section 5, we give a short summary and propose future research directions.

2. Related Work

In this section, we introduce the widely applied classification models (2.1)and the research that promotes the explainability for point clouds (2.2).

2.1. Classification models for point clouds

Before the advent of models that act directly on raw point clouds, there were typically two ways for the classification tasks, namely the voxel-based [23, 31, 19] and the multi-view-based approaches [35, 6, 13, 5]. The voxel-based approaches are dedicated to organizing irregular point clouds so that they are spatially ordered, which enables the extraction of adjacent features between points using 3D convolutional kernels. The voxel-based methods, as early solutions to address the irregularity, suffer from limited processing speed due to the incorporation of additional pre-processings, such as voxelization [23]. The multi-view-based methods project point clouds onto planes, subtly downscaling them to two-dimensional images. Interestingly, in recent research [6], a simple 6-view-method achieves almost state-of-the-art accuracy by introducing tricks into the training process. With the proposal of PointNet [28], a series of classification methods that operate directly on raw point clouds come into view. PointNet establishes a pipeline for following studies, i.e., Eq. 1. Most subsequent models based on raw point clouds [29, 40, 17, 43] are subject to this pipeline, with more complex tricks attached to the individual modules to achieve higher accuracies. However, all the aboves are based on neural networks, which are black-box models, and humans struggle to understand their decision-making principles [24].

2.2. Explainability research for point clouds

Post-hoc explanability methods: There are currently only few explainability studies on point cloud models. [7] and [37] explain the decision attributions by incorporating gradient-based and surrogate model-based explainability methods [24] to point clouds respectively. For the research on model intrinsic attribution, [46] observes changes in the prediction confidence by filtering and incrementally flipping the key points. However, all the aforementioned methodologies are post hoc attributions, and the explanations obtained may be biased due to flaws in explainability or perturbation approaches [38, 15, 11].

Interpretable models: Before DL was widely applied, point cloud classification was usually accomplished by manually extracting features and trivial ML models, e.g. LDA [45] or Markov network [25]. Nevertheless, these approaches were promptly replaced by newly emerged point cloud DNNs due to the limited performance. To address interpretability, a recent study [1] proposes a prototype-based model that accomplishes classification by clustering the features in the latent space. Although the decision basis of their approach is intuitive, it is less extensible, and we argue that the extraction of latent features with neural networks aggravates the opacity. In addition, PointHop [44] is also a non-DL method that utilizes a random forest to learn adjacent features of points extracted by a module called PointHop Unit. However, although PointHop improves the interpretability of the model, the geometric features between points are still incomprehensible for humans. In addition, PointHop performs significantly degraded on real scanned datasets.

3. Methods

In this section, we introduce the mechanism and structure of FPF. Section 3.1 outlines how fractal-based features operate, Section 3.2 details the internal structure of FPF, and Section 3.3 illustrates how FPF generates reliable explanations.

3.1. Fractal features

This work is inspired by *Hausdorff dimension* [8], whose formula can be seen in eq. S1. Hausdorff dimension reflects the smoothness of geometry and can be estimated by counting the total number of fractals in different sizes and applying an exponential fit. Similarly, we create features of point clouds manually with multi-size fractal windows. Fig. 1 illustrates an overview of fractal features. With progressively increased window sizes, the point set is sampled into different subgroups. We then extract relevant statistical information from each subgroup, which varies according the distribution of points within the subgroups. For instance, from the two point sets (first column), we can extract the feature sequences $[4, 2, 2]$, $[1, 2, 2]$ (top) and $[4, 4, 4]$, $[1, 1, 1]$ (bottom) with respect to n and $\bar{p}$, respectively. Note that we only list the total number of windows (n) and the average number of points contained ($\bar{p}$) here, additional information will be discussed in Sec. 3.2.

3.2. Model architecture

Fig. 2 shows the architecture of FPF. FPF consists of the following components: projection, fractal sampling, feature generation & concatenation and prediction modules. Consider the input as a D dimensional point cloud instance with N points: $\{x_i | i = 1, \dots, N\} \in \mathbb{R}^{N \times D}$, and we illustrate with the simplest case, i.e., $N = 3$.

Projection: Inspired by the multi-view classification methods [35, 42, 6], we first project the point cloud on each axis (x,y and z) and planes (xy, xz and yz). Subsequently, three 1-D and 2-D projections are obtained, they are denoted as: $P_1 \in \mathbb{R}^N$ and $P_2 \in \mathbb{R}^{N^2}$.

Fractal sampling: According to the mechanism in 3.1, we perform fractal sampling for each of the six projections. Let the number of multi-scale sampling be I. The start of the sampling window size is half of the entire spatial value range, i.e. $W_{max} = \frac{1}{2}\{\max x_i - \min x_i\}$, which is scalable and 1 for ModelNet40 [41]. The window size at the i-th sampling is:

$$W_i = e^{(-i \times \alpha)} \tag{2}$$

where α is a smoothing coefficient that flattens the changes in the sampling window sizes. We chose exponential because the points sampled by the equidistant window size are prone to alter drastically when i is small and almost constant when i is large (see Sec. S1.3 for analysis). The return of the sampling is which window each point belongs to, and two sampling results are obtained, $S_1 \in \mathbb{N}^N$ and $S_2 \in \mathbb{N}^{N^2}$ for P_1 and P_2, respectively. Note that the records of those windows that do not sample any points are marked as 0, which are also important features.

Feature generation & concatenation: Compared with raw point clouds, the advantage of tabularized features is that they are more intuitive when explaining attributions. Therefore, we extract statistics from point clouds in the form of tabular features as training data. Similar to other DNN models, FPF requires global and local features for prediction as well.

Global features: Three global features are involved: the number of non-zero fractal windows: $G_n = |\{s \in S_1, S_2 | s > 0\}|$, basic statistics for all sampling windows, and the Gaussian parameters. The base statistics include the maximum and minimum (non-zero) values of points within a single window, and the corresponding window indexes. To obtain the parameters, we employ Gaussian fits to approximate the distribution of the sampled points in corresponding projection space. This feature is based on the assumption that most objects are normalized to be located in the middle of the spatial coordinates. We estimate the parameters using one- and two-dimensional Gaussian models, respectively. The former contains two parameters, i.e., the mean (μ) and the variation (σ), and the latter consists of five, i.e., the means along each axis μ_x and μ_y, the standard deviations σ_x and σ_y, and an amplitude bias α. Fig. S2 shows an example of a Gaussian feature that captures the distribution differences across classes and windows sizes.

Local features: Two local features are involved: (averaged) distributions of the points within all windows as well as several specified windows. For the former, we calculate the extremes, means and variances of the points within each window separately and obtain their global averages. For the latter, we selectively monitor the windows of particular indexes and compute the internal distributions of points. However, since the total number of windows obtained after sampling with different sizes is inconsistent, we specify a proportion $m \in [0, 1)$ and select $|S| \times m$ ($|S|$ is total number of windows) as the monitored window. We record the number, mean and variance of the points in this window as features. In addition, multiple monitored targets can be appended.

Feature concatenation: We simply concatenate global and local features as final inputs. Note that our features are arbitrarily expandable, and different statistical features can be explored for specific datasets to achieve optimal accuracu.

Others: FPF incorporates other modules to further enhance performance.

Rotation augmentation: Rotating objects R times to create R new training data is an augmentation method proposed in [28, 29]. In this work, we set the angle of each rotation to $2\pi/R$, i.e., the object is rotated by an identical angle R times, and the $R + 1st$ time returns to the original one. There are two types of rotation augmentation available in FPF: *feature* and *quantity* augmentation. We first perform the same tabular feature extraction on the rotated object to obtain R additional feature series. In feature augmentation, we horizontally concatenate R sequences ($R+1$ times lengthened), which provides each training data with additional rotation information and enhances rotation invariance. The quantity augmentation is similar as in [28, 29]. The rotated features are vertically aggregated to the dataset as new training data, which increases the amount of training data to mitigate overfitting.

Attribution filtering: In decision making, there are features that contain either positive or negative attributions. Positive attributions reinforce the predictions while negative ones diminish the confidence. Therefore, after training a raw model, we obtain attributions of each feature based on the explanation (see Sec. 3.3) and filter out those features whose attributions are below a threshold TH_p to further enhance the accuracy.

3.3. Explainability

FPF offers two perturbation-based explanations: Gini Importance and grouped feature ablation.

Gini Importance is calculated as the average of the decreasing impurity of all nodes over all trees [21]. The advantage of this approach is the calling simplicity, which can be obtained directly through $RF.feature\_importances\_$ from *sklearn* in few milliseconds. One drawback of impurity attribution is the bias towards continuous or high cardinality variables [26]. Another concern is the bias in datasets containing highly correlated features, which leads to suboptimal predictor variables being assigned greater weights [33, 32].

Grouped feature ablation is an alternative method that addresses the above issues. Traditional feature ablation approaches suffer from out-of-distribution problem [11], i.e., when certain features are ablated, the data consisting of the remaining ones are outside the distribution from original dataset, resulting in unreliable prediction analyses. Therefore, we first group features according to types, i.e. features belonging to 1-D projection or sampling window size 1. Subsequently, we ablate all features belonging to the same type t to avoid any information residual. According to the methodology proposed by [11], we retrain the ablated dataset and record the accuracy of the entire testset as $Acc_{\bar{t}}'$. The attribution of features in type t can be represented as:

$$Atr_t \propto Acc - Acc_{\bar{t}}' \qquad (3)$$

where Acc is the accuracy of the original model on the unablated testset. Grouped feature ablation alleviates the bias in the explanations, while prolongs the processing time as the model has to be retrained as many times as the number of feature groups.

4. Experiments

In this section, we report the quantitative results of FPF through extensive experiments. Section 4.1 presents the performance of FPF on multiple datasets, and Section 4.2 provides visual explanations for the feature importances. In our experiments, we choose ModelNet40 [41] as the main dataset, which contains 9,843 and 2,468 CAD models belonging to 40 classes in the training and test sets, respectively. In addition, we test our approach on ScanObjectNN [39] and ShapeNet [2]. The former is a real-world dataset containing 15,000 objects in 15 categories, and the latter is a dataset containing 35,708 and 10,261 training and testing instances, respectively. Specifically, ShapeNet is originally proposed for 3D reconstruction and is extended to classification tasks in our experiments. All calculations are performed on an Intel(R) Core(TM) i7-4650U CPU @ 1.70GHz CPU except for the training of DNNs for the processing time comparison. For FPF, we set the smoothing coefficient α to 0.135, the number of fractal windows I to 30, and the number of rotational enhancements R to 3, which empirically yields the best efficiency-performance trade-off.

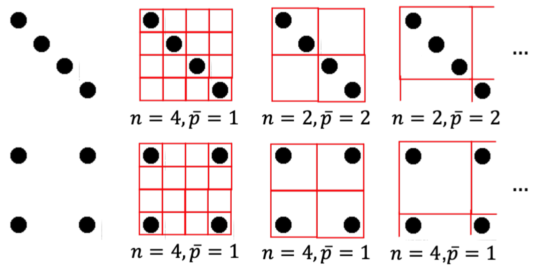

Figure 1. A simple illustration of fractal feature extraction. Sampling the input using fractal windows of different sizes yields sequences of statistics, which vary depending on the distribution of input points.

4.1. Quantitative Results

Results on ModelNet40. We compare FPF with two NN baselines (FC and CNN), four DL-based models (PointNet [28], PointNet++ [29], DGCNN [40] and PointMLP [22]) and one non-DL model. The selected DNN-based models are representative, where PointNet can be considered as a baseline for deep learning (the components in Equ. 1 are max-pooling, MLP and no point-wise correlation, respectively). PointNet++ replaces the last term with Multi-scale grouping (MSG), while DGCNN upgrades MLP to dynamic graph networks. PointMLP is the state-of-the-art model for point cloud classification. As reported in Table 1, our non-deep learning model FPF approximates DNNs in terms of accuracy (4.1% and 7.5% lower than PointNet and PointMLP respectively), with a significantly reduced processing time. However, FPF predicts the entire test set (2,468 instances) in less than 1 second with CPU, and the training time is approaching 1 minute. In addition, the raw FPF is enhanced in accuracy and speed with attribution filtering (empirically setting the threshold to $TH_p = 7e-5$), demonstrating the necessity of filtering out negative attribution features, which is unachievable in neural networks. Compared with the non-DL model PointHop, though FPF is slightly inferior in accuracy, its prediction is faster and the model size is almost 40 times shrunk.

Additionally, we attempt to train simple neural networks to learn the fractal features. We train a simple FC network and a CNN with the features extracted from the fractal win-

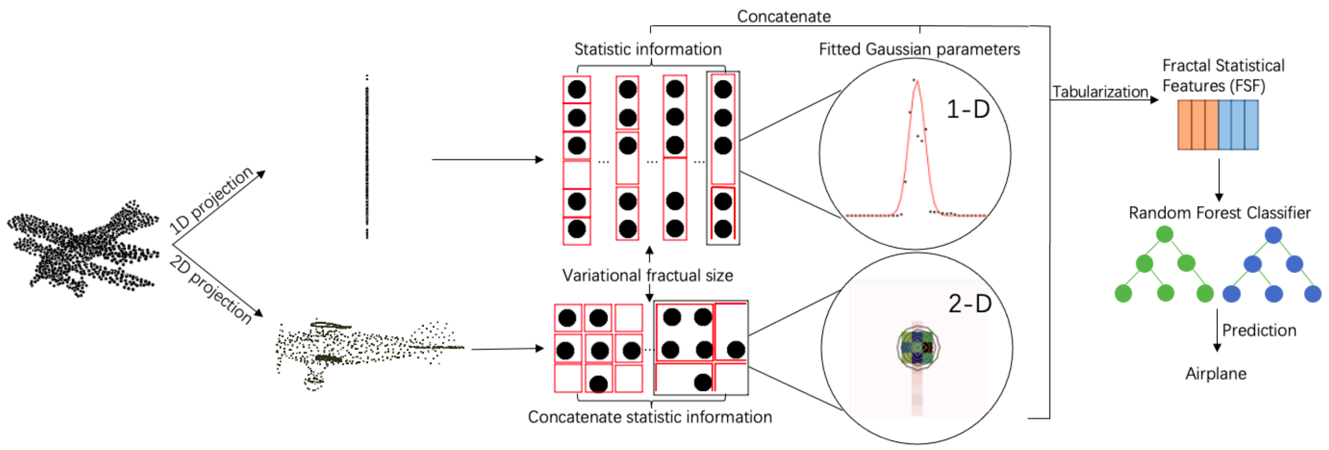

Figure 2. An overview of FPF architecture. Given a point cloud object, FPF first projects it onto the three axes and planes separately. For each projection, we record fractal feature extracted by the multi-size fractal windows. Under the assumption that the points in the fractal window overall subject to Gaussian distribution, we estimate the Gaussian parameters and concatenate the statistical information of other fractal windows together as features to train a random forest.

dows as NN baselines (the structure can be found in Section A). Interestingly, the performance of NNs is inferior to random forest. Interestingly, NNs perform inferior to random forests, but they are thousands of times faster compared to other DL methods. One possibility is that better network structures may exist that outperform random forests, but this would not only compromise speed, but also sacrifice interpretability.

Results on ScanObjectNN. We select the hardest variant (PB T50 RS) of ScannObjectNN as the training and test set, in which most of the objects are incomplete and retain only surface information. Table 2 reveals the quantitative results on ScanObjectNN. Note that FPF is potentially extensible and can concatenate more appropriate features for different data to achieve better performances. Here we follow the hand-crafted features in ModelNet40, which may not be optimal for ScanObjectNN, nevertheless, the accuracy of FPF outperforms PointNet, which is considered as the baseline for DNNs. We leave the methodology of crafting the most suitable features for different data as future work. Moreover, PointHop underperforms on this dataset (14% lower). We believe this is because the local region features extracted by the PointHop Unit are only suitable for hand-crafted datasets, and not for difficult ones like ScanobjectNN (most samples are available with surface information only). This restriction does not exist for FPF, since the hypothesis of FPF presupposes only that the distribution of the points from the train and test sets are similar.

Results on ShapeNet. Since ShapeNet is not a benchmark for classification, we only observe whether FPF suffers from accuracy collapse on different datasets. We report the accuracy of raw FPF as 79.1, and the accuracy of FPF after (empirically) filtering with $TH_p = 8e - 5$ as 79.3.

4.2. Explanations for Feature Importance

In addition to rapidity, a more important advantage of FPF is the interpretability. Aside from replacing DNNs with traditional models like PointHop [44] to enhance interpretability, we provide two explanations according to the properties of FPF, i.e., *Gini Importance* and *Grouped feature ablation*, which are shown in Fig. 3 and Fig. 4, respectively. The two explanations agree on the attribution of rotations, with one divergence in the attribution of features and a distributional discrepancy in the attribution of window sizes. The discrepancy stems mainly from the inherent deficiencies of existing explainability methods. Feature importance based on Gini impurity tends to assign more attributions to features with large cardinality [26], such as the fractal window sizes. In contrast, the importance of features with smaller cardinalities, such as the point statistics in windows, are prone to be underestimated (see Fig. 3, in "In_db" of the third subplots). The advantages of this explainability method are the simplicity and rapidity of invocation, which can be called directly in *sklearn*, and consumes an average computation time of approximately 0.03 seconds.

The post-hoc explanation based on grouped feature ablation is more plausible. The feature groups are independent of each other and the dataset is retrained based on ROAR [11] after each ablation round to prevent the out-of-distribution issue. However, ROAR is also controversial. The feature importance of ROAR depends on the magnitude of the decline in accuracies of the retrained models, while several studies questioned whether the explanations should be faithful to the original model or to the data [34, 36, 12]. One of the difficulties of explainability methods is the lack of ground truth, and exploring more accurate and plausible explanations is the potential research direction.

	Models	OA(%)	mAcc(%)	F1	T_{tr}	T_{te}	T_{avg}	Size
DL	Baseline FC	74.2	65.5	64.0	/	0.21	8.50×10^{-5}	6.8 MB
	Baseline CNN	81.4	75.0	74.4	/	4.22	1.71×10^{-3}	**4.2 MB**
	PointNet [28]	90.4	85.5	85.6	/	3068.07	1.22	39.8 MB
	PointNet++ [29]	92.5	89.8	89.5	/	3004.10	1.20	20.1 MB
	DGCNN [40]	92.7	89.6	89.7	/	968.99	0.39	6.9 MB
	PointMLP w/o vot. [22]	**93.8**	**90.2**	**89.9**	/	2039.05	0.83	101.3 MB
Non-DL	PointHop [44]	87.3	81.5	78.9	301.03	6.91	2.8×10^{-3}	3.16 GB
	FPF (raw)	85.4	79.0	79.7	68.88	0.31	1.25×10^{-4}	77.5 MB
	FPF (fltd)	86.3	80.4	81.1	**68.36**	**0.29**	$\mathbf{1.16 \times 10^{-4}}$	76.1 MB

Table 1. Classification results on ModelNet40. T_{tr}, T_{te}, T_{avg} and Size are the processing time for training the whole train set, validating the whole test set, the average time for predicting a single instance and the model size, respectively. FPF (fltd) and FPF (raw) denote FPF with/without attribution filtering, respectively. See section S1.7 a for detailed experimental configurations.

Model	DL				Non-DL		
	PN	PN++	DGCNN	PointMLP	PointHop	FPF(raw)	FPF(fltd)
OA(%)	68.0	77.9	78.1	**85.4**	54.2	68.2	68.8

Table 2. Comparison of the overall accuracy on the PB_T50_RS of ScanObjectNN (the hardest task). PN and PN++ denote PointNet and PointNet++, respectively.

4.3. Others

In this section we present additional results of FPF, including the feasibility of fractal features, ablation studies and salinity checks for the explanations.

Feasibility of fractal features. To further confirm the effectiveness of fractal features, we train a simple decision tree ($max\_depth = 20$) with multiple fractal features and observe whether the accuracies outperform the baselines. We consider two baselines, uniform and weighted random guesses. The former assumes that each label has the same probability of being guessed, while the latter is weighted according to the amount of data in each class. As shown in Table 3, each fractal feature significantly outperforms the random guess baseline.

Rotation vote. Rotation vote is a technique to further improve accuracy by rotating and predicting an object multiple times and then performing a majority voting, which exhibited superior results in several point cloud models [20, 29]. For FPF, we consider two candidates: rotation votes with/without rotation augmentation (see the last subsection of 3.2), which represent whether rotating information of the object is incorporated in the training, respectively. Note that since the selection of rotation features is only possible before the attribution filtering, we only compare the performance with *raw* model. Surprisingly, rotation vote yields no performance boost for FPF and the accuracy even collapses if rotation augmentation is not employed (see Table S7). We believe the reason is that the fractal features dramatically change with rotations, evidenced by the degradation of performance after ablation of the rotation augmentation in Table 4. The solution is to learn more fractal features at different rotation angles, which however leads to more time consuming.

Ablation study. We decompose and ablate each module of FPF in turn, and record the corresponding accuracies. For ablating multiple fractal series, we calculate the results for fractal windows with different sizes individually and take the average. For the rotation enhancement, we simply set $R = 1$. For the remaining fractal features, we remove them sequentially from the aggregated features. We train a new random forest model after each ablation in order to avoid the out-of-distribution issue. The results are demonstrated in Table 4. The absence of each module results in a degradation of accuracy, while the drop is more significant when employing single size of fractal windows.

Sanity checks. Due to the lack of ground truth, there are few metrics available to assess the plausibility of explanations. Among them, salinity check [38] is an important indicator. The fundamental idea is that the generated explanation should be relevant to the model, and the collapse of the explanation is supposed to be observed as the model is randomized.

Due to the utilization of random forest, we can hardly modify arbitrary layers of the model as in [38]. Instead, we randomize certain percentages (from 10% to 100%) of decision trees that are randomly selected from the forest. However, it is challenging to edit the weights of decision trees directly, we therefore randomize the chosen trees by retraining them with random labels. For evaluating the similarity of explanations, we follow [38] using *Krippendorff's* α [16], which is formulated as:

$$\alpha = 1 - \frac{D_o}{D_e} \tag{4}$$

where D_o and D_e are the disagreements observed and ex-

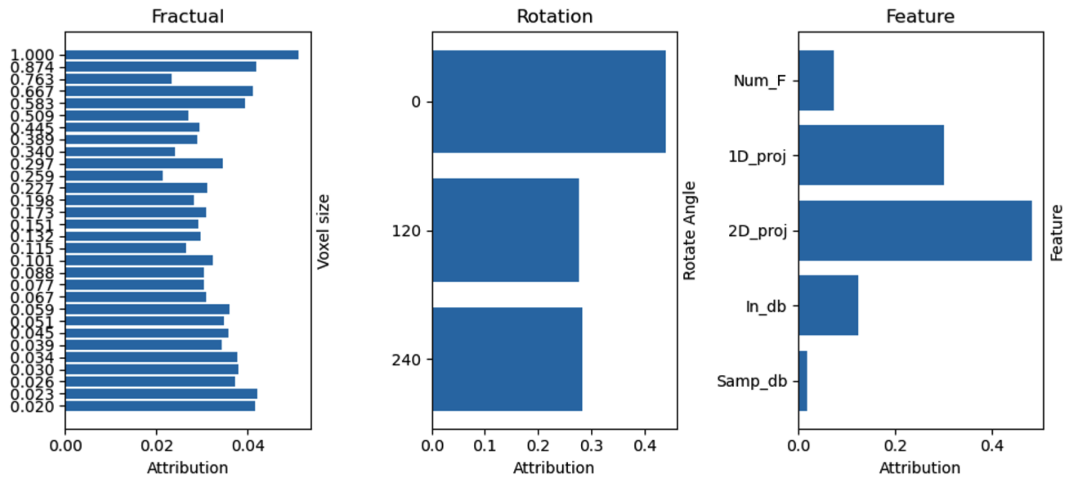

Figure 3. Intrinsic feature importance explanations based on Gini impurity. Fractal, Rotation and Feature represent the feature attributions of each fractal sizes, rotation degrees and hand-crafted features, respectively. Among Features, Num_F, $1D\_proj$, $2D\_proj$, In_db and $Samp\_db$ indicate the number of fractions, statistics and estimated Gaussian parameters for 1D and 2D projections, statistics over all fractions and inside specifically sampled fractions, respectively.

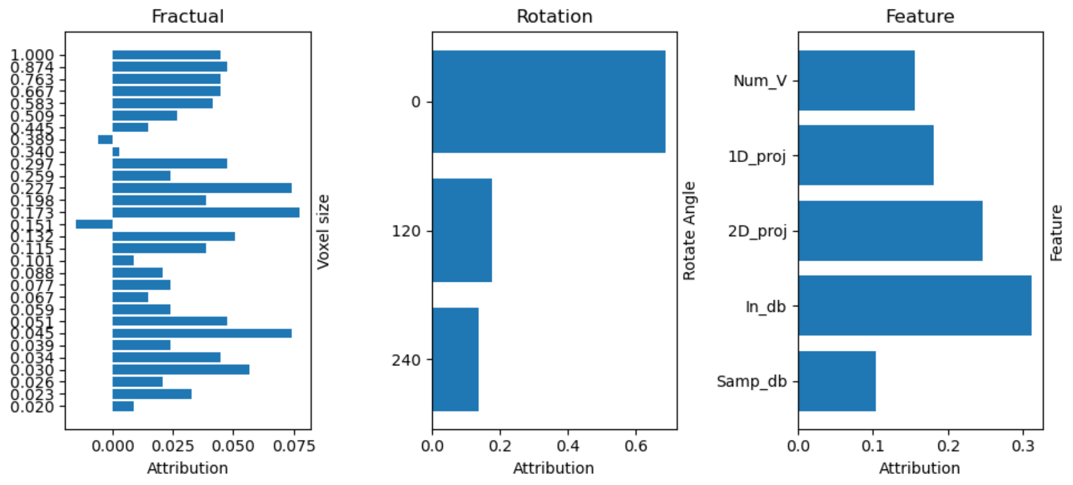

Figure 4. Explanation of Grouped feature ablation.

pected by chance respectively. If the Krippendorff's α approaches 1, the two explanations are highly analogous, while if it approaches 0, they are almost independent (Negative values indicate an inverse proximity). As the results in Fig. 5 demonstrate, the grouped feature ablation passed the salinity check, where the more trees are randomized, the closer Krippendorff's α converges to zero. However, the intrinsic feature importance is shown to be flawed. Though the α-value of the fractal window feature is rapidly corrupted, there is minimal decrease with respect to the rotation feature, and almost no changes in the hand-crafted feature. This can be attributed to that the interpretation based on Gini impurity suffers from neglect of low cardinality features, while the correlation between features also raises problem due to the absence of retraining.

5. Conclusion

In this work, we propose a non-deep learning point cloud classification pipeline FPF. By extracting statistical features from fractal windows, FPF enables traditional machine learning models to achieve comparable performance to deep learning. Compared to DNNs, traditional models are not only faster but, more importantly, possess better interpretability. We hope that our ideas will inspire more explainable-oriented work for point cloud recognition.

Acknowledgements. This research has been funded by the Federal Ministry of Education and Research of Germany and the state of North-Rhine Westphalia as part of the Lamarr-Institute for Machine Learning and Artificial Intelligence, LAMARR22B.

	Random Baselines		Fractal Features				
	Baseline_U	*Baseline_W*	*Num_F*	*1D_Proj*	*2D_Proj*	*In_db*	*Sample_db*
Accuracy	2.5	3.0	65.5	65.4	66.5	63.3	61.1
Precision	2.6	2.4	59.8	56.8	57.7	54.4	54.1
Recall	2.4	2.4	60.1	57.5	58.4	55.4	54.4

Table 3. Feasibility tests for fractal features. The baselines are: Uniform random guess and weighted random guess. The fractal features from left to right: number of fractions, 1D projection, 2D projection, intra-fraction distribution, sampled distributions.

Module	Augmentations		Fractal features					
	~~*Multi_Frac*~~	~~*Rotat_Aug*~~	~~*Num_F*~~	~~*1D_Proj*~~	~~*2D_proj*~~	~~*In_db*~~	~~*Sample_db*~~	*All*
OA(%)	81.9	85.4	85.8	85.7	85.5	85.3	85.9	**86.3**
mAcc(%)	75.0	79.2	79.9	79.4	79.4	79.2	79.6	**80.4**
F1	75.3	79.8	80.3	80.2	79.9	79.5	79.9	**81.1**

Table 4. Ablation study for modules. From left to right, the absent modules are: Multiple fractal series, rotation augmentation, number of fractions, 1D projection, 2D projection, intra-fraction distribution, sampled distributions. The last column indicates that all modules are integrated.

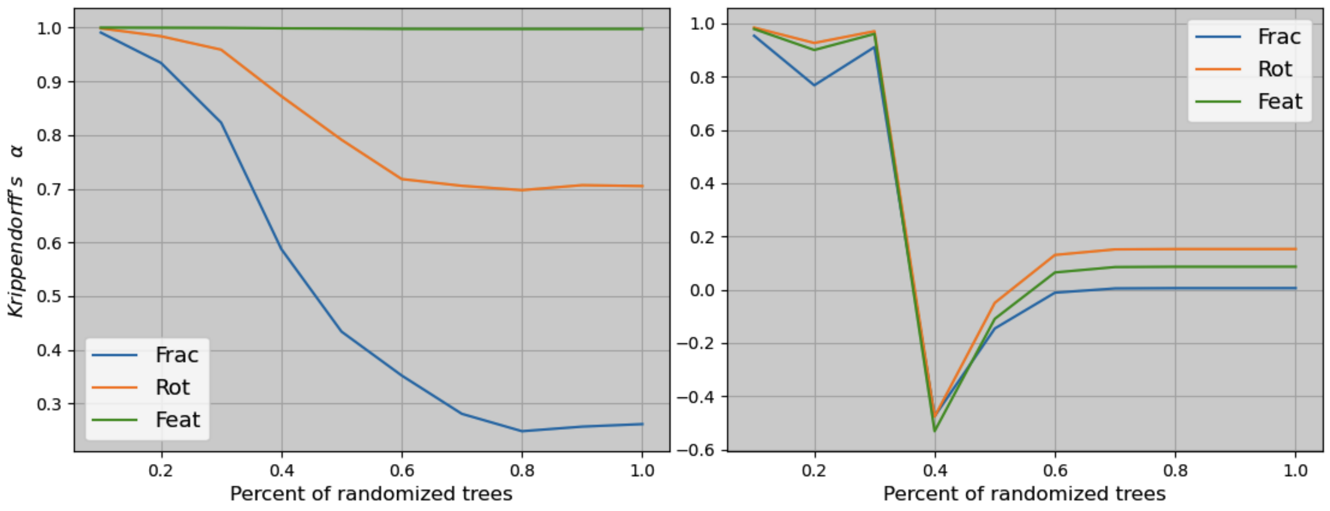

Figure 5. Results of sanity checks. The left side is the intrinsic feature importance, and the right side is the group feature ablation. The blue, orange and green lines denote fractal windows, rotation and hand-crafted features respectively.The x and y axes denote the percentage of randomized trees and Krippendorff's α score, respectively.

References

[1] Nicholas I Arnold, Plamen Angelov, and Peter M Atkinson. An improved explainable point cloud classifier (xpcc). *IEEE Transactions on Artificial Intelligence*, 2022.

[2] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

[3] Yaodong Cui, Ren Chen, Wenbo Chu, Long Chen, Daxin Tian, Ying Li, and Dongpu Cao. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[4] Mengnan Du, Ninghao Liu, and Xia Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019.

[5] Yifan Feng, Zizhao Zhang, Xibin Zhao, Rongrong Ji, and Yue Gao. Gvcnn: Group-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 264–272, 2018.

[6] Ankit Goyal, Hei Law, Bowei Liu, Alejandro Newell, and Jia Deng. Revisiting point cloud shape classification with a simple and effective baseline. In *International Conference on Machine Learning*, pages 3809–3820. PMLR, 2021.

[7] Ananya Gupta, Simon Watson, and Hujun Yin. 3d point cloud feature explanations using gradient-based methods. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[8] Felix Hausdorff. Dimension und äußeres maß. *Mathematische Annalen*, 79(1):157–179, 1918.

[9] Congjie He, Meng Ma, and Ping Wang. Extract interpretability-accuracy balanced rules from artificial neural networks: A review. *Neurocomputing*, 387:346–358, 2020.

[10] Robert R Hoffman, Shane T Mueller, Gary Klein, and Jordan Litman. Metrics for explainable ai: Challenges and prospects. *arXiv preprint arXiv:1812.04608*, 2018.

[11] Sara Hooker, Dumitru Erhan, Pieter-Jan Kindermans, and Been Kim. A benchmark for interpretability methods in deep neural networks. *Advances in neural information processing systems*, 32, 2019.

[12] Dominik Janzing, Lenon Minorics, and Patrick Blöbaum. Feature relevance quantification in explainable ai: A causal problem. In *International Conference on artificial intelligence and statistics*, pages 2907–2916. PMLR, 2020.

[13] Asako Kanezaki, Yasuyuki Matsushita, and Yoshifumi Nishida. Rotationnet: Joint object categorization and pose estimation using multiviews from unsupervised viewpoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5010–5019, 2018.

[14] Ben Kehoe, Sachin Patil, Pieter Abbeel, and Ken Goldberg. A survey of research on cloud robotics and automation. *IEEE Transactions on automation science and engineering*, 12(2):398–409, 2015.

[15] Pieter-Jan Kindermans, Sara Hooker, Julius Adebayo, Maximilian Alber, Kristof T Schütt, Sven Dähne, Dumitru Erhan, and Been Kim. The (un) reliability of saliency methods. In *Explainable AI: Interpreting, Explaining and Visualizing Deep Learning*, pages 267–280. Springer, 2019.

[16] Klaus Krippendorff. *Content analysis: An introduction to its methodology*. Sage publications, 2018.

[17] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. *Advances in neural information processing systems*, 31, 2018.

[18] Ying Li, Lingfei Ma, Zilong Zhong, Fei Liu, Michael A Chapman, Dongpu Cao, and Jonathan Li. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems*, 32(8):3412–3432, 2020.

[19] Yong Li, Guofeng Tong, Xingang Li, Liqiang Zhang, and Hao Peng. Mvf-cnn: Fusion of multilevel features for large-scale point cloud classification. *IEEE Access*, 7:46522–46537, 2019.

[20] Yongcheng Liu, Bin Fan, Shiming Xiang, and Chunhong Pan. Relation-shape convolutional neural network for point cloud analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8895–8904, 2019.

[21] Wei-Yin Loh. Classification and regression trees. *Wiley interdisciplinary reviews: data mining and knowledge discovery*, 1(1):14–23, 2011.

[22] Xu Ma, Can Qin, Haoxuan You, Haoxi Ran, and Yun Fu. Rethinking network design and local geometry in point cloud: A simple residual mlp framework. *arXiv preprint arXiv:2202.07123*, 2022.

[23] 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. IEEE, 2015.

[24] Christoph Molnar. *Interpretable Machine Learning*. 2 edition, 2022.

[25] Daniel Munoz, Nicolas Vandapel, and Martial Hebert. Directional associative markov network for 3-d point cloud classification. 2008.

[26] Terence Parr, Kerem Turgutlu, Christopher Csiszar, and Jeremy Howard. Beware default random forest importances, 2018.

[27] François Pomerleau, Francis Colas, and Roland Siegwart. A review of point cloud registration algorithms for mobile robotics. *Foundations and Trends in Robotics*, 4(1):1–104, 2015.

[28] 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.

[29] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017.

[30] Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature Machine Intelligence*, 1(5):206–215, 2019.

[31] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020.

[32] Carolin Strobl, Anne-Laure Boulesteix, Thomas Kneib, Thomas Augustin, and Achim Zeileis. Conditional variable importance for random forests. *BMC bioinformatics*, 9(1):1–11, 2008.

[33] Carolin Strobl, Anne-Laure Boulesteix, Achim Zeileis, and Torsten Hothorn. Bias in random forest variable importance measures: Illustrations, sources and a solution. *BMC bioinformatics*, 8(1):1–21, 2007.

[34] Pascal Sturmfels, Scott Lundberg, and Su-In Lee. Visualizing the impact of feature attribution baselines. *Distill*, 2020. https://distill.pub/2020/attribution-baselines.

[35] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015.

[36] Mukund Sundararajan and Amir Najmi. The many shapley values for model explanation. In *International conference on machine learning*, pages 9269–9278. PMLR, 2020.

[37] Hanxiao Tan and Helena Kotthaus. Surrogate model-based explainability methods for point cloud nns. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2239–2248, 2022.

[38] Richard Tomsett, Dan Harborne, Supriyo Chakraborty, Prudhvi Gurram, and Alun Preece. Sanity checks for saliency metrics. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6021–6029, 2020.

[39] Mikaela Angelina Uy, Quang-Hieu Pham, Binh-Son Hua, Thanh Nguyen, and Sai-Kit Yeung. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1588–1597, 2019.

[40] 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)*, 38(5):1–12, 2019.

[41] 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.

[42] Haoxuan You, Yifan Feng, Rongrong Ji, and Yue Gao. Pvnet: A joint convolutional network of point cloud and multi-view for 3d shape recognition. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 1310–1318, 2018.

[43] Jianhui Yu, Chaoyi Zhang, Heng Wang, Dingxin Zhang, Yang Song, Tiange Xiang, Dongnan Liu, and Weidong Cai. 3d medical point transformer: Introducing convolution to attention networks for medical point cloud analysis. *arXiv preprint arXiv:2112.04863*, 2021.

[44] Min Zhang, Haoxuan You, Pranav Kadam, Shan Liu, and C-C Jay Kuo. Pointhop: An explainable machine learning method for point cloud classification. *IEEE Transactions on Multimedia*, 22(7):1744–1755, 2020.

[45] Zhenxin Zhang, Liqiang Zhang, Xiaohua Tong, P Takis Mathiopoulos, Bo Guo, Xianfeng Huang, Zhen Wang, and Yuebin Wang. A multilevel point-cluster-based discriminative feature for als point cloud classification. *IEEE Transactions on Geoscience and Remote Sensing*, 54(6):3309–3321, 2016.

[46] Tianhang Zheng, Changyou Chen, Junsong Yuan, Bo Li, and Kui Ren. Pointcloud saliency maps. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1598–1606, 2019.

[47] Jianlong Zhou, Amir H Gandomi, Fang Chen, and Andreas Holzinger. Evaluating the quality of machine learning explanations: A survey on methods and metrics. *Electronics*, 10(5):593, 2021.