Bytes Of Intelligence
Exploring AI's Secrets

Cassava Leaf Disease Classification by Custom Model

Email abdullahsakib33@gmail.com GitHub abdullahsakib LinkedIn abdullah sakib33
Phone +8801829356820

Table of Contents

1. Introduction

1.1 Program Overview

This project was done as a part of "Bytes of Intelligence: Data Science and AI Internship Program," This program provides a comprehensive learning experience in data science and AI through workshops, challenges, and mentorship. Which is an innovative platform designed to propel aspiring data scientists and AI enthusiasts into the forefront of technological advancement and real-world problem-solving.

2. Data Source

2.1 Dataset Overview

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 porvides 5 folder/class of data, some data are incorrect as inside one named folder one may find data of another folder. Dataset is highly imbalace. Most of them are Cassava Mosaic Disease (CMD).

2.2 Accessing the Dataset

The dataset is hosted on Kaggle, a popular platform for data science competitions and collaborative projects.

3. Task Specifications

3.1 Data Management

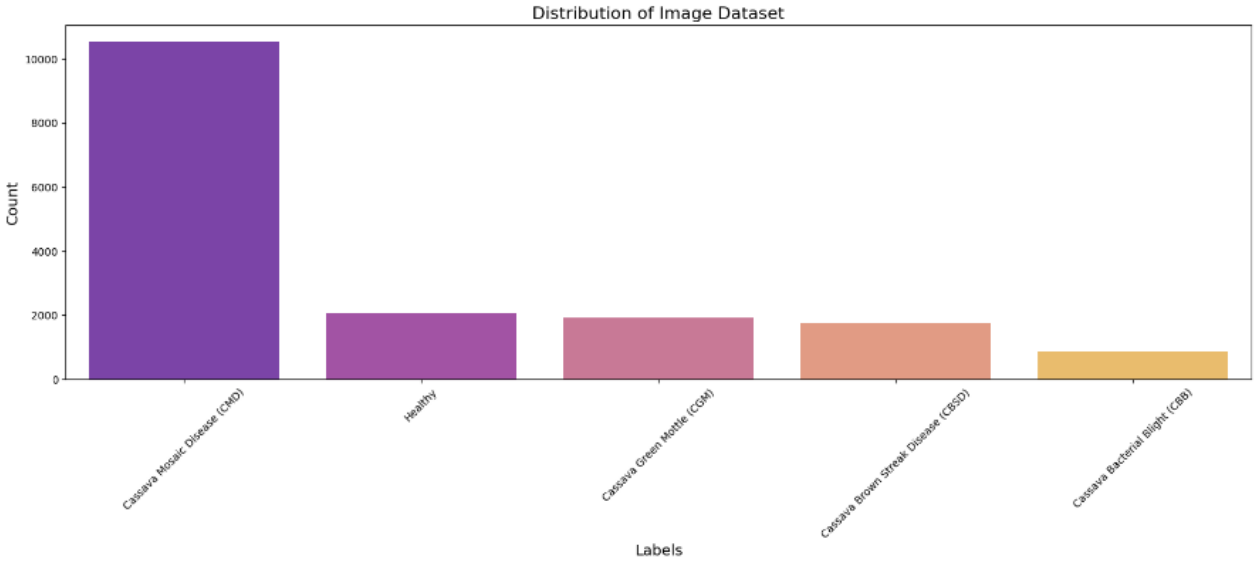
3.1.1 Data Acquisition

Data was downloaded from Kaggle

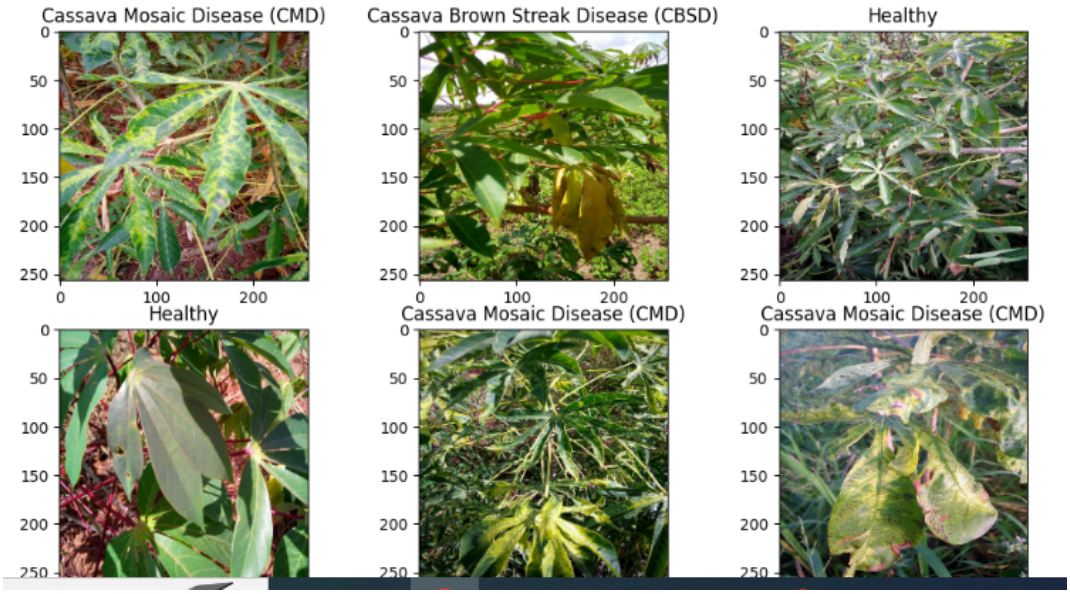
3.1.2 Exploratory Data Analysis (EDA)

Conduct an in-depth EDA to understand the dataset's characteristics:

- **Distribution of Classes:** There was 5 class:
    - 'Cassava Mosaic Disease (CMD)': 10526,
    - 'Healthy': 2061,
    - 'Cassava Green Mottle (CGM)': 1909,
    - 'Cassava Brown Streak Disease (CBSD)': 1751,
    - 'Cassava Bacterial Blight (CBB)': 870}



- **Image Quality and Variability:** Most of the image was high resulation having 800*600 shape.
- **Data Insights:** Some data are incorrect as inside one named folder one may find data of another folder. Dataset is highly imbalace. Most of them are Cassava Mosaic Disease (CMD).



3.1.3 Data Preprocessing

Preparation of the dataset for modeling: Data set was turned into a pandas dataframe having label and image data. Data was balanced and model was tested on both balanced and unbalanced data.

As data was imbalance augmentation was slightly incresing the performance in the compensation of huge amount of time.

- **Image Resizing:** image was resized to 256*256 Standardize image sizes while maintaining aspect ratios.
- **Rescaling:** Normalization was done to 0 to 1 range.
- and other augmentation like Randomflip, RandomRotation, RandomZoom, RandomConstrast was done.

3.2 Model Engineering

Test was done on both custom model and pretrained model.

Mostly used Convolutional and Maxpooling layer were used repetedly. Then in the second part dense and dropout layer used after flattening.

Layer (type)	Output Shape	Param #
input_layer_4 (InputLayer)	(None, 256, 256, 3)	0
sequential (Sequential)	(None, 256, 256, 3)	0
conv2d_19 (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d_19 (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_20 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_20 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_21 (Conv2D)	(None, 60, 60, 64)	36,928
max_pooling2d_21 (MaxPooling2D)	(None, 30, 30, 64)	0
conv2d_22 (Conv2D)	(None, 28, 28, 64)	36,928
max_pooling2d_22 (MaxPooling2D)	(None, 14, 14, 64)	0
conv2d_23 (Conv2D)	(None, 12, 12, 128)	73,856
max_pooling2d_23 (MaxPooling2D)	(None, 6, 6, 128)	0
conv2d_24 (Conv2D)	(None, 4, 4, 128)	147,584
max_pooling2d_24 (MaxPooling2D)	(None, 2, 2, 128)	0
flatten_3 (Flatten)	(None, 512)	0
dense_9 (Dense)	(None, 256)	131,328
dropout_6 (Dropout)	(None, 256)	0
dense_10 (Dense)	(None, 128)	32,896
dropout_7 (Dropout)	(None, 128)	0
dense_11 (Dense)	(None, 5)	645

Total params: 479,557 (1.83 MB)
Trainable params: 479,557 (1.83 MB)

3.2.1 Dataset Splitting

The dataset was divided into three subsets: Train dataset provided by kaggle divided into 2 set

- Training Set: The largest portion, used to train the model. 80% of the Train dataset provided by kaggle was used for training.
- Validation Set: Used to tune model parameters and prevent overfitting. 20% of the Train dataset provided by kaggle was used as validation dataset.
- Test dataset provided by kaggle was reserved for evaluating the model's performance on unseen data.

3.2.2 Model Architecture

Layer Structure:

- In the first section 7 Convolutional layer followed by 7 Maxpooling layer used , number of neurons were 32, 64 & 128. kernel size were (3,3) .

```
input_shape = (256, 256, 3)

inputs = Input(shape=input_shape)

x = augment(inputs)
x = Conv2D(32, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(64, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
x = Conv2D(128, (3, 3), activation='relu')(x)
x = MaxPooling2D((2, 2))(x)
```

- In the second part 2 dense layer were used followed by dropout layer after flattening. In this case number of neurons or filters were 256, 128 and 35% and 30% dropout was done to prevent overfitting.

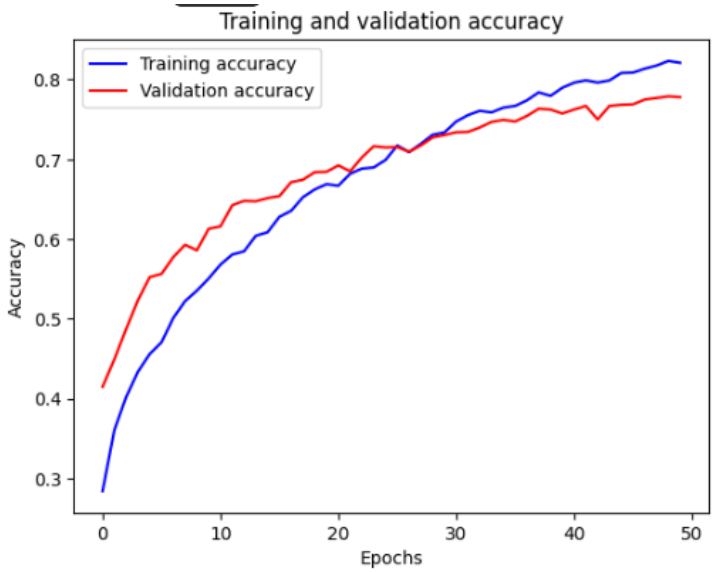
```
x = Flatten()(x)
x = Dense(256, activation='relu')(x)
x = Dropout(0.35)(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.30)(x)
outputs = Dense(5, activation='softmax')(x)
```

- **Activation Functions:** Except the last layer where "Softmax" activation function was used in all other case "Relu" activation function was used.
- **Transfer Learning:** Pretrained EfficientNetB0 model was used to compare the performance.

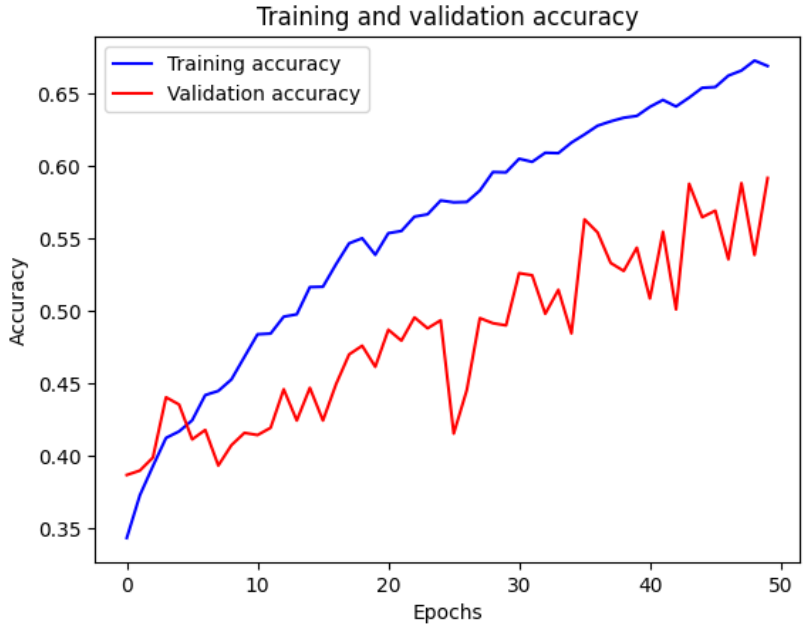
3.2.3 Model Training and Validation

Model was tranied on both balanced and unbalanced data for 50 epoch.

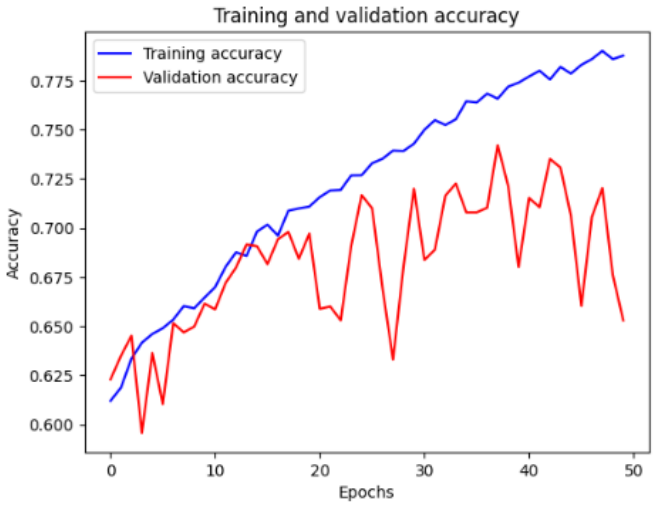
- Training and validation accuracy by EfficientNetB0 on balanced dataset



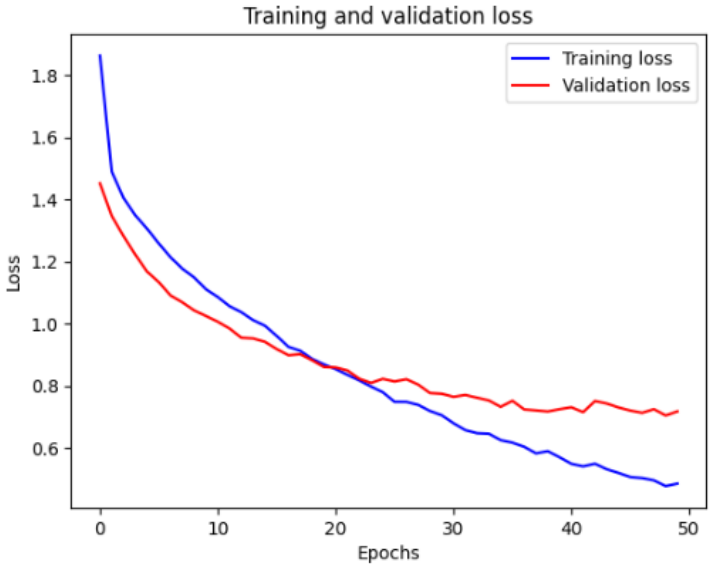
- Training and validation accuracy by custom model on balanced dataset



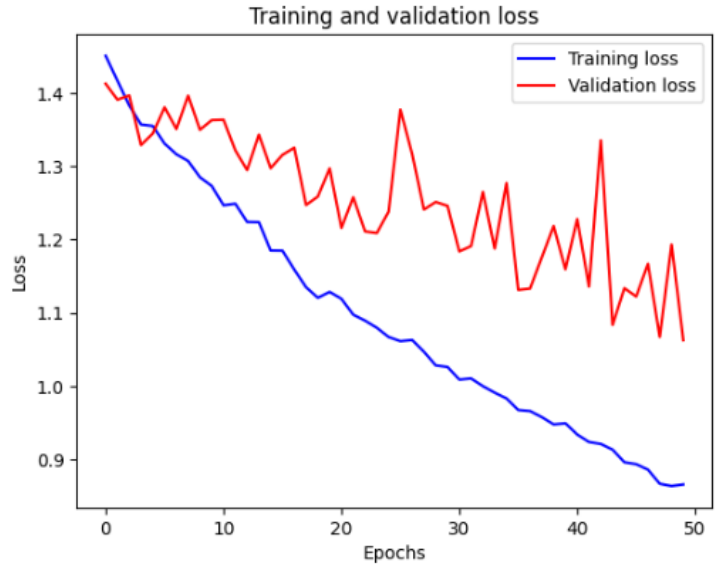
- Training and validation accuracy by custom model on un-balanced dataset



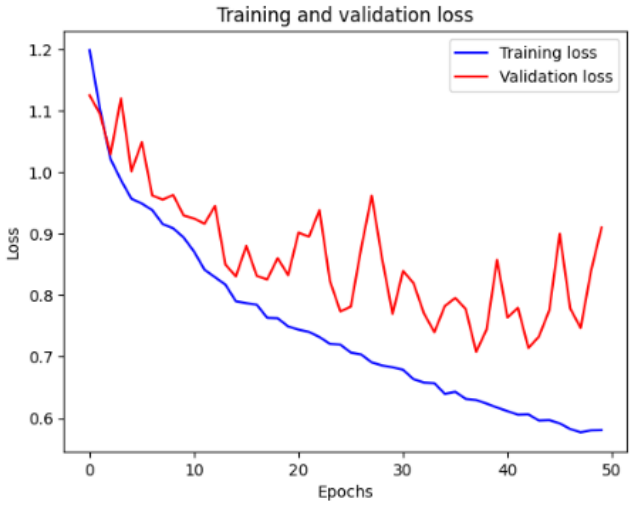
- Training and validation loss by EfficientNetB0 on balanced dataset



- Training and validation loss by custom model on balanced dataset



- Training and validation loss by custom model on un-balanced dataset



3.3 Evaluation and Analysis

- Validation and test accuracy by EfficientNetB0 on balanced dataset

```
results = model.evaluate(val_images, verbose=0)

print("    VAL Loss: {:.5f}".format(results[0]))
print("VAL Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    VAL Loss: 0.71788
VALAccuracy: 77.80%
```

+ Code + Markdown

```
results = model.evaluate(test_images, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 1.11245
Test Accuracy: 64.77%
```

- Validation and test accuracy by custom model on balanced dataset

```
results = model.evaluate(val_images, verbose=0)

print("    Val Loss: {:.5f}".format(results[0]))
print("Val Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Val Loss: 1.06270
Val Accuracy: 59.15%
```

```
results = model.evaluate(test_images, verbose=0)

print("    Test Loss: {:.5f}".format(results[0]))
print("Test Accuracy: {:.2f}%".format(results[1] * 100))
```

```
    Test Loss: 1.12340
Test Accuracy: 54.81%
```

- Validation and test accuracy by custom model on un-balanced dataset

```
results = model.evaluate(val_images, verbose=0)

print("Val Loss Unbalanced data: {:.5f}".format(results[0]))
print("Val Accuracy Unbalanced data: {:.2f}%".format(results[1] * 100))
```

```
Val Loss Unbalanced data: 0.90991
Val Accuracy Unbalanced data: 65.30%
```

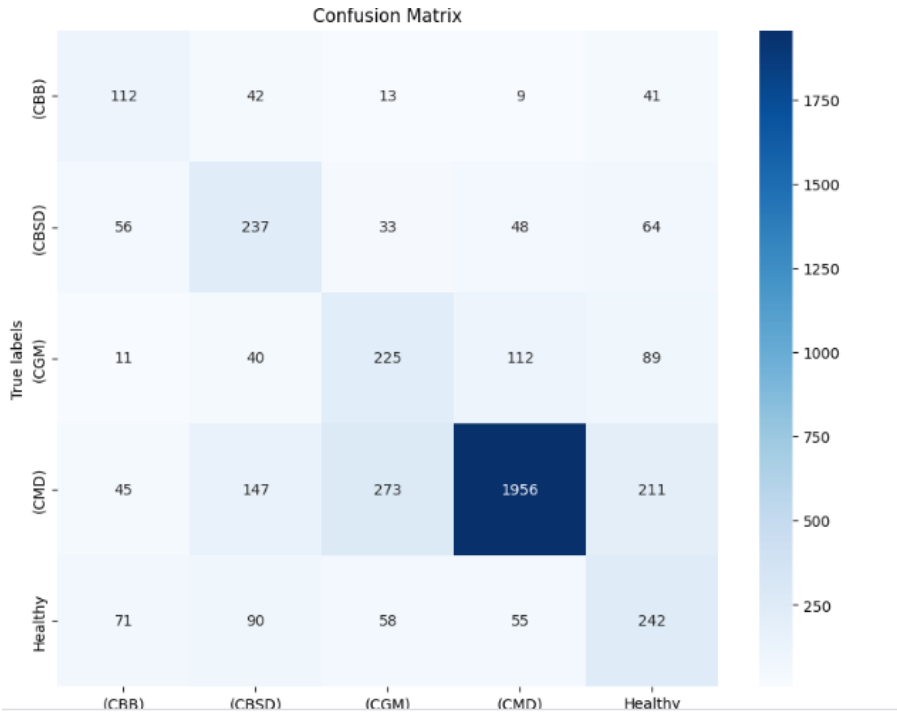
```
results = model.evaluate(test_images, verbose=0)

print("Test Loss Unbalanced data: {:.5f}".format(results[0]))
print("Test Accuracy Unbalanced data: {:.2f}%".format(results[1] *100))
```

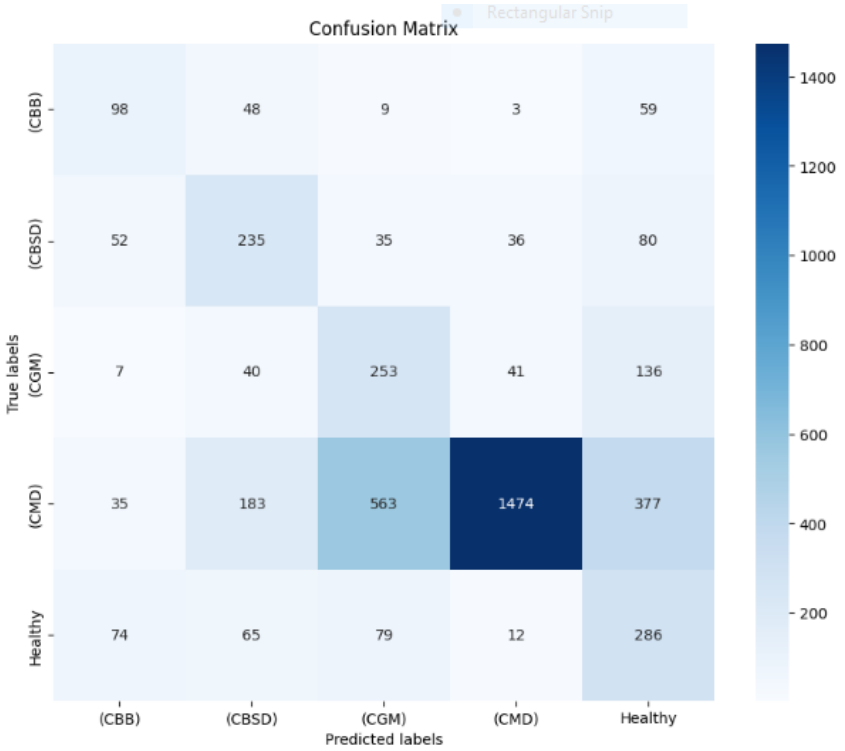
```
Test Loss Unbalanced data: 0.93445
Test Accuracy Unbalanced data: 64.18%
```

3.3.1 Performance Testing

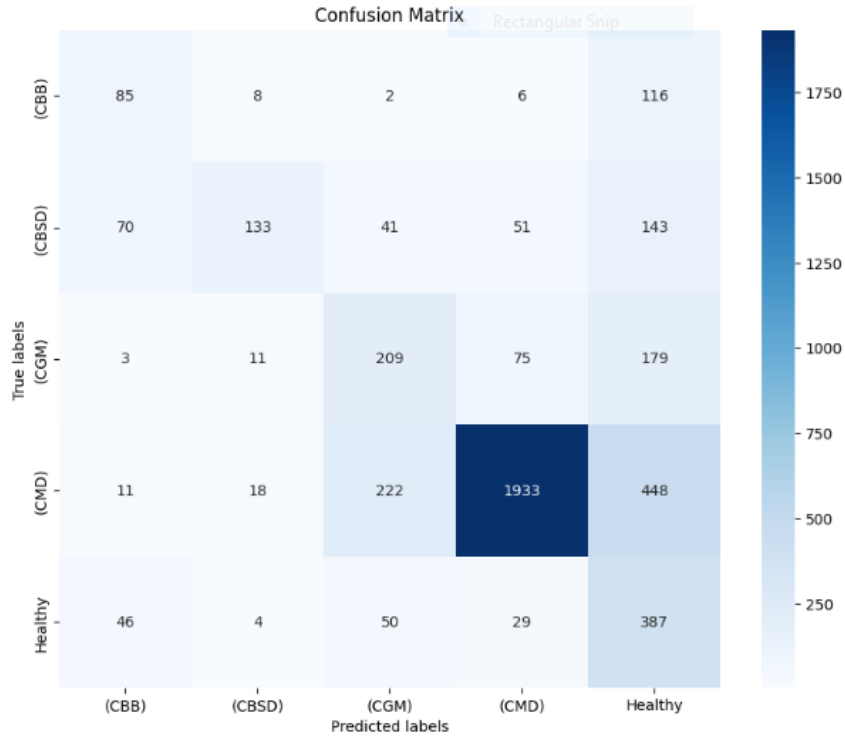
- Confusion Matrix by EfficientNetB0 on balanced dataset



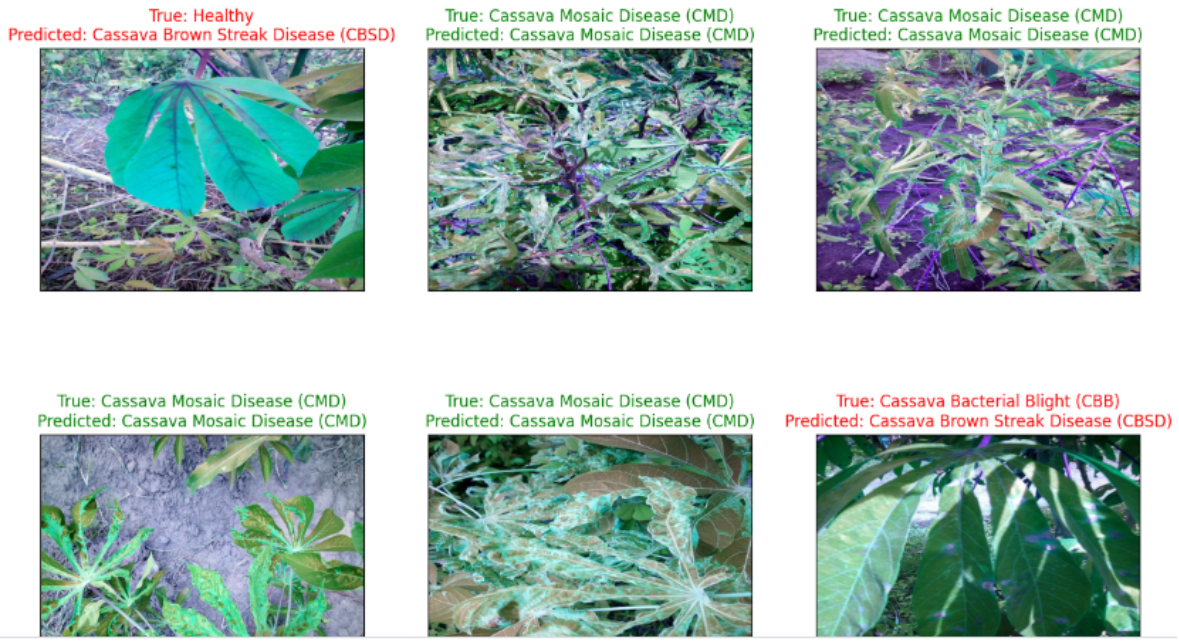
- Confusion Matrix by custom model on balanced dataset



- Confusion Matrix by custom model on un-balanced dataset

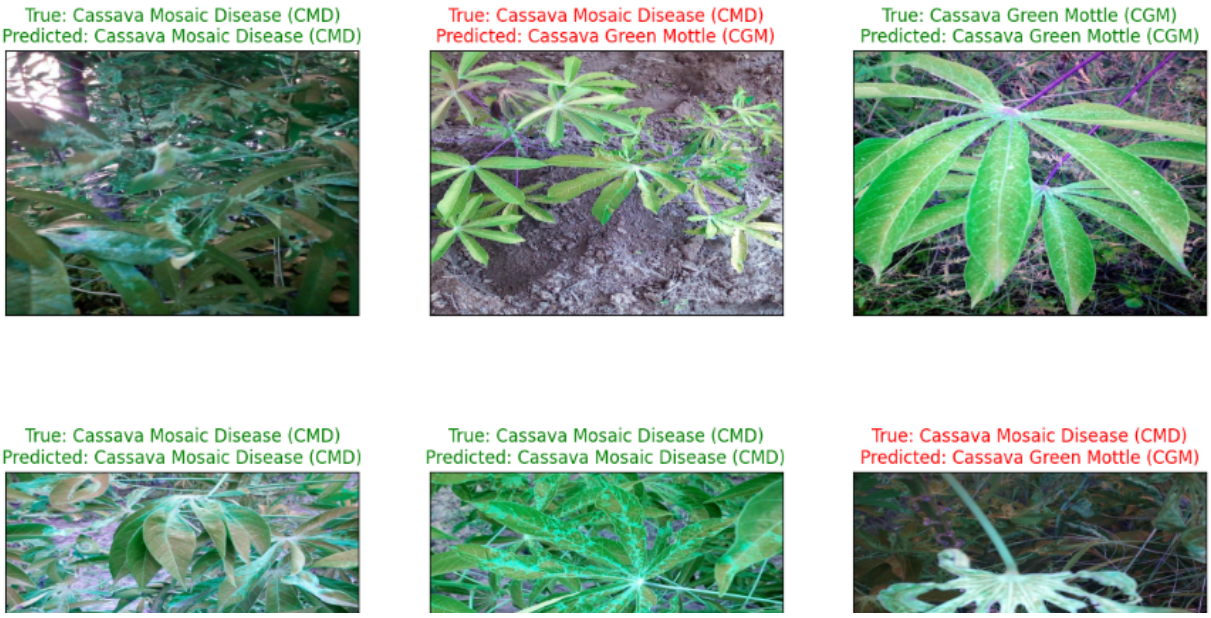


- image prediction by EfficientNetB0



- image prediction by custom model trained on balanced dataset

- image prediction by custom model trained on un-balanced dataset



Though custom model trained on un-balanced dataset seems performed better on accuracy but confusion matrix and image predtion shows that our custom model trained on balanced dataset actually performs batter.Custom model trained on balanced dataset predicts all classes where as Custom model trained on un-balanced dataset predicts on Cassava Mosaic Disease (CMD). On the other hand trainable data of unbalanced dataset was 13693 compared to 8000 balanced data. If Custom model trained on balanced dataset were trained for more epoch and more data surely it would performed better.

Certainly pretrained model performed betten than custom model. Because of lackings of knowledge my custom model is quiet simple.

3.3.2 Metrics Reporting

- precision, recall, and F1 score of EfficientNetB0

	precision	recall	f1-score	support
Cassava Bacterial Blight (CBB)	0.38	0.52	0.44	217
Cassava Brown Streak Disease (CBSD)	0.43	0.54	0.48	438
Cassava Green Mottle (CGM)	0.37	0.47	0.42	477
Cassava Mosaic Disease (CMD)	0.90	0.74	0.81	2632
Healthy	0.37	0.47	0.42	516
accuracy			0.65	4280
macro avg	0.49	0.55	0.51	4280
weighted avg	0.70	0.65	0.67	4280

- precision, recall, and F1 score of custom model trained on balanced dataset

	precision	recall	f1-score	support
Cassava Bacterial Blight (CBB)	0.37	0.45	0.41	217
Cassava Brown Streak Disease (CBSD)	0.41	0.54	0.47	438
Cassava Green Mottle (CGM)	0.27	0.53	0.36	477
Cassava Mosaic Disease (CMD)	0.94	0.56	0.70	2632
Healthy	0.30	0.55	0.39	516
accuracy			0.55	4280
macro avg	0.46	0.53	0.46	4280
weighted avg	0.71	0.55	0.59	4280

- precision, recall, and F1 score of custom model trained on un-balanced dataset

	precision	recall	f1-score	support
Cassava Bacterial Blight (CBB)	0.40	0.39	0.39	217
Cassava Brown Streak Disease (CBSD)	0.76	0.30	0.43	438
Cassava Green Mottle (CGM)	0.40	0.44	0.42	477
Cassava Mosaic Disease (CMD)	0.92	0.73	0.82	2632
Healthy	0.30	0.75	0.43	516
accuracy			0.64	4280
macro avg	0.56	0.52	0.50	4280
weighted avg	0.75	0.64	0.67	4280

3.4 Conclusion and Future Work:

Though the performance was not satisfactory but as my first project i am happy with that. My custom model was designed to have the best performane by fine tuning the number of neurons, number of layers, drop-out percentage but for various reason performance was average.

However the dataset and model performance was visualized nicely and performance of EfficientNetB0 was quite similar to custom model trained on un-balanced dataset. The dataset was imbalance and my sampling technique was not up-to-date. Hope my future work will be satisfactory.