An Improved eXplainable Point Cloud Classifier (XPCC)

Nicholas I. Arnold, Plamen P. Angelov *Fellow, IEEE*, Peter M. Atkinson

***Abstract*—Classification of objects from 3D point clouds has become an increasingly relevant task across many computer vision applications. However, few studies have investigated explainable methods. In this paper, a new prototype-based and explainable classification method called eXplainable Point Cloud Classifier (XPCC) is proposed. The XPCC method offers several advantages over previous explainable and non-explainable methods. First, the XPCC method uses local densities and global multivariate generative distributions. Therefore, the XPCC provides comprehensive and interpretable object-based classification. Furthermore, the proposed method is built on recursive calculations, thus, is computationally very efficient. Second, the model learns continuously without the need for complete re-training and is domain transferable. Third, the proposed XPCC expands on the underlying learning method, xDNN, and is specific to 3D. As such, three new layers are added to the original xDNN architecture: i) the 3D point cloud feature extraction, ii) the global compound prototype weighting, and iii) the SoftMax function. Experiments were performed with the ModelNet40 benchmark which demonstrated that XPCC is the only explainable point cloud classifier to increase classification accuracy relative to the base algorithm when applied to the same problem. Additionally, this paper proposes a novel prototype-based visual representation that provides model- and object-based explanations. The prototype objects are superimposed to create a prototypical class representation of their data density within the feature space, called the Compound Prototype Cloud. They allow a user to visualize the explainable aspects of the model and identify object regions that contribute to the classification in a human-understandable way.**

***Impact Statement*—The classification of 3D point cloud data has become a significant topic in recent years, in part because of the popularization of various unmanned robotics, augmented reality, and 3D mapping software. Such applications often involve decisions with direct consequences to individuals and society, yet very little research has been done towards explainable 3D classification algorithms. This paper proposes an inherently explainable prototype-based classification and visualization method for 3D point cloud objects. Experiments demonstrate that the proposed method is not only competitive with the state-of-the-art, but that it is also transferable and improves accuracy over the base algorithm.**

***Index Terms*—3D, AI, Classification, Deep Learning, Explainable, Point Cloud Data.**

February 8, 2022. This work was supported in part by the Cumbria Innovations Platform (CUSP) at Lancaster University.

Nicholas I. Arnold is with School of Computing & Communications Lancaster University, Lancaster University, Lancaster, UK (e-mail: n.arnold@lancaster.ac.uk).

Plamen P. Angelov is with School of Computing & Communications Lancaster University, Lancaster University, Lancaster, UK (e-mail: p.angelov@lancaster.ac.uk).

Peter M. Atkinson is with School of Computing & Communications Lancaster University, Lancaster University, Lancaster, UK (e-mail: pma@lancaster.ac.uk)

This paragraph will include the Associate Editor who handled your paper.

I. Introduction

CLASSIFICATION of 3D point cloud data has become an important research goal in response to the widespread adoption of 3D sensor technologies such as LiDAR and RGB-depth cameras. 3D point cloud data are a sparse collection of unordered coordinates in 3D space. They offer a fine-grained representation of real-world objects and accurately preserve intrinsic geometric 3D shape, surface and depth information. Point clouds have become an increasingly relevant data structure across a range of computer vision applications including remote sensing, autonomous driving, and robotics. In many cases such applications may have important real-world consequences (e.g., misperception of the environment may lead to the collision of an autonomous vehicle). Therefore, it is critical that a 3D object classification model is not only efficient in terms of accuracy and speed, but that the model's decisions can be understood intuitively and interpreted correctly by a human. In this paper, we propose a new prototype-based classification method called eXplainable Point Cloud Classifier (XPCC) for object classification of 3D point cloud objects. For clarity, this paper adopts the terminology supported by [1] to describe explainable machine learning models.

Early methods for 3D point cloud object classification rely on handcrafted features extracted directly from local neighborhood regions or through estimation of the surface around each point. For example, the Fast Point Feature Histogram (FPFH) [2] algorithm encodes the local geometric shape based on the normal angle between points and their neighbors, [3] uses binning to extend the FPFH into a global object descriptor and [4] builds a histogram of point locations summed along the bins of an accumulator constructed around each point to create an image. These features are designed to be invariant to shape transformations. Classification based on theses handcrafted features can then be achieved through classical supervised machine learning algorithms. These handcrafted features are explainable at a human level precisely because they were handcrafted to describe specific (local or global) properties of the shape. However, it is not trivial to find the most effective feature combination for a specific task. In this regard, the move to features learned through artificial neural network (NN) algorithms, such as convolutional neural networks (CNN) and other deep neural networks (DNN), was a breakthrough. The XPCC method benefits from the transfer learning paradigm [5] to incorporate learned features while retaining human interpretability.

Deep learning applied to 3D point clouds is far from straightforward, as point cloud data are unordered and non-

structured [6]. That is, there are no defined neighborhoods to connect each point in space. This contrasts with 2D images, where each pixel sits on a grid and has explicitly defined neighboring pixels. Recently however, deep learning classifiers, such as PointNet [7] and its derivatives [8], have been proposed and adapted to the specific properties of 3D point cloud data. These classifiers learn an embedding for each point and aggregate this information into a global shape descriptor. Classification is then achieved by feeding the global descriptor into several fully connected layers. By removing these last fully connected layers, the XPCC method uses a fixed pre-trained CNN to act directly on the point cloud data by extracting a global feature vector from the 3D point cloud objects. The choice of fixed feature extractor is not linked to any method in particular, it is therefore modular and can updated as research into DNN on point clouds progress.

Several characteristics of deep learning algorithms limit their wider real-world application. Firstly, DNNs are domain-specific; that is, they make classifications through learned properties as determined by the data on which they were trained. The addition of new classes or even additional data that do not follow the same statistical characteristics as the training data requires a complete retrain of the network. Secondly, training of DNNs is computationally demanding and require substantial numbers of training data, computational resources and time. While several 3D point cloud benchmark datasets have been published, the classes available are far from exhaustive. Therefore, training on atypical and uncommon classes is problematic. XPCC overcomes these limitations through both task domain transfer and learning domain transfer. Adding a new class requires only training the model on the new data samples, rather than a complete retrain, and the new classes are not required to be known to the feature extraction method. Furthermore, classification can be achieved with only a few training samples per class.

Deep learning methods, such as CNNs, involve and require a large number (millions or more) of model parameters (network weights) which have no direct link to the physicality of the problem. In addition, the architectures of DNNs such as CNNs involve several *ad hoc* decisions about the number and type of layers, stride and kernel size. Due to this complexity, and the opacity of the link between the inputs (point cloud coordinates) and the output (class label), such solutions are considered as 'black-box' [9]. The output is a multi-level embedded function (i.e., a function of a function of a function…) of inputs. This makes it difficult to explain the cause – effect relationship and the intuition of how the final decision is arrived at to a human user. A lack of transparency is a particular drawback in the case of 3D point cloud object classification. 3D point clouds are often used for real-world applications and the actions informed by the point cloud data have the potential to adversely affect results or endanger human life, if an incorrect decision is made (e.g., in self-driving cars). As algorithmic decisions become more consequential to individuals it becomes crucial that the algorithms are explainable in human terms. Efforts in explainable AI have focused on explaining deep learning methods [10], [11], but very little has been done to introduce explainability specifically to point-set learning on 3D point clouds.

This research builds on the image-based explainable deep neural networks (xDNN) framework [12] by extending it to object classification on 3D point sets. It is specific to point cloud data and offers several layers of human-interpretable explainability. By design, the XPCC internal architecture is algorithmically transparent, simple and, thus, easy to explain to a human user: the prototypes are the highly representative data samples and are learned incrementally after the first encounter of a specific class. The proposed method is non-iterative: instead, XPCC is an incremental, greedy learning algorithm that self-develops autonomously. It evolves the internal structure with the addition of new prototypes that reflect the changes of the data pattern represented by the local data density. In this study, we use a fixed KP-CNN, pre-trained on the ModelNet40 benchmark without limitation to the generality of the proposed concept. Experiments show that the proposed XPCC is not only explainable and computationally more efficient than the state-of-the-art in explainable point set deep learning classifiers, but also superior in terms of classification accuracy. To the best of our knowledge, XPCC is the only explainable point set classifier that achieves a higher overall accuracy compared to the benchmark deep networks used.

The main contributions of this paper are summarized as follows:

- A novel explainable point cloud classifier network is proposed that addresses the lack of transparent object classification algorithms for 3D point cloud data.
- A new prototype-based visual representation is proposed that explores explanations within the 3D space.
- An evaluation of the proposed classification network to improve classification accuracy over existing methods.

The rest of this paper is organized as follows. In section 2, a brief overview of relevant related work is provided. Section 3 details the proposed XPCC classifier and CPC method. Then, in section 4, we describe the experiments conducted and analyze the results. Finally, in section 5 we provide a conclusion.

II. Related Work

A. Deep Learning for Point Cloud Object Classification

Recently, deep learning classifiers were proposed that adapt to the properties of 3D point cloud data. These classifiers use error-correction to learn an embedding for each point and aggregate this information into a global shape descriptor. Hard classification is performed by feeding the global descriptor into several fully connected layers. Deep learning classifiers on point set data can be divided broadly into three types. The first of these is multi-view approaches, a technique pioneered by MVCNN [13] whereby the 3D object is projected into multiple 2D representations. However, it is difficult to design an efficient and robust strategy for choosing viewpoints. Recently, [14] utilized a graph convolutional network to optimize viewpoint sampling. The second type is volumetric-based methods. These methods divide the point cloud into voxels; for example, VoxNet [15] structures point cloud data into a volumetric occupancy grid as input to a 3D CNN. Originally, volumetric

methods were limited to point clouds with a relatively small number of points. However, octree structures have been used to reduce memory usage and increase the computational speed [16]. Nevertheless, volumetric approaches suffer from undesirable bias due to grid axis alignment and it is not clear if the advantages to processing 3D data directly in this manner are worth the additional overhead accrued [17]. The third type of classifiers is point-based methods. These methods are capable of learning directly on the point cloud structure without intermediate representations. This contrasts with the previous two types, where the point cloud data are converted and structured to apply mature 2D or 3D CNNs: a process that inherently results in information loss. Prominent network architectures for the point-based methods include graph convolution networks [18], [19], pointwise multi-layer perceptron (MLP) type NN [7], [8], and kernel point CNN [20], [21].

A main goal of the method proposed in this paper was to explore explanations uniquely possible within the 3D space. Therefore, the multi-view and volumetric approaches were not used for feature extraction. Instead, a pre-trained kernel point convolutional network, KP-CNN was used in the 3D feature extraction layer. Unlike grid convolution, the kernel point convolutions define continuous convolution kernels composed of a series of kernel points with weights. Specifically, the weights for neighboring points are related to the spatial distribution with respect to the center point; formulated as an optimization problem [20].

B. Explainable Deep Learning On Point Clouds

Previous literature on explainable deep learning on point cloud data focuses on techniques to better understand the representations learned by the network. [7], [20], [22] demonstrated how to visualize information learned by the point-cloud based NN through projecting back a coloring based on the level of activation of the point functions onto the input point cloud. Additionally, [7] used *t*-SNE to embed point cloud global features into a 2D space and visualize the correlation between the point clouds. [22] modified the PointNet network to create class-attentive mappings and specified in [23] the model agnostic 3DCAM; however, these representations are not always intuitive to non-experts. [24] proposed a two-stage method of local-to-global attributes for explainable point cloud classification. Specific to kernel point-based methods, in [20] the Effective Receptive Field is computed as the gradient of kernel point responses to measure the influence of each input point in relation to the result at a particular location. PointMask [25] introduces a differentiable layer before the encoder that learns to mask out points by maximizing mutual information between masked points and the class labels. [26] demonstrated visualizations using kernel correlation as an affinity measure between two different point sets: the neighboring points and kernel points. These explanation techniques are limited primarily to *post-hoc* interpretations and, in comparison to their base architecture, the explainability negatively impacts classification accuracy.

In contrast to the above, the proposed method is explainable by design, and an overall increase in accuracy over deep learning methods. Furthermore, previous explainable point set classification methods do not use the 3D medium itself, beyond color visualization, for explanations. The proposed CPC is a novel approach to using the inherent nature of 3D to offer explication not possible in 2D.

C. Prototype Learning

Prototype-based methods learn a set of highly representative samples (i.e., the prototypes) which themselves represent the probability distribution in the feature space [27]. One or more prototypes represent each class in the dataset, with new samples assigned to a class based on a similarity metric. Prototype-based models have long demonstrated their high efficiency and versatility in classification problems and have an obvious interpretation. These models can be manipulated through the addition, removal or adaptation of prototypes. This makes them well suited for incremental learning. Examples of well-known prototype-based approaches are the learning vector quantization model [28], the radial basis function (RBF) network [29], Gaussian Mixtures, and the self-organizing map (SOM) [30] model. Additionally, *k*-means, support vector machine (SVM), and particle filtering (Sequential Monte Carlo methods), may be considered as kinds of prototype-based methods. The *k*-nearest neighbors (kNN) algorithm is related to many prototype-learning methods. However, because the classic technique for *k*NN stores all data rather than selective exemplars it is only loosely considered a prototype-based method and is, strictly speaking, not a learning method. The XPCC can, like other prototype learning methods, be viewed as a type of feedforward NN. In particular, it is similar to SOM in that the proposed method does not use error-correction learning. Instead, a type of greedy competitive learning is applied. XPCC is based on local densities and empirically-derived global multivariate generative distributions.

III. Proposed Method

In the following sections, we consider the input to the XPCC classification method as a set of $\mathcal{N}$ point cloud objects, $\mathcal{O} = \{\mathcal{O}_i \,|\, i = 1, \ldots, \mathcal{N}\}$, and the output is the set of predicted labels. Each object is a separate point cloud, and these, in turn, represent the shape of each object as a set of 3D (x, y, z) coordinates. Depending on the specific acquisition technology, the points may contain additional observed information such as color and intensity. The proposed method classifies based only on the objects' shape and, thus, only the coordinates are taken as the input. This means that the classification will not be affected by inconsistent color values between different objects. As such, object point cloud models are considered to contain only one entity. Functionally, point cloud objects do not need to have the same number of points.

A. Feature Extraction

Feature extraction encodes the global shape of each point cloud object into a global descriptor. To do this, the layers of a pre-trained CNN are used as a fixed feature extractor, and the global descriptor obtained as the feature vector computed

from the final fully connected layer. The set of feature vectors define a feature space that is optimised for the separation of the training objects. From the transfer learning concept, it is assumed that this feature space can also effectively separate objects from a different domain. The features extracted by the DNN are denoted as $X = \mathrm{DNN}(\mathcal{O}) \in \mathbb{R}^{\mathcal{N} \times d_f}$, where $\mathrm{DNN}(\cdot)$ is the fixed DNN and d_f is the dimension of the extracted features. In the case of the KP-CNN, $d_f = 1024$. We use $\mathbf{x}_i$ to refer to an object represented as it's feature vector extracted by the fixed DNN.

The point-based KP-CNN with rigid Kernel Point Convolution (KPConv) blocks [20] pre-trained on the ModelNet40 [31] dataset is used as the feature extractor to produce a 1-by-1024 dimensional global feature vector per object. It incorporates a method to perform downscaling or upscaling to the input point clouds, as required, so that they do not need to have the same number of points. This also means that the network remains robust to varying point densities, particularly across objects from different scenes or scanning technologies. Furthermore, it has been shown that the KP-CNN identifies simple geometric structures (lines, planes, and spherical regions) at lower layers of the network, and more complex characteristics at further layers [20]. Not only does this provide some transparency to the CNN, but is also an important indication of generalizing to object types that are not in the CNN training data.

The feature vectors are individually scaled to their unit norm (i.e., $L2$ normalization is performed on each element):

$$\mathbf{x}_i = \frac{\mathbf{x}_i}{\max(\|\mathbf{x}_i\|, \epsilon)}, \tag{1}$$

where ϵ is a small constant.

B. Training

Training the XPCC starts by performing a filtering operation where the prototypes are identified from the training data. This is done directly through a non-iterative 'one pass' process; the prototypes themselves are the most representative training samples belonging to a particular class. Thus, meta-parameters for the XPCC are trained per-class: all the calculations are performed separately for each class and can be performed simultaneously (in parallel). Each classes' parameters are initialized with the first observation sample of that class,

$$k \leftarrow 1, M \leftarrow 1, \mu \leftarrow x_1, p_1 \leftarrow x_1, N_1 \leftarrow 1, r_j \leftarrow r^* \tag{2}$$

where k is the current instance (number of training samples seen), M is the number of prototypes identified for a class, μ is the recursive global mean of all data samples as yet observed, p_1 is the first prototype $\{p_j \,|\, j = 1, \ldots, M\}$, N_j is the number of member points around each prototype, and r_j is the radius of the area of influence of the corresponding prototype. r^* is the initial degree of similarity of the prototype member space and is defined as:

$$r^* = \sqrt{2\left(1 - cos\left(30^\circ\right)\right)} = \left\| \frac{\mathbf{x}_i}{\|\mathbf{x}_i\|} - \frac{p_i}{\|p_i\|} \right\| \tag{3}$$

After initialization, the parameters of each class are updated recursively, absorbing only the samples that belong to them. The following pseudo-code demonstrates the process for each new training sample of the class type. First, the instance count is updated through:

$$k \leftarrow k + 1, \tag{4}$$

then μ is updated as follows:

$$\mu \leftarrow \frac{\mu\,(k-1) + \mathbf{x}_i}{k}. \tag{5}$$

Data samples (i.e., the objects) that are closer to the global mean have higher density values. Therefore, the data density indicates how strongly data samples influence one another in the data space. The density function is defined as a Cauchy function [32]:

$$D\left(x_i\right) = \frac{1}{\frac{1+\|\mathbf{x}_i - \mu\|^2}{\sigma}} \tag{6}$$

where μ is the global mean and $\sigma = 1 - \|\mu\|^2$.

The prototypes are determined by partitioning the labeled training data based on the data density and area of influence within the latent feature space. The prototypes are the local peaks of the data density in the feature space for their corresponding class. It is important to note that the prototypes are independent from each other, such that the addition of a new prototype does not influence the already existing prototypes.

$$\begin{aligned} \alpha &= \max D(p_j) \\ \beta &= \min D(p_j) \\ j^* &= \arg\min(\mathbf{x}_i - p_j) \end{aligned} \tag{7}$$

$$\begin{gathered} \textbf{IF } D(\mathbf{x}_i) > \alpha \textbf{ OR } D(\mathbf{x}_i) < \beta \textbf{ OR } \left(\|\mathbf{x}_i - p_{j^*}\| \leq r_{j^*}\right) \\ \textbf{THEN } \text{add a new prototype} \end{gathered} \tag{8}$$

The prototypes with maximum and minimum density, α and β respectively, and the index j^* which denotes the prototype closest to the current sample, are used to control the addition of new prototypes [32]. If either of the first two conditions in (8) is met, or if the sample lies outside the area of influence of the closest prototype (the third condition in (8)), then the new data sample is added as a new prototype to its respective class. If the conditions are not met, then new samples are assigned as a support member of the prototype nearest in the feature space.

The prototype is then updated recursively as follows [32]:

$$p_{j^*} \leftarrow \frac{p_{j^*}\,(N_j - 1) + \mathbf{x}_i}{N_j}. \tag{9}$$

The support, or the number of data samples associated with a certain prototype, is updated $S \leftarrow S + 1$ and the radius is updated recursively using (10) [32].

$$r_{j^*} \leftarrow \frac{r_{j^*} + (1 - p_{j^*})}{2} \tag{10}$$

After the initial training process, the model can learn continuously by absorbing new training samples of previously seen classes or of new unseen classes. Alternatively, a user can manipulate and fine-tune the model through the addition, removal, or adaption of prototypes manually.

Explanation of prototypes can be represented in the form of linguistic logical IF...THEN rules where the density D can

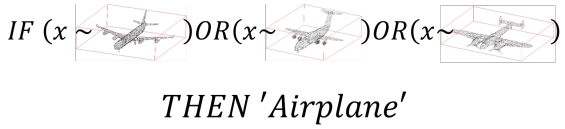

Fig. 1. Visual illustration of linguistic IF…THEN rules, where $\sim$ stands for similarity and x is the queried sample. If the sample is within the degree of similarity for the set of prototypes belonging to a class, then the class label is applied.

be seen as a fuzzy degree of membership [12]. All rules, per class, can be combined using the logical OR operator (see Fig.1).

C. Compound Prototype Cloud

After training, the XPCC model contains several class containers, one per class (n.b. that the containers are referred to as 'data clouds' in [12]. However, we call them class containers to avoid confusion with 'point clouds'). Each container consists of the per-class meta-parameters, including the prototypes identified for that class. The number of prototypes is much less than the number of training samples of that type that were seen (i.e., $P \ll N$). The prototypes themselves represent the 3D point cloud object. The CPCs are composed of all point cloud-based prototypes superimposed and, as a result, one such aggregated prototype per class. The CPC creation process is as follows. Starting with the first identified prototype as the reference object, perform principal component analysis to obtain the three main directions of the point cloud and deduce the main axis. Second, take the extent along the main axis to scale the clouds to the reference (reference objects are chosen as the initial prototype). Then, perform a fine registration using a point-to-point iterative closest point algorithm [33].

D. Classification Algorithm

In this section, we describe the classification procedure for the XPCC method. The principle of the XPCC classification approach is based on the intuition that people learn by comparing similarity between objects, but only remember a few distinct objects during decision-making (i.e., the prototypes) - the so-called anthropomorphic approach to machine learning [34]. If a new object is encountered, a person is likely to assume that it belongs to the class which it most closely resembles. Following this logic, the learning of the proposed method revolves around the position and properties of the prototypes in the feature space.

Given a new test sample $\mathbf{x}_t$, the proposed XPCC method first finds the local similarity to each class's prototypes:

$$S_j = \text{Similarity}(\mathbf{x}_t, p_j) = \frac{1}{(1 + \|\mathbf{x}_t - p_j\|^2)} \tag{11}$$

and then determines the global similarity to each class as $\text{S}_j^* = \max \text{S}_j$. With these, the proposed method then performs a global weighting (13), where $\odot$ is the Hadamard product, based on similarity between the new sample and the CPC feature vector. The CPC feature vector is estimated as the recursively updated global mean of the class, that is,

$$S_{\text{CPC}} = \text{Similarity}(\mathbf{x}_i, \text{CPC}_j). \tag{12}$$

$$\gamma_i = \text{S}_{\text{CPC}} \odot \text{S}_j^* \tag{13}$$

In the proposed method, the SoftMax function (14) is used to normalize the output of the previous layer to a probability distribution over the predicted output classes.

$$\gamma_i^* = \frac{\exp(\gamma_i)}{\sum_k \exp(\gamma_k)} \tag{14}$$

Finally, the hard classification is conducted through the $\arg\max(\gamma_i^*)$ function, thus providing the label of the most likely class.

IV. Experiments

We compared the proposed XPCC classifier against both classical machine learning classifiers and state-of-the-art point set learning classifiers. Unless stated otherwise, the term XPCC refers specifically to the use of the proposed method with the KP-CNN feature extractor network. In the experiment comparing the proposed method against classical machine learning algorithms, the

$$\text{Precision} = TP/\left(TP + FP\right),$$
$$\text{Recall} = TP/\left(TP + FN\right),$$
$$\text{and F1-score} = 2\frac{(Precision * Recall)}{(Precision + Recall)},$$

were used as evaluation metrics, where TP, FP and FN are the number of true positives, false positives and number of false negatives, respectively. For all experiments we also give the overall accuracy,

$$OA = \frac{TN + TP}{TP + FP + TN + FN},$$

where TN is true negatives. For our experiments comparing the proposed method against point set deep learning algorithms, we also use Mean Accuracy ($mAcc$),

$$mAcc = \frac{1}{M}\sum_{i=1}^{M} acc_i,$$

where acc_i is the accuracy for the samples class i and M is the total number of samples.

The Princeton ModelNet project provides a collection of synthetic 3D CAD object models split into two benchmarks: a 40-class subset and 10-class subset known as ModelNet40 and ModelNet10, respectively. We believe ModelNet is the only publicly available 3D object benchmark specific to object classification. Other 3D model and point cloud datasets exist, but they are either a collection of objects without a defined test/train split [35], have very few models per class [36] or are intended for different tasks like semantic segmentation [37], [38], [39] or 3D object detection [40]. The ScanObjectNN [41] benchmark is an interesting dataset made up of real-world object models; but at the time of writing, it is not available for public access. It will be a topic of future consideration. To evaluate the XPCC we used the ModelNet40 shape classification benchmark. There are 12,311 CAD models from 40 categories of human-made objects, split into 9,843

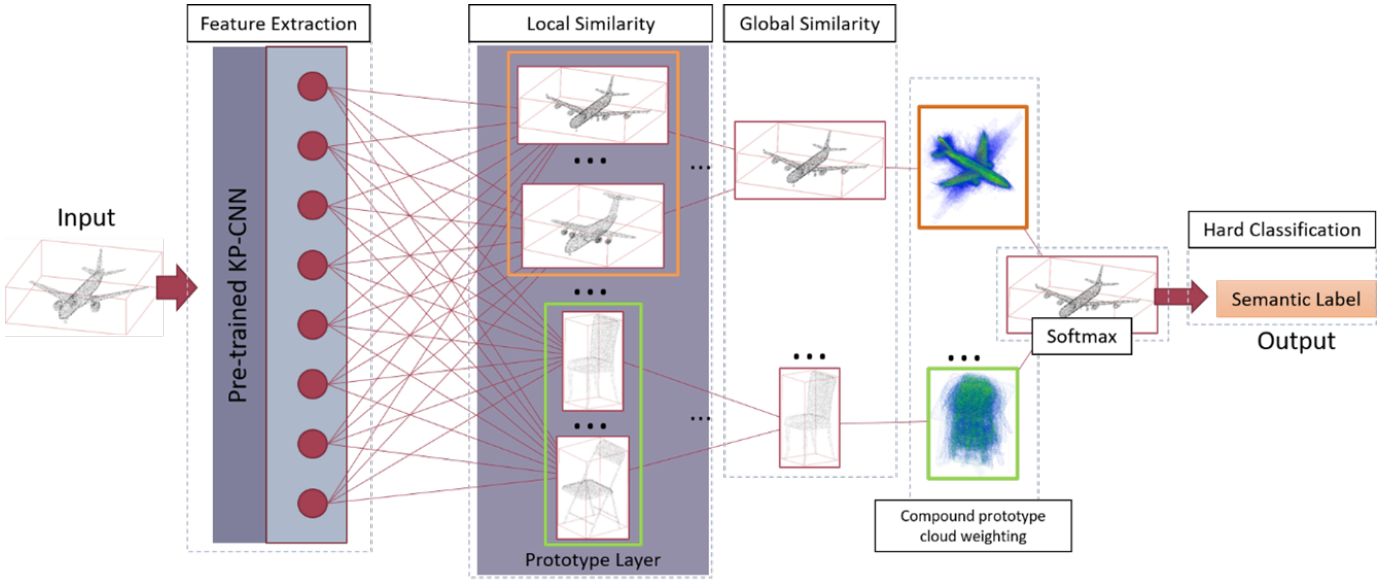

Fig. 2. The XPCC classification architecture represented as network layers. First, the point cloud object is input to the Feature Extraction layer and the pre-trained CNN is used to generate a global descriptor that encodes the global shape of the object into a feature vector. Second, the Local Similarity layer compares the object's global descriptor against the prototypes of each class. This is called the Prototype Layer in this figure and is represented by the colored rectangles within the local similarity layer. Third, the similarity score for each class is extracted by the Global Similarity layer as the classes' most similar prototype. Fourth, the classes' similarity score is weighted by the object's similarity to the respective classes' compound prototype. Last, the Softmax layer normalizes the output of the previous layer to a probability distribution over the predicted output classes. Hard classification is then performed on the output to produce the predicted class label.

for the training set and 2,468 for the testing set. The CAD models were converted into 3D point clouds by sampling points randomly along the model surfaces.

Experiments with the XPCC were run on a Unix machine with 3.6 GHz AMD Ryzen 5 3600 6-Core CPU, 16GB RAM and SSD. The KP-CNN feature extractor network was trained using an NVIDIA 2070S GPU. Training the KP-CNN took 5 hours. Once trained, the fixed network acts as a generalized point cloud feature extractor. In our comparison studies we also evaluated the PointNet++ network as a feature extractor; this was trained under the same conditions. The points are restricted to contain only the (x, y, z) coordinate information. Training of, and inference with, the XPCC is highly parallelizable and can be conducted on either CPU or GPU hardware.

A. CPC-Demonstration

The motivation for the CPC is to represent what the model has learned by visualizing the per-class parameters as a system of superimposed prototypes in 3D space. This is comparable to the idea of an object that a human might imagine when thinking of a particular object class. For example, if asked to think of an 'airplane', a cylindrical capsule shape with wings and tail rudder is likely to come to mind. Although this is the general shape, there are other aspects that might be different depending on the person's experiences with the object. For the airplane example, attributes such as the wing positions, angle or number might vary, and, additionally, the plane might have propellers or jet engines. Much like the human idea of an object, the CPC contains these aspects as physical options. We illustrate the CPC representation in Fig.3. By encoding the object's similarity (the normalized data density) in the feature space as a color, the CPC indicates the areas that contribute to the classifier's decisions when examining new objects. As can be seen in Fig.3, green and red regions are areas with greater density within the model's decision space and correspond to the areas of an object that contribute to the model's knowledge of that class and, ultimately, the decision-making process.

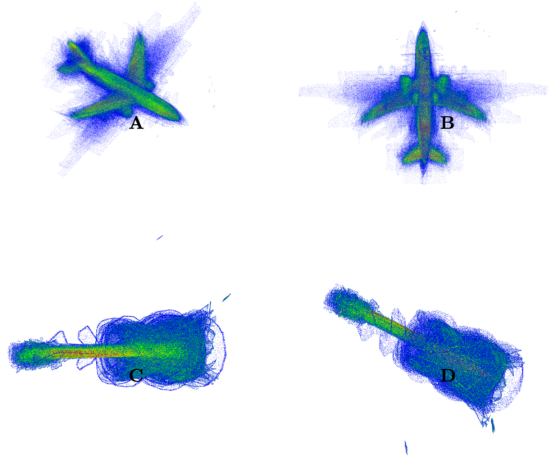

Fig. 3. Visual depiction of the compound prototype clouds for selected ModelNet40 classes. A and B are the CPC for class Airplane. C and D are the CPC for the class Guitar. The color represents the distribution of confidence per data point extracted from the data alone. Blue areas are those with low confidence, green areas are those with medium confidence, and red areas are those with high confidence.

TABLE I
PERFORMANCE COMPARISON WITH CLASSICAL CLASSIFIERS ON THE MODELNET40 BENCHMARK.

Method	Metric			
	Accuracy	Precision	Recall	F1
*k*NN (brute-force) [42]	90.04	88.23	87.64	87.78
L-SVM (hinge-loss) [43]	90.36	87.89	87.27	87.25
C-SVM (linear-kernel) [44]	91.17	88.16	88.51	88.67
C-SVM (poly-kernel) [44]	91.05	88.97	88.03	88.35
C-SVM (rbf-kernel) [44]	91.00	89.04	88.39	88.53
C-SVM (sigmoid-kernel) [44]	90.88	87.63	88.49	87.88
Decision Tree (gini) [42]	81.97	76.89	77.77	77.00
Decision Tree (entropy) [42]	78.72	73.67	73.15	72.75
Random Forest (gini) [42]	90.43	87.79	87.71	87.61
Random Forest (entropy) [42]	90.55	88.69	88.09	88.25
MLP (relu) [42]	90.15	87.38	87.05	86.93
MLP (tanh) [42]	90.07	87.86	87.81	87.69
MLP (sigmoid) [42]	90.76	87.62	88.32	87.79
Gaussian Naive Bayes [45]	87.76	84.78	84.91	84.44
XPCC (ours)	**91.82**	**92.10**	**91.82**	**91.83**

All classical methods were implemented with the scikit-learn [42] python library. The classical explainable classifiers are trained with the same feature vectors as XPCC, generated by the fixed-CNN feature extractor. Normalization was performed as described in (1) for all experiments.

B. Comparative Results with Classical Methods

In this section, we compared the proposed XPCC method against classical machine learning approaches to classification including: *k*NN, SVM, decision trees, random forest, multi-layer perceptron, and Gaussian Naive Bayes. All methods were implemented using the scikit-learn python library [42], however we cite the underlying algorithm or library if applicable. The *k*NN method is formulated as a *brute force* problem. Five variations of support vector machine classifier are reported. L-SVM is underpinned by [43] and utilizes squared-hinge loss to train one-vs-rest classifiers. Conversely, C-SVM is underpinned by [44] and trains one-vs-one classifiers; results are reported using the linear, polynomial (poly), radial basis function (rbf), and sigmoid kernels. Two variations of the decision tree and random forest algorithms are recorded, the first using the gini-impurity and the second using entropy-impurity. Additionally, results are compared against three variations of MLP. Each consist of the standard 3-layers configuration and use the ReLU, tanh, sigmoid activation functions, respectively. Optimization of the MLP is performed using [46]. Lastly, we compare against a Gaussian Naive Bayes classifier. Further specifications of the comparative classifier implementations can be found in [42].

From Table I, it is clear that the XPCC method achieved higher scores across all metrics in comparison to these classical approaches. The proposed method increased the overall accuracy by 0.65 percentage points (p.p.), increased F1-score by 3.16 p.p., and increased Recall by 3.30 p.p., compared to the subsequently leading C-SVM with linear-kernel method. Similarly, Precision was increased by 3.06 p.p. over the subsequently leading C-SVM with rbf kernel. The proposed method shares the ability to perform online machine learning with the Gaussian Naive Bayes method. A comparison of these two methods shows that the XPCC method increases accuracy by 4.06 p.p., precision by 7.32 p.p., recall by 6.91 p.p., and F1-score by 7.39 p.p.. Furthermore, the XPCC method does not incur performance or stability overhead when performing online learning because it is built on recursive calculations.

C. Comparative Results with State-of-the-art

We performed a classification test on the ModelNet40 benchmark and compared the XPCC classifier against the state-of-the-art in explainable point set learning algorithms, including PointHop, CLAIM, and PointMask. Additionally, we compared the proposed method against baseline deep learning algorithms including PointNet, PointNet++ and the base KP-CNN. The results from this experiment can be found in Table II.

Prior explainable point cloud classifiers use either PointNet or PointNet++ as their base method. These have an overall accuracy score of 89.2 p.p. and 90.7 p.p., respectively (as presented on the ModelNet40 benchmark rankings). However, compared to the base methods the mechanisms by which the explainable methods provide explanation result in a decrease in overall accuracy: PointHop incurs a 0.6 p.p. decrease from PointNet and 2 p.p. decrease from PointNet++, CLAIM sustains a 2.1 p.p. decrease from PointNet and 3.6 p.p. decrease from PointNet++, and PointMask experiences a 7 p.p. decrease from PointNet and 8.5 p.p. decrease from PointNet++. With our implementation of the KP-CNN, the base network alone achieved baseline accuracies of 91.80% (overall accuracy) and 88.75% (mean accuracy). In comparison with this baseline the XPCC produced on average an 0.02% increase to overall accuracy and an 0.12 p.p. increase to mean accuracy. Therefore, we believe XPCC to be the only explainable point set classifier that leads to an increase in accuracy relative to the base algorithm. In comparison with the previous explainable approaches, the proposed method increased classification accuracy by 2.7 p.p. (versus PointHop), 4.6 p.p. (versus CLAIM) and 9.64 p.p. (versus PointMask).

As a further experiment for accuracy, we reversed the test/train split (i.e., train on 2,468 samples and test on 9,843). The XPCC achieved an overall accuracy of 96.97% and mean accuracy of 95.46%. This would be far too few training samples to produce accurate results with the deep learning point-set learning methods. This demonstrates that the proposed method can achieve high accuracy with significantly less training data, although we do note that this particular experiment negates any difficulties purposefully encoded within the test set of the benchmark.

D. Analysis

Improvements to benchmark results are often marginal and relying on these scores alone can sometimes ignore other aspects of an algorithm which may be equally beneficial. As such, we stress that accuracy is only one aspect of explainable classifiers. Specifically, it has been shown that scores over 91% on the ModelNet40 benchmark are sensitive to optimizations that do not necessarily transfer to real world experiments [17], [41]. To highlight this effect, we also pre-trained a PointNet++ network as feature extractor using the optimized data augmentation described in [17]. In terms of overall accuracy

TABLE II
PERFORMANCE COMPARISON AGAINST STATE-OF-THE-ART AND EXPLAINABLE POINT-SET DEEP LEARNING METHODS ON THE MODELNET40 BENCHMARK.

Method	OA	mAcc	Training Time	Device	# of parameters	Transparent	Reproducibility	Retraining
PointNet[7]	89.2	86.2	5+ h	GPU	3.5 M	No	No	Yes
PointNet++[8]	90.7	–	5+ h	GPU	1.5 M	No	No	Yes
PointHop[24]	88.65	83.3	∼20 m	CPU	–	Yes	No	Yes
CLAIM[47]	87.1	–	–	GPU	–	Yes	No	Yes
PointMask[25]	82.18	–	–	–	–	Yes	–	–
KP-CNN (rigid KPConv)[20]	91.80*	88.75	5+ h	GPU	14.3 M	No	No	Yes
xDNN[12]	89.42	86.37	∼7s	CPU	**P** × 2	Yes	Yes	No
XPCC & KPConv (ours)	91.82	**88.87**	**∼2s (GPU) ∼6s (CPU)**	CPU/GPU	**P** × 2	Yes	Yes	No
XPCC & PointNet++ (ours)	**92.18**	88.43	**∼2s (GPU) ∼4s (CPU)**	CPU/GPU	**P** × 2	Yes	Yes	No

The highest result for each accuracy metric listed is in bold, where OA is the overall accuracy and mAcc is the mean accuracy. Training time is the approximate time needed to train a method. The Device column reports the device typically required by the algorithm. In the number of parameters column, **P** is the number of prototypes; M is million. For both XPCC and xDNN there are two parameters (μ and σ) per prototype. The Transparent metric is conditional on the network's ability to be interpreted or if it is a 'black box' method. The Reproducibility metric is conditional on if, given the same training data, the network will always conclude the same predictions. The Retraining is conditional on if the network must be completely retrained in order to add a new training sample. Metrics that are not reported by or cannot be understood from the source literature are marked using the '–' notation.

* The authors of KPConv record an accuracy score of 92.9 in their publication, however we were unable to reproduce this score; this is not unique to our experience [41]. Our retrained KP-CNN achieves an average accuracy score of 91.8.

this formulation of the XPCC does perform better than the KPConv feature extractor on the benchmark's defined test/train split. However, the XPCC & PointNet++ configuration was less accurate under the mean accuracy metric which takes class balance into consideration. As shown in the Domain Transfer sub-section, the features extracted by the KP-CONN are demonstrated to be discriminative when applied generally. For this reason, we opted for the more sophisticated KP-CNN as feature extractor.

We list the number of parameters for all methods (if available) in Fig.II. The deep learning methods that rely on error-correction learning (e.g., backpropagation with gradient descent) have millions of parameters, a characteristic that extends to their derivative explainable methods. In comparison, the xDNN and XPCC methods have two parameters for each prototype, where the number of prototypes for our proposed method is around 10% of the number of training samples seen for a class.

The XPCC is significantly faster than all other methods. Training took on average 6 seconds on a CPU and 2 seconds on a GPU. This is thanks to the highly parallelizable structure of the model: all the calculations are performed separately for each class and, thus, can run simultaneously. In relation to the other methods this translates into the XPCC being at least 9000 times faster to train than methods that took 5 hours, and 600 times faster than methods that took 20 minutes to train. This, compounded with the fact that the XPCC does not need to be completely retrained to add new classification types (i.e., it learns continuously), demonstrates the efficiency and practical applicability.

E. Domain Transfer

The XPCC method incorporates two varieties of domain transfer: task-wise domain transfer and learning domain transfer.

Task-wise domain transfer is intrinsic to the method. The prototype-based internal structure of XPCC allows for a trained model to be transferred to a new domain without a complete retrain. Specifically, adding a new class requires training the model on only the new data samples, rather than a complete retrain. For example, if an XPCC model has been trained to classify chairs and tables, the model only needs to be updated with training samples of a new class, such as televisions. Similarly, the model does not need to forget previously learned classes when transferred.

To investigate the effectiveness of the learning domain transfer and the approach to transfer learning, we assessed the transferability of the fixed-CNN feature extractor in the XPCC method. This experiment was conducted using the ModelNet10 benchmark, by examining accuracy results when applied on classes of objects not seen by the feature extraction network. The fixed-CNN feature extractor network was trained without access to one of the object classes. Then, the XPCC model was trained on the full dataset (including that object class) and validated using the ModelNet10 train-test data split. In this way, only the XPCC model had any knowledge of the hidden class and, therefore, not reliant on the fixed-CNN having knowledge of that class of object. This process was systematically repeated for each class within the ModelNet10 benchmark, and the average accuracy score recorded. In this experiment, the XPCC & KP-CNN feature extractor achieved an overall accuracy score of 91.38% for the hidden classes. We performed the same experiment with the XPCC & PointNet++ feature extractor. This configuration achieved only an overall accuracy score of 89.96% for the hidden class, giving a 1.42 p.p. decrease in accuracy; these results suggest that the PointNet++ feature extraction method is not as discriminative when applied generally. For this reason, we selected the more sophisticated KP-CNN feature extractor, which makes the XPCC method more viable for training and classifying atypical 3D point cloud object model classes.

F. Error Analysis

From examining the similarity scores between negatively predicted samples and prototypes, it is clear there were some limitations to the proposed method. A primary bottleneck of

the method is discerning between objects that share many similar characteristics. As such, errors in the classification are predominately from objects with subjective classification labels, such as the distinction between a glass and vase or table and desk. Other errors were from samples whose shapes are similar, such as bench and sofa. This appears as a trend across explainable 3D methods on the ModelNet benchmarks and suggests that more contextual information is needed beyond geometric shape. Additionally, semantic limitations present within the ModelNet40 benchmark are arguably hard for a human to discern between. For example, shapes that have different semantic labels but are geometrically very similar, such as the difference between a 'flower pot' (with a plant in it) and a 'plant' (in a pot).

V. Conclusion

We proposed a new classification method, the explainable Point Cloud Classifier (XPCC), for object classification within 3D point clouds. The proposed method is algorithmically and structurally transparent, learns continuously, without the need to be completely retrained at the addition of new classes, and offers several layers of human-interpretable explainability. This paper also presents a novel technique to visualize the explainable aspects of the model, called Compound Prototype Clouds (CPC). The technique is unique to 3D point clouds and prototype-based learning and represents what the model has learned. Specifically, it identifies object regions which contribute to the classification. Experiments show that the proposed classifier method is computationally efficient, trainable on thousands of samples in seconds and is competitive with the state-of-the-art in point set deep learning classifiers in terms of classification accuracy. Furthermore, the proposed method is the only explainable point set classifier that achieves higher accuracy compared to the base network used. A limitation of the proposed method is that classification relies on point clouds containing only one object. In our future work, we will focus on applying the method to real world data and extending the method to other point-cloud specific objectives such as object detection within a scene.

References

[1] A. Barredo Arrieta, N. Díaz-Rodríguez, J. Del Ser, A. Bennetot, S. Tabik, A. Barbado, S. Garcia, S. Gil-Lopez, D. Molina, R. Benjamins, R. Chatila, and F. Herrera, "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI," *Information Fusion*, vol. 58, pp. 82–115, Jun. 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253519308103

[2] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D registration," in *2009 IEEE International Conference on Robotics and Automation*, May 2009, pp. 3212–3217, iSSN: 1050-4729.

[3] R. B. Rusu, G. Bradski, R. Thibaux, and J. Hsu, "Fast 3D recognition and pose using the Viewpoint Feature Histogram," in *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Oct. 2010, pp. 2155–2162, iSSN: 2153-0866.

[4] A. E. Johnson and M. Hebert, "Surface matching for object recognition in complex three-dimensional scenes," *Image and Vision Computing*, vol. 16, no. 9, pp. 635–651, Jul. 1998. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0262885698000742

[5] S. Bozinovski, "Reminder of the First Paper on Transfer Learning in Neural Networks, 1976," *Informatica*, vol. 44, no. 3, Sep. 2020, number: 3. [Online]. Available: https://www.informatica.si/index.php/informatica/article/view/2828

[6] M. Zaheer, S. Kottur, S. Ravanbakhsh, B. Poczos, R. R. Salakhutdinov, and A. J. Smola, "Deep Sets," *Advances in Neural Information Processing Systems*, vol. 30, 2017. [Online]. Available: https://papers.nips.cc/paper/2017/hash/f22e4747da1aa27e363d86d40ff442fe-Abstract.html

[7] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 77–85.

[8] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: deep hierarchical feature learning on point sets in a metric space," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., Dec. 2017, pp. 5105–5114.

[9] C. Rudin, "Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead," *Nature Machine Intelligence*, vol. 1, no. 5, pp. 206–215, May 2019. [Online]. Available: https://www.nature.com/articles/s42256-019-0048-x

[10] E. A. Soares, P. P. Angelov, B. Costa, M. Castro, S. Nageshrao, and D. Filev, "Explaining Deep Learning Models Through Rule-Based Approximation and Visualization," *IEEE Transactions on Fuzzy Systems*, pp. 1–1, 2020, conference Name: IEEE Transactions on Fuzzy Systems.

[11] W. Samek, A. Binder, G. Montavon, S. Lapuschkin, and K. Müller, "Evaluating the Visualization of What a Deep Neural Network Has Learned," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 11, pp. 2660–2673, Nov. 2017, conference Name: IEEE Transactions on Neural Networks and Learning Systems.

[12] P. Angelov and E. Soares, "Towards explainable deep neural networks (xDNN)," *Neural Networks*, vol. 130, pp. 185–194, Oct. 2020. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0893608020302513

[13] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view Convolutional Neural Networks for 3D Shape Recognition," *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015.

[14] X. Wei, R. Yu, and J. Sun, "View-GCN: View-Based Graph Convolutional Network for 3D Shape Analysis," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Seattle, WA, USA: IEEE, Jun. 2020, pp. 1847–1856. [Online]. Available: https://ieeexplore.ieee.org/document/9156567/

[15] D. Maturana and S. Scherer, "VoxNet: A 3D Convolutional Neural Network for real-time object recognition," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Hamburg, Germany: IEEE, Sep. 2015, pp. 922–928. [Online]. Available: http://ieeexplore.ieee.org/document/7353481/

[16] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-CNN: Octree-based Convolutional Neural Networks for 3D Shape Analysis," *ACM Transactions on Graphics*, vol. 36, no. 4, pp. 1–11, Jul. 2017, arXiv: 1712.01537. [Online]. Available: http://arxiv.org/abs/1712.01537

[17] A. Goyal, H. Law, B. Liu, A. Newell, and J. Deng, "Revisiting point cloud shape classification with a simple and effective baseline," 2021.

[18] Q. Xu, X. Sun, C.-Y. Wu, P. Wang, and U. Neumann, "Grid-GCN for Fast and Scalable Point Cloud Learning," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[19] L. Landrieu and M. Simonovsky, "Large-Scale Point Cloud Semantic Segmentation with Superpoint Graphs," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4558–4567, iSSN: 2575-7075.

[20] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. Guibas, "KPConv: Flexible and Deformable Convolution for Point Clouds," in *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2019, pp. 6410–6419, iSSN: 2380-7504.

[21] A. Boulch, "ConvPoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, vol. 88, pp. 24–34, May 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0097849320300224

[22] B. Zhang, S. Huang, W. Shen, and Z. Wei, "Explaining the PointNet: What Has Been Learned Inside the PointNet?" in *CVPR Workshops*, 2019.

[23] Y. Cao, M. Previtali, and M. Scaioni, "Understanding 3d Point Cloud Deep Neural Networks by Visualization Techniques," *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B2-2020, pp. 651–657, Aug. 2020. [Online]. Available: https://www.int-arch-photogramm-remote-sens-spatial-inf-sci.net/XLIII-B2-2020/651/2020/

[24] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo, "PointHop: An Explainable Machine Learning Method for Point Cloud Classification," *IEEE Transactions on Multimedia*, vol. 22, no. 7,

pp. 1744–1755, Jul. 2020, arXiv: 1907.12766. [Online]. Available: http://arxiv.org/abs/1907.12766

[25] S. A. Taghanaki, K. Hassani, P. K. Jayaraman, A. H. Khasahmadi, and T. Custis, "PointMask: Towards Interpretable and Bias-Resilient Point Cloud Processing," *arXiv:2007.04525 [cs]*, Jul. 2020, arXiv: 2007.04525. [Online]. Available: http://arxiv.org/abs/2007.04525

[26] Y. Shen, C. Feng, Y. Yang, and D. Tian, "Mining Point Cloud Local Structures by Kernel Correlation and Graph Pooling," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun. 2018, pp. 4548–4557, iSSN: 2575-7075.

[27] J. Bien and R. Tibshirani, "Prototype selection for interpretable classification," *The Annals of Applied Statistics*, vol. 5, pp. 2403–2424, 2011.

[28] T. Kohonen, J. Kangas, J. Laaksonen, and K. Torkkola, "Lvqpak: A software package for the correct application of learning vector quantization algorithms," in *[Proceedings 1992] IJCNN International Joint Conference on Neural Networks*, vol. 1, 1992, pp. 725–730 vol.1.

[29] Meng Joo Er, Shiqian Wu, Juwei Lu, and Hock Lye Toh, "Face recognition with radial basis function (RBF) neural networks," *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 697–710, May 2002, conference Name: IEEE Transactions on Neural Networks.

[30] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990, conference Name: Proceedings of the IEEE.

[31] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A deep representation for volumetric shapes," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2015, pp. 1912–1920.

[32] P. P. Angelov and X. Gu, *Empirical Approach to Machine Learning*, ser. Studies in Computational Intelligence. Springer International Publishing, 2019. [Online]. Available: https://www.springer.com/gp/book/9783030023836

[33] P. J. Besl and N. D. McKay, "A Method for Registration of 3-D Shapes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 14, no. 2, pp. 239–256, Feb. 1992. [Online]. Available: https://doi.org/10.1109/34.121791

[34] P. P. Angelov and X. Gu, "Toward Anthropomorphic Machine Learning," *Computer*, vol. 51, no. 9, pp. 18–27, Sep. 2018, conference Name: Computer.

[35] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu, "ShapeNet: An Information-Rich 3D Model Repository," *arXiv:1512.03012 [cs]*, Dec. 2015, arXiv: 1512.03012. [Online]. Available: http://arxiv.org/abs/1512.03012

[36] M. Deuge, A. Quadros, C. Hung, and B. Douillard, "Unsupervised feature learning for classification of outdoor 3D Scans," *Australasian Conference on Robotics and Automation, ACRA*, 2013.

[37] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuScenes: A Multimodal Dataset for Autonomous Driving," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[38] A. Dai, A. X. Chang, M. Savva, M. Halber, T. Funkhouser, and M. Nießner, "ScanNet: Richly-Annotated 3D Reconstructions of Indoor Scenes," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul. 2017, pp. 2432–2443, iSSN: 1063-6919.

[39] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d.net: A New Large-Scale Point Cloud Classification Benchmark," *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1/W1, pp. 91–98, May 2017. [Online]. Available: http://www.isprs-ann-photogramm-remote-sens-spatial-inf-sci.net/IV-1-W1/91/2017/

[40] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The KITTI vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 3354–3361, iSSN: 1063-6919.

[41] M. A. Uy, Q.-H. Pham, B.-S. Hua, D. T. Nguyen, and S.-K. Yeung, "Revisiting Point Cloud Classification: A New Benchmark Dataset and Classification Model on Real-World Data," in *International Conference on Computer Vision (ICCV)*, 2019.

[42] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[43] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "Liblinear: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, p. 1871–1874, jun 2008.

[44] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Trans. Intell. Syst. Technol.*, vol. 2, no. 3, may 2011. [Online]. Available: https://doi.org/10.1145/1961189.1961199

[45] H. Zhang, "The optimality of naive bayes," in *FLAIRS Conference*, 2004.

[46] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2017.

[47] S. Huang, B. Zhang, W. Shen, and Z. Wei, "A CLAIM approach to understanding the PointNet," in *Proceedings of the 2019 2nd International Conference on Algorithms, Computing and Artificial Intelligence*, ser. ACAI 2019. Association for Computing Machinery, 2019, pp. 97–103. [Online]. Available: https://doi.org/10.1145/3377713.3377740