Comprehensive Project Report: Cassava Leaf Disease Classification

Table of Contents

---

1. Introduction

Cassava is a crucial crop for food security in many tropical regions. Detecting diseases in cassava leaves is vital for maintaining crop health and ensuring food production. This project aims to classify cassava leaves into different disease categories or identify them as healthy using various deep learning models. The models mainly used include Convolutional Neural Network (CNN), if computational resources are limited can consider MobileNetV2, ResNet50, VGG16, EfficientNetB0/B1/B2, and InceptionNet pretrained model.

---

2. Dataset

The dataset for the Cassava Leaf Disease Classification Challenge is a comprehensive collection of annotated images representing various common diseases affecting cassava plants, one of the most crucial crop resources in tropical and subtropical regions. It includes thousands of high-

resolution images categorized into several disease classes, as well as a category for healthy leaves. This dataset provides a rich resource for developing and testing machine learning models aimed at identifying and classifying plant diseases based on visual evidence.

The dataset [1] used for this project is the Cassava Leaf Disease Classification dataset from Kaggle.

It consists of images of cassava leaves categorized into five classes:

1. Cassava Bacterial Blight (CBB)
2. Cassava Brown Streak Disease (CBSD)
3. Cassava Green Mottle (CGM)
4. Cassava Mosaic Disease (CMD)
5. Healthy

The dataset is divided into separate directories for training and testing, each containing subdirectories for each class.

---

3. Exploratory Data Analysis (EDA)

3.1 Distribution of Classes

- Assess Balance: Check the distribution of images across different disease categories and healthy leaves. This can be done by counting the number of images per category and visualizing this information. Imbalanced datasets may require strategies like oversampling, undersampling, or generating synthetic samples to address class imbalance. It was found that the dataset had an imbalanced distribution across the different classes, which could affect the model performance.

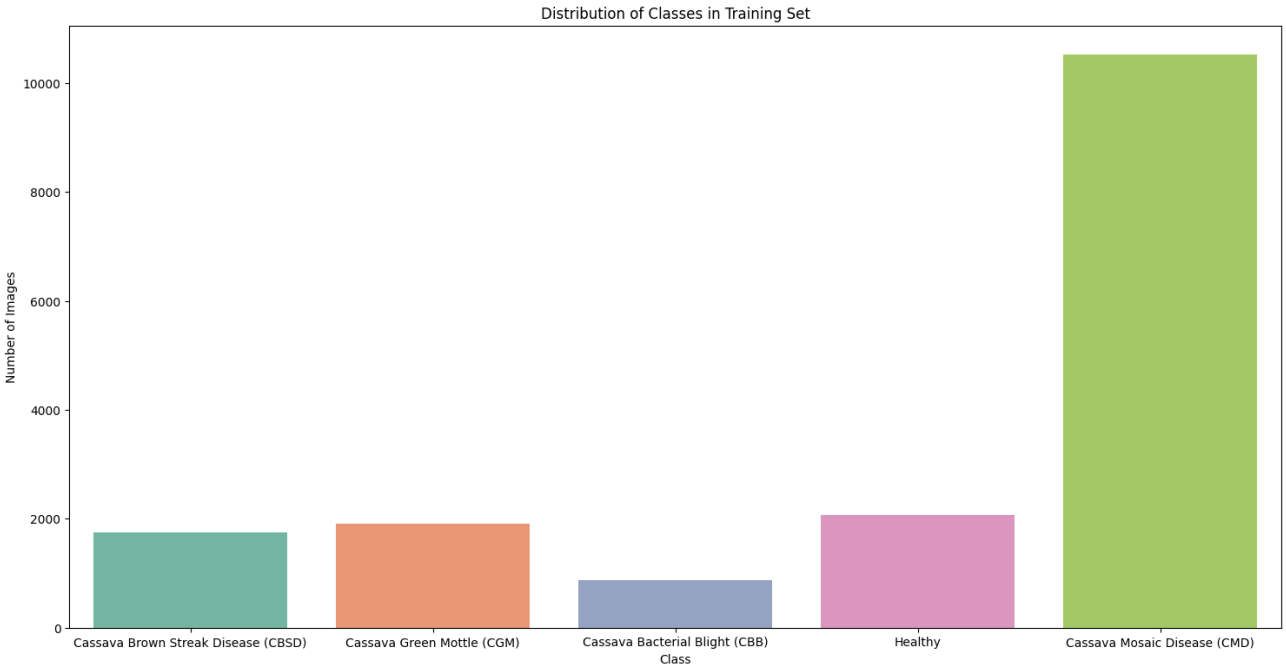

Figure 1: Distribution of Classes in Training Set

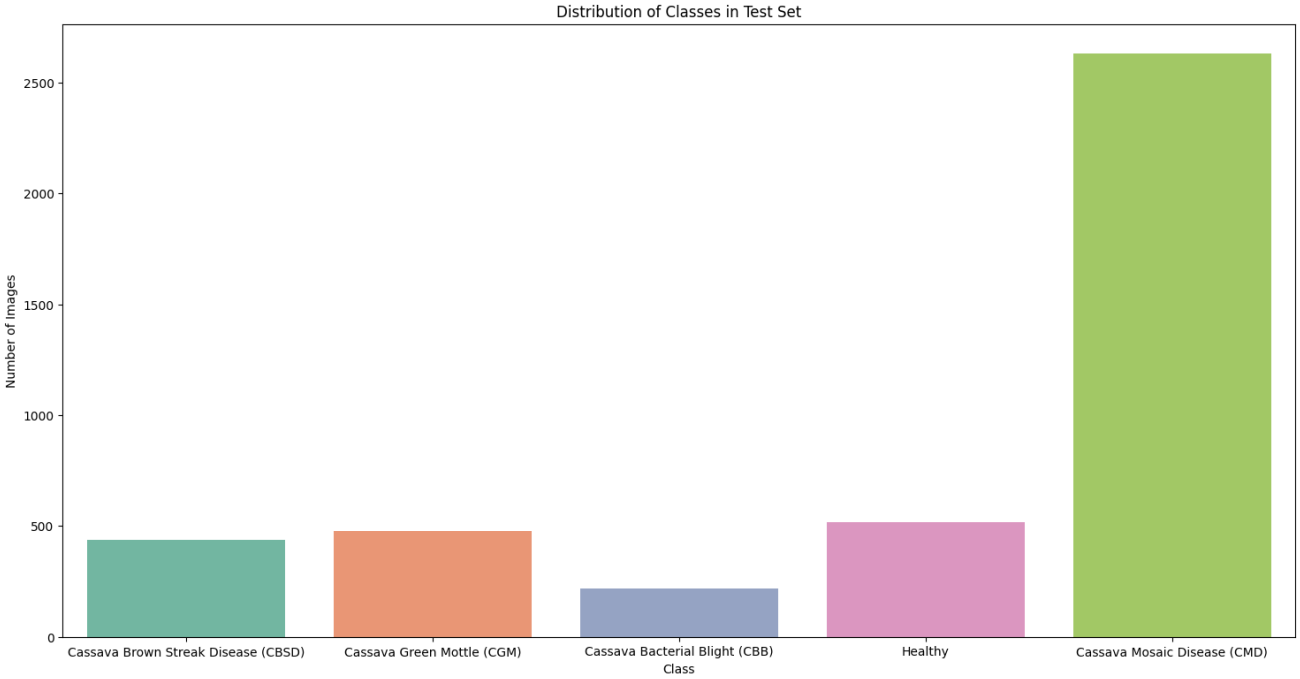

Figure 2: Distribution of Classes in Test Set

- Visual Inspection: Sample images from each class were visually inspected to assess image quality and variability. The images varied in resolution, lighting conditions, and angles, presenting a challenge for model training. This helps in identifying potential issues that might need preprocessing steps like resizing, normalization, or augmentation.

Figure 3: Sample images from train data set

3.2 Data Insights

- Anomaly Detection: Look for unusual patterns such as extremely high-resolution images, very low-quality images, or images that do not clearly represent cassava leaves. These could affect model training negatively.

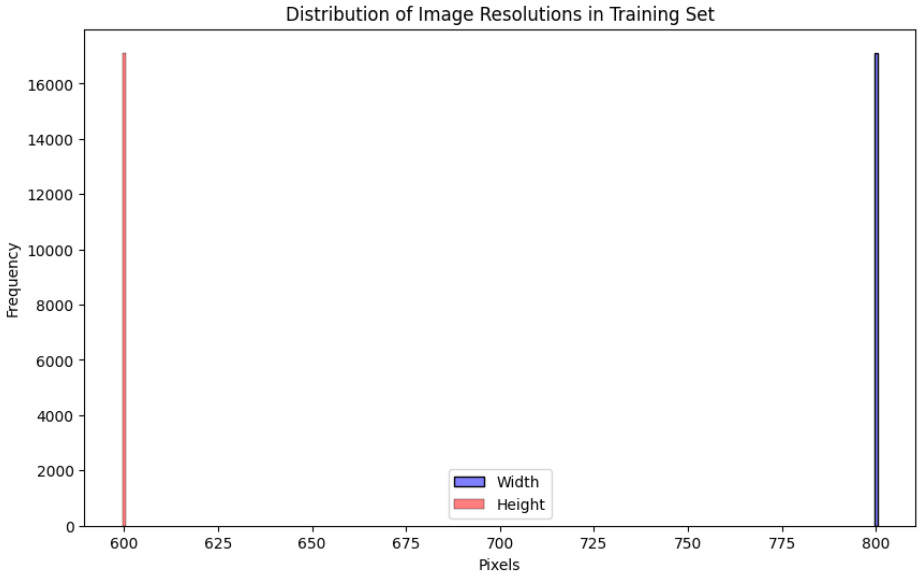

Figure 4: Image Resolutions in Training Set

We have already performed EDA as outlined in the previous steps. The key findings are:

1. **Class Distribution**: The dataset is imbalanced with certain classes having more images than others.
2. **Image Quality**: The images vary in resolution and quality.
3. **Anomalies**: Some images have significantly different resolutions.

---

4. Data Preprocessing

4.1 Image Resizing

- Standardize the size of all images to a fixed dimension while preserving the aspect ratio. This ensures that the input to the neural network is consistent, reducing the computational overhead during training. All images were standardized to a fixed size of 224x224 pixels.

4.2 Normalization

- Normalize pixel values to a range between 0 and 1 or -1 and 1. This scales the input data to a standard range, facilitating faster convergence during training. All images were normalized to have pixel values with a mean of [0.485, 0.456, 0.406] and a standard deviation of [0.229, 0.224, 0.225].

4.3 Augmentation

- Implement data augmentation techniques such as rotation, scaling, flipping, and cropping to artificially increase the size of the dataset. This helps in improving the model's robustness and preventing overfitting. In this project applied techniques such as random horizontal and vertical flips, random rotations, and color jitter were applied to increase dataset diversity and reduce overfitting.

---

5. Model Engineering

5.1 Dataset Splitting

- Divide the train dataset into approximately 85% for training, 15% for validation. This split ensures that the model has enough data to learn from while also having separate sets to validate and test its performance.

5.2 Model Architecture

- Design a Convolutional Neural Network (CNN) architecture with several convolutional layers followed by max-pooling layers, and finally, one or more fully connected layers at the end. Activation functions like ReLU can be used for hidden layers, and softmax for the output layer to handle multi-class classification.
- Consider using transfer learning with pre-trained models like VGG16, ResNet, or MobileNet, especially if computational resources are limited. Fine-tuning these models involves training them on the new dataset while keeping some of the lower-level layers frozen.

---

6. **Model Training and Validation**

- Train the model using the training set and evaluate its performance on the validation set after each epoch. Each model was trained for 25 epochs using the Adam optimizer and a learning rate of 0.001. The training process involved monitoring the training and validation loss and accuracy at each epoch.

7. Evaluation and Analysis

7.1 Performance Testing

- Evaluate the trained model on the test set to measure its generalization capability. This gives an unbiased estimate of how well the model will perform on unseen data.

7.2 Metrics Reporting

- Calculate and report various performance metrics such as accuracy, precision, recall, and F1 score for each class and overall. These metrics provide a comprehensive view of the model's performance, highlighting areas where the model excels and where improvements are needed.

Classification Report

Detailed classification reports were generated for each model, showing the precision, recall, and F1-score for each class.

	precision	recall	f1-score	support
Cassava Bacterial Blight (CBB)	0.43	0.38	0.40	217
Cassava Brown Streak Disease (CBSD)	0.57	0.32	0.41	438
Cassava Green Mottle (CGM)	0.57	0.33	0.42	477
Cassava Mosaic Disease (CMD)	0.80	0.96	0.87	2632
Healthy	0.47	0.39	0.43	516
accuracy			0.72	4280
macro avg	0.57	0.47	0.51	4280
weighted avg	0.69	0.72	0.70	4280

Figure 5: Classification Report for CNN

Additionally, the Receiver Operating Characteristic (ROC) curve and Area Under the Curve (AU C) were plotted for each model to assess their performance.

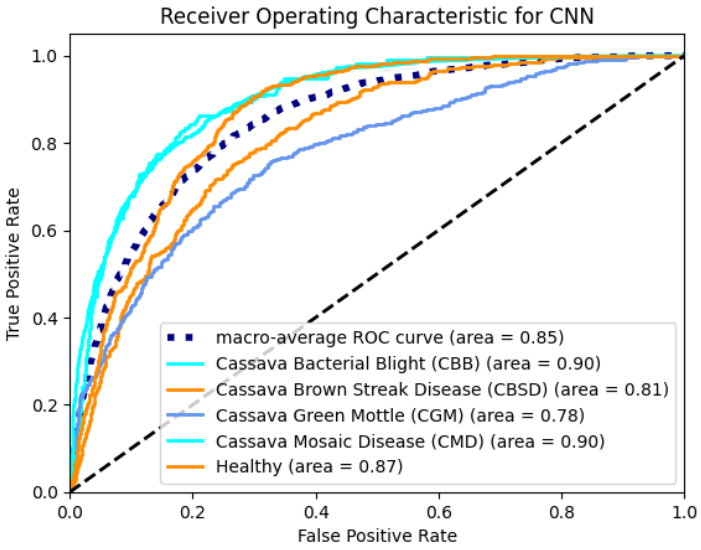

Figure 6: ROC curve for CNN model

Sample images from the test set were displayed with their true and predicted labels to visually inspect the model performance. This helped in understanding the types of errors made by the models.

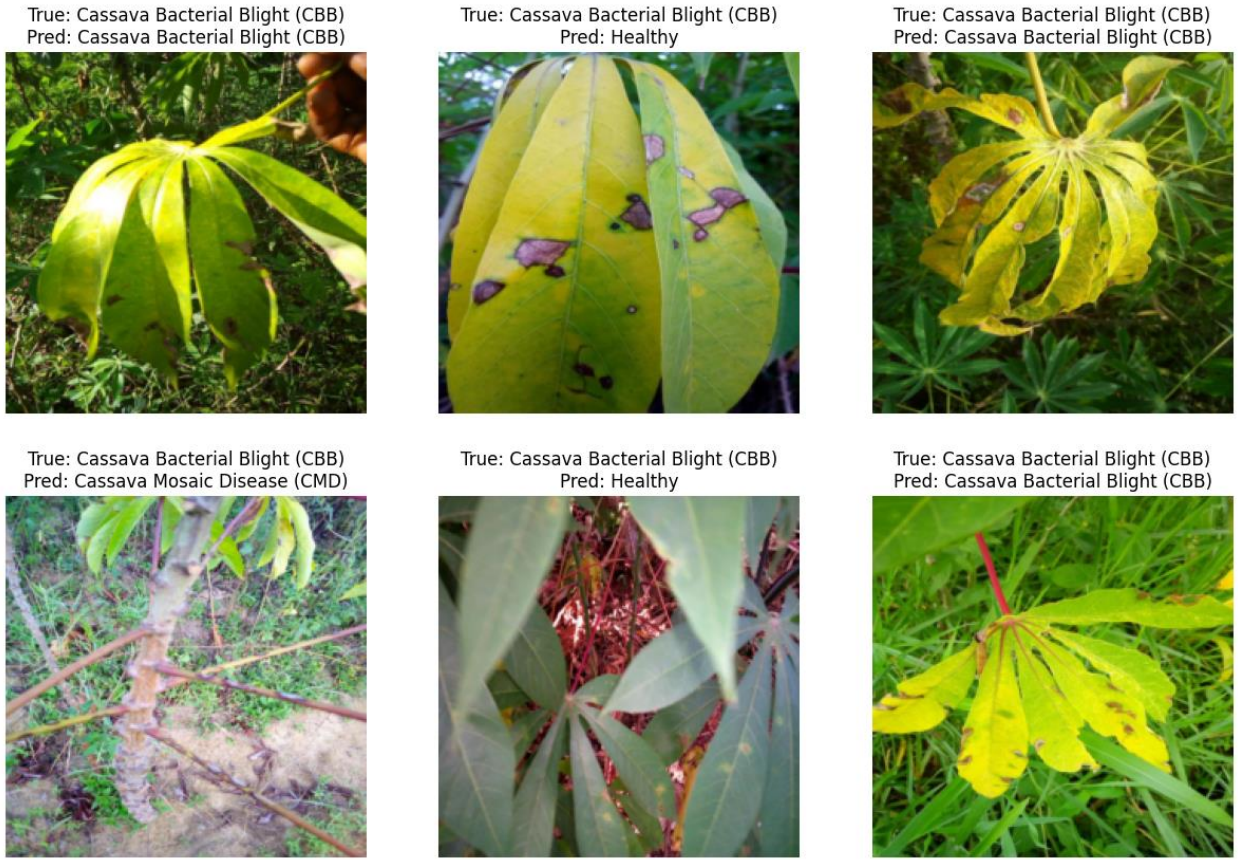

Figure 7: Display Some Test & predicted Images with Labels

8. **Conclusion**

This structured approach, combining EDA, preprocessing, model engineering, and evaluation, provides a solid foundation for tackling the Cassava Leaf Disease Classification problem effectively. Here in this, CNN model showing accuracy 0.72 percent though transformer model may provide higher accuracy. Such as, I trained the with model MobileNetv2 and got highest accuracy 80.96 percent.

References

[1] Cassava Leaf Disease Classification Dataset
https://www.kaggle.com/datasets/gauravduttakiit/cassava-leaf-disease-classification