Approach 2 - Image Classification

i. First implementation 15 channels

1. Initial Data Scenario:

Our dataset comprises point cloud data acquired through airborne laser scanning (LAS). Each point in these clouds possesses a rich array of 22 attributes, including local and global coordinates, color values, intensity measurements, planarity, linearity, sphericity, and verticality indicators. Additional features include mean intensities along both X and Y axes, edge area, and intensity principal gradient positions. These point clouds have been meticulously labeled into six distinct classes, each exhibiting varying sample sizes: 2 lanes (473 samples), 3 lanes (193 samples), crossing (133 samples), split 4 lanes (164 samples), split 6 lanes (220 samples), and transition (242 samples).

2. Data Preprocessing Strategy:

Having delved into the intricacies of our dataset and the underlying classification challenge, we embarked on a transformational journey by reframing the problem as an image classification task. This shift was catalyzed by the realization that our point clouds essentially represent 2D planes with diverse surface attributes and intensity characteristics. Our preprocessing pipeline unfolded as follows:

2.1. Constructed an unlinked grid, partitioning the data into pixels, where each pixel encapsulates the mean values of the individual features of the points.

2.2. Generated image representations, efficiently stored as numpy files, characterized by a shape of (Height x Width x 15), encapsulating the 15 essential attributes.

2.3. Standardized the images by resizing them to a uniform dimension of (124 x 124 x 15), ensuring consistency in the subsequent analysis.

3. CNN Model Implementation:

The transformed data was seamlessly integrated into a Convolutional Neural Network (CNN) architecture, meticulously crafted within the framework of the Keras_CNN_15Features.ipynb code.

ii. Implementation of 3-Channel Image Processing for Point Cloud Data Visualization and Classification

Step 1: Conversion of Point Cloud Data to Txt Format

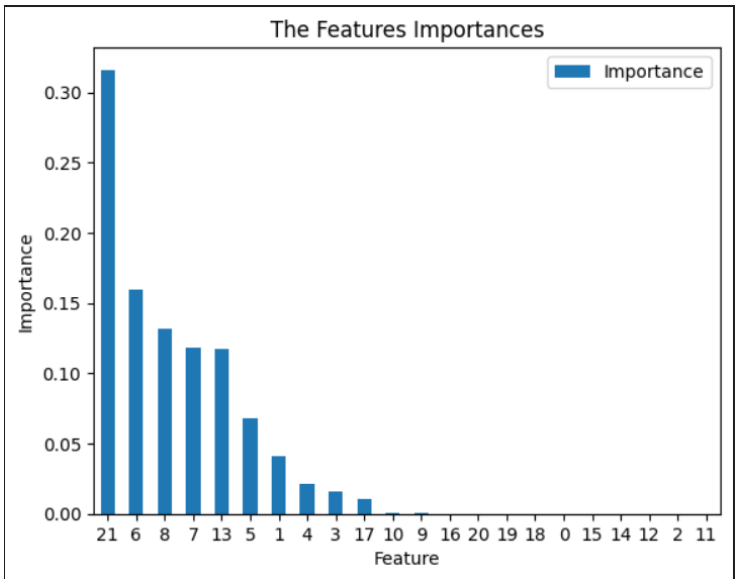
The initial step involves converting the point cloud data, stored in the NumPy format, into a text file using the "Num2TXT.py" script.

Step 2: Data Visualization and Preprocessing

The subsequent phase encompasses data visualization and preprocessing through the utilization of various scripts and tools.

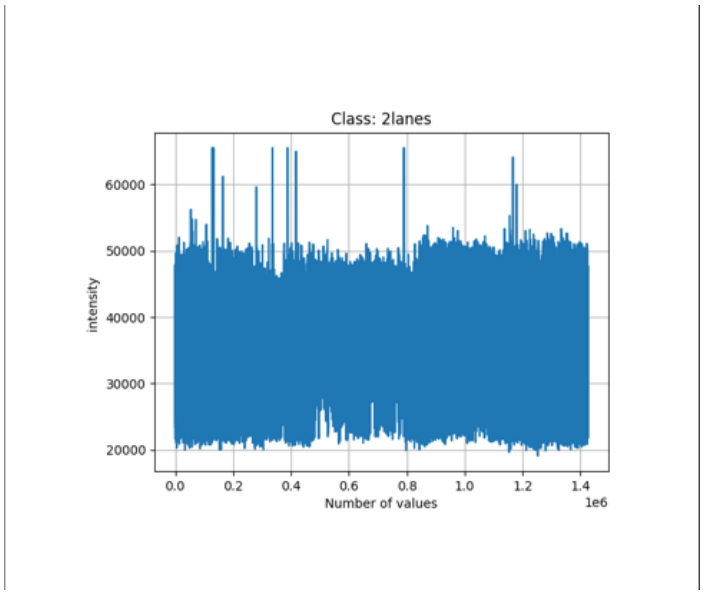
1. Visualization of Decision Tree Results from Approach 1

   Integrating the outcome of the decision tree from the preceding approach, an inset image is generated. This provides a visual representation of the decision tree's outcomes.

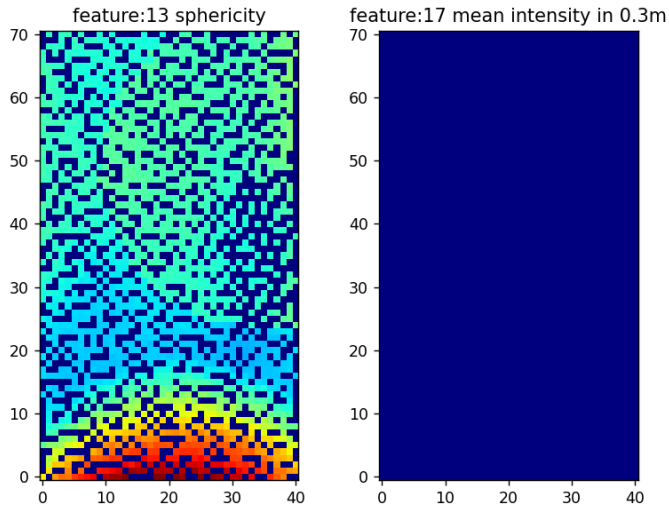


2. Utilization of Feature Value Range Analysis

Leveraging the "features_value_range.pptx" PowerPoint file and, insights into the value ranges of features are gained. This analysis assists in understanding the distribution and scope of feature values.



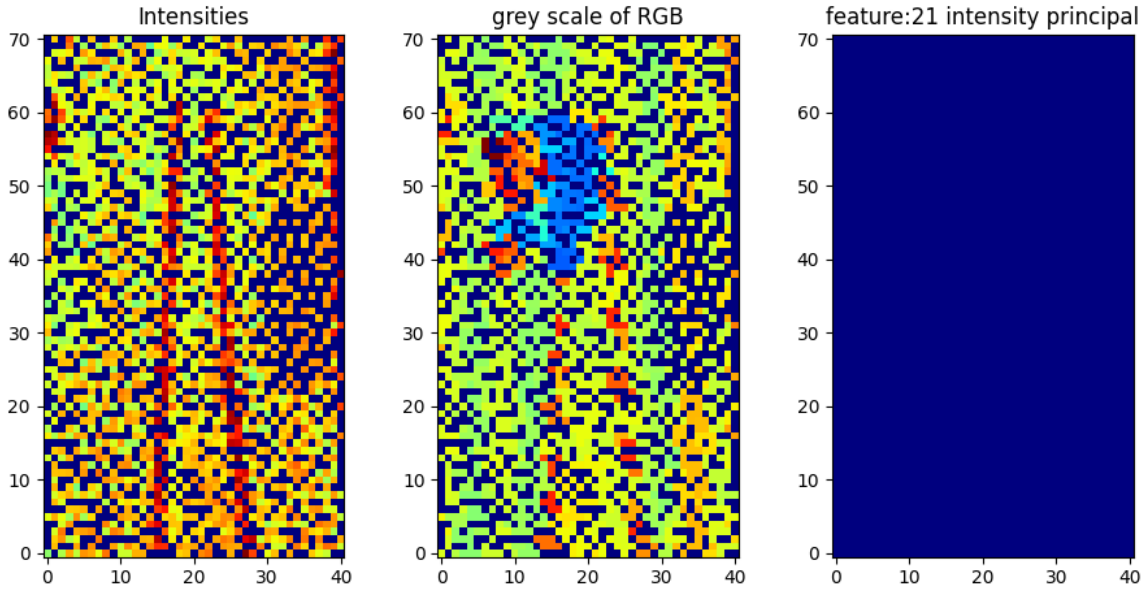
3. Utilizing the "txt2jpg_visualize.py" script to gain insights into the data and facilitate the process of feature selection.



This image from transition folder, Tile6

4. Conversion of Data to JPG for Channel Analysis

To facilitate feature selection for image channels, a selection of images is converted to JPG format using the "txt2jpg.py" script.



Step 3: Generation of Multi-Channel Images and Classification

1. Conversion of Data to JPG Images

The "txt2jpg.py" script is further employed in this step, involving several stages:

a. Projecting the 3D point cloud onto an occupancy grid using the (x, y) values.

b. Selection of relevant features based on graphs and the decision tree outcomes.

c. Extraction of intensity values, which are then normalized by dividing by 65535 (the maximum intensity value across the dataset) to form Channel 1.

d. Conversion of RGB colors from the point cloud into grayscale for Channel 2.

e. Extraction of intensity principal gradient positions, guided by the decision tree outcomes, for Channel 3.

2. Implementation of Classification Models

2.1. Classification from Scratch

Utilizing the "classification_from_scratch.ipynb" notebook, the following steps are taken:

a. Resizing images to dimensions (64, 64).

b. Partitioning data into an 80% training set and a 20% validation set.

c. Applying data augmentation techniques, such as rotation and flipping.

d. Incorporating augmented data into the training set.

e. Defining the machine learning model.

f. Executing the machine learning model.

g. Evaluation of model performance.

2.2. ResNet18 Classification

Implementation of the ResNet18 classification model is carried out using the "RESNET18.ipynb" notebook:

a. Resizing images to dimensions (256, 256).

b. Division of data into an 80% training set and a 20% validation set.

c. Application of data augmentation techniques, including rotation and flipping.

d. Integration of augmented data into the training set.

e. Definition of the ResNet18 model architecture.

f. Configuration of model parameters.

g. Assessment of model performance.

Iterative Exploration of Multi-Channel Datasets

The experimentation process involves iterating through different datasets containing 1, 2, and 3 channels. This exploration aims to analyze the impact of channel composition on image visualization and classification outcomes.