Chapter 3

Binary Classification on Columnar Cacti in Point Clouds Using Neural Networks

3.1 Data Preprocessing

3.1.1 Data Acquisition

The point cloud data in which this research is built upon was gathered via photogrammetric means in the Sonoran Desert with aid from the Seri Tribe. Altogether, there were five saguaro (CG) and five organ pipe (OP) specimens captured, each in their own point cloud, with either a DJI Mavic Pro drone with a 15 degree nadir camera angle flown in a circular path at around 10 meters altitude, or in the case that the specimen was occluded by other vegetation and below one meter in height, with a Google Pixel 6. Once gathered, the point cloud data was fed into Agisoft Metashape Professional to generate a point cloud and 3D model for each plant. Among the seven specimens gathered with the drone, there were an average of 45.4 photos used to generate the point cloud. Among the remaining three point clouds gathered with the Pixel 6, there were an average of 18.3 photos used to generate the point cloud. Stored in the .las format, these ten files encapsulate raw spatial information

captured across various terrains containing cacti. Finally, the points representing the cactus bodies were manually isolated and reclassified inside the software and then extracted into new point cloud files. Each point in these clouds carries information on its position (XYZ coordinates), its color attributes (RGB values), and its class (cactus or non-cactus).

3.1.2 Manual Classification

To train a neural network model, this study employs a supervised learning approach, requiring that the point cloud data possess class labels on which the model can train itself. Thus, the columnar cacti had to be manually isolated and reclassified inside of Agisoft Metashape Professional before exporting the point clouds for preprocessing and model training.

3.1.3 Experimental Setup

Subsequent data processing and model construction was developed in Python with the commonly used pandas, numpy, tensorflow, sklearn, and matplotlib packages. The code was developed and tested on a machine containing an AMD Ryzen 7 5800X 8-Core processor and an Nvidia 3080 GPU running Windows 11.

3.1.4 Reading and Initial Processing

The initial step in preprocessing involves the transformation of these raw point clouds into a more conducive format. Leveraging the laspy library, a Python tool designed for reading and writing LAS files, the data is read into the environment and manipulated into a pandas DataFrame, offering a tabular representation that is both intuitive and efficient for analysis.

3.1.5 Feature Selection and Culling

Upon import, the point clouds contained many columns of information which did not provide further insight to the structure of the point clouds. The primary focus lies on the spatial coordinates (XYZ) and color values (RGB). A culling process removes extraneous columns from the DataFrame. This step not only streamlines the dataset, but also significantly

reduces computational overhead, focusing the model's attention on features with the highest relevance.

3.1.6 Data Normalization and Scaling

Neural networks are highly sensitive to the scale of input data [18]. Thus, normalizing the spatial coordinates and scaling the RGB color values are necessary steps. The normalization process follows the standard, or z-score, normalization strategy where the X, Y, and Z coordinates each have their means subtracted and then are divided by their standard deviation. This yields features with zero mean and unit variance. Concurrently, RGB values are scaled to a [0, 1] range. With algorithms like gradient descent, which neural networks are built upon, features with larger magnitudes can heavily bias the learning process. These adjustments ensure that all features are enclosed in a common boundary, without the loss of any information or altering the inherent shape of the data.

Tests were conducted where the X, Y, and Z coordinates were instead scaled similarly to the RGB values, whereby their minimum and maximum values are used to shift the range of the data to [0, 1]. With all other things being equal, the standard scaling outperformed this min-max scaling strategy. It is worth noting that the standard scaling strategy is not necessarily suitable for all data sets, as it is sensitive to regions in the point cloud with higher point density. If some region had a significantly higher point density, then that area would disproportionately influence the mean and standard deviation calculations. This strategy does lead to some amount of warping on a global scale if the point cloud is not normally distributed, which is true in this case, however the relationship between points is maintained through this normalization process.

Furthermore, the positive class label was adjusted for points that were given the "cactus" classification. As an artifact of how the classification process worked in Agisoft, the positive class label upon importing the data was a value of 5. At first, this value was reset to a value of 1 according to convention for binary classification. However, subsequent tests showed very poor model performance. Tests without this adjustment of the positive class label, i.e. leaving it at 5, surprisingly showed much more promising results. This observation led to the

hypothesis that the computed errors for false negative predictions increased as the positive class label increased, leading to the positive class instances carrying more weight for the loss function. Put more simply, increasing the magnitude of the positive class labels penalized the model more for misclassifying points that are supposed to be a cactus, effectively increasing the weight of the cactus points. Because of the severe class imbalance in the data (discussed in the next subsection 3.1.7), this led to very favorable outcomes for model performance. This hypothesis is supported by observations that with positive class labels even higher in magnitude, the model misclassifies more and more of the background as being a cactus, thus leading to over-weighting of the positive class instances. After more testing, a positive class label of 3 was decided, as this value struck the right balance of enabling the model to learn effectively from the limited positive class instances and not having too much of the background misclassified.

3.1.7 Downsampling and Data Balancing

Given the vast number of points in each cloud, downsampling the data was necessary to manage computational resources and training time effectively. After verifying that downsampling would not meaningfully affect the model's results, random downsampling was applied to each of the ten point clouds. This brought each point cloud from millions of points (varying from 6 to 12 million) to just 100,000. In doing this, training times for the models tested were kept to a minimum without sacrificing structural features in the data.

However, since the number of cactus points represents such a small proportion of the entire point cloud, an additional step was needed to help offset the severe class imbalance in the data. The percentage of positive class instances among the ten point clouds range from 0.28% to 4.49%, with an average of 1.97$\pm$1.22%. So, to give the neural network more positive class instances to learn from, I controlled the positive class instances to be exactly 3$\pm$0.1% of the 100,000 downsampled points. A random deviation of $\pm$0.1% is added to help avoid the model overfitting to an exact number of points.

3.2 Neural Network Model for Binary Classification

3.2.1 Architectural Design

In designing the neural network model for this classification task, a deliberate choice was made to prioritize speed and efficiency over model complexity. This decision was informed by the specific requirements of the project, where rapid processing and simplicity were valued highly, as well as local hardware limitations. Given this, it was decided that this work would focus on two different neural network architectures notable for their speed and relative simplicity: the multilayer perceptron (MLP) and one-dimensional convolutional neural network (1D CNN). Moreover, based on a thorough review of the literature, voxelization methods, which convert point clouds into 3D voxel grids, were deliberately avoided. Such methods introduce additional complexity and computational overhead without guaranteed improvement in performance for this specific application, though the spatial relationships in the data would be able to be more thoroughly analyzed by the model. Instead, a direct input approach to the network was chosen, where raw point features are fed into the model, maintaining the integrity and granularity of the data. The dataset is organized into arrays with six features for training: the three spatial coordinates and three color values. This preparation ensures that the model receives direct, unaltered input, maximizing the efficiency of data processing and aligning with the goal of prioritizing speed and simplicity.

3.2.2 Training Methodology

Being that there were ten point clouds in the overall dataset, a 10-fold cross-validation strategy was employed to ensure a robust evaluation of the models and enhance the reliability of the results. In this approach, the dataset is divided into ten equal parts, or "folds," where each fold contains one of the point clouds. Then, one fold is set aside as a test set, while the remaining nine folds are combined to form the training set. The model then is trained using the training set, and evaluated on its performance using the test set. This process repeats so that each fold is used for testing once and for training nine times, ensuring that all data contribute to both training and validation processes. This method nets ten distinct models,

each tested on a different fold. Cross-validation significantly mitigates bias and enhances the detection of overfitting by systematically rotating the train-test splits. This not only allows each model to be trained and validated across diverse data subsets but also stabilizes the evaluation by averaging the performance across different iterations, providing a more comprehensive assessment of the model's effectiveness and generalizability.

Each training session is limited to five epochs, following observations that the model converges rapidly, typically within the first one or two epochs. Extending the training to more epochs, even up to 100, did not yield any improvements in performance, indicating that the chosen architecture efficiently captures the relevant patterns in the data without the need for prolonged training.

During the development process, a key observation was made regarding the balance between accuracy and recall as the percentage of cactus points in the training set was adjusted. Increasing the proportion of cactus points led to higher recall but lower accuracy. After extensive experimentation, a ratio of approximately 3% cactus points to 97% non-cactus points was identified as optimal. Another factor which played a major role in increasing the recall but decreasing the accuracy was the positive class value, as discussed in section 3.1.6. By adjusting this value upwards, it effectively put more weight on the limited number of cactus points by making false negative classifications more costly to the model's loss function. Similar to the class imbalance rate, setting this positive class value too high would lead to very high recall scores, but lower accuracy. By having the positive class value set to 3 and having a class imbalance ratio of approximately 3% cactus points to 97% non-cactus points, this strikes a balance for the model being able to effectively identify cactus points while maintaining reasonable overall accuracy, aligning with the project's objectives of extracting cacti from point clouds to simplify volume and surface area estimates.

3.2.3 Final Models

After extensive testing, the best model was built using an MLP architecture detailed in Figure 3.2, where the layers used Rectified Linear Unit (ReLU) activation functions (except in the last layer, which had a sigmoid activation to suit the binary classification being done), used

the Adam optimizer, had a learning rate of 0.001, used the binary crossentropy loss function, and trained for five epochs with a batch size of 10,000 points. Using these parameters, I performed 10-fold cross-validation where the model was trained on nine of the ten point clouds, each containing 100,000 points, and used the tenth point cloud as the validation set.

The best model built using the 1D CNN architecture, detailed in Figure 3.1, started out based on the PointNet++ architecture [5], but was subsequently optimized for this specific application. It used the same activation functions, optimizer, learning rate, loss function, epochs, and batch size as listed above, and used kernel sizes of 25, 13, and 7 for its three convolutional layers, in that order. The 1D max pooling layers used a pool size of 2, and both the pooling and convolutional layers' outputs were padded to have the same spatial dimensions as the inputs.

3.3 Model Evaluation and Results

3.3.1 Performance Metrics

For this application, I put the highest priority on the recall metric, and the second highest on overall accuracy. While accuracy gives a comprehensive view of the model's performance, it is not sufficient on its own, especially for datasets with large class imbalances such as this one. Recall provides an additional layer of insight into how well the model is identifying positive instances, which is critical for the target application of this research. With a recall value of close to 1.0, it ensures the model is capturing almost the entire cactus body and will help curtail underestimates for volume and surface area. Additionally, visualization of the model's predictions compared with actual classifications through 3D scatter plots provides a spatial understanding of its performance, highlighting the practical outcomes in the model design.

3.3.2 Results

By iteratively testing models with varying hyperparameters, the MLP architecture proved to be only marginally better than the 1D CNN architecture for this application. This result is especially notable as CNNs are known to be well-suited for spatial data [5]. However, this outcome underscores the effectiveness of the MLP in handling the given task, despite its relative simplicity, and supports conclusions drawn in [11]. The results from the 10-fold cross-validation, which indicate a generally high level of performance for both architectures can be seen in Tables 3.1 and 3.2. The specific model architectures which yielded those results are detailed in Figures 3.1 and 3.2. The MLP architecture manages to perform slightly better when averaging the results from all 10 folds, as the average accuracy and recall for the 1D CNN architecture were 88.05±8.43% and 76.16±27.61 respectively, but the average accuracy and recall for the MLP architecture were 88.49±7.40% and 78.49±29.39% respectively.

For the MLP model, the folds for CG_01, CG_02, CG_03, CG_05, OP_01, OP_03, and OP_04 all achieve at least a 90% recall with accuracies ranging from 84.26% to 94.91%. Examining Figures 3.4, 3.6, 3.8, 3.12, 3.14, 3.18, and 3.22 shows that the model effectively captures the columnar cactus bodies contained within the point clouds, but also captures clusters of undesirable points on the ground and surrounding vegetation. The folds with lower recall values, namely CG_04, OP_02, and OP_05, reflect the model's inconsistency in identifying cactus points across all datasets. Overall, the model can produce a point cloud in which the columnar cacti are captured and significantly more isolated the majority of the time. Therefore, it is hypothesized that using the resulting point cloud for subsequent volume calculations would substantially speed up the process of isolating the cacti the majority of the time. However, when the model does not perform so well on a certain cactus, this would lead to large underestimates of its volume and surface area.

In the following subsection 3.3.3, there are plots showing the original point clouds with their corresponding RGB values, alongside plots showing the true class labels and the MLP model's predicted class labels for each of the ten point clouds used in this study. The plots for the 1D CNN model's predicted class labels are only marginally different than the MLP's predicted labels and are excluded for brevity.

Plants	Accuracy	Recall
CG_01	0.8425	0.9118
CG_02	0.8878	0.9638
CG_03	0.9072	0.9817
CG_04	0.9506	0.6735
CG_05	0.9468	0.7122
OP_01	0.9389	0.9266
OP_02	0.6546	0.0826
OP_03	0.9227	0.9933
OP_04	0.8361	0.8936
OP_05	0.9173	0.4769

Table 3.1: Accuracy and recall results from the best performing 1D CNN model.

Layer (type)	Output Shape	Param #
conv1d (Conv1D)	(None, 6, 64)	1664
max_pooling1d (MaxPooling1D)	(None, 3, 64)	0
conv1d_1 (Conv1D)	(None, 3, 128)	106624
max_pooling1d_1 (MaxPooling1D)	(None, 2, 128)	0
conv1d_2 (Conv1D)	(None, 2, 256)	229632
global_max_pooling1d (GlobalMaxPooling1D)	(None, 256)	0
dense_8 (Dense)	(None, 128)	32896
dense_9 (Dense)	(None, 64)	8256
dense_10 (Dense)	(None, 32)	2080
dense_11 (Dense)	(None, 1)	33

Total params: 381185 (1.45 MB)
Trainable params: 381185 (1.45 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 3.1: A description of the model architecture for the 1D CNN, indicating the number of convolutional filters and pooling layers, followed by the fully-connected layers at the end. The "None" in each ordered set is for the variable batch size, which in this case was 10,000.

3.3.3 Visualizations

Plants	Accuracy	Recall
CG_01	0.8426	0.9626
CG_02	0.9111	0.9348
CG_03	0.9096	0.9935
CG_04	0.9421	0.5394
CG_05	0.9491	0.9000
OP_01	0.9309	0.9482
OP_02	0.6838	0.0205
OP_03	0.9127	0.9964
OP_04	0.8613	0.9119
OP_05	0.9056	0.6412

Table 3.2: Accuracy and recall results from the best performing model, which had an MLP architecture.

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 64)	448
dense_1 (Dense)	(None, 128)	8320
dense_2 (Dense)	(None, 256)	33024
dense_3 (Dense)	(None, 512)	131584
dense_4 (Dense)	(None, 256)	131328
dense_5 (Dense)	(None, 128)	32896
dense_6 (Dense)	(None, 64)	8256
dense_7 (Dense)	(None, 1)	65

Total params: 345921 (1.32 MB)
Trainable params: 345921 (1.32 MB)
Non-trainable params: 0 (0.00 Byte)

Figure 3.2: A description of the model architecture for the multilayer perceptron, detailing the number of nodes and parameters in each layer. The "None" in each ordered pair is for the variable batch size, which in this case was 10,000.

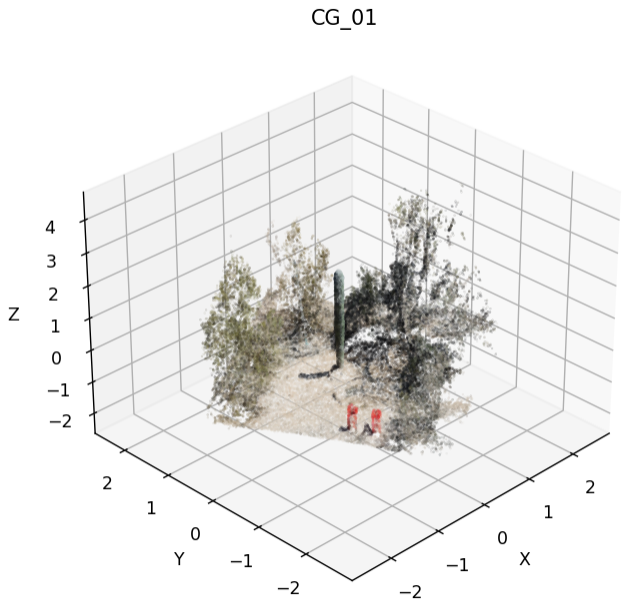

Figure 3.3: Downsampled rendering of CG_01

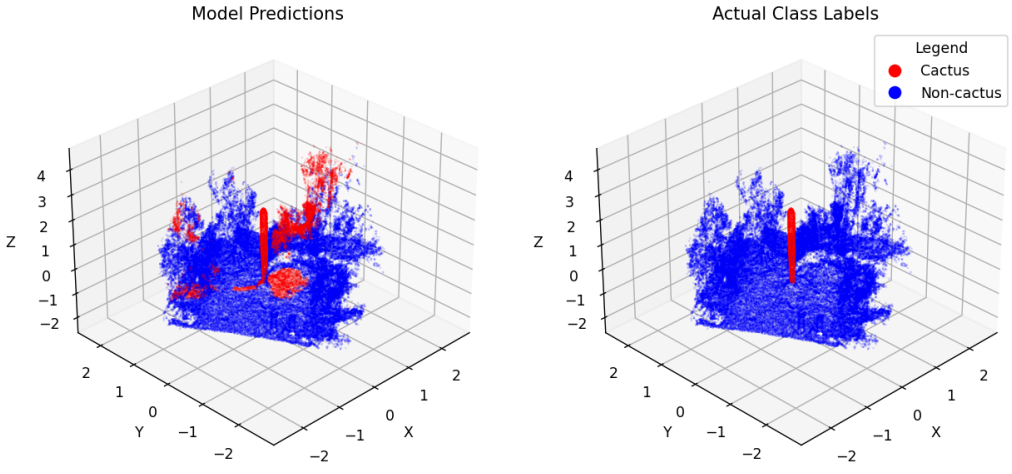

Figure 3.4: MLP model performance on CG_01. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

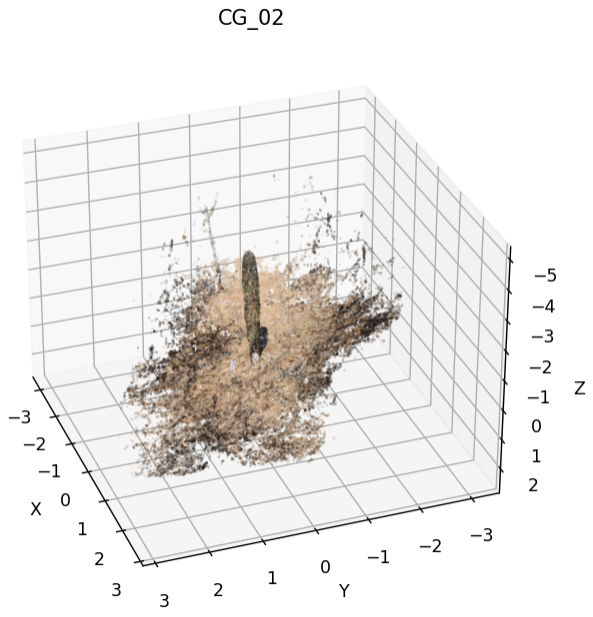

Figure 3.5: Downsampled rendering of CG_02

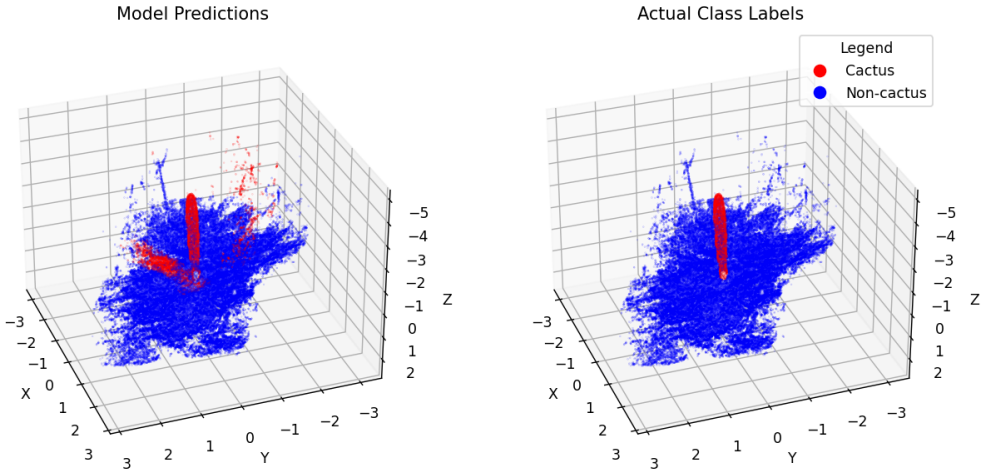

Figure 3.6: MLP model performance on CG_02. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

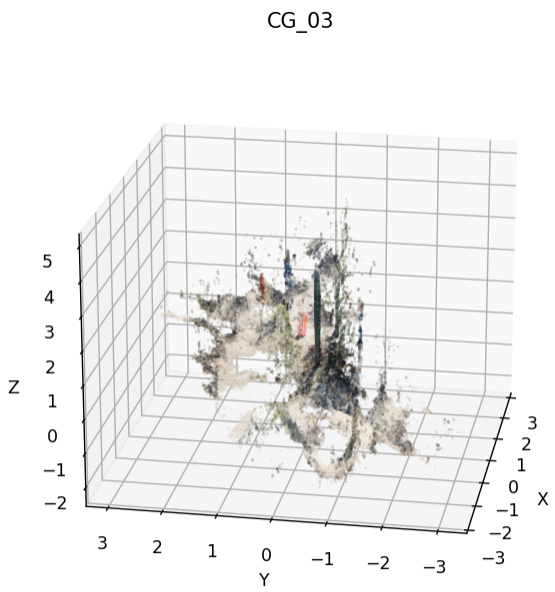

Figure 3.7: Downsampled rendering of CG_03

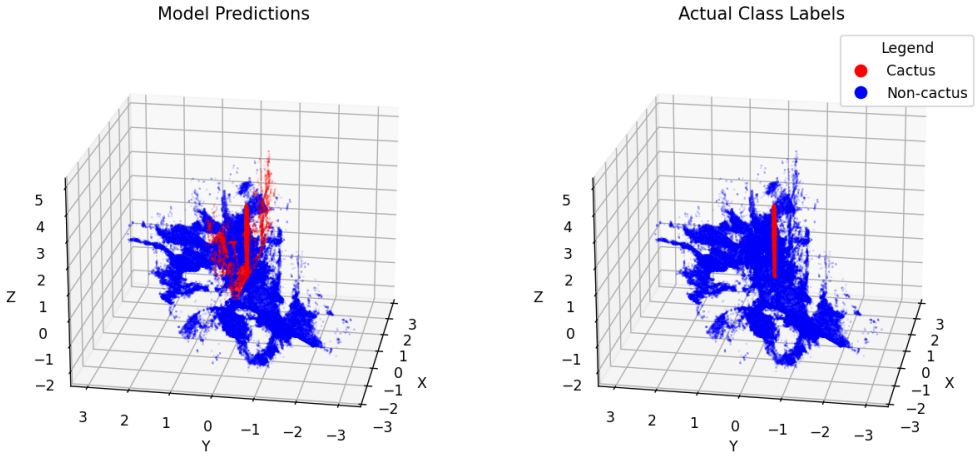

Figure 3.8: MLP model performance on CG_03. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

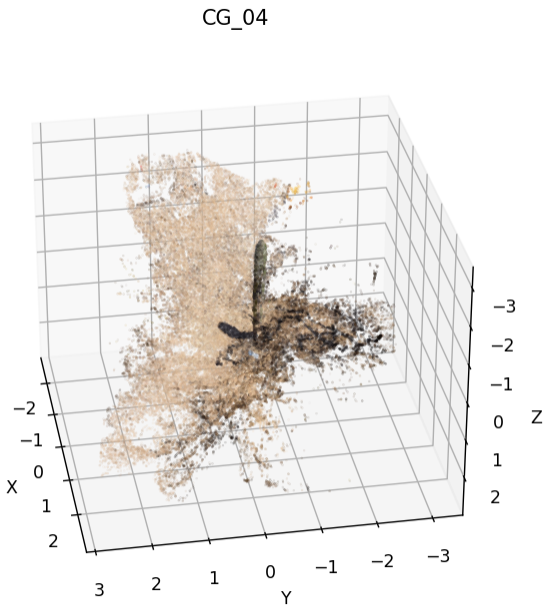

Figure 3.9: Downsampled rendering of CG_04

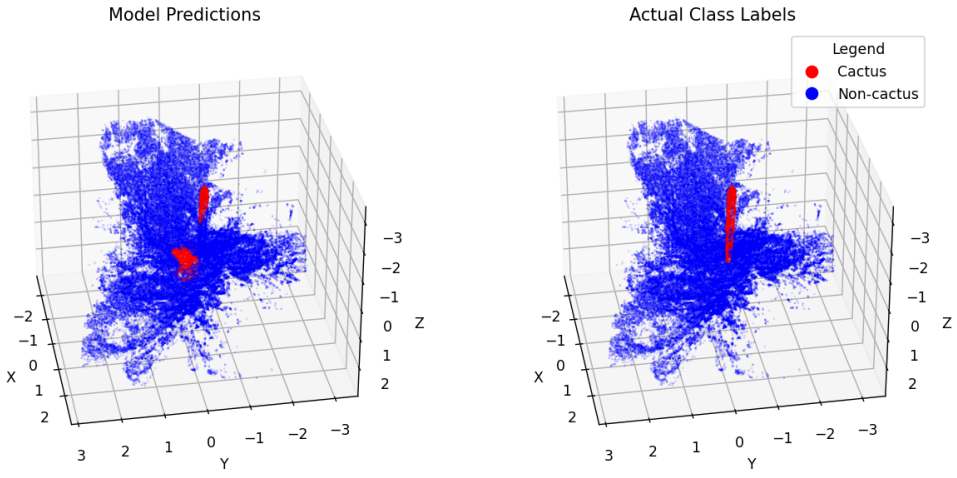

Figure 3.10: MLP model performance on CG_04. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

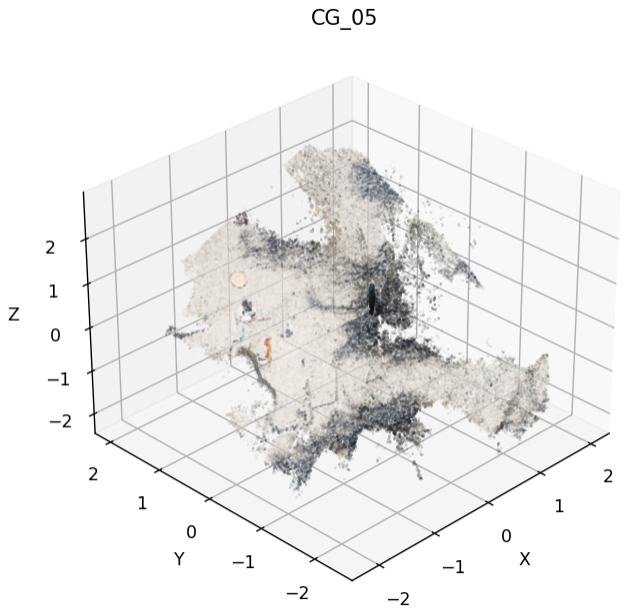

Figure 3.11: Downsampled rendering of CG_05

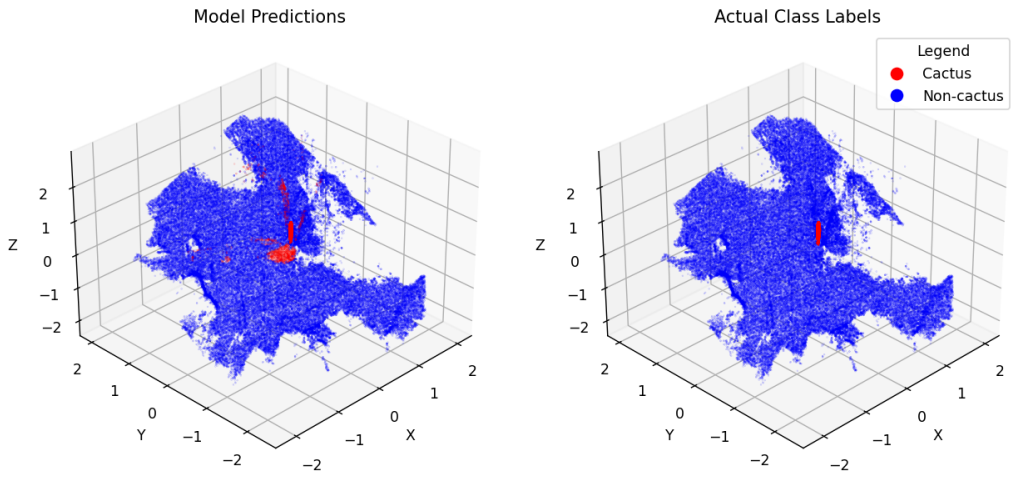

Figure 3.12: MLP model performance on CG_05. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

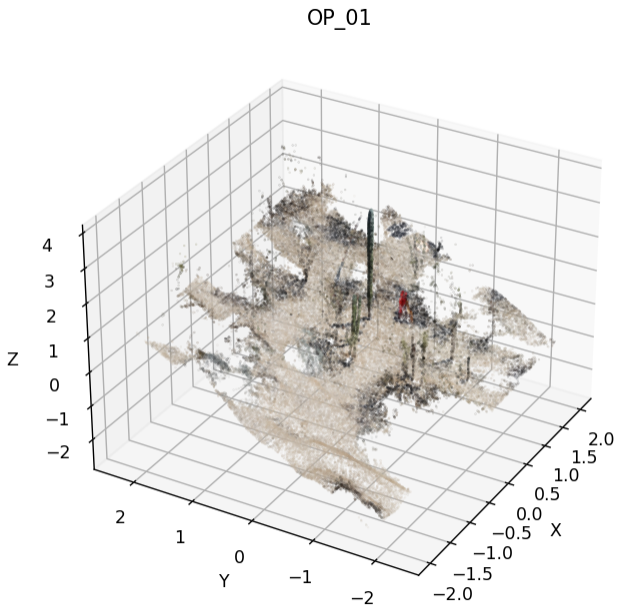

Figure 3.13: Downsampled rendering of OP_01

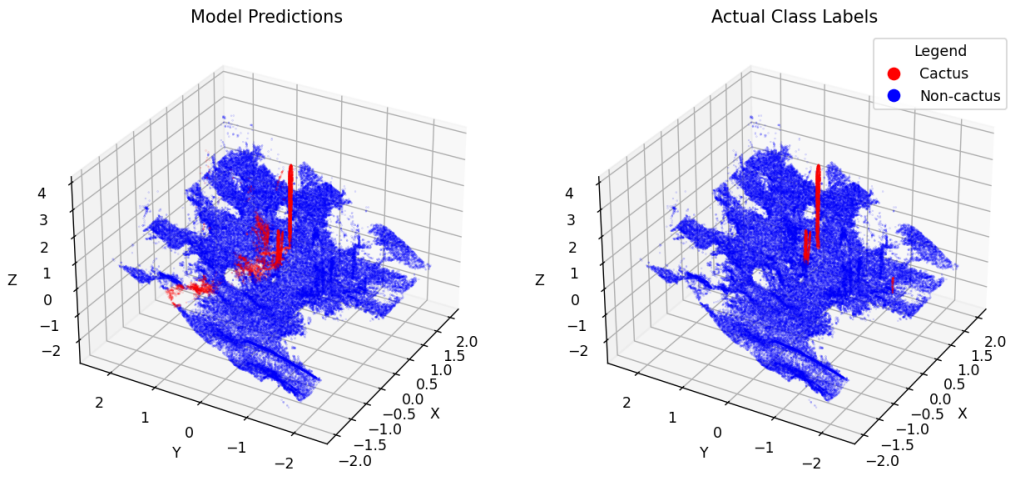

Figure 3.14: MLP model performance on OP_01. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

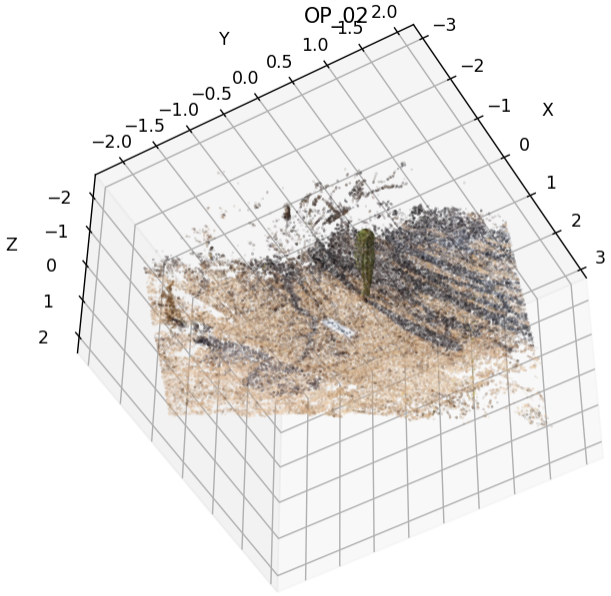

Figure 3.15: Downsampled rendering of OP_02

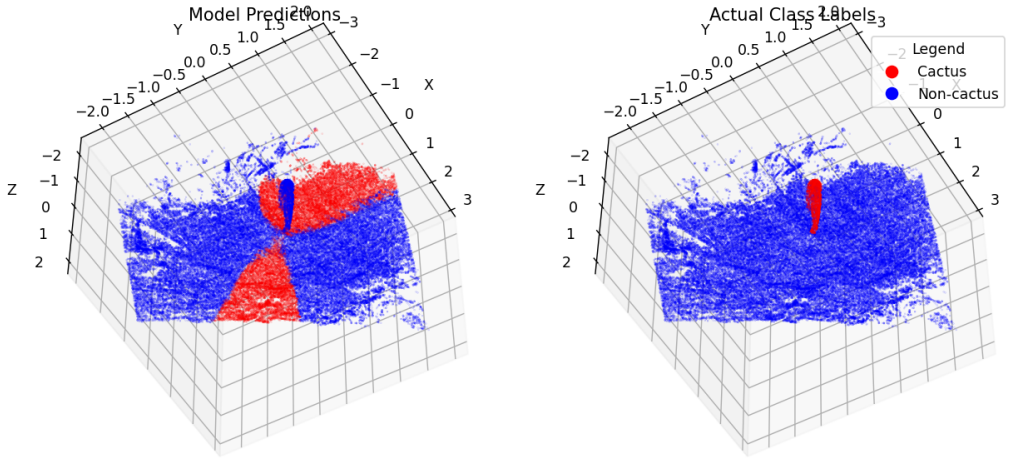

Figure 3.16: MLP model performance on OP_02. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

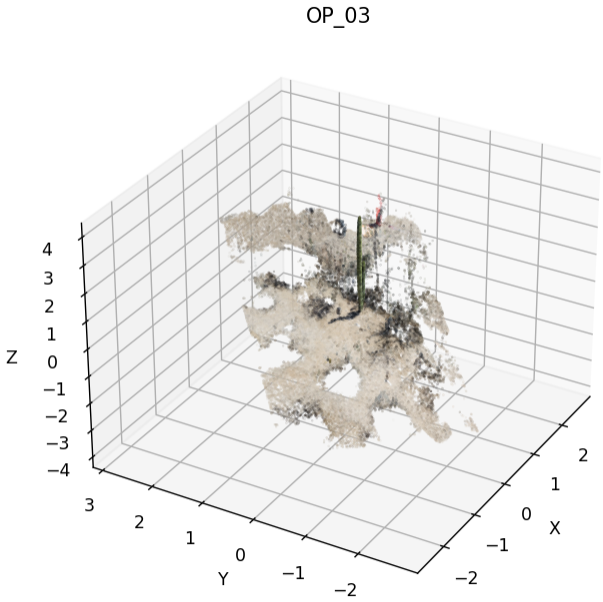

Figure 3.17: Downsampled rendering of OP_03

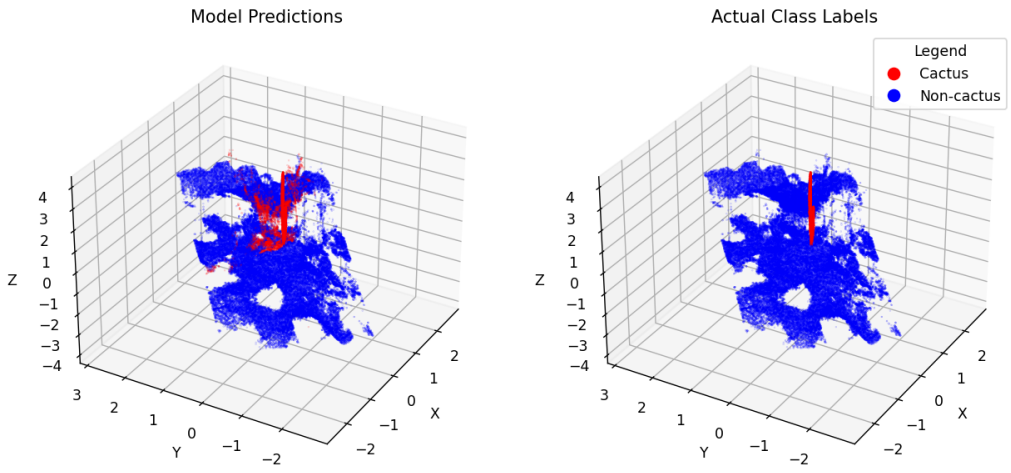

Figure 3.18: MLP model performance on OP_03. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

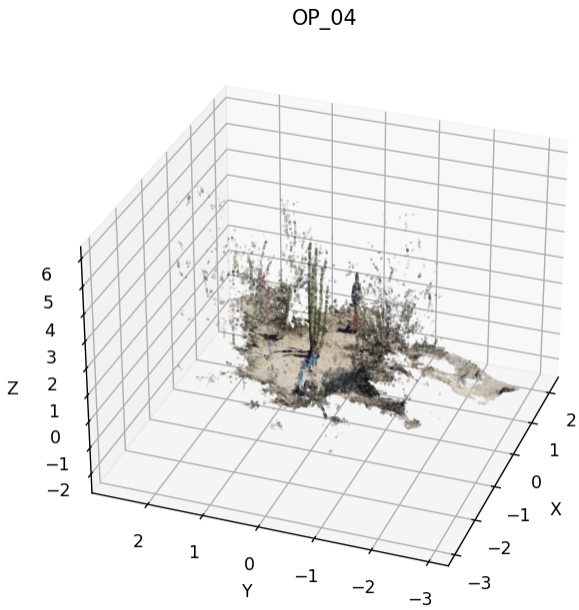

Figure 3.19: Downsampled rendering of OP_04

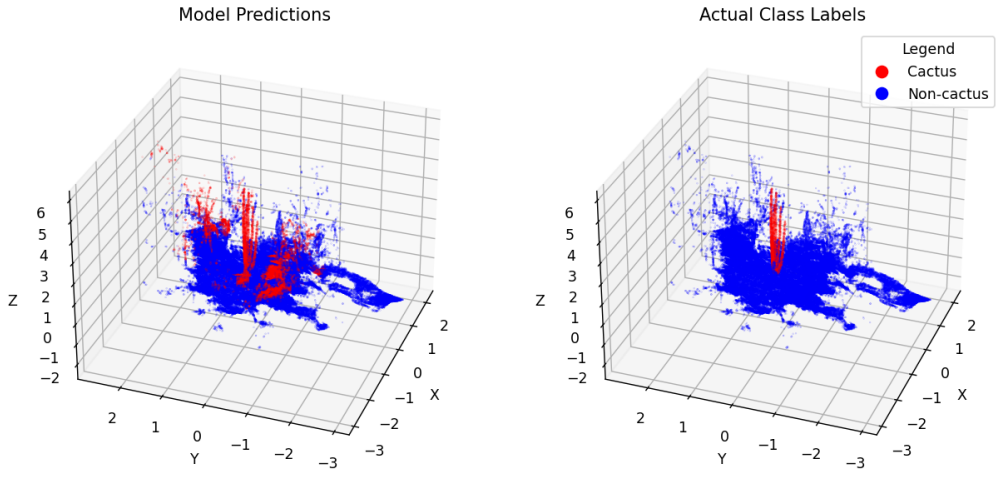

Figure 3.20: MLP model performance on OP_04. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

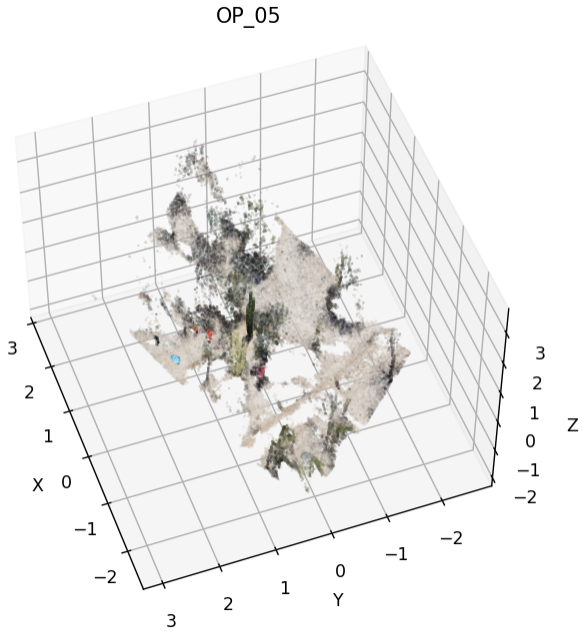

Figure 3.21: Downsampled rendering of OP_05

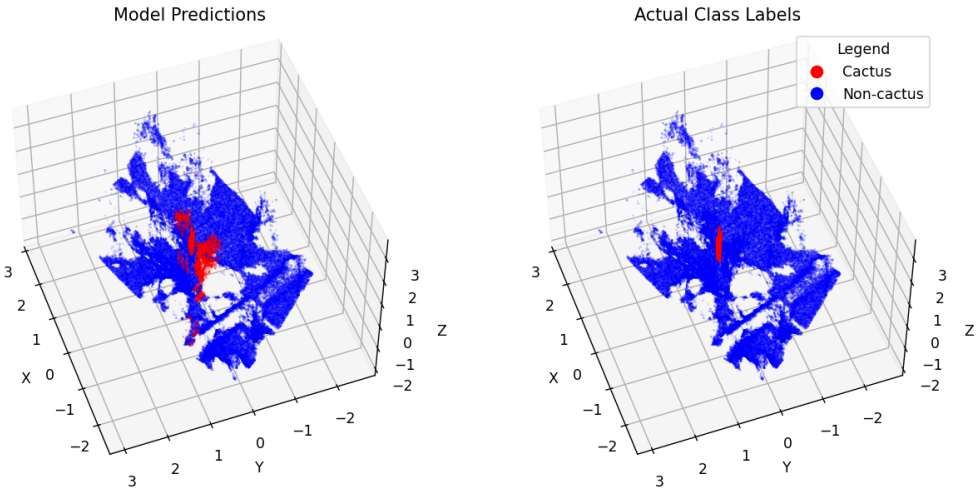

Figure 3.22: MLP model performance on OP_05. The left plot shows the MLP model performance when trained on the other nine point clouds, and the right shows the true labels.

3.4 Discussion

3.4.1 Model Performance Variability

The variability in model performance across different folds, particularly the wide range of accuracies and recalls, underscores the heterogeneity of the point cloud datasets. Certain datasets may contain more easily distinguishable features between cactus and non-cactus points, leading to higher accuracies and recalls. Additionally, the relative coarseness of the point clouds which were gathered with the Google Pixel 6 (see Figures 3.5, 3.9, and 3.15) instead of the DJI Mavic Pro offers an insight as to just how important it is to use high quality data for point cloud classification. The fact that those point clouds were generated with far fewer images on average (18.3 compared to 45.4 with the drone), points to a deficiency in data collection. Additionally, cacti CG_02 and CG_04 have Z-axes that are flipped upside down, i.e. the cactus and other plants grow up from the ground in the negative Z direction instead of the positive Z direction. Only one of those three attained with the Pixel 6 attained a score of above 90% in both accuracy and recall, making it feasible for volume calculations. Conversely, six of the seven attained with the DJI Mavic Pro attained scores high enough to make them feasible for volume calculations.

The significantly low recall in one of the folds (1.2155%) indicates that the model was completely ineffective in identifying the cactus points in that particular dataset. This could be attributed to several factors, including the quality of the data, the distribution of cactus points, or the irregular orientation of the coordinates that made classification more challenging. Specifically, I hypothesize that this particular fold performed so poorly because the ground plane was at a significantly different angle than the other nine point clouds. In all other instances, the columnar cactus bodies are oriented roughly parallel to the Z-axis, whereas in this instance the cactus body is significantly skewed (see Figure 3.15). It is also worth noting that this point cloud was one of three which were collected with a Google Pixel 6 as opposed to with a drone flying a circular path around the cactus. So, at some point in the process of building the point clouds from the images gathered with the Pixel 6, all three of those point clouds had some error in their coordinates. The first two, CG_02 and

CG_04 having a flipped Z-axis, and OP_02 being completely skewed relative to all the other point clouds. Being that the point clouds were originally measured with latitude, longitude, and meters above sea level, something must have gone awry in the process of generating the point clouds from the data gathered with the Pixel 6.

3.4.2 Data Imbalance and Model Training

Finding a model that had the right balance of recall and accuracy scores required extensive testing. As discussed at the end of section 3.2.2, the class imbalance rate and positive class value were directly correlated with the recall score but inversely correlated with accuracy, i.e. increasing those values led to higher recall but lower accuracy. This trade-off reflects the model's increasing tendency to classify points as cactus, at the expense of correctly identifying non-cactus points. Balancing accuracy and recall is crucial in applications where both identifying the object of interest and correctly classifying the background are important, as the less background points there are, the faster the volume calculation workflow can be.

This optimization process was more complex than simply tuning these two values however, because like any neural network architecture, there exists numerous other hyperparameters. Alongside this class imbalance rate, I additionally tested many different values for the downsampling rate, number of layers in the model, types of layers, number of nodes/filters per layer, kernel size (for the CNN), activation function, optimizer function, learning rate, loss function, epochs, and batch size, before settling on the settings which ultimately yielded the highest performing models.

Observations that the model converged after only one or two epochs, with no significant improvements in extending training up to 100 epochs, suggest that the chosen architecture and training methodology are efficient in capturing the relevant patterns in the data with minimal training. However, this also raises questions about the potential for overfitting and the model's ability to generalize across unseen data. Future work could explore methods to enhance model generalizability, however this would likely necessitate the collection of landscape-scale point cloud datasets that contain many more columnar cacti to train and test subsequent models.

Chapter 4

Limitations

4.1 Potential Bias Due to Central Cacti Placement

A specific limitation related to the dataset used for training these models is the potential bias introduced by the consistent placement of cacti roughly in the center of each point cloud. This central placement can lead to models that are overly optimized, or overfitted, to recognize cacti primarily in central positions and less effective at detecting cacti located in peripheral areas of the point cloud. Overfitting occurs when a model learns the details of the training data to an extent that it negatively impacts the performance of the model on new data. Such bias can hurt the generalizability of the model, affecting its performance in practical scenarios where cacti may not be centrally located. Models, whether MLP or CNN, trained on this dataset might develop a dependency on this central positioning, assuming it as a standard feature, thereby failing to learn more generalized, robust features that are indicative of cacti regardless of their position.

To mitigate this issue, data augmentation strategies can be particularly effective. These strategies involve modifying the original training data to create artificially varied versions, which can help the model learn to recognize cacti in different positions and orientations. For instance, by shifting the entire point cloud so that the cactus appears off-center can help the model learn to detect cacti regardless of their position. Alternatively, parts of the point cloud can be cropped to simulate incomplete views of cacti, which is common

in real-world scenarios where a cactus might be partially obscured or cut off at the edges of the sensor's range. Implementing these data augmentation techniques can significantly reduce the model's bias towards cacti centered in point clouds, enhancing its robustness and improving its utility for the practical applications this work aims to facilitate.

4.2 Multilayer Perceptron Architecture

The multilayer perceptron (MLP) model encounters significant limitations when applied to the classification of point cloud data. A critical shortfall of MLPs is their inability to capture relationships between data points in a point cloud. In an MLP, each input data point is processed independently, without any regard for its spatial or temporal context. This independent processing means that MLPs cannot learn the structural dependencies or the arrangement of points within the point cloud, which can be vital for recognizing gneralizable patterns such as the shapes or configurations specific to columnar cacti. Given the nature of point clouds, where the spatial arrangement of points provides essential information about object shapes and sizes, MLPs lack the architectural support to leverage this spatial information. The model's architecture does not allow it to perceive the point cloud as a cohesive entity, thereby potentially focusing excessively on non-generalizable features found in the training set and misinterpreting or overlooking critical geometric cues that are crucial for accurate classification.

4.3 One-Dimensional Convolutional Neural Network Architecture

The 1D CNN model attempts to address the issue of capturing spatial relationships within point clouds by applying 1D convolutions. However, a fundamental challenge arises from the nature of point clouds, where the points are not arranged in a sequential manner. Within the computer, the point cloud is stored as a two-dimensional array that has the shape (P, F), where P represents how many points are in the point cloud, and F represents how many

features each point has. The convolution being applied in only one-dimension means that it is applied across the P points for each feature F. This gives the model the ability to learn points' spatial relationships, but only across an individual feature. Furthermore, the ordering of the P points in the array is the order in which convolution is done, meaning that the 1D convolution searches for patterns in points that are adjacent to one another in its array representation, and not necessarily points that are adjacent to one another in its point cloud representation. Figure 4.1 shows a visualization of how the array representation of the point cloud maps onto the actual 3D plot of the point cloud for the CG_01 cactus. The nine other point clouds have similar looking plots, where the ordering of the array seems to separate the point cloud into several different regions. As a result, the effectiveness of the 1D CNN in capturing spatial relationships among points is not always reliable in this context. The majority of the time, the convolution will contain adjacent points, but if these points lie close to the boundary of one of the regions, then the points that are being convolved may be spatially separated by a large amount and may lead to the model learning spurious relationships.

4.4 Potential for Three-Dimensional Convolutional Neural Networks

4.4.1 Three-Dimensional Convolution on Point Clouds

In contrast to MLPs and 1D CNNs, 3D convolutional neural networks (3D CNNs) are better equipped to handle the spatial data of point clouds. By performing convolutions in three dimensions as opposed to one, 3D CNNs can process volumetric data, effectively capturing the spatial relationships between points in a point cloud. This capability allows 3D CNNs to recognize and learn from the geometric and topological properties of the data, making them particularly suitable for applications like point cloud classification where the shape and structure of objects are critical. The use of 3D convolution enables the model to preserve the contextual relevance of each point and potentially improve the accuracy and robustness

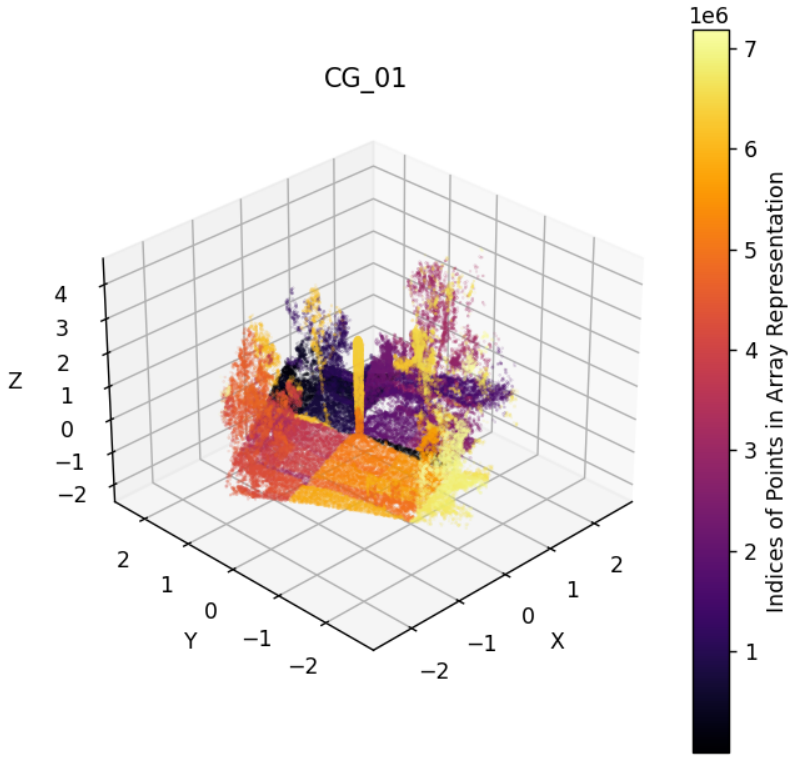

Figure 4.1: This plot shows how the points in the CG_01 point cloud are arranged in its array representation. The array has shape (100000, 6) due to its downsampling, three spatial coordinates, and three color values. Here the points are colored according to their index in that array. The maximum value is greater than 7 million because that was the original amount of points in the point cloud, and the indices were not adjusted during downsampling. The plot indicates that the points are arranged in distinct regions, and thus adjacent points in the array are very commonly adjacent in the point cloud. However, this relationship falls apart for points which lie on the boundaries of these regions, such as the points which fall at roughly X=0.

of the classification results.

Applying a 3D CNN to a point cloud with features such as XYZ coordinates and RGB colors involves treating the spatial dimensions (XYZ) and color dimensions (RGB) differently. The XYZ coordinates dictate how the points are arranged in three-dimensional space, whereas the RGB values would serve as feature channels similar to color channels in a conventional image. Before performing 3D convolution, the point cloud would first need to be

arranged on a regular grid, which can be accomplished through a process called voxelization. The entire 3D space of the point cloud would need to be subdivided into individual cells, called voxels, where each cell represents whether the space is occupied and by what features (colors). Due to the sparse nature of point clouds, very few of the voxels in the overall grid will be occupied. After this is done, convolutions are then performed across this voxel grid, where each 3D convolutional filter slides across the three spatial dimensions (XYZ) of the voxelized point cloud to process the RGB features and incorporate spatial relationships between the occupied voxels. This method allows the network to learn from both the spatial arrangement of points and their associated color features, making it powerful for tasks that require an understanding of the full context of the 3D space, such as object recognition through binary classification in point clouds.

4.4.2 Considerations of Computational Complexity

Computational demands of 3D CNNs are significantly higher than those of 1D CNNs or MLPs. The increased complexity presents several challenges arising from the volumetric nature of the data and the dimensionality of the convolutions. Firstly, storing and processing volumetric data requires substantial memory resources. Each convolutional layer in a 3D CNN must handle a 3D array whose size can grow exponentially with the addition of more layers or deeper networks, leading to exceedingly high memory consumption. Secondly, the computational cost of performing 3D convolutions across such a large dataset is substantial. Each filter must move across three dimensions and apply weighted operations at each position in the input volume, typically necessitating the use of powerful GPUs or even distributed computing systems to manage the workload efficiently. Thirdly, due to the extensive computations required, training times for 3D CNNs can be extremely lengthy. This makes model experimentation and tuning a slow process, hindering rapid development and iteration.

While 3D CNNs offer distinct advantages for handling complex spatial data like point clouds, the computational cost associated with these models is considerable. Teams or individuals considering the use of 3D CNNs must weigh these computational demands against the expected gains in model performance and ensure that sufficient computational resources

are available to support the development and deployment of these models. Due to the relatively small scope of this project, limited computational resources, and not wanting to introduce unneeded complexity, 3D CNNs were avoided in this work to instead focus on lighter-weight, faster-training model architectures, like the MLP and 1D CNN, to see how well they could handle the task of identifying columnar cacti in point clouds.

Chapter 5

Future Work and Conclusion

5.1 Future Work

The findings from this study open several avenues for further research. The most obvious avenue forward is to put this model to the test by feeding it point cloud data from an entire landscape that ecologists wish to analyze, instead of using it on individual specimens. If this model could effectively isolate the columnar cacti specimens in that case, it could mean massive amounts of time and effort saved for ecologists studying these plants. Gathering the volumes of these columnar cacti on a landscape-scale is currently a job that spans weeks to months, but given an effective enough model, could be transformed into one which lasts just a few days.

Another approach that was theorized but not tested in this work was to instead perform multi-class classification on the point clouds. With this approach, an additional class for the ground points would be added, in hopes that the model could improve upon its false positive rate by not confusing the ground around the cactus for being part of a cactus. Alternatively, one could implement a ground segmentation algorithm, such as in Huang, et. al. [19], on the point cloud and then feed the output into a binary classifier. Using an approach like this would likely lead to more isolated clusters of points, representing the sparse, above-ground plant communities in which these columnar cacti thrive, which may allow the model to classify the cacti more effectively.

Investigating alternative neural network architectures, like those incorporating spatially-aware layers [20], or 3D CNNs as discussed in section 4.4, could potentially improve the model's ability to understand the complex spatial relationships within point cloud data. Additionally, exploring advanced data preprocessing techniques, such as feature engineering, may be able to extract more informative attributes from the data and enhance model performance. Also, addressing the challenges of overfitting and bias through the usage of data augmentation techniques could lead to more robust classification outcomes. Finally, the methodologies developed in this study could easily be adapted and applied to the analysis of other ecosystems given the generalizability of neural networks.

5.2 Conclusion

This study embarked on the development of a neural network model for the binary classification of point clouds into cactus and non-cactus categories, aimed at facilitating volume calculations for ecohydrological studies. Two neural network architectures were tested for this application: the MLP and 1D CNN. Their performance matched closely, with the MLP having a slight advantage. These architectures were tested using a 10-fold cross-validation process, and demonstrated promising results, albeit with variability across different folds. The variability in performance across different datasets highlights the complexities of this task and opens up questions about model overfitting. Trade-offs between model simplicity and performance, challenges posed by data imbalance, and the importance of efficient training strategies are key considerations for future research in this area. As of right now, I believe that this model will save time for the volume and surface area calculation workflow for these columnar cacti species, but further work is needed to both test if the resulting volumes and surface areas are accurate and to improve the false positive rate without sacrificing the recall score. Even with the relatively few number of individual cacti in this dataset, this work marks a significant step forward in the application of neural network models to the classification of point cloud data, providing insights for future investigations and applications in which scientists aim to harness the power of machine learning in environmental contexts.

References

[1] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[2] K. Alhamzi, M. Elmogy, and S. Barakat, "3d object recognition based on local and global features using point cloud library," *International Journal of Advancements in Computing Technology*, vol. 7, pp. 43–54, 05 2015.

[3] G. Pang and U. Neumann, "Fast and robust multi-view 3d object recognition in point clouds," in *2015 International Conference on 3D Vision*, 2015, pp. 171–179.

[4] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," 2017.

[5] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," 2017.

[6] C. W. Sia, K. H. Lim, and J. T. S. Phang, "Comparative study of point cloud classification using deep learning neural networks," in *2023 International Conference on Digital Applications, Transformation Economy (ICDATE)*, 2023, pp. 1–5.

[7] S. A. Bello, S. Yu, C. Wang, J. M. Adam, and J. Li, "Review: Deep learning on 3d point clouds," *Remote Sensing*, vol. 12, no. 11, 2020. [Online]. Available: https://www.mdpi.com/2072-4292/12/11/1729

[8] L. Wang, W. Meng, R. Xi, Y. Zhang, C. Ma, L. Lu, and X. Zhang, "3d point cloud analysis and classification in large-scale scene based on deep learning," *IEEE Access*, vol. 7, pp. 55 649–55 658, 2019.

[9] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun, "Deep learning for 3d point clouds: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 12, pp. 4338–4364, 2021.

[10] A. Hazer and R. Yildirim, "Deep learning based point cloud processing techniques," *IEEE Access*, vol. 10, pp. 127 237–127 283, 2022.

[11] X. Ma, C. Qin, H. You, H. Ran, and Y. Fu, "Rethinking network design and local geometry in point cloud: A simple residual mlp framework," 2022.

[12] Y. Chen, M. Fu, and K. Shen, "Point-bls: 3d point cloud classification combining deep learning and broad learning system," in *2022 34th Chinese Control and Decision Conference (CCDC)*, 2022, pp. 2810–2815.

[13] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, and B. Ghanem, "Pointnext: Revisiting pointnet++ with improved training and scaling strategies," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 23 192–23 204. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/9318763d049edf9a1f2779b2a59911d3-Paper-Conference.pdf

[14] J. P. Carbonell-Rivera, J. Estornell, L. A. Ruiz, J. Torralba, and P. Crespo-Peremarch, "Classification of uav-based photogrammetric point clouds of riverine species using machine learning algorithms: A case study in the palancia river, spain," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B2-2020, pp. 659–666, 2020. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLIII-B2-2020/659/2020/

[15] S. Dersch, M. Heurich, N. Krueger, and P. Krzystek, "Combining graph-cut clustering with object-based stem detection for tree segmentation in highly dense airborne lidar point clouds," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 172, pp. 207–222, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924271620303233

[16] Z. Chen, H. Lin, H. Sun, C. Li, J. Li, and D. Kai, "Extraction of pinus massoniana tree crown from unmanned aerial images," in *2018 Fifth International Workshop on Earth Observation and Remote Sensing Applications (EORSA)*, 2018, pp. 1–5.

[17] E. López-Jiménez, J. I. Vasquez-Gomez, M. A. Sanchez-Acevedo, J. C. Herrera-Lozada, and A. V. Uriarte-Arcia, "Columnar cactus recognition in aerial images using a deep learning approach," *Ecological Informatics*, vol. 52, pp. 131–138, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1574954119300895

[18] Z. Li, F. Liu, W. Yang, S. Peng, and J. Zhou, "A survey of convolutional neural networks: Analysis, applications, and prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, no. 12, pp. 6999–7019, 2022.

[19] W. Huang, H. Liang, L. Lin, Z. Wang, S. Wang, B. Yu, and R. Niu, "A fast point cloud ground segmentation approach based on coarse-to-fine markov random field," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 7841–7854, 2022.

[20] C. Kaul, N. Pears, and S. Manandhar, "Sawnet: A spatially aware deep neural network for 3d point cloud processing," 05 2019.