SHAX-PC

Adapting XAI methodology SHAP for point cloud data

University of Twente
Bachelor Creative Technology

Supervisor Faizan Ahmed
Critical Observer Bram Ton
2024

Marc F. Harinck, Bram Ton, and Faizan Ahmed

Adapting XAI methodology SHAP for point cloud data structures

Creative Technology, University of Twente

Abstract **This research created a custom SHAP framework designed to explain a PointNet model's prediction. By calculating the marginal contribution of an individual point with respect to a coalition towards the model's prediction. Individual points' SHAP value shows the level of contribution that point has towards the classification of a PointNet Model in the Model Net 10 dataset. This research successfully implements the custom SHAP framework and showcases which points contribute to the classification of a chair.**

1. ACKNOWLEDGEMENT

I want to thank my supervisor Dr. Faizan Ahmed for lending me his trust, time and expertise during this study. Faizan has guided me through this thesis and allowed me to make this significant discovery and introducing me to the world of data and XAI, a career path I see myself taking in the future. A sense of pride was felt during my presentation when Faizan said "Until last week I wasn't sure it was possible, now we know it is. You did a good job".

I also want to thank Ronald Oosting for lending his experiences, time and ears as I was able to use rubber duck debugging of my work and results and ensuring me to critique my own work and staying skeptical.

2. INTRODUCTION

It may seem counterintuitive to trust a prediction of a machine learning model. These models are commonly perceived as a "black box" system [1], as the absence of decision-making explanations leads to a lack in interpretability. Explainable AI (XAI) aims to clarify the "black box" nature of machine learning models [2] by incorporating transparency and interpretability into the models. Explainable AI (XAI) refers to the capacity of an artificial intelligence (AI) system or model to offer clear and comprehensible justifications for its choices [3]. The use of XAI for point cloud models is advantageous due to their inherent complexity and interpretability challenges [4]. Therefore, future studies in XAI in this sector are highly recommended. An example of social significance is the need for enhanced transparency in autonomous driving. This safety-critical application [5] needs comprehensive testing and validation to instil trust.

This research specifically addresses explainable artificial intelligence (XAI) techniques applied to point cloud data structures. The XAI model employed is Shapley Additive Explanations (SHAP), which has been modified to accommodate the data structure of a point cloud and the associated PointNet model. Which classifies point clouds into categories based on the given set of points. This study introduces a custom Explainable Artificial Intelligence (XAI) framework designed to explain the predictions made by the PointNet model. This study primarily investigates the following questions:

How to create a custom XAI framework which adapts SHAP values to a PointNet model and point cloud data?

The first section discusses background research addressing the concepts used in this research. The second section addresses the methodology, which explains the steps taken to achieve the result. The following section analyzes the results obtained form the methodology. The second to last and last sections are the discussion and conclusion respectively. In these sections the adequacy of the study and the results are discussed followed by concluding remarks.

3. BACKGROUND RESEARCH

3.1. Related work

This section provides an overview of the relevant research and is mostly based on two comprehensive literature studies. The section serves as a concise overview of the key subjects in XAI, providing reviews instead of comprehensive coverage of all elements.

Alejandro et. al. [6] examines the principles of several XAI models, such as Local Interpretable Model-Agnostic Explanations (LIME), Deep Learning Importance FeaTures (DeepLIFT), and SHAP. A wide range of models can use LIME and SHAP as versatile techniques. Tabular data types are frequently used for these models. Conversely, DeepLIFT specifically caters to natural language processing models, and computer vision models requiring text and images data.

LIME works by picking a small part of the model and a small part of the data and estimating them locally. By changing the data sample, it finds out how each characteristic affects the model subset, which leads to specific explanations for individual events instead of the whole model.

The DeepLIFT algorithm dissects a neural network's output prediction for a given input by propagating the influence of each neuron in the network backward to each feature of the input [1]. By comparing the activation levels of each neuron for the expected and actual predictions, this method determines the individual contribution of each feature to a prediction.

Grounded in the principles of cooperative game theory, SHAP clarifies the extent to which a feature contributes to a prediction based on cooperative game theory principles. This is a mechanism for providing explanations at a local level. Section 3 elaborates on the expansion of SHAP.

All three XAI models operate on a per-instance basis, focusing on individual data points rather than the entire dataset as a whole. SHAP and LIME apply to any type of model, whereas DeepLIFT is specifically designed for deep neural networks.

In a separate study, Clement et. al. [7] have extended Aljandro's et. Al. findings by exploring a broader range of explainable artificial intelligence (XAI) methodologies. The paper examines the concepts of Anchors & LORE and Permutation Feature Importance (PFI).

Anchors is an explainable artificial intelligence (XAI) technique that works with IF-THEN statements that are based on numerical ranges in the feature space and don't depend on the model. A greedy beam search algorithm achieves this. LORE is a model-agnostic tool that uses a genetic algorithm to generate synthetic neighbourhood data sets. It then trains a decision tree classifier and derives decision rules from it.

Permutation Feature Importance (PFI) is a technique that is independent of the model used and identifies predictive features by altering feature values and quantifying the effect on the model's prediction. When a feature's modification significantly impacts the model's performance, it is considered essential. Both XAI approaches enhance the ability to provide localized explanations for predictions and increase the interpretability of any model, regardless of its prior knowledge or assumptions.

3.2. Point cloud

A point cloud is a collection of individual data points which are coordinates in a three-dimensional Euclidean space [8] and stored as a unordered list. A Light Detection and Ranging (LiDAR) scanner can capture and analyze an item or environment to obtain a point cloud. It generates a three-dimensional coordinate map by producing pulses of infrared light and calculating the time it takes for the light to bounce back [9]. One major benefit of utilizing point clouds is the inclusion of depth information. Advanced computer vision systems can incorporate this information by integrating data from several cameras [10]. This added complexity is mitigated when using a single LiDAR scanner that collects depth information. Decreasing the complexity and processing required.

A point in a point cloud is obtained by sampling from a finite three-dimensional Euclidean space $\mathbb{R}^3$. A point is represented by a tuple $P_i(x_i, y_i, z_i) \in \mathbb{R}^3$, where $x_i, y_i, and\ z_i$ are the values of the point's coordinate a

point and point cloud can be plotted in a three-dimensional Cartesian coordinate systems for visualizations. Furthermore, a point can possess other information, such as a label or colour value and lacks any volume.

3.3. Machine learning with point clouds

This section provides an analysis of the application of machine learning to point clouds, the specific model utilized in this study, and the conceptual framework of the model.

Comprehending point cloud data without machine learning necessitates the expertise of skilled engineers and the time-consuming process of manually capturing and labelling data [9]. Machine learning has automated the processing stage that involves point clouds. This study primarily focuses on the PointNet model [4] because of its ability to handle unordered data, such as point clouds, without the need for a predefined structure like voxels.

The PointNet model utilizes a Convolutional Neural Network that emulates the cognitive processes of the human brain. A Convolutional Neural Network (CNN) specifically uses several interconnected layers to process visual input like photos or videos. Neural network-based image processing employs filters at each layer to evaluate and extract distinctive characteristics from pictures. A filter traverses the picture, performing element-wise multiplication between its values and the image's corresponding pixel values at every position. This results in the creation of a feature map. Subsequently, an activation function is employed to inject non-linearity into the network. Facilitating its ability to acquire more intricate patterns. To enhance computational efficiency and decrease the size of the feature map, pooling layers, such as max pooling, are employed to aggregate the values. The feature maps obtained are transformed into a single-dimensional vector by flattening them. This vector is then passed through fully connected layers that carry out the final classification. The output layer employs the softmax activation function to transform class scores into probabilities, hence yielding the ultimate classification outcome [4].

3.4. SHAP

The Shapley Additive eXplanations (SHAP) methodology assists in understanding machine learning models. Cooperative game theory's Shapley values serve as the foundation for the concept. The Shapley values, as described by Molnar [11], is “the weighted average of a player’s marginal contributions to all possible coalitions.” This section draws on Christoph Molnar's book "Interpreting Machine Learning Models With SHAP" [11] for its theories on SHAP and Shapley values.

The Shapley value is a method of allocating rewards in a coalition game where numerous players contribute to an outcome, but not all players participate equally. The Shapley values propose a method of distributing the payoff in a coalition game based on the participants weighted average marginal contributions.

$$\Phi_j(\mathrm{N}, v) = \sum_{S \subseteq \mathrm{N} \setminus \{\mathrm{j}\}} \frac{|S|!\,(N - |S| - 1)!}{N!} \big(v(\mathrm{S} \cup \{j\}) - v(S)\big) \tag{1}$$

$$j = Player\ index$$

$$N = \text{Set of players with (n) } players$$

$$S = coalition\ of\ any\ combination\ of\ players$$

Equation (1) gives the weighted marginal contribution of a single player for all coalition games. The equation contains three components.

1) The two terms $v(\mathrm{S} \cup \{j\})$ and $v(S)$ represent value functions of a collation including point *j* and the value of the coalition excluding point *j*. Subtracting them: $v(\mathrm{S} \cup \{j\}) - v(S)$ gives the marginal contribution of player *j* towards a coalition S.

2) The weight term $\frac{|S|!(N-|S|-1)!}{N!}$ determines the weight of the marginal contribution as it accounts for different coalition sizes.

3) The summation term $\sum_{S \subseteq \mathrm{N} \setminus \{\mathrm{j}\}}$, sums over all coalitions not containing the player of interest.

The marginal contribution provides a method for determining the distribution of a reward based on the individual contributions of players in a cooperative game. The difference between Shapley values and SHAP values is the inclusion of a machine learning model for the SHAP framework, translating the concepts of game theory to machine learning prediction concepts.

Equation (2) is the marginal contribution of the

$$\left(v(\mathrm{S} \cup \{\mathrm{j}\}) - v(S)\right) = \int f\left(x_{S\cup \mathrm{j}}^{(i)} \cup X_{C\setminus j}\right) d\mathbb{P}_{X_{c\setminus j}} - \int f\left(x_S^{(i)} \cup X_C\right) d\mathbb{P}_{X_c} \quad (2)$$

PointNet model and shows the contribution of point j towards coalition S. The terms $f\left(x_{S\cup \mathrm{j}}^{(i)} \cup X_{C\setminus j}\right)$ and $f\left(x_S^{(i)} \cup X_C\right)$ represent the value functions of the marginal contribution of point j towards coalition S and the marginal contribution of coalition S without point j respectively. $x_{\mathrm{S\cup j}}^{(i)}$ and $x_s^{(i)}$ is the known feature (point) value to the model and X_C and $X_{c\setminus j}$ are unknown feature (point) values which are randomly distributed according to distribution $\mathbb{P}_{\cdot}$.

Terms $\int d\mathbb{P}_{X_{c\setminus j}}$ and $\int d\mathbb{P}_{X_c}$ are the integral taken over the probability distribution $\mathbb{P}_{\cdot}$ for unknown features values X_C. Integrating over the distribution of a random variable is known as marginalization [11].

Calculating the interval for each value of a feature (point) is not possible for two primary reasons outlined by Molnar [11]. Firstly, the overall number of features in a model is excessively large since the number of coalitions grows exponentially with the number of features, namely 2^p.
The second major problem is that the feature distributions are unknown, namely the distributions of $P\left(X_{c\setminus j}\right)$ and $P(X_c)$. The book [11] advocates for the use of the Monte-Carlo approach to estimate the SHAP value.

Various approximation approaches can be used for the SHAP value function. Simon [12] describes a method which uses a projected stochastic gradient algorithm to estimate the SHAP values. The approach for approximation is chosen to be the Monte-Carlo approximation based on its ease of implementation and its alignment with the theoretical framework proposed by Molnar [11], yet Simon's method will be discussed in section 6.3. The Monte Carlo approximation operates by generating a substantial number of random samples from a given dataset. As the number of random samples increases, the average gradually approaches the expected value of the distribution. The background data utilized for sampling might consist of the identical data employed for training or validating the model. The Monte-Carlo approximation method substitutes an integral calculation with a summation and replaces the distribution P with data samples. The equation for this approximation is as follows:

$$\left(v(\mathrm{S} \cup j) - v(S)\right) \approx \frac{1}{m}\sum_{k=1}^{m}\left[f\left(x_{S\cup j}^{(i)} \cup X_{C\setminus j}^{k}\right) - f\left(x_S^{(i)} \cup X_C^k\right)\right] \quad (3)$$

$$m = Monte\ Carlo\ sample\ size$$

Equation (3) Is the marginal contribution of player j towards coalition S approximated according to the Monte-Carlo approximation. The Monte-Carlo approximation is subject to over and under sampling which is addressed when discussing the results.

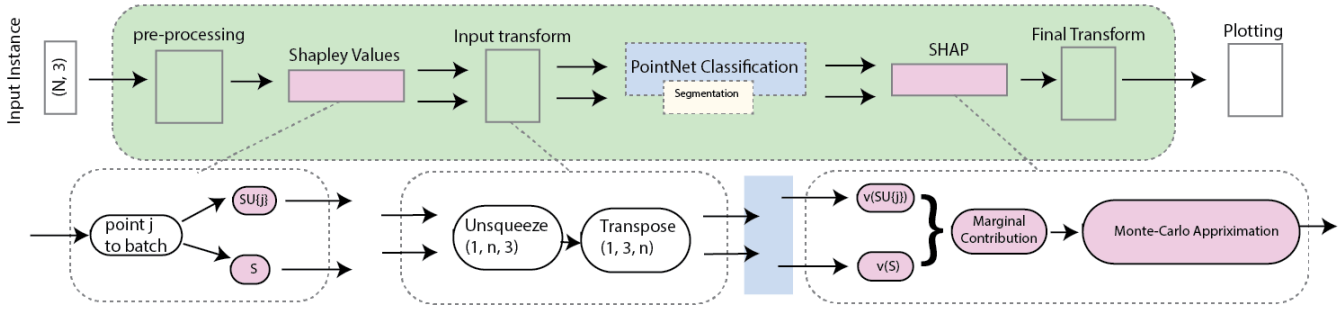

Figure 1, The pipeline integrates the custom SHAP framework for PointNet with point cloud data. The pipeline's key steps include pre-processing, creating two lists of S and S with j, transforming these lists to align with the model's expectations, calculating the model's marginal contribution, converting the SHAP value to colour, and finally plotting the results.

4. METHODOLOGY

This section describes how to compute the SHAP value using a custom framework. Figure 1 provides an overview of the custom framework, which depicts the study's flow. The discussion starts with the raw data set, then moves on to the pre-processing stage, the computation of Shapley values, the transformation of inputs for the model, the calculation of SHAP values, and a final transformation for the output plot.

4.1. Data Set

The research utilizes data obtained from the Princeton ShapeNet ModelNet10 library [13], which consists of a collection of 10 distinct man-made home objects, including chairs, desks, and 8 other categories. Figure 2 depicts a specific item with the item identifier 888, extracted from the library. The model's vertices comprise 1479 points that make up the chair mesh. Four pre-processing phases yield the final point cloud: sampling, normalizing, rotating, and generating noise.

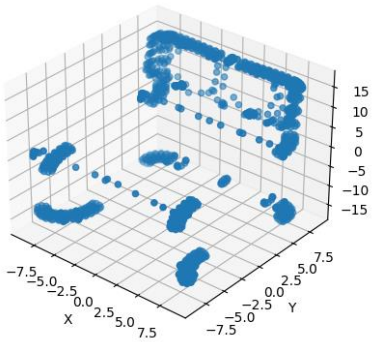

Figure 2, Raw point cloud data instance of item 888 collected from the ModelNet10 folder of ShapeNet [13].

4.2. Pre-processing

The four pre-processing steps taken are sampling, normalizing, rotating and inducing Gaussian noise.

4.2.1. Sampling

The mesh is uniformly sampled at 512 points across its surfaces. We assign a weight to each surface proportional to its size, ensuring that surfaces have similar point densities. See Figure 3a.

4.2.2. Normalization

The sampled data points are normalized into a unit sphere [4] to account for variations in scale among different meshes. This normalization process ensures consistency across all data instances of various sorts, and by positioning the cloud in the centre, it facilitates its rotation in the subsequent stage. See Figure 3b.

4.2.3. Rotating

The cloud is normalized and then randomly rotated along the Z axis to ensure the rotational invariance of each point cloud. The goal of this process is to create a point cloud that is more realistic, akin to what a LiDAR scanner would produce. See Figure 3c.

4.2.4. Induce noise

A Gaussian noise, characterized by a mean of 0 and a standard deviation of 0.002 [4], is applied to the point cloud. This stage mimics a LiDAR scan and is a more realistic representation of real-world data, as real-world data is noisy. See Figure 3d.

Figure 3 provides a graphic representation of all four pre-processing phases. Qi et. al. [4], implemented these procedures in their research to develop a robust model for categorizing point clouds. This study employs the same model previously trained by Qi [4]. Therefore, a data instances is created to fit the SHAP framework destined for PointNet. The instance undergoes the same pre-processing procedures as the model's initial training, helping the model's capacity to predict.

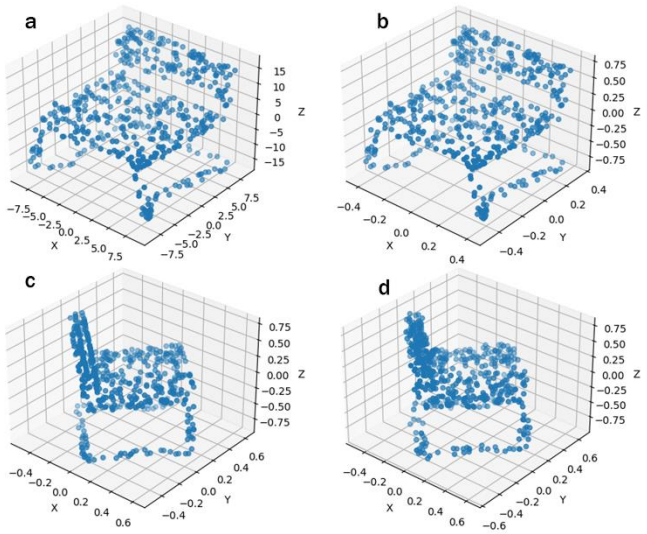

Figure 3, Four pre-processing steps to augment the point cloud. a uniformly samples the points along the surfaces. b normalized the cloud int the unit sphere. c randomly rotates the entire cloud around the z axis. d induces gaussian noise to the point cloud.

4.3. SHAP values for point cloud

The calculation of the SHAP value for point clouds involves two distinct algorithms. The first involves iterating over all points in the point cloud and sampling using the Monte-Carlo method. The second strategy utilizes the prediction function of the PointNet model to determine the marginal contribution. Figure 4 and Figure 5 display the pseudo-code for separate algorithms, with each step outlined in the subsequent subsections.

4.3.1. Algorithm 1

Figure 4 uses the "Monte Carlo Estimation of Shapley Values" algorithm to calculate the SHAP values for each point in the point cloud data instance. The function requires two parameters: `point_cloud`, which is a tensor array representing a multi-dimensional array of the point cloud, and `num_samples`, which determines the number of Monte-Carlo samples used for the estimation. The function samples the data instance with 512 points, resulting in a value of 512. The function assigns the variable `num_points` a value equal to the number of points in `point_cloud`. Additionally, the function initializes the array `shapley_values`, setting all elements to zero.

In `point_cloud`, the outer loop iterates over each point `j`. The variable `marginal_contributions` is initialized to record the marginal contributions of point `j` across various permutations.

To perform Monte Carlo sampling, the inner loop executes `num_samples` times for each point `j`. A permutation `perm` of point indices is generated to mimic different coalitions. The list `subset` is initialized to temporarily hold points as they are included based on the permutation parameter `perm`.

A separate loop is used to iterate over the permuted indices `perm`. If the current index `m` equals `j`, the subset should include point `j`. The variable `subset_with_j` represents the subset that includes point `j`, whereas the variable `subset_tensor` represents the subset that does not include point `j`. The `value_function` invokes Algorithm 2, which transforms the subsets for the PointNet model and calls the predict function.

Algorithm 1 Monte Carlo Estimation of Shapley Values

```
Require: point_cloud, num_samples
num_points ← point_cloud.shape[0]
shapley_values ← array of zeros of size num_points
for j ← 0 to num_points − 1 do
    marginal_contributions ← empty list
    for sample ← 0 to num_samples − 1 do
        perm ← random permutation of num_points
        subset ← empty list
        for k in perm do
            if k == j then
                if subset is empty then
                    subset_with_j ← point_cloud[k].numpy().reshape(1, -1)
                else
                    subset_with_j ← vstack(subset, point_cloud[k].numpy())
                end if
                if subset is empty then
                    subset_tensor ← empty array
                else
                    subset_tensor ← vstack(subset)
                end if
                marginal_contribution ← value_function(subset_with_j) −
value_function(subset_tensor)
                ALGORITHM 2
                append marginal_contribution to marginal_contributions
                break
            end if
            append point_cloud[k].numpy() to subset
        end for
    end for
    shapley_values[j] ← mean(marginal_contributions)
end for
return shapley_values
```

Figure 4, Algorithm 1 which loops through all points in the point cloud, adds them into two lists and calls the predict function of the mode. With the predict value return from Algorithm 2 it calculates the marginal contribution and stores the SHAP value of each point.

This marginal contribution is added to the `marginal_contributions` list, and the loop is terminated to begin the next sample. If the value of `j` is not equal to the value of `k`, the current point is transformed into a NumPy array and added to the subset.

After evaluating all samples for point `j`, compute the mean of the marginal contributions. The average value for point `j`, calculated using SHAP, is placed in the `shapley_values` array.

4.3.2. Algorithm 2

In figure 5, algorithm 2 takes `subset_with_j` and `subset_tensor` from Algorithm 1 and uses the predict function to compute the marginal contribution of the two lists for each Monte-Carlo sample.

If the variable `point_cloud` (the two subset lists) is a 2D array with a shape of (N, 3), it signifies that there is a single point cloud consisting of N points, each defined by 3 coordinates. The `unsqueeze(0)` method converts the input into a PyTorch tensor and includes a batch dimension. This operation produces a tensor with a shape of (1, N, 3).

If the form of `point_cloud` is already three-dimensional (B, N, 3), it indicates that there are B batches of point clouds, each containing N points with 3 coordinates. This tensor is transformed straight into a PyTorch tensor. If `point_cloud` contains dimensions different than the anticipated ones, a value error is raised to indicate the presence of unexpected dimensions.

The reason for implementing error handling is for troubleshooting and because PointNet requires the input tensor shape to be (B, N, 3) for the model. For individual data instances, the batch number is consistently 1. Here, N has a value of 512. The tensor shape is transposed to align with the input specifications. PointNet requires the input tensor to shape like (B, 3, N).

The `transpose(1, 2)` operation, swaps the final two dimensions of a tensor. If the original form was (1, N, 3), this produces a tensor with a shape of (1, 3, N).

Subsequently, using `pointnet.eval()`, the code switches the PointNet model to evaluation mode. This function deactivates specific layers, such as the dropout layer unique to training and inference processes.

The `torch.no_grad()` function avoids the computation of gradients, which reduces memory consumption and enhances inference performance. The model then generates predictions using the received `subset_with_j` and `subset_tensor` tensor arrays. The results tensor comprises the unprocessed predictions (logits) generated by the model. The `softmax` function transforms the logits into probabilities along the class dimension. Finally, the code retrieves the highest probability from the probability tensor and turns it into a Python float using the `.item()` method. The function returns the highest probability as its outcome.

Algorithm 2 Predict Maximum Probability from Point Cloud

```
Require: point_cloud
    if point_cloud.ndim == 2 then
        point_cloud_tensor ← torch.tensor(point_cloud).unsqueeze(0)
    else if point_cloud.ndim == 3 then
        point_cloud_tensor ← torch.tensor(point_cloud)
    else
        raise ValueError("Unexpected point cloud dimensions")
    end if
    point_cloud_tensor ← point_cloud_tensor.transpose(1, 2)
    if point_cloud_tensor.shape[-1] == 0 then
        raise ValueError("Point cloud tensor is empty or has insufficient points")
    end if
    pointnet.eval()
    with torch.no_grad() do
    outputs, _, _ ← pointnet(point_cloud_tensor)
    probabilities ← torch.nn.functional.softmax(outputs, dim=1)
    max_prob ← probabilities.max().item()
    return max_prob
```

Figure 5, Algorithm 2, transforms lists subset_with_j and subset_tensor and runs it through the predict function of the model, and returns the max probability used in Algorithm 1.

4.4. Final transform and plotting

Combining algorithms 1 and 2 results in a list named "`shapley_values`," which holds the SHAP values for each point in the point cloud. From the "`shapley_values`" list, the minimum and maximum are found, and using those extremes, a colormap using Matplotlib maps all values to a colour representation, resulting in Figure 6.

5. RESULTS

This section showcases the results obtained from the methodology. The text discusses three distinct outcomes, with the first being a graphical representation that illustrates the SHAP value for the given data instance. The second figure represents the same data instance, but it uses different samples for the Monte-Carlo approximation. The final graphic showcases independent data instances sampled using the same Monte-Carlo sample.

Figure 6 displays four distinct viewports of the identical data instance. This instance corresponds to item 888 and falls under the chair classification. This point cloud consists of 512 points. Each point is defined by its x, y, and z coordinates and colour value.

The x, y, and z information represent the point's coordinates on the graph. The hue of each data point corresponds to its SHAP value, with a higher value indicating a greater contribution to the prediction and a lower value indicating a lesser contribution. The colour represents the conversion of the numerical SHAP value into a corresponding colour value. The selected colour range spans from yellow to red with yellow having lesser importance towards the prediction and red having increased importance towards the prediction. The colour bar to the right of the four subplots displays the numerical value of the SHAP value and its corresponding colour. The Monte-Carlo approximation contains 1024 steps. The points that are of great significance and interest have an orange-red colour, and fall within the uppermost region of the SHAP value spectrum.

Figure 6 displays the SHAP values of 512 data points, where the largest SHAP value is 0.0086 and the lowest SHAP value is -0.001. Points with a positive SHAP value indicate that they provide a positive contribution to the prediction when compared to the other points in the instance. Points with a SHAP value equal to zero have no influence on the model's prediction, and points with a negative SHAP value have a negative contribution on the prediction.

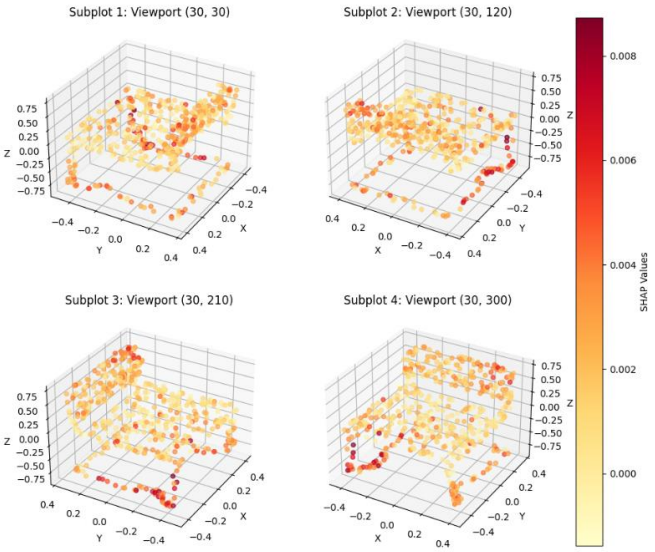

Figure 6, Four subplots of a single data instance generated from 512 points with x, y, z, and SHAP value. Dark red colored points have high model prediction contribution and light-yellow colored points a low model prediction contribution.

Figure 6 has Monte-Carlo sampling of 512 steps. The accuracy of this sample size increases as more steps are used to approximate. The approximation converges towards the true function when the number of steps approaches infinity. Figure displays how increasing the number of sampling steps affects the SHAP values.

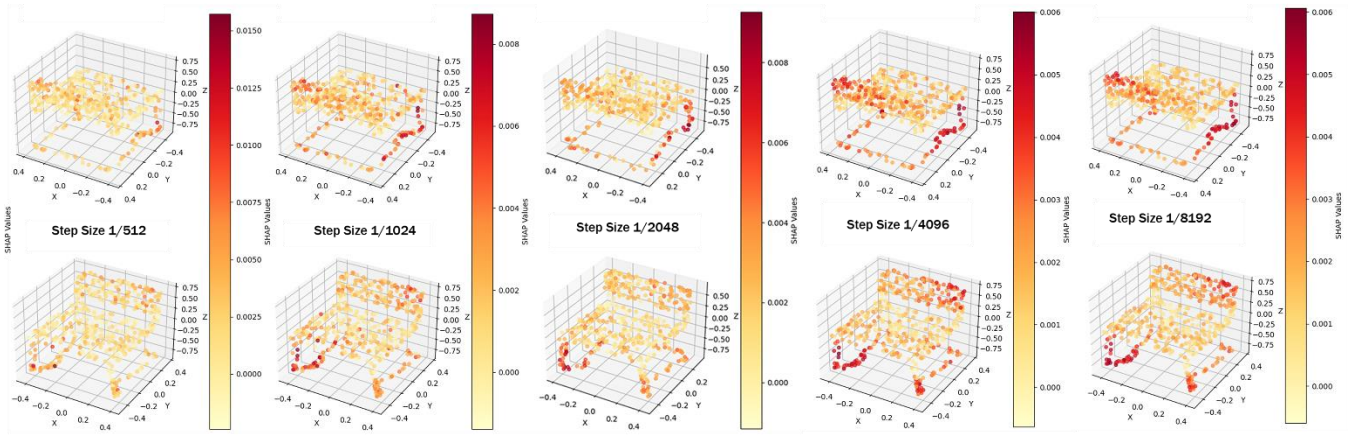

Figure 7, Five subplots representing various Monte-Carlo approximation samplings at varying step sizes. From left to right, the number of steps increases, and the size of each step decreases. All subplots depict the same data instance.

Figure 7 exhibits five subplots, each with larger number of steps and smaller step size. The subplots one through five, arranged from left to right, have step sizes of $\frac{1}{512}, \frac{1}{1024}, \frac{1}{2048}, \frac{1}{4096}, and \frac{1}{8192}$, respectively. Notably as the number of steps grows and the step size decreases. The distribution of SHAP values becomes narrower, resulting in a decrease in the extremity of the SHAP values for both negative and positive SHAP values. There is also an increasing relative proportion of points with higher SHAP values as the number of steps increases. Another finding is the emergence of clusters, where all surrounding points within the cluster are dark red. The final key observation is that each subplot identifies different points to be important, rather than having commonality across subplots for important points.

Once we reach a specific sampling threshold, further sampling becomes redundant, merely increasing computational power and complexity without significantly affecting the conclusion. The choice between attaining precise results through a greater number of steps or emphasizing efficiency varies based on the application. Section 6.1 discusses the adequacy of Figure 7's results.

Further analysis of the custom SHAP framework is tested by subjecting it to various point clouds and asking it to predict and explain why it believes each point cloud to be a chair. Figure 8 demonstrates this.

Figure 8 exhibits four unique chair representations, each composed of 512 points. A Monte Carlo sampling

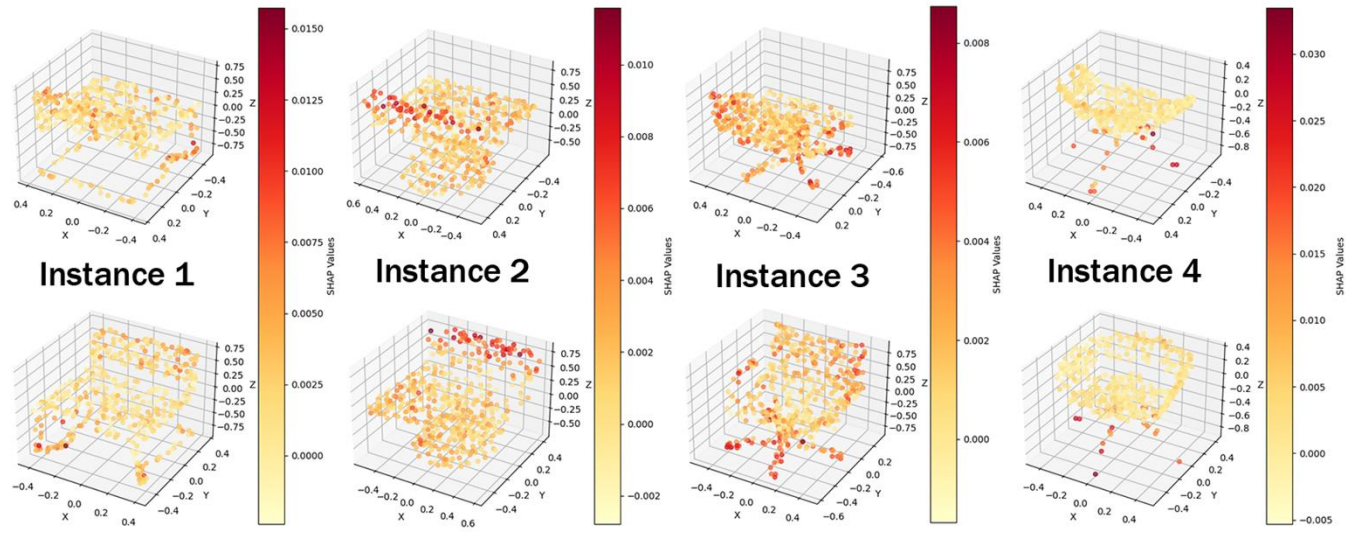

Figure 8, Four different data instances of classification chair, all sampled at 512 points and 512 Monte-Carlo steps

technique with 512 samples was used to obtain the points. The SHAP values provide insights into the unique characteristics of each chair. In the second subplot, the points located at the top of the backrest primarily contribute to the prediction. In the third case, the legs and armrests significantly influence the outcome, as they display shades of red. For the same reasons mentioned above, the legs play a crucial role as the primary contributing element in the fourth case. The custom framework has successfully shown why different point clouds are classified as chairs, each with its own set of characteristics. Moreover, the framework elucidates the process of classifying each data instance as a chair.

6. DISCUSSION

6.1. Different sizes of Monte Carlo

By increasing the number of Monte Carlo samples, the subplots in Figure 7 became more comprehensible. The goal of this study is to provide a thorough explanation of the model's prediction. The initial sample size of 1024 steps is not considered comprehensive. However, these crucial aspects become more evident as the sample size increases. The significance of the model's key principles becomes much more evident. At 512 points, the model took around 2 hours to complete 4096 steps of approximation. 4096 steps can adequately analyse a point cloud with 512 points, providing a comprehensive understanding of the PointNet model and this point cloud. As the number of points in the point cloud increases, further investigation may indicate the necessity of a higher sample rate. Future studies can explore this aspect. The same principle applies to determining the threshold for sampling.

6.2. Different chairs

The sampling rate for Figure 8 is 512 steps. These subplots don't emphasize an adequate explanation, as they require a larger sample size. However, the figure does provide insight into cluster formation. Despite the reduced sample size, the cluster formation and increasing step numbers in Figure 7 allow us to approximate the appearance of each instance in Figure 8. Nevertheless, the SHAP framework in these 512 samples clarifies this point cloud's classification.

6.3. The choice between Monte-Carlo and Markov Chain Monte-Carlo

During the course of this research, The Monte-Carlo approximation method is employed to estimate the value function. This approach is employed based on the premise that the points collected during pre-processing follow a random normal distribution, which is necessary for the Monte-Carlo approximation to function well. A projected stochastic gradient technique [9] can be used as an alternative approach to approximate the value function. The Markov Chain Monte Carlo method relies on a Poisson distribution of variables. In this study, the specific allocation of points remains unknown. Monte-Carlo is a more straightforward and simpler method to implement. One drawback is that it assumes that the data are independent. The choice of approximation method has an impact on the SHAP value itself, but it does not have any bearing on the validity of this research as it is unrelated to the objective.

6.4. Point sampling

For object recognition, a minimum of 512 points is sufficient, both for the model and for a human interpreter of the data. Reducing the number of sampling points also improves computational performance and lowers complexity. If an object has been segmented or is so large that it requires an increasing number of points to capture, sampling may become more difficult. However, this research finds that categorizing only one individual item at a time using instances of 512 points is adequate.

6.5. Model choice

The custom SHAP framework was specific to the PointNet model form Qi et al. [4] model was chosen based on its straightforwardness, ease of implementation, and lack of an ordered structure suitable for raw point cloud points. Alternative models might also be suitable. Adapting different models to the framework may require modifying the input transformation. After making these changes, a variety of models could potentially utilize this framework.

6.6. Bug fixes and other first time research problems.

This study marks the first attempt to utilize SHAP in three dimensions, having previously restricted its use to text and images. Applying SHAP in this new way allows for higher dimensional and complex data to become interpretable. Tis framework allows for interpretability on the smallest scale of the data instance, allowing for the maximum interpretability of the model, the data, and the prediction. By understanding individual points contribution, the entire model is understood. Its real-world scenario would be for autonomous driving. Autonomous cars use LiDAR sensors to scan its surroundings and have a model decide if the car should stop or go. When the model decides to ignore a stop sign, debugging the model and data becomes a hassle. Applying this custom framework in such application allows for massive leap forwards for the interpretability and trust in autonomous driving. Truly this development marks a start of a new combined branch for both point clouds and XAI.

Given the challenge of developing a self-contained explainer, it is reasonable to anticipate certain bugs and inefficiencies within the framework's code. Nevertheless, this serves as a promising starting point and might potentially lead to the development of a new field of explainers focused on point clouds and point cloud models in the future.

7. CONCLUSION

This study has effectively developed a tailored framework for explaining the inner workings of the machine learning model PointNet. In the context of point cloud data, SHAP values were utilized to analyse the weighted average marginal contribution of individual points in a certain prediction, resulting in the visualization presented in the results. This study's PointNet model, which explains various scatterplots, serves as an excellent candidate to test the SHAP framework. Custom code is developed by recognizing the PointNet classification model specifications as well as the fundamental principles of Shapley values in order to obtain a custom SHAP XAI model. When analyzing five different Monte Carlo sampled plots, increasing the sampling rate revealed that larger sample sizes result in a clearer result and a better explanation given a fixed data entry. and at a lesser resolution, four distinct occurrences of a chair are analyzed to provide valuable insights into the model's classification process, revealing the specific spots it relies on to produce chair classifications. The results clearly demonstrated the efficient utilization of SHAP values for point cloud data interpretation.

Ultimately, this study effectively expanded the application of explainable artificial intelligence (XAI) to point cloud machine learning models by utilizing a customized SHAP design. The established framework and methodology form a strong basis for future efforts and contribute to the overarching objective of enhancing the transparency and interoperability of machine learning models.

8. Bibliography

1] A. Shrikumar, P. Greenside, A. Shcherbina, and A. Kundaje, ‘Not Just a Black Box: Learning Important Features Through Propagating Activation Differences’, *arXiv [cs.LG]*. 2017.

2] M. T. Ribeiro, S. Singh, and C. Guestrin, ‘“Why Should I Trust You?”: Explaining the Predictions of Any Classifier’, *arXiv [cs.LG]*. 2016.

3] X. A. I. Foundation, ‘What is XAI?’ 2023.

4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, ‘PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation’, *arXiv [cs.CV]*. 2017.

5] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, ‘Multi-View 3D Object Detection Network for Autonomous Driving’, *arXiv [cs.CV]*. 2017.

6] A. B. Arrieta *et al.*, ‘Explainable Artificial Intelligence (XAI): Concepts, Taxonomies, Opportunities and Challenges toward Responsible AI’, *arXiv [cs.AI]*. 2019.

7] T. Clement, N. Kemmerzell, M. Abdelaal, and M. Amberg, ‘XAIR: A Systematic Metareview of Explainable AI (XAI) Aligned to the Software Development Process’, *Machine Learning and Knowledge Extraction*, vol. 5, no. 1, pp. 78–108, 2023.

W. Lihua and K.-H. Jo, ‘Fully

8] Convolutional Neural Networks for 3D Vehicle Detection Based on Point Clouds', 07 2019, pp. 592–601.

9] A. Conner-Simons, 'Deep learning with point clouds'. 2019.

10] J. Xu *et al.*, 'Multi-Camera Collaborative Depth Prediction via Consistent Structure Estimation', in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022.

11] C. Molnar, *Interpreting Machine Learning Models With SHAP: A Guide With Python Examples And Theory On Shapley Values*. Independently published, 2023.

12] Z. Wu *et al.*, '3D ShapeNets: A Deep Representation for Volumetric Shapes', *arXiv [cs.CV]*. 2015.