Interpretable3D: An Ad-Hoc Interpretable Classifier for 3D Point Clouds

Tuo Feng[1], **Ruijie Quan**[2*], **Xiaohan Wang**[2], **Wenguan Wang**[2], **Yi Yang**[2]

[1]ReLER, AAII, University of Technology Sydney
[2]ReLER, CCAI, Zhejiang University
feng.tuo@student.uts.edu.au, quanruij@hotmail.com, {wxh1996111, wenguanwang.ai}@gmail.com, yi.yang@uts.edu.au

Abstract

3D decision-critical tasks urgently require research on explanations to ensure system reliability and transparency. Extensive explanatory research has been conducted on 2D images, but there is a lack in the 3D field. Furthermore, the existing explanations for 3D models are *post-hoc* and can be misleading, as they separate explanations from the original model. To address these issues, we propose an *ad-hoc* interpretable classifier for 3D point clouds (*i.e.*, Interpretable3D). As an intuitive case-based classifier, Interpretable3D can provide reliable *ad-hoc* explanations without any embarrassing nuances. It allows users to understand how queries are embedded within past observations in prototype sets. Interpretable3D has two iterative training steps: *1)* updating one prototype with the mean of the embeddings within the same sub-class in Prototype Estimation, and *2)* penalizing or rewarding the estimated prototypes in Prototype Optimization. The mean of embeddings has a clear statistical meaning, *i.e.*, class sub-centers. Moreover, we update prototypes with their most similar observations in the last few epochs. Finally, Interpretable3D classifies new samples according to prototypes. We evaluate the performance of Interpretable3D on four popular point cloud models: DGCNN, PointNet2, PointMLP, and PointNeXt. Our Interpretable3D demonstrates comparable or superior performance compared to softmax-based black-box models in the tasks of 3D shape classification and part segmentation. Our code is released at: github.com/FengZicai/Interpretable3D.

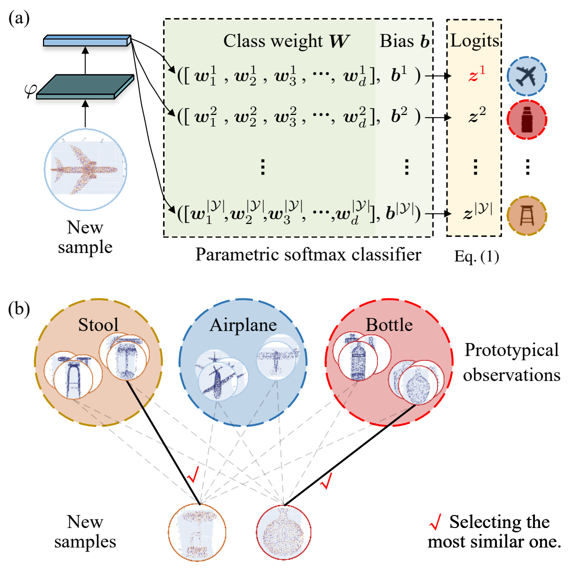
(a)
φ
New sample
Class weight $\boldsymbol{W}$
Bias $\boldsymbol{b}$
Logits
$([\boldsymbol{w}_1^1, \boldsymbol{w}_2^1, \boldsymbol{w}_3^1, \cdots, \boldsymbol{w}_d^1], \boldsymbol{b}^1)$
$([\boldsymbol{w}_1^2, \boldsymbol{w}_2^2, \boldsymbol{w}_3^2, \cdots, \boldsymbol{w}_d^2], \boldsymbol{b}^2)$
$([\boldsymbol{w}_1^{|\mathcal{Y}|}, \boldsymbol{w}_2^{|\mathcal{Y}|}, \boldsymbol{w}_3^{|\mathcal{Y}|}, \cdots, \boldsymbol{w}_d^{|\mathcal{Y}|}], \boldsymbol{b}^{|\mathcal{Y}|})$
z^1
z^2
$z^{|\mathcal{Y}|}$
Parametric softmax classifier
Eq. (1)
(b)
Stool
Airplane
Bottle
Prototypical observations
New samples
√ Selecting the most similar one.

Figure 1: (a) The softmax-based DNN models employ parametric classifiers. However, they lack a direct and intuitive interpretation of the decision-making processes. (b) Interpretable3D selects the most similar prototype (*i.e.*, the one with the maximum cosine similarity) for new samples.

Introduction

Over the past decade, significant progress has been made in deep neural networks (DNNs). However, serious concerns have been raised over DNNs' safety and trustworthiness (Rudin 2019; Rudin et al. 2022), when applied to 3D real-world decision-critical scenarios, such as self-driving cars (Meng et al. 2021; Yin et al. 2022a; Zheng et al. 2020) and medical diagnosis systems. Interpretability has emerged as a critical concern regarding the trustworthiness of predictions generated by DNNs. However, understanding DNNs proves to be particularly challenging due to their complex, non-linear, and high-dimensional nature. They lack natural explanations that humans can easily comprehend. Similarly, the parametric softmax classifier learns highly abstract parameters and lacks a direct and intuitive interpretation (Wang et al. 2022; Li et al. 2018; Angelov and Soares 2020). Incorrect data fed into black-box models may lead to harmful consequences. As a result, the demand for interpretable 3D models has become increasingly urgent.

Extensive research has been conducted on the interpretability/explainability of 2D images (Wang et al. 2022; Zhang, Rao, and Yang 2021; Akhtar and Jalwana 2023; Hu et al. 2023), but there is a lack of interpretable 3D point cloud research. Not to mention the few existing explanation studies on 3D models have been conducted with *post-hoc* explanations and only play an auxiliary role in 3D systems. For instance, saliency/attention maps (Chen et al. 2021; Schinagl et al. 2022) have been applied to existing voxel-based 3D object detection networks. However, the *post-hoc* saliency maps

[*]Corresponding author: Ruijie Quan.

cannot explain the roles of highlighted parts in the decision-making process (Rudin 2019). The various saliency methods even produce conflicting saliency maps, making it difficult for researchers to determine which one actually reflects attention (Rudin et al. 2022). Moreover, *post-hoc* explanations are derived from a separate modeling process with strong priors. These priors, however, are not part of training (Li et al. 2018). The nuances (minor differences) between *post-hoc* modeling process and DNNs are also unknown (Rudin 2019). So *post-hoc* analysis may result in problematic and misleading explanations (Rudin et al. 2022; Arrieta et al. 2020). In contrast, *ad-hoc* interpretability methods can provide reliable explanations without any unknown nuances (Rudin 2019).

In this work, we design an inherently interpretable classifier for 3D point clouds (*i.e.*, Interpretable3D). It is inherently an intuitive case- or instance-based classifier, as it reveals what the representation means and how the embedding queries typical past observations from the prototype sets. As shown in Fig. 1, new observations are classified based on their proximity to a prototype observation within the dataset. Therefore, Interpretable3D can explain its own reasoning process in a human-understandable way. The learned models naturally come with explanations for each prediction, and the explanations are loyal to what the network actually computes. Our Interpretable3D provides a level of interpretability that is absent in existing *post-hoc* 3D explanation models.

In each iteration, training sample embeddings are clustered and averaged to estimate prototypes (*i.e.*, Prototype Estimation). Compared to the softmax classifier, the mean of embeddings holds distinct statistical significance, representing class sub-centers. Furthermore, the estimated prototypes are then either penalized or rewarded based on their performance in the prediction (*i.e.*, Prototype Optimization). By taking the within-class clustering as a dual *optimal transport problem* (Feng et al. 2023; Wang et al. 2022), Prototype Estimation can proficiently uncover diverse intra-class variations in an online mode. Moreover, it can automatically extract discriminative prototypes to handle complex real-world disparities. The following Prototype Optimization can enhance the representativeness of the estimated prototypes by applying penalties or rewards. Furthermore, we enhance the interpretability of prototypes by updating them with their most similar observations in the final few epochs.

We evaluate Interpretable3D with four point cloud models: DGCNN (Phan et al. 2018) (Graph-based), PointNet2 (Yan 2019), PointMLP (Ma et al. 2021), and PointNeXt (Qian et al. 2022) (MLP-based). Our experiments are conducted on three well-known public benchmarks (*i.e.*, ModelNet40 (Wu et al. 2015) and ScanObjectNN (Uy et al. 2019) for shape classification and ShapeNetPart (Yi et al. 2016) for part segmentation). Interpretable3D achieves comparable or even better performance than softmax-based black-box models. Additionally, our approach provides intrinsic interpretability to the classification and part segmentation results. This is a key advantage of our approach, as it allows for better transparency and comprehensibility of the AI decision-making process. This is our response to the current lack of *ad-hoc* interpretable research within the field of 3D vision.

Related Work

In 3D vision, existing research of *post-hoc* analysis includes activation maximization (Tan 2023) and saliency/attention maps (Zheng et al. 2019; Ziwen et al. 2020; Chen et al. 2021; Schinagl et al. 2022). However, *post-hoc* explanations are problematic and misleading (Rudin et al. 2022; Laugel et al. 2019; Arrieta et al. 2020) for the following reasons: ***i)*** *Post-hoc* analysis requires a separate modeling effort, which is not completely faithful to the original model, with unknown nuances (Rudin 2019). Such a nuance makes it hard to guarantee the correctness of their interpretations (Fan et al. 2021). ***ii)*** *Post-hoc* explanations can vary depending on the chosen explanation models (Li et al. 2018). This can lead to numerous conflicting yet seemingly convincing explanations for the same classification decision (Li et al. 2018), none of which may actually be the correct reason for the classification (Hong et al. 2023; Rudin et al. 2022). ***iii)*** *Post-hoc* explanations cannot provide a reasoning process for the network's decision-making (Li et al. 2018; Chen et al. 2019; Zhu and Yang 2020). For instance, saliency maps cannot explain how the highlighted pixels are used (Rudin 2019). ***iv)*** *Post-hoc* methods may produce explanations that are not interpretable to humans, necessitating extra modeling to ensure understandability (Li et al. 2018; Wang et al. 2023a). In a sense, *post-hoc* explainability methods are often regarded as an excuse to deploy black-box models (Rudin and Radin 2019; Min et al. 2023), explaining and losing accuracy. Whereas an *ad-hoc* interpretable model does not. Interpretability should promote accuracy and not the other way around.

As far as we know, we are the first attempt to design an *ad-hoc* interpretable classifier for point cloud parsing. Although neural networks have complex structures and it is unclear how the data are mapped into features, we can use interpretable machinery to gain insights into the decision-making mechanism (Rudin et al. 2022; Wang et al. 2023b). To pursue *ad-hoc* interpretability, our Interpretable3D has integrated interpretable machinery into black-box models. It intrinsically relies on intuitive case- or instance-based paradigms, revealing the meaning of the representation and how that information influences decision-making. During training, Interpretable3D defines prototypes as representative class centers (Zhou et al. 2022; Yin et al. 2022b; Liang et al. 2023) within the same space as the observed data. This unique approach allows for more meaningful discussions with domain experts, as prototypes can be easily interpreted, just like the original observations. This contrasts starkly with the widely used unexplainable parametric softmax classifier. More importantly, Interpretable3D not only provides self-explanation without *post-hoc* analysis but also achieves comparable performance when compared to the softmax-based DNN models.

Proposed Algorithm

We develop an interpretable nearest neighbor based prototype classifier, the overview of which is illustrated in Fig. 2. Suppose that the input is $\mathcal{X}=\{x_k : k=1,2,\dots,K\}$, *i.e.*, a set with K training samples $\{x_k : x_k \in \mathbb{R}^{N\times 3}\}$. Each sample consists of N points. $\mathcal{L}=\{l_k \in \mathcal{Y} : k=1,2,\dots,K\}$ is a set of object labels, where $\mathcal{Y}$ is the label set. The point-based methods $\phi \circ \varphi : \mathcal{X} \mapsto \mathcal{L}$ take $\mathcal{X}$ as input. $\varphi : \mathcal{X} \mapsto \mathcal{F}$ learns the underlying representations $\mathcal{F} = \{f_k : k = 1, 2, \ldots, K\}$, and then $\phi : \mathcal{F} \mapsto \mathcal{L}$ predicts the labels.

Preliminary: Softmax Classifier

Currently, the prevailing approach for ϕ is the parametric softmax classifier, it can be formulated as:

$$\hat{l} = \arg\max\nolimits_{l \in \mathcal{Y}} z^l, \tag{1}$$

where $\hat{l}$ denotes class prediction; z^l is an unnormalized l class output of the last fully-connected (FC) layer, composing the logits vector $\boldsymbol{z} = [z^l]_l \in \mathbb{R}^{|\mathcal{Y}|}$. A softmax function is then applied to $\boldsymbol{z}$. While training, the learnable parameters in ϕ and φ are updated by minimizing the cross-entropy loss:

$$\underset{w^*, b^*, \theta}{\arg\min} \sum_{k=1}^{K} -\log \frac{\exp\left(\left(\boldsymbol{w}^{l_k}\right)^\top \varphi_\theta(x_k) + b^{l_k}\right)}{\sum_{l \in \mathcal{Y}} \exp\left(\left(\boldsymbol{w}^{l}\right)^\top \varphi_\theta(x_k) + b^{l}\right)}, \tag{2}$$

where the parameters θ are in φ, the weight matrix $\boldsymbol{W} = [\boldsymbol{w}^l]_l \in \mathbb{R}^{d \times |\mathcal{Y}|}$ and bias vector $\boldsymbol{b} = [b^l]_l \in \mathbb{R}^{|\mathcal{Y}|}$ are learnable parameters. The softmax classifier ϕ has been trained purely to optimize the accuracy. However, these learned highly abstract parameters are disconnected from the physical characteristics of the problem being modeled (Angelov and Soares 2020) and lack a direct or intuitive interpretation (Li et al. 2018; Wang et al. 2022). From a human perspective, they do not provide any clues about what drives the classifier to reach its decisions.

Interpretable3D

Previous point cloud methods either utilize a non-transparent parametric softmax classifier or attempt to understand the network in *post-hoc* paradigms. However, these paradigms can be problematic and misleading. To offer a human-readable explanation for point clouds, we propose Interpretable3D. In contrast to the softmax classifier, Interpretable3D is built upon the intuitive concept of selecting the most similar prototype for new samples. Note that this contrast between parametric and non-parametric classifiers is important. It enables researchers to comprehend that ϕ can be learned without relying on non-transparent weight vectors.

Here we utilize S within-class prototypes for $\mathcal{Y}$ to represent past observations, *i.e.*, $|\mathcal{Y}| \times S$ prototypes $\boldsymbol{M} = [\boldsymbol{m}_s^l]_{l,s} \in \mathbb{R}^{d \times |\mathcal{Y}| \times S}$ in total. This is due to the presence of intra-class differences; each class cannot be accurately represented by just one prototype. In most cases, the number of prototypes surpasses that of categories (Nova and Estévez 2014). Therefore, the prediction $\hat{l}$ is made by assigning the labels of the most similar prototypes to the samples (Wang et al. 2022):

$$\hat{l} = l^*, \;\; (l^*, s^*) = \arg\max\nolimits_{l \in \mathcal{Y}, s \in \{1, \cdots, S\}} \langle f, \boldsymbol{m}_s^l \rangle, \tag{3}$$

where $\langle \cdot, \cdot \rangle$ is cosine similarity. Our model is fully *online* and *end-to-end* trained. We use the most standard and simple initialization strategy (Biehl, Hammer, and Villmann 2016): randomly selecting S data samples per class as the initial prototypes. During each training iteration, the prototypes are first updated and then optimized. Firstly, training samples within the same category are clustered and assigned sub-class labels. The $|\mathcal{Y}| \times S$ prototypes are updated with the mean of embeddings from the same subclass in a momentum manner. We then assign the labels of the most similar prototypes to the samples (Eq. (3)), and $\hat{l}$ is obtained. Next, Prototype Optimization involves penalizing or rewarding $\boldsymbol{m}_s^l$ according to $\hat{l}$ and l (Eq. (7,8)). Moreover, in the final few epochs, the prototypes are updated with the features of the most representative training samples, rather than the mean of embeddings. As a result, the prototypes are connected to typical examples and can be seen as typical observations for classification. Finally, all the prototypes are stored as classification evidence for inference.

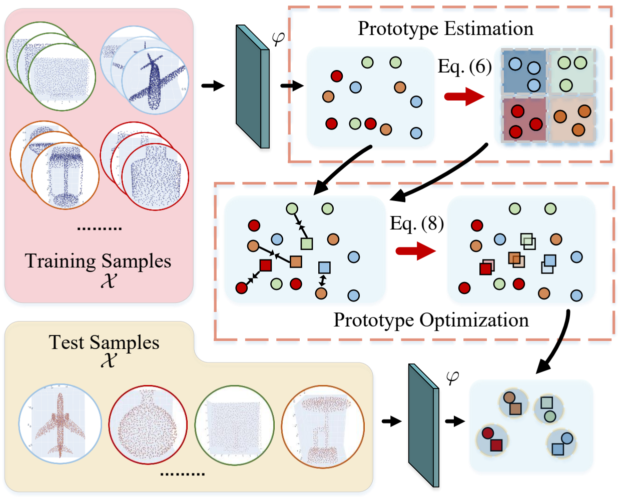

Figure 2: The overview of our Interpretable3D. At each training iteration, Interpretable3D first estimates the prototypes and then optimize them. Sample features are represented by points, and prototypes are represented by squares. In Prototype Estimation, sample features are regarded as an *optimal transport problem* (*i.e.*, Eq (6)). Intra-class clustering is then conducted to estimate the mean of embeddings that are within the same subclass. In Prototype Optimization, prototypes that have a higher cosine similarity to the correct prediction receive a reward (→←), while those that are misclassified receive a punishment (↔).

Interpretable3D predicts by evaluating the similarity between samples and prototypes, offering a clear depiction of the decision-making process. This aligns with the *prototype theory* in psychological cognitive sciences (Rosch 1975; Knowlton and Squire 1993). In *prototype theory*, an object can be classified based on a single representative example or a set of items that are typical for the category (Taylor 2003).

Prototype Estimation

We cluster intra-class representations in the latent space $\mathcal{F}$ to mine typical prototypes. Suppose there are N^l samples for class l, underlying representation $\boldsymbol{F}^l \in \mathbb{R}^{d \times N^l}$ are mapped to prototypes $\boldsymbol{M}^l \in \mathbb{R}^{d \times S}$. We denote $\boldsymbol{A}^l \in \{0,1\}^{S \times N^l}$ as the assignment matrix of $\boldsymbol{F}^l$ among S subclasses. The (s,n)-*th* element of $\boldsymbol{A}^l$ indicates whether to assign the n-*th* feature of $\boldsymbol{F}^l$ to the *s-th* sub-cluster center. The optimization of $\boldsymbol{A}^l$ can be accomplished by taking the similarity between the representations and cluster centers as a cost matrix, *i.e.*, $\boldsymbol{Q}^l = \text{softmax}(-\boldsymbol{M}^{l\top}\boldsymbol{F}^l)$. Thus, the label assignment task can be viewed as an instance of the *optimal transport problem* (Asano, Rupprecht, and Vedaldi 2019; Wang et al. 2022):

$$\begin{aligned} \boldsymbol{A}^{l*} &= \underset{\boldsymbol{A}^l \geq 0}{\arg\min} \langle \boldsymbol{Q}^l, \boldsymbol{A}^l \rangle_F, \\ \text{s.t. } \boldsymbol{A}^l \mathbf{1}_{N^l} &= \boldsymbol{r}, \boldsymbol{A}^{l\top} \mathbf{1}_S = \boldsymbol{c}, \end{aligned} \tag{4}$$

where $\boldsymbol{A}^{l*}$ is the *optimal transportation plan* between the representations and cluster centers, which can be also viewed as the transportation plan of the sample and prototype. $\langle\cdot\rangle_F$ is the Frobenius dot-product. The vectors $\boldsymbol{r}$ and $\boldsymbol{c}$ are marginal projections of $\boldsymbol{A}^l$. They define constraints for $\boldsymbol{A}^l$, *i.e.*, $\boldsymbol{r} = \frac{1}{S}\mathbf{1}_S, \boldsymbol{c} = \frac{1}{N^l}\mathbf{1}_{N^l}$. The former equipartition constraint guarantees that, on average, each prototype is selected at least $1/S$ times. The latter unique assignment constraint ensures that every point is exclusively assigned to one prototype. The entropic regularization (Cuturi 2013) of problem 4 is formulated as:

$$\begin{aligned} &\min_{\boldsymbol{A}^l \geq 0} \langle \boldsymbol{Q}^l, \boldsymbol{A}^l \rangle_F - \zeta H(\boldsymbol{A}^l), \\ &\text{s.t. } \boldsymbol{A}^l \mathbf{1}_{N^l} = \boldsymbol{r}, \boldsymbol{A}^{l\top} \mathbf{1}_S = \boldsymbol{c}, \zeta > 0, \end{aligned} \tag{5}$$

where ζ is the regularization parameter, and $H(\boldsymbol{A}^l)$ is the entropic regularization. Problem 5 is constrained by affine constraints, while the dual problem is unconstrained, making it simpler to design algorithms and analyze complexity (Lin, Ho, and Jordan 2019). With the Lagrangian function, the dual form (Lin, Ho, and Jordan 2019) of problem 5 can be simplified as follows:

$$\min_{\boldsymbol{u} \in \mathbb{R}^S, \boldsymbol{v} \in \mathbb{R}^{N^l}} \left\{ \mathbf{1}_S^\top \boldsymbol{A}^{l*} \mathbf{1}_{N^l} - \langle \boldsymbol{u}, \boldsymbol{r} \rangle_F - \langle \boldsymbol{v}, \boldsymbol{c} \rangle_F \right\}, \tag{6}$$

where $\boldsymbol{A}^{l*} = \text{diag}\left(e^{\boldsymbol{u}}\right) e^{-\boldsymbol{Q}^l/\gamma} \text{diag}\left(e^{\boldsymbol{v}}\right)$. $\boldsymbol{u}$ and $\boldsymbol{v}$ are two vectors of scaling coefficients, resulting from a small number of matrix multiplications. γ trades off convergence speed with closeness to the original transport problem. This task can be solved by Sinkhorn-Knopp algorithm (Knight 2008) and APDAGD (Lin, Ho, and Jordan 2019). During training, the latent representation space $\mathcal{F}$ undergoes changes. This attribute means that the prototypes need be recalculated after each batch using the entire dataset. However, this process can be costly and time-consuming. To address this, we employ a momentum update strategy (He et al. 2020). It updates each prototype with the average of embeddings assigned to each sub-cluster of the training samples. However, in the last few epochs, we substitute the average of embeddings with the features of the closest training sample.

Prototype Optimization

Among prototype-based classification methods, a particularly attractive approach is Learning Vector Quantization (LVQ) (Kohonen 1990, 2012; Nova and Estévez 2014). The LVQ family employs the winner-takes-all (WTA) principle, where prototypes compete for updates according to labels and predictions (Bishop and Nasrabadi 2006; Ritter et al. 1992). One fundamental design within LVQ family is the LVQ1 algorithm. This algorithm aims to rectify the decision boundary by adjusting the prototypes. To avoid suboptimal local minima or so-called "dead unit" in the WTA training process (Biehl, Hammer, and Villmann 2016), we take the estimated prototypes $\boldsymbol{M}^l$ in Prototype Estimation as the initial set. Therefore, each class is also represented by S prototypes. Similar to LVQ1, we crisply assign each representation in $\boldsymbol{F}^l = [f^l \in \mathbb{R}^d]_l$ to the closest prototype $\boldsymbol{M}^w$, the so-called *winner*. Only the winning prototypes $\boldsymbol{M}^w$ are altered according to the *competitive learning* update equation:

$$\begin{aligned} \boldsymbol{M}^w &\leftarrow \boldsymbol{M}^w + \eta\, \psi\big(l, \hat{l}_w\big)\big(\boldsymbol{F}^l - \boldsymbol{M}^w\big), \\ \psi(l, \hat{l}_w) &= \begin{cases} +1 & \text{if } l = \hat{l}_w \\ -1 & \text{else} \end{cases}, \end{aligned} \tag{7}$$

where η is the update rate, it controls the magnitude of updates; $\hat{l}_w$ denotes class prediction of the winning prototypes. This update strategy implements a rewarding mechanism for the *winner*, which is based on its ability to correctly classify the input, by migrating the *winner* towards $\boldsymbol{F}^l$. Conversely, it applies a punishment to the *winner* when it fails to correctly label the input, *i.e.*, moving the *winner* away from $\boldsymbol{F}^l$. Upon repeated presentation of each training sample, the *winner* is moved in the direction of the example feature if they share the same label and in the opposite direction if they don't. An enhanced LVQ algorithm, known as LVQ2.1 (Kohonen 1990, 2012), is often favored because it is effective in *Bayesian decision theory*. Starting with properly estimated initial prototypes, we can update $\boldsymbol{M}^w$ as follows:

$$\begin{aligned} &\begin{cases} \boldsymbol{M}_p^w \leftarrow \boldsymbol{M}_p^w - \eta\left(\boldsymbol{F}^l - \boldsymbol{M}_p^w\right), \\ \boldsymbol{M}_q^w \leftarrow \boldsymbol{M}_q^w + \eta\left(\boldsymbol{F}^l - \boldsymbol{M}_q^w\right), \end{cases} \\ &\text{if } \min\Big(\frac{\langle \boldsymbol{F}^l, \boldsymbol{M}_p^w \rangle}{\langle \boldsymbol{F}^l, \boldsymbol{M}_q^w \rangle}, \frac{\langle \boldsymbol{F}^l, \boldsymbol{M}_q^w \rangle}{\langle \boldsymbol{F}^l, \boldsymbol{M}_p^w \rangle}\Big) > \frac{1-\mu}{1+\mu}, \end{aligned} \tag{8}$$

where $\langle\cdot,\cdot\rangle$ is cosine similarity. $\boldsymbol{M}_p^w$ and $\boldsymbol{M}_q^w$ are the nearest prototypes to $\boldsymbol{F}^l$; $\boldsymbol{F}^l$ and $\boldsymbol{M}_p^w$ belong to the same class, while $\boldsymbol{M}_q^w$ doesn't. Moreover, μ refers to relative window width, defined around the midplane of the two nearest prototypes. Decision boundaries (*i.e.*, the midplane) are directly shifted toward the Bayes limits with attractive and repulsive forces from $\boldsymbol{F}^l$. Finally, prototypes can represent their respective class by assuming class-typical positions in $\mathcal{F}$.

Algorithm Details

Training Objective. In Interpretable3D, we utilize $|\mathcal{Y}| \times S$ prototypes $\left[\boldsymbol{m}_s^l\right]_{l,s} \in \mathbb{R}^{d \times |\mathcal{Y}| \times S}$ (*i.e.*, mean feature vectors of the training data) to involve in training objective:

$$\arg\min_\theta \sum_{k=1}^{K} -\log \frac{\exp\left(\max\left(\left\{\langle \varphi_\theta(x_k), \boldsymbol{m}_s^{l_k} \rangle\right\}_{s=1}^S\right)\right)}{\sum_{l \in \mathcal{Y}} \exp\left(\max\left(\left\{\langle \varphi_\theta(x_k), \boldsymbol{m}_s^l \rangle\right\}_{s=1}^S\right)\right)}. \tag{9}$$

Comparing Eq. (2) and Eq. (9), $\left[\boldsymbol{m}_s^l\right]_{l,s}$ are derived solely from data features, and the model can minimize training objective only by optimizing the vector $\varphi_\theta(x_k)$ instead of the

Method	OA(%)	mAcc(%)
PointNet (Qi et al. 2017a)	89.2	86.0
PointNet++ (Qi et al. 2017b)	90.7	-
PointNet2 (Yan 2019)	92.2	-
PointNet2 + **`Ours`**	93.2	89.3
DGCNN (Phan et al. 2018)	92.9	90.2
DGCNN + **`Ours`**	93.5	90.3
PointMLP (Ma et al. 2021)	94.1	91.3
PointMLP + **`Ours`**	94.1	92.0
PointNeXt (Qian et al. 2022)	94.0	91.1
PointNeXt + **`Ours`**	94.3	91.8

Table 1: Classification results on ModelNet40.

Method	OA(%)	mAcc(%)
PointNet (Qi et al. 2017a)	68.2	63.4
PointNet++ (Qi et al. 2017b)	77.9	75.4
DGCNN (Phan et al. 2018)	78.1	73.6
DGCNN + **`Ours`**	78.0	74.3
PointNet2 (Yan 2019)	79.1	77.6
PointNet2 + **`Ours`**	79.3	78.4
PointMLP (Ma et al. 2021)	85.4	83.9
PointMLP + **`Ours`**	85.6	84.5
PointNeXt (Qian et al. 2022)	87.7	85.8
PointNeXt + **`Ours`**	88.0	86.5

Table 2: Classification results on ScanObjectNN.

parametric softmax classifier. Moreover, with such a non-parametric, distance-based scheme, Interpretable3D builds a closer link to metric learning in the adaptive latent space.

Typical Past Observations. While updating with the average of embeddings, the working mechanism remains intuitive — classify data to the class of closest sub-center, and the prototypes have a clear statistical meaning — class sub-centers. In contrast, the class weights of softmax classifier are learnable parameters that are difficult to understand. During the last 20 epochs, we find the typical past observation that has maximum similarity for each estimated prototype, and the prototypes are updated with the typical samples' features instead of average embeddings. This implements the classic *prototype theory* in cognition: human refer to past exemplar observations for classification decision-making (Rosch 1975; Knowlton and Squire 1993; Taylor 2003). Once trained, these prototypes are stored (like the learnable weights of the softmax classifier) as classification evidence. By showing these prototypical training samples, human can understand how our model actually works — it performs a direct comparison between test data and these prototypical samples for classification. Thus our model is *ad-hoc* interpretable.

Online Clustering. Intra-class data samples are grouped into S subclasses for exploiting latent structures of the entire dataset. We empirically set $S = 15$. The momentum coefficient and μ in Eq. (8) are set as 0.999 and 0.4, following (He et al. 2020; Kohonen 1990, 2012).

Backbones $\phi \circ \varphi$. We evaluate our algorithm on point-based and graph-based networks, including DGCNN (Phan et al. 2018), PointNet++ (Qi et al. 2017b; Yan 2019), PointMLP (Ma et al. 2021), PointNeXt (Qian et al. 2022). The training and testing configurations follow the default settings of the respective methods mentioned above. Our algorithm implements an interpretable prototype-based learning scheme for 3D shape classification and part segmentation. For the part segmentation task, we deploy Deep Hough Voting (Qi et al. 2019) to extract instance-level features. Therefore, Interpretable3D can be applied to any object recognition and part segmentation networks that can learn instance-wise features.

Experiment

In this section, we integrate point cloud networks with Interpretable3D to demonstrate its capabilities. We begin by presenting the results of 3D shape classification on the ModelNet40 dataset (Wu et al. 2015). Subsequently, we delve into the performance evaluation on the ScanObjectNN dataset (Uy et al. 2019). Moving forward, we evaluate our algorithm on a part segmentation benchmark, *i.e.*, the ShapeNetPart dataset (Yi et al. 2016). Additionally, we utilize qualitative results to examine the decision-making process, and then use MMD2 and Maximum witness value to evaluate the prototypes and data distribution. Finally, we provide the analysis of the core components.

Datasets and Metrics. For shape classification, the model takes 1,024 points as input, and for part segmentation, it takes 2,048 points as input. We report the class-average accuracy (mAcc) and overall accuracy (OA) for shape classification, along with the class mean intersection over union (mIoU) and instance mIoU for part segmentation.

Shape Classification on ModelNet40

We compare the classification results of various classic works in Table 1. The results achieved by Interpretable3D are just as good as those of competitors in terms of OA and mAcc. More specifically, PointNet2 + **`Ours`** achieves 1.0% higher OA than PointNet2. PointNeXt + **`Ours`** and PointMLP + **`Ours`** demonstrate comparable performance to their counterparts on OA. These results are promising as the ModelNet40 dataset is extensively studied and the results have been long-standing at around 94%. Moreover, PointNet2 + **`Ours`** outperforms the advanced variant of PointNet++ (Qi et al. 2017b) (91.9% OA) that incorporates normal vectors and highly dense points (5k). This verifies the effectiveness of PointNet2 + **`Ours`**.

Shape Classification on ScanObjectNN

Except for the OA metric of DGCNN + **`Ours`**, our approach, as detailed in Table 2, outperforms point-based methods by a margin of 0.2%-0.3% and 0.6%-0.8% in terms of OA and mAcc, respectively. Moreover, with Interpretable3D, all the methods narrow the gap between mAcc and OA, indicating a more decent level of robustness than their counterparts. Actually, ScanObjectNN is highly challenging due to occlusions, noise, *etc*. Through Prototype Estimation, Interpretable3D can automatically uncover real-world variations and extract distinctive prototypes. The consistent improvements of PointNet2 + **`Ours`**, PointMLP + **`Ours`**, and PointNeXt + **`Ours`**

Method	C. mIoU	I. mIoU
PointNet (Qi et al. 2017a)	80.4	83.7
PointNet++ (Qi et al. 2017b)	81.9	85.1
DGCNN (Phan et al. 2018)	82.3	85.2
DGCNN + **Ours**	82.9	85.5
PointNet2 (Yan 2019)	82.5	85.4
PointNet2 + **Ours**	82.9	85.7
PointMLP (Ma et al. 2021)	84.6	86.1
PointMLP + **Ours**	84.9	86.2
PointNeXt (Qian et al. 2022)	85.2	87.0
PointNeXt + **Ours**	85.6	87.2

Table 3: Segmentation results on ShapeNetPart. C. mIoU refers to Class mIoU, and I. mIoU indicates Instance mIoU.

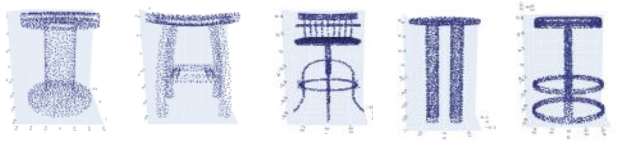

Figure 3: The prototypes for 'stool' on ModelNet40.

strongly confirm the effectiveness of Interpretable3D.

Part Segmentation on ShapeNetPart

We deploy Deep Hough Voting (Qi et al. 2019) to extract instance-level features. With Interpretable3D, each instance is assigned a part label, and all points belonging to the same instance are assigned the same category label. The outcomes are presented in Table 3. Our approach has achieved comparable results among all the tested methods, demonstrating its suitability for the part segmentation task.

Interpretability

As mentioned above, we have demonstrated the effectiveness of Interpretable3D. Here, we showcase its capabilities in transparency and interpretability. Following prior literature (Chen et al. 2019; Li et al. 2018) in other domains, we examine *ad-hoc* interpretability by presenting prototypes and assessing the similarity between prototypes and samples. **Interpretability for Predictions and Prototypes.** To make it easier to understand, we visualize some prototypical observations of Interpretable3D trained on the ModelNet40 dataset (Wu et al. 2015). Fig. 3 shows the prototypes for 'stool'. In addition, Interpretable3D allows users to see how the model comes to its predictions by visualizing the prototypes based on the similarity scores between test sample representatives and prototypes. In Fig. 4, we can understand how Interpretable3D makes decisions on the ModelNet40 dataset and ScanObjectNN. Taking the results on ModelNet40 for example, a bookshelf (test sample in the first row) is correctly classified, it also looks close to the prototype of 'bookshelf' (Prototype 1). However, in the failure case, Interpretable3D has difficulty in accurately determining whether the observation represents a 'flower pot' or a 'bottle'. It eventually makes the wrong decision. Even though users are unsure how

Model	OA(%)↑	mAcc(%)↑	MMD2↓	Witness_{max}↓
ProtoPNet	92.83	89.70	0.243	0.577
$\text{PointNeXt}_{\texttt{PE}}$	94.21	91.27	0.101	0.328
$\text{PointNeXt}_{\texttt{PO}}$	94.29	91.77	0.085	0.225

Table 4: Quantitative results on data distribution.

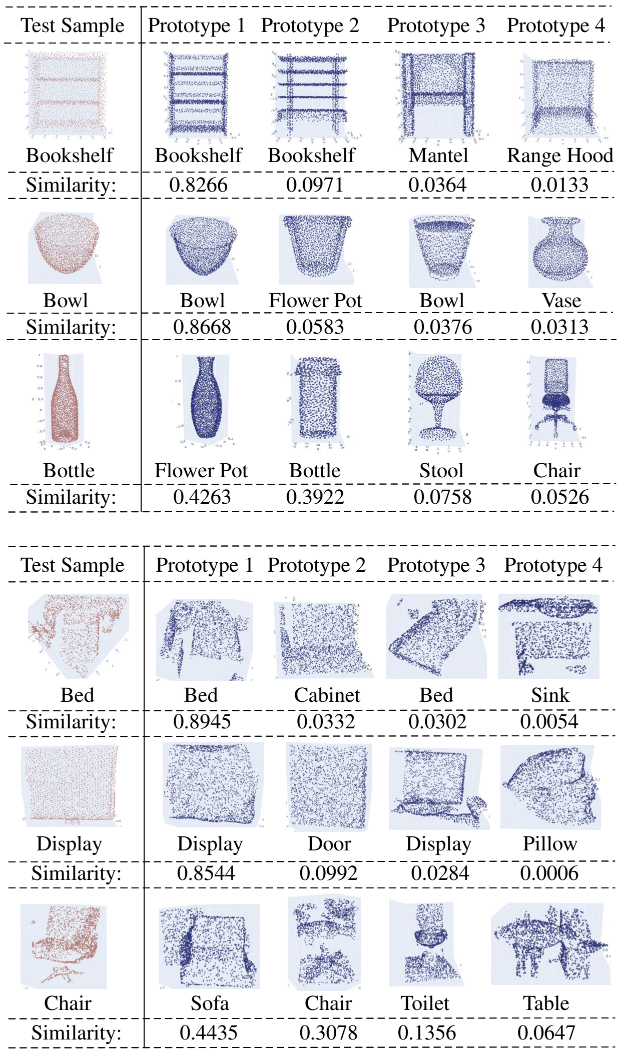

Figure 4: The upper part presents the results of Interpretable3D on ModelNet40, while the lower part displays the results on ScanObjectNN. Here, we interpret predictions with normalized similarity and visualized prototypes, including success and failure cases. For each part, the first two rows show the success cases that are correctly classified, while the last row shows the failure case. Test sample is in the left column, with the four prototypes sorted by similarity scores in the right columns. Test samples are categorized based on prototypes that have maximum similarity.

Interpretable3D maps point clouds to features, the decision-making mode (Rudin et al. 2022) is straightforward for users.

	PE1	PE2	PO1	PO2	PointMLP OA(%)	PointNeXt OA(%)	Train Speed ↑	Test speed ↑
Random Init					68.0±1.5	70.4±1.2	422.8	1441.0
PE1	✓				80.3±0.5	84.2±0.4	97.6	1441.0
PE2		✓			81.2±0.3	84.8±0.4	228.3	
PE1 + PO1	✓		✓		83.6±0.2	85.5±0.2	92.8	
PE1 + PO2	✓			✓	84.1±0.1	86.4±0.3	91.3	
PE2 + PO1		✓	✓		85.2±0.2	87.0±0.1	223.8	
PE2 + PO2		✓		✓	**85.6±0.1**	**88.0±0.2**	219.5	

Table 5: Study of Prototype Estimation/Optimization process on ScanObjectNN (Uy et al. 2019). PE1, PE2, PO1, PO2 refer to Sinkhorn-Knopp algorithm, APDAGD, Eq. (7), Eq. (8), respectively. We report the speed by throughput (instances per second).

Because it provides explanations for its decisions, humans can trust it and make decisions in high-stakes situations.

Understanding Prototypes and Data Distribution. Next, we use the MMD2 metric and the maximum witness value (Kim, Khanna, and Koyejo 2016; Molnar 2020) to quantitatively measure prototype distribution and test sample distribution. We use the cosine similarity as a kernel function to approximate the densities of prototypes and test samples. This is different from the implementation in (Kim, Khanna, and Koyejo 2016). Based on the kernel function, Maximum Mean Discrepancy (MMD) is a measure of the difference between two probability distributions. When the MMD2 value is close to zero, the prototype distribution is well-suited for the dataset. This is because the prototypes are evenly distributed among the test samples within the latent space. Additionally, the witness function quantitatively represents the relationship between each test sample and all the prototypes. We treat the test sample with the maximum witness value (*i.e.*, Witness$_{max}$) as a criticism. Specifically, criticisms are data points where the distribution of prototypes and data diverges.

On the top of PointNeXt, we adapt ProtoPNet (Chen et al. 2019) to 3D vision for comparison. Following (Chen et al. 2019), ProtoPNet employs three combined models and relies on voting for results (6000 prototypes in total). Moreover, two Interpretable3D models, PointNeXt$_{\mathtt{PE}}$ and PointNeXt$_{\mathtt{PO}}$ (600 prototypes), are trained on ModelNet40. PointNeXt$_{\mathtt{PE}}$ is trained with Prototype Estimation, and PointNeXt$_{\mathtt{PO}}$ is trained with Prototype Estimation and Optimization. As shown in Table 4, PointNeXt$_{\mathtt{PO}}$ exhibits better performance in both MMD2 and OA compared to PointNeXt$_{\mathtt{PE}}$ and ProtoPNet. PointNeXt$_{\mathtt{PO}}$ and PointNeXt$_{\mathtt{PE}}$ surpass ProtoPNet with significantly fewer prototypes. This shows that our method learns more representative prototypes; we don't need to assemble multiple models to enlarge the representative capability. Comparing PointNeXt$_{\mathtt{PO}}$ and PointNeXt$_{\mathtt{PE}}$, the better performance of PointNeXt$_{\mathtt{PO}}$ can be attributed to the fact that the LVQ-type algorithm optimizes the distribution of prototypes, aligning them more closely with the original distribution of the training data. In other words, it is also understandable for humans that Prototype Optimization gives a "penalty" or "reward" signal to prototypes based on object labels, making the prototypes closer to the density distribution of the training data. A test sample can be identified as a criticism if it has the largest witness value, indicating that it deviates the most from the prototype distribution. The smaller Witness$_{max}$ value of PointNeXt$_{\mathtt{PO}}$ reflects that all test samples fit the prototype distribution better. The performance of PointNeXt$_{\mathtt{PE}}$ and PointNeXt$_{\mathtt{PO}}$ is also reported based on typical samples rather than average embeddings on ModelNet40. What's even more remarkable is that PointNeXt$_{\mathtt{PO}}$ outperforms the softmax-based black-box PointNeXt (94.0% OA and 91.1% mAcc) while enhancing interpretability.

Ablation and Analysis

To further analyze the core components in Interpretable3D, we conduct ablation studies on the ScanObjectNN (Uy et al. 2019) test set. We set two baselines: PointMLP and PointNeXt. In Table 5, the mean results from three random runs are reported. All the results are obtained without voting strategies. The OA metric for the baselines (random initialization for prototypes) are 68.0% and 70.4%, respectively. Performance improvement is observed when both Prototype Estimation and Optimization are employed. However, the best results come from combining both processes, yielding 85.6% and 88.0%. This indicates that Prototype Estimation and Optimization have the potential to achieve superior performance by modifying the prototype feature space and preserving the most prototypical examples. Furthermore, the reported training and testing speeds confirm the high efficiency of our approach.

Conclusion

We developed an *ad-hoc* interpretable classifier, *i.e.*, Interpretable3D, specifically designed for point cloud classification and part segmentation tasks. By performing Prototype Estimation and Optimization, Interpretable3D benefits from a reshaped prototype feature space and the preserved typical prototypes. The revealed decision-making mode enables users to understand how the system works and how decisions are made. Prototype Optimization further enhances the interpretability by illustrating how prototypes can be modified in a manner easily comprehensible for humans. Our algorithm consistently produces promising results across three datasets. In the future, our will explore the application of this *ad-hoc* style Interpretable3D to other 3D tasks.

Acknowledgments

This work was supported by China Scholarship Council (CSC) scholarship.

References

Akhtar, N.; and Jalwana, M. A. A. K. 2023. Rethinking interpretation: Input-agnostic saliency mapping of deep visual classifiers. In *AAAI*.

Angelov, P.; and Soares, E. 2020. Towards explainable deep neural networks (xDNN). *Neural Networks*, 130: 185–194.

Arrieta, A. B.; Díaz-Rodríguez, N.; Del Ser, J.; Bennetot, A.; Tabik, S.; Barbado, A.; García, S.; Gil-López, S.; Molina, D.; Benjamins, R.; et al. 2020. Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI. *Information fusion*, 58: 82–115.

Asano, Y.; Rupprecht, C.; and Vedaldi, A. 2019. Self-labelling via simultaneous clustering and representation learning. In *ICLR*.

Biehl, M.; Hammer, B.; and Villmann, T. 2016. Prototype-based models in machine learning. *Wiley Interdisciplinary Reviews: Cognitive Science*, 7(2): 92–111.

Bishop, C. M.; and Nasrabadi, N. M. 2006. *Pattern recognition and machine learning*, volume 4. Springer.

Chen, C.; Li, O.; Tao, D.; Barnett, A.; Rudin, C.; and Su, J. K. 2019. This looks like that: deep learning for interpretable image recognition. In *NeurIPS*.

Chen, Q.; Li, P.; Xu, M.; and Qi, X. 2021. Sparse Activation Maps for Interpreting 3D Object Detection. In *CVPR*.

Cuturi, M. 2013. Sinkhorn distances: Lightspeed computation of optimal transport. In *NeurIPS*.

Fan, F.-L.; Xiong, J.; Li, M.; and Wang, G. 2021. On interpretability of artificial neural networks: A survey. *IEEE Transactions on Radiation and Plasma Medical Sciences*, 5(6): 741–760.

Feng, T.; Wang, W.; Wang, X.; Yang, Y.; and Zheng, Q. 2023. Clustering based point cloud representation learning for 3d analysis. In *ICCV*.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *CVPR*.

Hong, J.-H.; Nam, W.-J.; Jeon, K.-S.; and Lee, S.-W. 2023. Towards Better Visualizing the Decision Basis of Networks via Unfold and Conquer Attribution Guidance. In *AAAI*.

Hu, B.; Tunison, P.; RichardWebster, B.; and Hoogs, A. 2023. Xaitk-Saliency: An Open Source Explainable AI Toolkit for Saliency. In *AAAI*.

Kim, B.; Khanna, R.; and Koyejo, O. O. 2016. Examples are not enough, learn to criticize! criticism for interpretability. In *NeurIPS*.

Knight, P. A. 2008. The Sinkhorn–Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 30(1): 261–275.

Knowlton, B. J.; and Squire, L. R. 1993. The learning of categories: Parallel brain systems for item memory and category knowledge. *Science*, 262(5140): 1747–1749.

Kohonen, T. 1990. Improved versions of learning vector quantization. In *IJCNN*.

Kohonen, T. 2012. *Self-organizing maps*, volume 30. Springer Science & Business Media.

Laugel, T.; Lesot, M.-J.; Marsala, C.; Renard, X.; and Detyniecki, M. 2019. The dangers of post-hoc interpretability: unjustified counterfactual explanations. In *IJCAI*.

Li, O.; Liu, H.; Chen, C.; and Rudin, C. 2018. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*.

Liang, J. C.; Zhou, T.; Liu, D.; and Wang, W. 2023. CLUSTSEG: Clustering for Universal Segmentation. In *ICML*.

Lin, T.; Ho, N.; and Jordan, M. 2019. On efficient optimal transport: An analysis of greedy and accelerated mirror descent algorithms. In *ICML*.

Ma, X.; Qin, C.; You, H.; Ran, H.; and Fu, Y. 2021. Rethinking Network Design and Local Geometry in Point Cloud: A Simple Residual MLP Framework. In *ICLR*.

Meng, Q.; Wang, W.; Zhou, T.; Shen, J.; Jia, Y.; and Van Gool, L. 2021. Towards a weakly supervised framework for 3d point cloud object detection and annotation. *IEEE TPAMI*, 44(8): 4454–4468.

Min, Z.; Luo, Y.; Yang, W.; Wang, Y.; and Yang, Y. 2023. Entangled View-Epipolar Information Aggregation for Generalizable Neural Radiance Fields. *arXiv preprint arXiv:2311.11845*.

Molnar, C. 2020. Interpretable machine learning. https://christophm.github.io/interpretable-ml-book/.

Nova, D.; and Estévez, P. A. 2014. A review of learning vector quantization classifiers. *Neural Computing and Applications*, 25(3): 511–524.

Phan, A. V.; Le Nguyen, M.; Nguyen, Y. L. H.; and Bui, L. T. 2018. Dgcnn: A convolutional neural network over large-scale labeled graphs. *Neural Networks*, 108: 533–543.

Qi, C. R.; Litany, O.; He, K.; and Guibas, L. J. 2019. Deep hough voting for 3d object detection in point clouds. In *ICCV*.

Qi, C. R.; Su, H.; Mo, K.; and Guibas, L. J. 2017a. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*.

Qi, C. R.; Yi, L.; Su, H.; and Guibas, L. J. 2017b. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*.

Qian, G.; Li, Y.; Peng, H.; Mai, J.; Hammoud, H.; Elhoseiny, M.; and Ghanem, B. 2022. PointNeXt: Revisiting PointNet++ with Improved Training and Scaling Strategies. In *NeurIPS*.

Ritter, H.; Martinetz, T.; Schulten, K.; et al. 1992. *Neural computation and self-organizing maps: an introduction*. Addison-Wesley Reading, MA.

Rosch, E. 1975. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3): 192.

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

Rudin, C.; Chen, C.; Chen, Z.; Huang, H.; Semenova, L.; and Zhong, C. 2022. Interpretable machine learning: Fundamental principles and 10 grand challenges. *Statistics Surveys*, 16: 1–85.

Rudin, C.; and Radin, J. 2019. Why are we using black box models in AI when we don't need to? A lesson from an explainable AI competition. *Harvard Data Science Review*, 1(2): 1–9.

Schinagl, D.; Krispel, G.; Possegger, H.; Roth, P. M.; and Bischof, H. 2022. OccAM's Laser: Occlusion-Based Attribution Maps for 3D Object Detectors on LiDAR Data. In *CVPR*.

Tan, H. 2023. Visualizing Global Explanations of Point Cloud DNNs. In *WACV*.

Taylor, J. R. 2003. *Linguistic categorization*. OUP Oxford.

Uy, M. A.; Pham, Q.-H.; Hua, B.-S.; Nguyen, T.; and Yeung, S.-K. 2019. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In *ICCV*.

Wang, W.; Han, C.; Zhou, T.; and Liu, D. 2022. Visual Recognition with Deep Nearest Centroids. In *ICLR*.

Wang, X.; Zheng, Z.; He, Y.; Yan, F.; Zeng, Z.; and Yang, Y. 2023a. Progressive local filter pruning for image retrieval acceleration. *IEEE TMM*.

Wang, Y.; Zeng, Z.; Guan, T.; Yang, W.; Chen, Z.; Liu, W.; Xu, L.; and Luo, Y. 2023b. Adaptive Patch Deformation for Textureless-Resilient Multi-View Stereo. In *CVPR*.

Wu, Z.; Song, S.; Khosla, A.; Yu, F.; Zhang, L.; Tang, X.; and Xiao, J. 2015. 3d shapenets: A deep representation for volumetric shapes. In *CVPR*.

Yan, X. 2019. Pointnet/Pointnet++ Pytorch.

Yi, L.; Kim, V. G.; Ceylan, D.; Shen, I.-C.; Yan, M.; Su, H.; Lu, C.; Huang, Q.; Sheffer, A.; and Guibas, L. 2016. A scalable active framework for region annotation in 3d shape collections. *ACM TOG*, 35(6): 1–12.

Yin, J.; Fang, J.; Zhou, D.; Zhang, L.; Xu, C.-Z.; Shen, J.; and Wang, W. 2022a. Semi-supervised 3D object detection with proficient teachers. In *ECCV*.

Yin, J.; Zhou, D.; Zhang, L.; Fang, J.; Xu, C.-Z.; Shen, J.; and Wang, W. 2022b. Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. In *ECCV*.

Zhang, Q.; Rao, L.; and Yang, Y. 2021. A novel visual interpretability for deep neural networks by optimizing activation maps with perturbation. In *AAAI*.

Zheng, T.; Chen, C.; Yuan, J.; Li, B.; and Ren, K. 2019. Point-cloud saliency maps. In *ICCV*.

Zheng, Z.; Ruan, T.; Wei, Y.; Yang, Y.; and Mei, T. 2020. VehicleNet: Learning robust visual representation for vehicle re-identification. *IEEE TMM*, 23: 2683–2693.

Zhou, T.; Wang, W.; Konukoglu, E.; and Van Gool, L. 2022. Rethinking semantic segmentation: A prototype view. In *CVPR*.

Zhu, L.; and Yang, Y. 2020. Label independent memory for semi-supervised few-shot video classification. *IEEE TPAMI*, 44(1): 273–285.

Ziwen, C.; Wu, W.; Qi, Z.; and Fuxin, L. 2020. Visualizing Point Cloud Classifiers by Curvature Smoothing. In *BMVC*.