A project report on

CASSAVA LEAF DISEASE CLASSIFICATION USING TRANSFER LEARNING

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

SRINATH K R

(17BCE1217)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

June, 2021

CASSAVA LEAF DISEASE CLASSIFICATION USING TRANSFER LEARNING

Submitted in partial fulfillment for the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

SRINATH K R

(17BCE1217)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

June, 2021

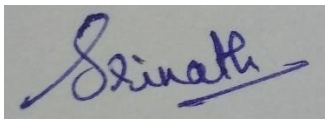
VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

DECLARATION

I hereby declare that the thesis entitled "CASSAVA LEAF DISEASE CLASSIFICATION USING TRANSFER LEARNING" submitted by me, for the award of the degree of Bachelor of Technology in Computer Science Engineering to VIT is a record of bonafide work carried out by me under the supervision of **Dr. Janaki Meena M**.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place: Chennai



Date: 30/06/2021 Signature of the Candidate

School of Computer Science and Engineering

CERTIFICATE

This is to certify that the report entitled **"Cassava Leaf Disease Classification using Transfer Learning"** is prepared and submitted by **Srinath K R** (**17BCE1217**) to VIT Chennai, in partial fulfilment of the requirement for the award of the degree of **B.Tech. CSE** programme is a bonafide record carried out under my guidance. The project fulfils the requirements as per the regulations of this University and in my opinion meets the necessary standards for submission. The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma and the same is certified.

Signature of the Guide:

Name: Dr. Janaki Meena M

Date: 30/06/2021

Signature of the Internal Examiner

Name:

Date:

Signature of the External Examiner

Name:

Date:

Approved by the Head of Department, **B. Tech CSE**

Name: Dr. Justus S

Date:

(Seal of SCOPE)

ABSTRACT

Cultivated extensively in the tropical regions, Cassava is a major source of energy and contains a lot of essential vitamins and minerals. In Uganda, it is grown both as cash and food security crops. However, it is prone to viral and bacterial diseases which causes heavy crop losses and affects food security. So, identifying and detecting these diseases is very important for the livelihood of farmers. Using a dataset containing more than 21,000 annotated photos taken during a routine survey in Uganda, this work aims to categorise each cassava image into four illness categories or a fifth category that indicates a healthy leaf. Transfer learning was used to train EfficientNet, a Convolutional Neural Network (CNN) model. Neural network models extract the required features automatically to classify an image into its respective class. Transfer learning is a machine learning technique in which a model created for one task is used to do another task that is related to it. The developed model was deployed as an interactive web application using Streamlit and this proposed system also attained an overall accuracy score of 0.8887 and 0.8851 on the private and public Kaggle competition leaderboard respectively, indicating the efficiency of the approach.

ACKNOWLEDGEMENT

It is my pleasure to express with deep sense of gratitude to Dr. Janaki Meena M, Professor, School of Computer Science and Engineering, Vellore Institute of Technology, for her constant guidance, continual encouragement and understanding; more than all, she taught me patience in my endeavor. My association with her is not confined to academics only, but it is a great opportunity on my part to work with an intellectual and expert in the field of Big Data, Machine Learning and Deep Learning.

I would like to express my gratitude to the Chancellor Dr. G. Viswanathan, Vice President Mr. Sankar Viswanathan, Assistant Vice President Ms. Kadhambari S. Viswanathan, Vice Chancellor Dr. Rambabu Kodali and Pro-Vice Chancellor Dr. Kanchana Bhaaskaran V.S for providing with an environment to work in and for their inspiration during the tenure of the course.

In jubilant mood I express ingeniously my whole-hearted thanks to Dr. Justus S, Head of the Department & Co-ordinator, Dr. Karmel A, and Project Coordinator Dr. B V A N S S Prabhakar Rao, B. Tech. Computer Science and Engineering, SCOPE, VIT Chennai Campus, for their valuable support and encouragement to take up and complete the thesis.

Special mention to Dean, Dr. Jagadeesh Kannan R, Associate Dean, Dr. Geetha S, SCOPE, VIT Chennai, for spending their valuable time and efforts in sharing their knowledge and for helping us in every aspect.

All teaching staff and members working as limbs of our university for their not-self-centered enthusiasm coupled with timely encouragements showered on me with zeal, which prompted the acquirement of the requisite knowledge to finalize my course study successfully. I would like to thank my parents for their support.

It is indeed a pleasure to thank my friends who persuaded and encouraged me to take up and complete this task. Last but not least, I express my gratitude and appreciation to all those who have helped me directly or indirectly toward the successful completion of this project.

Place: Chennai

Date: 30/06/2021

SRINATH K R

17BCE1217

TABLE OF CONTENTS

CONTENTS ……. vii

LIST OF FIGURES ……. ix

LIST OF TABLES ……... ix

LIST OF ABBREVIATIONS ……. x

CHAPTER 1

INTRODUCTION ……. 1

1.1 WHAT IS CASSAVA? ……..... 1

1.2 OBJECTIVE OF THIS STUDY ……... 2

CHAPTER 2

RELATED WORK ……..... 3

CHAPTER 3

TOOLS AND TECHNOLOGIES USED …… 6

3.1 TOOLS USED ……...…... 6

3.2 PYTHON LIBRARIES USED ……. 6

CHAPTER 4

PROPOSED METHODOLOGY …….……..... 9

4.1 DATA ……..... 11

4.1.1 CASSAVA BACTERIAL BLIGHT (CBB) ……...…..... 11

4.1.2 CASSAVA BROWN STREAK DISEASE (CBSD) …… 12

4.1.3 CASSAVA GREEN MOTTLE (CGM) …… 13

4.1.4 CASSAVA MOSAIC DISEASE (CMD) ……. 13

4.1.5 HEALTHY …… 14

4.2 MODEL ……..……. 14

4.2.1 INTRODUCTION …… 14

4.2.2 ARCHITECTURE OF EFFICIENTNET-B0 ……... 15

4.2.3 COMPOUND SCALING ……. 15

4.2.4 TRANSFER LEARNING …… 17

4.3 IMPLEMENTATION …….……. 17

4.3.1 EXPERIMENTAL SETTINGS …… 17

4.3.2 DEPLOYMENT USING STREAMLIT ……... 18

CHAPTER 5

PROJECT CODE AND DEMONSTRATION ……... 19

5.1 MODEL TRAINING …… 19

5.2 DEPLOYMENT …… 30

CHAPTER 6

RESULTS AND DISCUSSION …… 37

CHAPTER 7

CONCLUSION AND FUTURE WORK ……. 40

REFERENCES ……..…. 41

LIST OF FIGURES

1.1 CASSAVA ROOTS WITH LEAVES ...…….. 1

4.1 OVERALL PROPOSED SYSTEM ...……..... 10

4.2 CLASS DISTRIBUTION OF IMAGES ...……..... 11

4.3 CASSAVA BACTERIAL BLIGHT ……..…. 12

4.4 CASSAVA BROWN STREAK DISEASE ……..... 12

4.5 CASSAVA GREEN MOTTLE ……13

4.6 CASSAVA MOSAIC DISEASE ……. 13

4.7 HEALTHY LEAF ……..... .14

4.8 ARCHITECTURE OF EFFICIENTNET-B0 BASELINE MODEL ...…… 15

4.9 PROPOSED COMPOUND SCALING METHOD ……. 16

4.10 IMAGE BEFORE PERFORMING AUGMENTATION …….. 18

4.11 IMAGES GENERATED AFTER PERFORMING DATA AUGMENTATION ……... 18

6.1 PLOT SHOWING TRAINING AND VALIDATION ACCURACY AGAINST EPOCHS …… ……37

6.2 PLOT SHOWING TRAINING AND VALIDATION LOSS AGAINST EPOCHS ……... 38

6.3 HOME PAGE OF THE DEPLOYED APP …… 39

6.4 PREDICTION RESULT DISPLAYED BY THE WEB APP …… 39

LIST OF TABLES

6.1 RESULTS AND ANALYSIS ……..... 37

LIST OF ABBREVIATIONS

ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network

Chapter 1

Introduction

1.1 WHAT IS CASSAVA?

Cassava (*Manihot esculenta*) is a South American tuberous woody plant [1]. It is commonly grown throughout the world's tropical and subtropical parts. It is the third-largest carbohydrate source in the tropical countries, next only to rice and corn [1]. It is well-known for its carbohydrate-rich tuberous roots and is a key staple crop in third-world nations, feeding more than half a billion people worldwide [1]. For this reason, it is often called the "Bread of the Tropics". It is also known to be a crop with high drought-tolerance. Nigeria, the Democratic Republic of the Congo, Thailand, Indonesia, and Brazil account for over 70% of global cassava production. [2].

Figure 1.1 Cassava roots with leaves

Uganda was the 11th largest producer of cassava in the world in 2010 [3]. In Uganda, cassava is grown as a commercial crop as well as a food crop. Cassava has been widely accepted and the output has increased rapidly since its debut in the nation. Pests and disease outbreaks, however, have wreaked havoc on the crops, resulting in significant yearly production losses for farmers. Pests such as Cassava Green Mite, as well as viruses such as Cassava Mosaic Virus and Cassava Brown Streak Virus, have become

the most significant restrictions on cassava production and food security [4]. Early detection and identification are critical steps in controlling the spread of these illnesses. However, current disease detection methods need farmers enlisting the assistance of agrarian experts arranged by the government to physically visit, assess and analyze the plants. This is a time-consuming, low-supply, and costly approach [16].

1.2 OBJECTIVE OF THIS STUDY

As a result, an intelligent system for early and precise identification of plant diseases is required. Farmers may only have access to low-bandwidth, mobile-quality cameras, thus these effective solutions must work effectively under substantial restrictions. To address these computational limitations, transfer learning, a machine learning process in which a model learned for one job is re-used to accomplish another related job, presents an alternative to training deep learning models. In this study, the use of a pre-trained deep learning convolutional neural network model to identify and classify leaf diseases was investigated using a dataset of 21,397 photos taken during a routine survey in Uganda, which included four cassava disease categories and one healthy category. Farmers may be able to swiftly detect unhealthy plants as a result of this, perhaps sparing their crops from irreversible damage.

Chapter 2

Related Work

Plant leaf disease detection is a significant and fascinating scientific research subject, and it has advanced in recent years.

The authors in [5] suggested a deep CNN model for classifying 26 diseases that affect 14 different plant crop species. Two neural network designs–AlexNet and GoogLeNet–were employed on the publicly accessible PlantVillage image dataset. Model training was performed in two ways- building the model from scratch and by using transfer learning. To test the models, various factors such as image color, grey scale, train-test split ratio, and so on were changed. The highest level of accuracy was 99.35 percent.

In the paper [6], the authors used a dataset of cassava affected and healthy leaf images to train a deep CNN model that identified 2 types of damages caused by pests and 3 types of diseases. For this, the authors employed a transfer learning strategy to do this, and the neural network model employed was InceptionV3. Three distinct architectures were utilised for the model's final layer: softmax layer, SVM, and KNN. Different train-test split ratios were explored to find the optimal one, which yielded a 93 percent overall accuracy.

In [7], the authors developed a system that performs both classification and object detection on the cassava leaf images dataset. A CNN classification model was built to categorize cassava leaf pictures as healthy or unhealthy, as well as a faster R-CNN detection model to detect the impacted part of the unhealthy leaves. Following the detection of the unhealthy section, a new CNN classification model was created to categorize the leaves as CBSD (Cassava Brown Streak Virus Disease) or CBSD (Cassava Brown Streak Virus Disease). The system achieved an accuracy and F-score of 0.96, according to the authors.

In [8], the authors developed a CNN model from scratch to detect and classify diseased cassava leaf images into 5 categories – 4 diseases and healthy. Because the dataset was small and had a significant class-imbalance with a large bias towards 2 disease classes, the authors used numerous strategies to counteract this, including SMOTE, changing

class weight, focus loss, and input picture dimensions. The most accurate model has a precision of almost 93 percent. The authors' major goal was to create a model that could overcome large class imbalance and accurately represent all classes.

To classify cassava leaf diseases, the authors in [9] created two models: plain convolutional neural network (PCNN) and deep residual neural network (DRNN). A special block processing methodology was included to counter the class imbalance since the dataset was skewed towards two disease classes. Other image processing techniques were also employed. After considering many performance indicators, it was determined that DRNN outdid PCCN by 9.25 percent, with the former obtaining an overall accuracy of 96.75 percent.

In [10], the authors employed the LeNet CNN technique to categorize banana leaves into three categories: banana sigatoka, banana speckle (all unhealthy), and healthy. Both color and grayscale images were employed in this method. Different train-test split ratios were used to test the model. The efficacy of the system was evaluated using a combination of measures such as F1-score, precision, recall, and accuracy. According to the authors, the model performed well under a variety of challenging settings, including varied picture resolutions, light, size, orientation, and a complex backdrop, while requiring less computation effort.

The authors in [11] developed a modified LeNet CNN model for categorizing maize leaves into four categories: Gray Leaf Spot, Northern Leaf Blight, Common Rust (all infected), and healthy. Principal Component Analysis (PCA) Whitening was used to prepare the pictures. According to the authors, the primary goal of this research was to examine the effectiveness of LeNet by adjusting factors such as kernel size and convolutional layer depth. The generated model was able to achieve a 97.89 percent accuracy.

In the paper [12], using the PlantVillage dataset and another supplementary dataset, the authors used the EfficientNet architecture with a transfer learning technique to classify plant leaf diseases. The performance of eight EfficientNet models (B0, B1, B2, B3, B4, B5, B6, and B7) was compared to that of AlexNet, InceptionV3, ResNet50, and VGG16, which are all modern CNN architectures. Even though the difference between the algorithms utilised was extremely little, it was found that EfficientNet B5 and B4

delivered the best results on the original and enhanced datasets using multiple metrics such as precision, accuracy, specificity, and sensitivity.

The authors in [13] suggested a new deep learning approach for detecting and classifying plant illnesses from leaf pictures. The dataset was built right from scratch by getting the necessary photos from internet. There were 15 classifications in total: 13 disorders, healthy, and background photos. To increase the dataset's variety, several data augmentation approaches were used. The model was trained using the CaffeNet CNN architecture. The model has an overall accuracy of 96.3 percent, with precision ranging from 91 percent to 98 percent for particular classes.

Using a modified version of the CNN model LeNet, the scientists devised a technique to detect and diagnose 10 distinct illnesses that occur in tomato leaves in [14]. A portion of the PlantVillage dataset's tomato leaf disease photos was used. The authors tested two deep learning architectures, AlexNet and GoogLeNet, and found that a modified version of LeNet outperformed the others. According to the authors, the goal of this project was to employ the fewest resources possible while achieving outcomes comparable to those reached by contemporary methodologies. The proposed model had an accuracy of 94-95 percent on average.

The author in paper [15] tried to figure out how the quantity of the dataset influences deep learning's ability to classify plant diseases. A dataset with photos of 12 plant species afflicted by 56 illnesses was utilised for this. The GoogLeNet CNN model was trained by transfer learning and evaluated with two picture types: original and background-removed pictures. It was discovered that picture backdrop has a significant impact on the outcome of a CNN model, especially in smaller datasets including images taken in real-life situations.

Chapter 3

TOOLS AND TECHNOLOGIES USED

3.1 TOOLS USED

Python

Python is a programming language with wide range of applications, including data analysis and the creation of deep learning algorithms. Because of its growing popularity as a programming language preferred for computational sciences and the abundance of numerous advanced tools image for processing, Python is a great choice for these sorts of image processing jobs.

Jupyter

Jupyter is a free and open-source, interactive notebook-for-computation web application that allows the integration of code, output, explanations, and other multimedia resources into a single document.

3.2 PYTHON LIBRARIES USED

NumPy

NumPy is a free Python library that provides support for large, multidimensional matrices and arrays along with a large number of mathematical functions at a high level to manipulate them. It is the most important Python module for scientific computing.

Pandas

Pandas is a Python open source library that has user-friendly data structures and tools for data analysis. It has procedures and data structures for manipulating numerical tables.

Scikit-learn

Scikit-learn is a Python Machine Learning package that includes methods for classification, regression, and grouping. It was developed to work with SciPy and NumPy, Python's scientific and numerical libraries. In Python, the term scikit-learn is written as sklearn.

Matplotlib

Matplotlib is a charting library that produces high-quality figures in a variety of interactive formats on a variety of platforms. It is a Python package that allows you to create interactive and animated visualisations. With just a few lines of code, charts may be created.

Seaborn

Seaborn is a Python data visualisation package based on Matplotlib. It offers a high-level interface that allows you to make aesthetically beautiful and helpful data visualisations. It was created with pandas data frame objects in mind.

Pillow (Python Image Library)

The image processing abilities of Python are enhanced by PIL, the Python Imaging Library. The library can read and write a broad variety of file formats, has a quick internal representation, and can do advanced processing of images.

Tensorflow

Tensorflow is a free and open source machine learning framework that uses data flow graphs to generate models. It enables programmers to build large-scale neural networks with several layers. TensorFlow is commonly used for classification, prediction, clustering, etc.

Streamlit

Streamlit is an open-source Python framework used for creating and sharing data web applications. In the data science community, it is steadily gaining traction. Because of the ease with which one can develop a data science web app, many developers use it in their daily workflow.

Chapter 4

Proposed Methodology

The flow diagram depicting the overall proposed system is shown in Figure 4.1. In this system, after understanding the data set and performing an exploratory data analysis, the problem is to be defined. Since the job is to categorise each cassava image into four illness categories or a fifth category that indicates a healthy leaf, this is an Image Classification problem. After identifying the type of problem, the next stage involves data preparation where major data preprocessing steps are performed. Later, the pre-trained ConvNet model is chosen, tweaked and re-trained on the cassava image dataset. Then, the performance of the model is assessed using metrics like accuracy and loss. Finally, the developed model is deployed as an interactive web app.

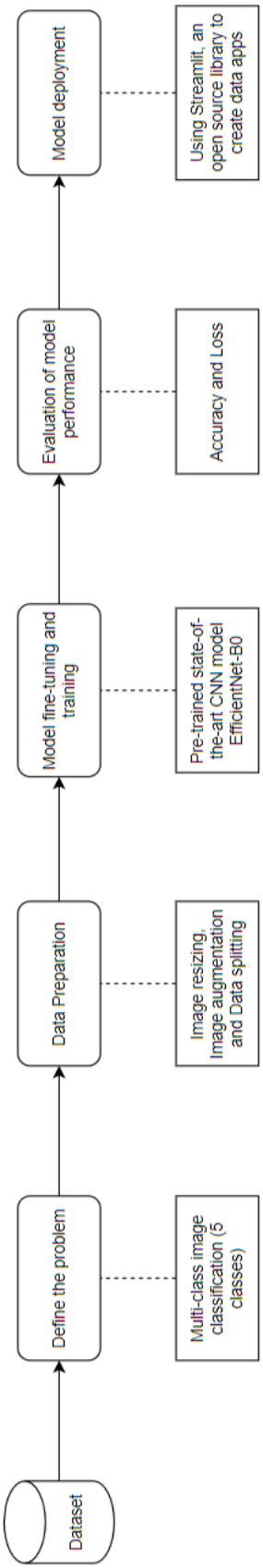


Figure 4.1 Overall proposed system

4.1 DATA

The photos of cassava leaves used in this experiment came from Kaggle [16]. There are 21,397 photos in the dataset: **Healthy** cassava leaf with 2,577 images and four categories of disease affected cassava leaves- **Cassava Bacterial Blight (CBB)** with 1,087 images, **Cassava Brown Streak Disease (CBSD)** with 2,189 images, **Cassava Green Mottle (CGM)** with 2,386 images, and **Cassava Mosaic Disease (CMD)** with 13,158 images gathered during a routine survey in Uganda. The images were collected from farmers taking pictures of their fields, and labeled by professionals at the National Crops Resources Research Institute (NaCRRI) in association with the Artificial Intelligence laboratory of Makerere University. This is in a form that most closely resembles what farmers would encounter on the field [16].

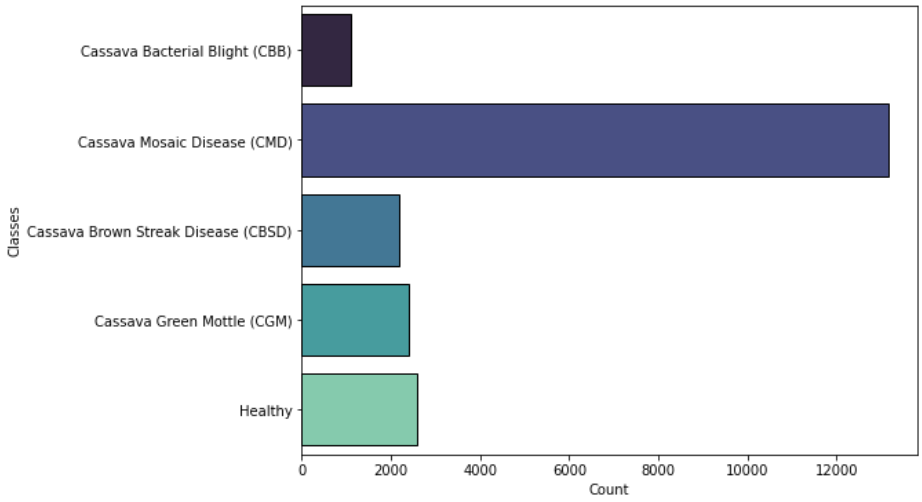


Figure 4.2 Class distribution of images

4.1.1 CASSAVA BACTERIAL BLIGHT (CBB)

The pathogen *Xanthomonas axonopodis pv. Manihotis* causes Cassava Bacterial Blight (CBB), a critical bacterial disease of cassava. Bacterial blight is the disease that causes the most yield losses in cassava across the world. Leaf wilting, spotting, dieback, tiny, water-soaked patches, bright green regions, or both on leaves are some of the symptoms.

Figure 4.3 Cassava Bacterial Blight

4.1.2 CASSAVA BROWN STREAK DISEASE (CBSD)

Cassava Brown Streak Disease (CBSD) is a major viral disease caused by two different types of ipomovirus species namely Cassava Brown Streak Virus (CBSV) and Ugandan Cassava Brown Streak Virus (UCBSV), both belonging to the *Potyviridae* family. CBSD-affected plants have symptoms on their older leaves, stems, and roots. Secondary veins sprouting from the midribs generate yellow patches, which frequently merge together.

Figure 4.4 Cassava Brown Streak Disease

4.1.3 CASSAVA GREEN MOTTLE (CGM)

Cassava Green Mottle (CGM) is a cassava virus disease caused by the Cassava Green Mottle Virus (CGMV), which belongs to the *Secoviridae* family. The leaves of CGMV-affected cassava plants are ruffled and feature faint to obvious yellow spots, green patterns, and twisted edges.

Figure 4.5 Cassava Green Mottle

4.1.4 CASSAVA MOSAIC DISEASE (CMD)

The Cassava Mosaic Virus (CMV), a generic term for many virus species in the *Geminiviridae* family, causes Cassava Mosaic Disease (CMD), a serious viral disease. Because cassava is vegetatively propagated, the illness is disseminated primarily through the movement of virus-infected cassava cuttings. The dots on an infected leaf are light yellow or white. Other signs include twisted, stunted, and deformed leaves.

Figure 4.6 Cassava Mosaic Disease

4.1.5 HEALTHY

Healthy leaves are cassava leaves that have not been impacted by any of the four illnesses listed above.

Figure 4.7 Healthy leaf

4.2 MODEL

4.2.1 INTRODUCTION

In this study, transfer learning on a convolutional neural network (CNN) model on cassava leaf image dataset was performed. When compared to conventional approaches of training classification models where features are extracted manually, CNNs extract the target features automatically. Because CNNs take longer to train due to their complexity, transfer learning is used to leverage the knowledge (features, weights, and so on) obtained by a previously trained model to train a new model with additional classes. By making use of the vast visual knowledge previously learnt by the model from the ImageNet picture collection, the weights of the advanced and powerful CNN model **EfficientNet B0** [17] were retrained to identify the cassava leaf pictures in this approach.

CNNs have dominated the area of computer vision since AlexNet won the ImageNet Challenge in 2012. However, model scaling – altering different dimensions of the model to increase accuracy – is one of the major difficulties that these models encounter. In the ImageNet challenge, **EfficientNets**, a family of 8 state-of-the-art CNN models built by Google and numbered B0 to B7, attained an impressive accuracy of

84.4 percent with 66M parameters. The number of parameters and accuracy improve dramatically as the number of models increases in this category. EfficientNet makes uses of Swish, a novel activation function, rather than the most popular and commonly used activation function, Rectifier Linear Unit (ReLU).

4.2.2 ARCHITECTURE OF EFFICIENTNET-B0

EfficientNet-B0, the EfficientNet family's baseline model, was created using a Neural Architecture Search (NAS), which maximises both accuracy and FLOPS (floating point operations per second). It includes 237 layers and 5.3 million parameters. The EfficientNet-B0 network is built on MobileNetV2's MBConv mobile inverted bottleneck blocks [18].

With a number of channels, an inverted bottleneck block has a narrow -> wide -> narrow structure. The input has a large number of channels, which are expanded using a 1x1 convolution before being processed with a 3x3 depth-wise convolution, which decreases the number of parameters greatly. After that, a 1x1 convolution is used to limit the number of channels so that input and output may be added.

Stage i	Operator $\hat{\mathcal{F}}_i$	Resolution $\hat{H}_i \times \hat{W}_i$	#Channels $\hat{C}_i$	#Layers $\hat{L}_i$
1	Conv3x3	224×224	32	1
2	MBConv1, k3x3	112×112	16	1
3	MBConv6, k3x3	112×112	24	2
4	MBConv6, k5x5	56×56	40	2
5	MBConv6, k3x3	28×28	80	3
6	MBConv6, k5x5	14×14	112	3
7	MBConv6, k5x5	14×14	192	4
8	MBConv6, k3x3	7×7	320	1
9	Conv1x1 & Pooling & FC	7×7	1280	1

Figure 4.8 Architecture of EfficientNet-B0 baseline model [17]

4.2.3 COMPOUND SCALING

A CNN model may be scaled up in three different ways: depth, breadth, and resolution.

• The network's depth (d) is the number of layers.

• The number of channels or filters in the convolutional layer is measured in width (w).

• The width and height of the pictures provided to the model are measured in resolution (r).

Scaling up any of the aforementioned dimensions enhances accuracy, but larger models reach a point where accuracy plateaus. So, in order to improve accuracy and efficiency, all of the above-mentioned dimensions must be balanced. Compound scaling [17] is the term for this. Unlike other recent and sophisticated CNN models, EfficientNet obtains more efficient outcomes by scaling these dimensions equally.

The first stage in this compound scaling approach is to conduct a grid search to determine the link between the multiple scaling dimensions of the baseline model when resources are limited. An appropriate scaling factor for the dimensions is calculated in this way. The baseline model is then scaled to reach the desired target network using these criteria.

The compound scaling technique proposed by the developers at Google is as follows:

$$\begin{aligned} \text{depth: } & d = \alpha^\phi \\ \text{width: } & w = \beta^\phi \\ \text{resolution: } & r = \gamma^\phi \\ \text{s.t. } & \alpha \cdot \beta^2 \cdot \gamma^2 \approx 2 \\ & \alpha \geq 1, \beta \geq 1, \gamma \geq 1 \end{aligned}$$

Figure 4.9 Proposed compound scaling method [17]

where ϕ is a coefficient specified by the user that governs available resources and γ, β, and α are constants that describe how the resources are assigned to network resolution, width and depth respectively.

The suggested compound scaling approach is implemented in two phases, starting with the baseline network.

- Fix ϕ = 1 and assume there are twice as many resources accessible. Now, perform a small search in the grid for γ, β, and α. The optimal parameters for

the baseline network EfficientNet-B0 are $\gamma = 1.15$, $\beta = 1.1$, and $\alpha = 1.2$ such that $\alpha * \beta^2 * \gamma^2$ is approximately 2.

- Fix γ, β, and α using the values reached in the previous step and try out different values of ϕ. EfficientNets B1-B7 are produced by making use of different values of ϕ.

4.2.4 TRANSFER LEARNING

The machine learning approach in which a model that has been trained for one job is retrained for a different but related task is called transfer learning. In this approach, the last few layers present in the pre-trained network are eliminated and can be replaced with required layers for the job to be achieved.

The taught characteristics via transfer learning can be applied to a variety of computer vision issues, even if the new challenges are in completely different classes than the original job. Transfer learning, like any other model, saves time and computing power for EfficientNet as compared to training a model from scratch. As a result, it is more accurate than many other models. This is because the compound scaling methodology has been enhanced in terms of depth, breadth, and picture resolution.

4.3 IMPLEMENTATION

4.3.1 EXPERIMENTAL SETTINGS

The Cassava Leaf Disease Classification Kaggle competition dataset [16] consists of 21,397 images categorized into 4 unique cassava leaf disease classes and one class of healthy leaves. 80 percent of the photos (17,118) were used for training the model and the 20 percent (4,279) were set aside for validating the model.

The original picture size was 800x600 pixels. To speed up the model training process, the photos were scaled to 512x512 pixels. Data augmentation techniques such as rotating the photos by 45 degrees, shearing transformations, horizontal and vertical flipping, and horizontal and vertical shifting of pictures were used to overcome the data imbalance. The loss function used was sparse categorical cross entropy, and the optimization was done with Adam optimizer. The model was trained for 20 epochs with

16 being the batch size. The learning rate was initially set at 0.001, and after the loss stopped falling, it was lowered by a factor of 0.3. Early stopping was also incorporated to keep track of validation loss and to halt the training once a certain level was reached.

Figure 4.10 Image before performing augmentation

Figure 4.11 Images generated after performing data augmentation

4.3.2 DEPLOYMENT USING STREAMLIT

The developed model was then deployed as an interactive web application using an open-source Python library **Streamlit** [19]. Streamlit is a popular open-source framework for machine learning and data science that makes it simple to design and share custom, beautiful and interactive web apps. It turns data scripts into shareable web apps.

Chapter 5

Project Code and Demonstration

5.1 MODEL TRAINING

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import os, cv2, json
from PIL import Image
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import tensorflow as tf
from tensorflow.keras import models, layers
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint,
EarlyStopping, ReduceLROnPlateau
from tensorflow.keras.applications import EfficientNetB0
from tensorflow.keras.optimizers import Adam
import warnings
warnings.simplefilter('ignore')
```

```
directory = '../input/cassava-leaf-disease-classification'

os.listdir(directory)


### Data

print('Number of train images: ',
len(os.listdir(os.path.join(directory, 'train_images'))))

print('Number of test images: ',
len(os.listdir(os.path.join(directory, 'test_images'))))


with open(os.path.join(directory,
'label_num_to_disease_map.json')) as file:

    mapping = json.loads(file.read())

    mapping = {int(k): v for k, v in mapping.items()}


print(mapping)


train_labels = pd.read_csv(os.path.join(directory,
'train.csv'))

train_labels['label_names'] =
train_labels['label'].map(mapping)

train_labels.head(10)


train_labels['label_names'].value_counts()


### Distribution of classes

plt.figure(figsize = (8, 6))

sns.countplot(y=train_labels['label_names'],
edgecolor="black", palette="mako")
```

```
plt.xlabel('Count')
plt.ylabel('Classes')
plt.show()


### Class 0: Cassava Bacterial Blight (CBB)
sample = train_labels[train_labels.label == 0].sample(1)
plt.figure(figsize = (12, 8))
for ind, (image_id, label) in enumerate(zip(sample.image_id,
sample.label)):
    plt.subplot(1, 3, ind + 1)
    img = cv2.imread(os.path.join(directory, 'train_images',
image_id))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
    plt.axis('off')


plt.show()


### Class 1: Cassava Brown Streak Disease (CBSD)
sample = train_labels[train_labels.label == 1].sample(1)
plt.figure(figsize = (12, 8))
for ind, (image_id, label) in enumerate(zip(sample.image_id,
sample.label)):
    plt.subplot(1, 3, ind + 1)
    img = cv2.imread(os.path.join(directory, 'train_images',
image_id))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
```

```
    plt.imshow(img)
    plt.axis('off')

plt.show()

### Class 2: Cassava Green Mottle (CGM)
sample = train_labels[train_labels.label == 2].sample(1)
plt.figure(figsize = (12, 8))
for ind, (image_id, label) in enumerate(zip(sample.image_id,
sample.label)):
    plt.subplot(1, 3, ind + 1)
    img = cv2.imread(os.path.join(directory, 'train_images',
image_id))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
    plt.axis('off')

plt.show()

### Class 3: Cassava Mosaic Disease (CMD)
sample = train_labels[train_labels.label == 3].sample(1)
plt.figure(figsize = (12, 8))
for ind, (image_id, label) in enumerate(zip(sample.image_id,
sample.label)):
    plt.subplot(1, 3, ind + 1)
    img = cv2.imread(os.path.join(directory, 'train_images',
image_id))
```

```
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
    plt.axis('off')

plt.show()

### Class 4: Healthy
sample = train_labels[train_labels.label == 4].sample(1)
plt.figure(figsize = (12, 8))
for ind, (image_id, label) in enumerate(zip(sample.image_id,
sample.label)):
    plt.subplot(1, 3, ind + 1)
    img = cv2.imread(os.path.join(directory, 'train_images',
image_id))
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.imshow(img)
    plt.axis('off')

plt.show()

### Modeling Preparation
BATCH_SIZE = 16
STEPS_PER_EPOCH = len(train_labels) * 0.8 / BATCH_SIZE
VALIDATION_STEPS = len(train_labels) * 0.2 / BATCH_SIZE
EPOCHS = 20
TARGET_SIZE = 512
```

```
train_labels.label = train_labels.label.astype('str')


train_datagen = ImageDataGenerator(validation_split = 0.2,
                                   preprocessing_function =
None,
                                   rotation_range = 45,
                                   zoom_range = 0.2,
                                   horizontal_flip = True,
                                   vertical_flip = True,
                                   fill_mode = 'nearest',
                                   shear_range = 0.1,
                                   height_shift_range = 0.1,
                                   width_shift_range = 0.1)


train_generator =
train_datagen.flow_from_dataframe(train_labels,
                          directory = os.path.join(directory,
'train_images'),
                          subset = 'training',
                          x_col = 'image_id',
                          y_col = 'label',
                          shuffle = True,
                          target_size = (TARGET_SIZE,
TARGET_SIZE),
                          batch_size = BATCH_SIZE,
                          class_mode = 'sparse')
```

```
validation_datagen = ImageDataGenerator(validation_split =
0.2)


validation_generator =
validation_datagen.flow_from_dataframe(train_labels,
                        directory = os.path.join(directory,
'train_images'),
                        subset = 'validation',
                        x_col = 'image_id',
                        y_col = 'label',
                        shuffle = True,
                        target_size = (TARGET_SIZE,
TARGET_SIZE),
                        batch_size = BATCH_SIZE,
                        class_mode = 'sparse')


### Before Image Augmentation
img_path = os.path.join(directory, 'train_images',
train_labels.image_id[20])
img = image.load_img(img_path, target_size = (TARGET_SIZE,
TARGET_SIZE))
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis = 0)
img_tensor /= 255.


plt.imshow(img_tensor[0])
plt.axis('off')
plt.figure(figsize = (12, 8))
```

```
plt.show()


### After Image Augmentation

generator =
train_datagen.flow_from_dataframe(train_labels.iloc[20:21],

                          directory = os.path.join(directory,
'train_images'),

                          x_col = 'image_id',

                          y_col = 'label',

                          target_size = (TARGET_SIZE,
TARGET_SIZE),

                          batch_size = BATCH_SIZE,

                          class_mode = 'sparse')


aug_images = [generator[0][0][0]/255 for i in range(10)]

fig, axes = plt.subplots(2, 5, figsize = (8, 6))

axes = axes.flatten()


for img, ax in zip(aug_images, axes):

    ax.imshow(img)

    ax.axis('off')


plt.tight_layout()

plt.show()


### Modeling using EfficientNet-B0
```

```
conv_base = EfficientNetB0(include_top = False, weights =
‘imagenet’, input_shape = (TARGET_SIZE, TARGET_SIZE, 3))
model = conv_base.output
model = layers.GlobalAveragePooling2D()(model)
model = layers.Dense(5, activation = 'softmax')(model)
model = models.Model(conv_base.input, model)

model.compile(optimizer = Adam(lr = 0.001),
              loss = 'sparse_categorical_crossentropy',
              metrics = ['acc'])

model.summary()

print('No. of layers in EfficientNetB0:', len(model.layers))

model_save =
ModelCheckpoint('./EffNetB0_512_16_best_weights.h5',
                            save_best_only = True,
                            save_weights_only = True,
                            monitor = 'val_loss',
                            mode = 'min', verbose = 1)

early_stop = EarlyStopping(monitor = 'val_loss', min_delta =
0.001,
                           patience = 5, mode = 'min', verbose
= 1,
                           restore_best_weights = True)
```

```
reduce_lr = ReduceLROnPlateau(monitor = 'val_loss', factor =
0.3,
                              patience = 2, min_delta = 0.001,
                              mode = 'min', verbose = 1)


### Training the model
history = model.fit(
    train_generator,
    steps_per_epoch = STEPS_PER_EPOCH,
    epochs = EPOCHS,
    validation_data = validation_generator,
    validation_steps = VALIDATION_STEPS,
    callbacks = [model_save, early_stop, reduce_lr]
)


acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']


### Plotting training and validation loss against epochs
plt.figure(figsize = [8, 6])
plt.plot(loss)
plt.plot(val_loss)
plt.xlim(xmin = 0)
plt.xticks(np.arange(0, 21, 2))
```

```
plt.title('Loss over epochs')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='best')

plt.show()

### Plotting training and validation accuracy against epochs
plt.figure(figsize = [8, 6])
plt.plot(acc)
plt.plot(val_acc)
plt.xlim(xmin = 0)
plt.xticks(np.arange(0, 21, 2))

plt.title('Accuracy over epochs')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Validation'], loc='best')

plt.show()

### Saving the model for deployment
model.save('./EffNetB0_512_16.h5')
```

5.2 DEPLOYMENT

```
import streamlit as st

import tensorflow.keras

from PIL import Image, ImageOps

import numpy as np

import warnings

warnings.filterwarnings('ignore')


# To make sure there are no file encoding warnings

st.set_option('deprecation.showfileUploaderEncoding', False)


def main():

    st.set_page_config(

    page_title = "Cassava Leaf Disease Prediction",

    layout = "centered",

    page_icon= ":four_leaf_clover:"

    )


    menu = ['Home', 'About']

    choice = st.sidebar.radio("Menu", menu)


    if choice == "Home":

        st.title('🍀 Cassava Leaf Disease Prediction')


        st.subheader("This app predicts diseases affecting
cassava leaves.")
```

```
        st.write("")

        uploaded_file = st.file_uploader("Choose a leaf
image", type = ["jpg", "jpeg", "png"])

        if st.button("Predict"):
            if uploaded_file is not None:
                image = Image.open(uploaded_file)

                st.write("")
                st.write("Uploaded image")

                st.image(image, use_column_width = True)
                st.write("")

                try:
                    with st.spinner("**Predicting...**"):

                        np.set_printoptions(suppress = True)



### Loading the saved EfficientNet-B0 model
                        model =
tensorflow.keras.models.load_model('model/EffNetB0_512_16.h5')

                        image = image.resize((512, 512))
```

```
                        image = np.expand_dims(image, axis =
0)

                        labels = {0: "Cassava Bacterial Blight
(CBB)", 1: "Cassava Brown Streak Disease (CBSD)", 2: "Cassava
Green Mottle (CGM)", 3: "Cassava Mosaic Disease (CMD)", 4:
"Healthy"}

                        predictions =
np.argmax(model.predict(image))

                        print(labels[predictions])

                        label = labels[predictions]

                    if(label == 'Healthy'):
                        st.success("Above leaf image is
predicted to be **" + label + "**.")

                    else:
                        st.success("Above leaf image is
predicted to be affected by **" + label + "**.")
                        st.write("")

                        if(label == 'Cassava Bacterial Blight
(CBB)'):
                            st.write("The pathogen
*Xanthomonas axonopodis pv. Manihotis* causes **Cassava
```

```
Bacterial Blight (CBB)**, a bacterial disease affecting
cassava. \
                                Bacterial blight is the
disease that causes the most yield losses in cassava across
the world.")


                        elif(label == 'Cassava Brown Streak
Disease (CBSD)'):
                            st.write("**Cassava Brown Streak
Disease (CBSD)** is a viral disease caused by two different
ipomovirus species, Cassava Brown Streak Virus (CBSV) and
Ugandan Cassava Brown Streak Virus (UCBSV), both of which
belong to the *Potyviridae* family.")


                        elif(label == 'Cassava Green Mottle
(CGM)'):
                            st.write("**Cassava Green Mottle
(CGM)** is a viral disease caused by Cassava Green Mottle
Virus (CGMV), which belongs to the *Secoviridae* family.")


                        elif(label == 'Cassava Mosaic Disease
(CMD)'):
                            st.write("**Cassava Mosaic Disease
(CMD)** is a viral disease caused by Cassava Mosaic Virus
(CMV), a generic term for many virus species in the
*Geminiviridae* family.")


                except:
                    st.error("Apologies! Something went wrong!
🤦")
            else:
```

```
                st.error("Could you please upload an image?
🐶")


    elif choice == "About":
        st.title("About the project")
        st.write("")

        cassava_with_leaves =
Image.open("./assets/cassava_with_leaves.jpg")
        st.image(cassava_with_leaves, use_column_width = True,
caption = "Cassava roots with leaves.")

        st.write("")
        st.write("As the second-largest provider of
carbohydrates in Africa, **Cassava** (*Manihot esculenta*) is
a key food security crop grown by smallholder farmers because
it can withstand harsh conditions. \
            At least 80% of household farms in Sub-Saharan
Africa grow this starchy root, but viral diseases are major
sources of poor yields. \
            Existing methods of disease detection require
farmers to solicit the help of government-funded agricultural
experts to visually inspect and diagnose the plants. \
            This suffers from being labour-intensive, low-
supply and costly. As an added challenge, effective solutions
for farmers must perform well under significant constraints,
since African farmers may only have access to mobile-quality
cameras with low-bandwidth. \
            With the help of data science, it may be possible
to identify common diseases so they can be treated.")
```

```
        st.write("")

        st.write("This application is aimed at classifying a
cassava leaf image, using a [**Kaggle Research Code
Competition**](https://www.kaggle.com/c/cassava-leaf-disease-
classification) dataset consisting of 21,397 labeled images
collected during a regular survey in Uganda, into four disease
categories (**Cassava Bacterial Blight**, **Cassava Brown
Streak Disease**, **Cassava Green Mottle** and **Cassava
Mosaic Disease**) or a fifth category indicating a **Healthy**
leaf. \

            Most images were crowdsourced from farmers taking
photos of their gardens, and annotated by experts at the
National Crops Resources Research Institute (NaCRRI) in
collaboration with the AI lab at Makerere University, Kampala.
\

            This is in a format that most realistically
represents what farmers would need to diagnose in real life.")


        st.write("")

        st.write("For this, [**EfficientNet-
B0**](https://arxiv.org/abs/1905.11946), a state-of-the-art
convolutional neural network model, was trained using
**Transfer Learning**. \

            EfficientNet is a convolutional neural network
architecture and scaling method that uniformly scales all
dimensions of depth/width/resolution using a compound
coefficient. \

            Unlike conventional practice that arbitrary scales
these factors, the EfficientNet scaling method uniformly
scales network width, depth, and resolution with a set of
fixed scaling coefficients.")


        st.write("")
```

```
        efficientnetb0_architecture =
Image.open("./assets/efficientnetb0-architecture.png")

        st.image(efficientnetb0_architecture, use_column_width
= True, caption = "EfficientNet-B0 baseline network")


        st.write("")

        st.write("The trained model was also made as a
submission to [**Cassava Leaf Disease
Classification**](https://www.kaggle.com/c/cassava-leaf-
disease-classification/), the Kaggle Research Code
Competition. \

        The model achieved an overall **accuracy score** of
**0.8887** and **0.8851** on the **private** and **public
competition leaderboard** respectively, indicating the
efficiency of the approach.")
```

Chapter 6

Results and Discussion

The categorization accuracy measure was used to measure the performance of the model. It is described as the proportion of rightly identified samples to the overall number of samples. The best training and validation accuracy values obtained till the appropriate epoch are shown in Table 6.1.

Table 6.1 Results and Analysis

No. of epochs	Training accuracy	Validation accuracy
5	0.8816	0.8663
10	0.9101	0.8831
14	0.9184	0.8831

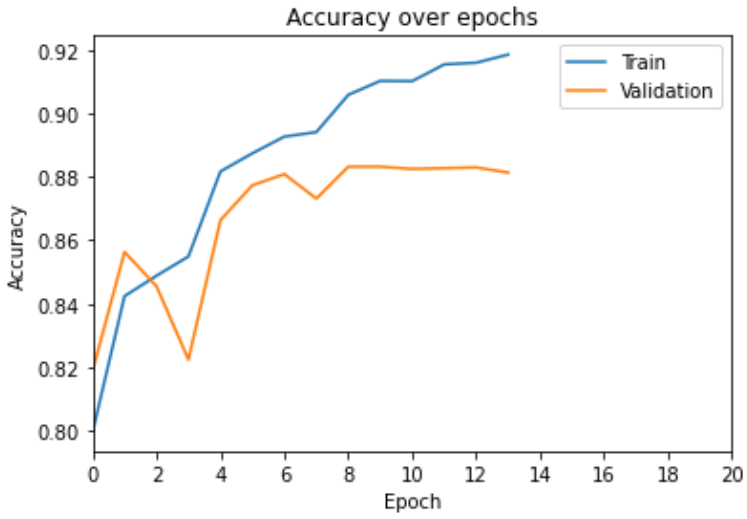


Figure 6.1 Plot showing training and validation accuracy against epochs

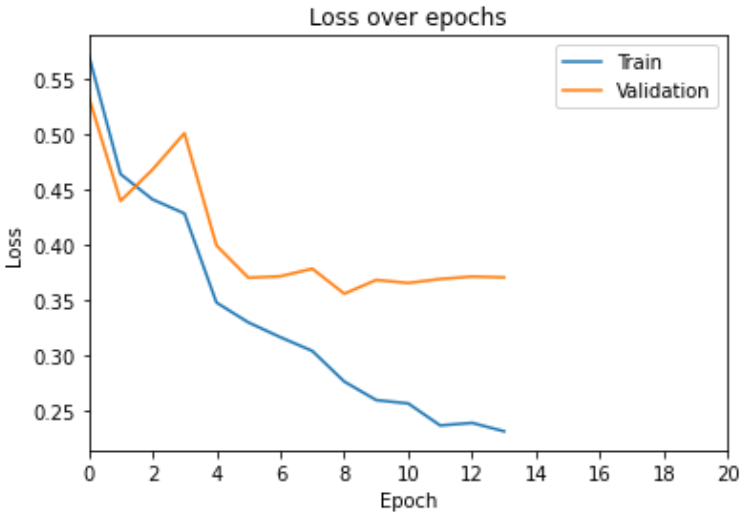


Figure 6.2 Plot showing training and validation loss against epochs

Since the validation loss did not improve, early stopping was triggered, and the model training process was terminated after 14 epochs. The maximum validation accuracy score achieved was 0.8831, whereas the greatest training accuracy score achieved was 0.9184.

Figures 6.1 and 6.2 show plots of train and validation accuracy and loss against the number of epochs. This model received an overall accuracy score of 0.8887 and 0.8851 on the private and public competition leaderboards, respectively, when it was submitted to the Cassava Leaf Disease Classification Kaggle Competition, whose dataset was used in this study. These findings indicate that the model worked well on the dataset and that it can be used to classify cassava leaf diseases.

The developed model was saved as a .h5 file and then loaded using *load_model*() present in *Keras* Python library. The application asks for an image of a cassava leaf; once an image is chosen and 'Predict' button is clicked, by making use of the developed model, it is identified whether the leaf is affected by either of the earlier mentioned four diseases or a healthy leaf, along with a small description for each class.

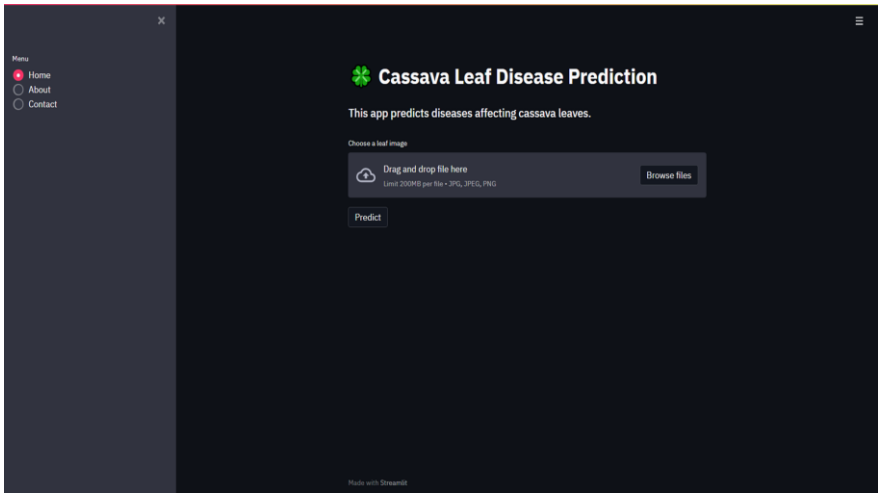


Figure 6.3 Home page of the deployed web app

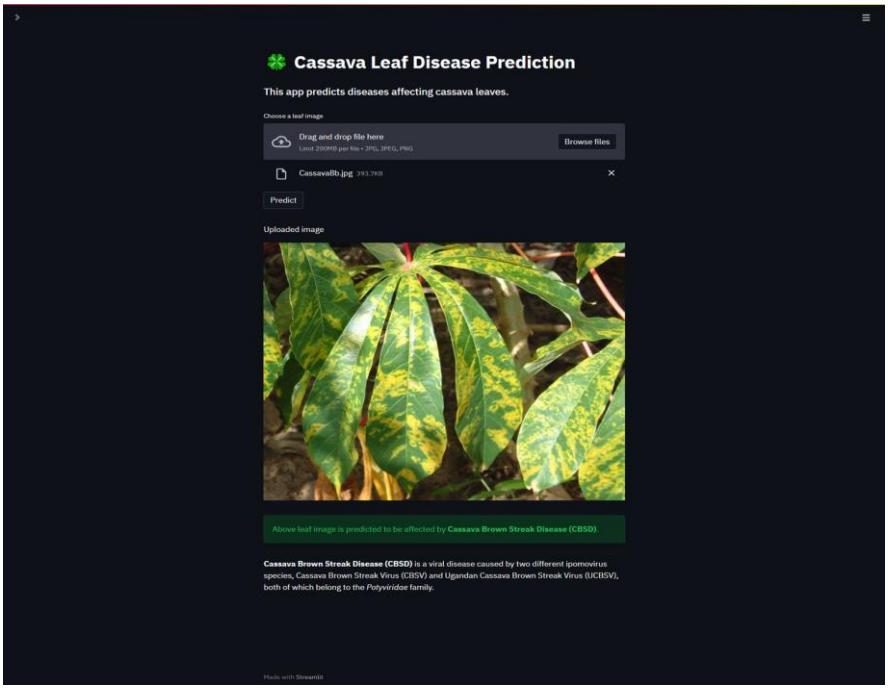


Figure 6.4 Prediction result displayed by the web app

The findings suggest that image classification with transfer learning using the CNN model EfficientNet-B0 is an effective strategy for detecting cassava leaf diseases. This technique avoids the time-consuming and work-intensive process involving extracting features from photos to train models, as well as training a model from scratch, and the model may be simply developed and deployed on a hand-held device as a mobile application.

Chapter 7

Conclusion and Future Work

Cassava is one of Uganda's most essential crops, providing millions of people with the calories and carbohydrates they need. However, a number of illnesses pose a significant danger to its output. The goal of this research is to discover and categorize four distinct leaf diseases in the cassava crop. To categorise cassava leaf pictures into four illness classes or a fifth class – healthy, the suggested approach used transfer learning on the state-of-the-art CNN model EfficientNet. This study reveals that transfer learning approach provides a rapid and effective solution to plant leaf disease identification, which is otherwise a difficult and expensive operation, with overall accuracy scores of 0.8887 and 0.8851 on the private and public competition leaderboards, respectively. An interactive web application for the same was also developed for easy and convenient usage.

Thus, in the future, this model can be used to create a simple smart phone application that can function effectively under severe limitations and be used by farmers or agricultural workers to monitor and diagnose unhealthy plants in real time.

REFERENCES

[1] Wikipedia contributors. (2021, April 9). **Cassava**. In Wikipedia, The Free Encyclopedia. Retrieved 10:15, April 17, 2021, from https://en.wikipedia.org/w/index.php?title=Cassava&oldid=1016772157

[2] IFAD, F. (2000). **The world cassava economy. Facts, trends and outlook**. Rome, IFAD, FAO.

[3] Ameu, M et al., (2013). **Cassava farmer field schools. FAO Plant Production and Protection Paper** (FAO) eng no. 218.

[4] Legg, J. P., Owor, B., Sseruwagi, P., & Ndunguru, J. (2006). **Cassava mosaic virus disease in East and Central Africa: epidemiology and management of a regional pandemic**. Advances in virus research, 67, 355-418.

[5] Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). **Using deep learning for image-based plant disease detection.** Frontiers in plant science, 7, 1419.

[6] Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., & Hughes, D. P. (2017). **Deep learning for image-based cassava disease detection.** Frontiers in plant science, 8, 1852.

[7] Sangbamrung, I., Praneetpholkrang, P., & Kanjanawattana, S. (2020). **A novel automatic method for cassava disease classification using deep learning.** Journal of Advances in Information Technology Vol, 11(4).

[8] Sambasivam, G., & Opiyo, G. D. (2021). **A predictive machine learning application in agriculture: Cassava disease detection and classification with imbalanced dataset using convolutional neural networks.** Egyptian Informatics Journal, 22(1), 27-34.

[9] Oyewola, D. O., Dada, E. G., Misra, S., & Damaševičius, R. (2021). **Detecting cassava mosaic disease using a deep residual convolutional neural network with distinct block processing.** PeerJ Computer Science, 7, e352.

[10] Amara, J., Bouaziz, B., & Algergawy, A. (2017). **A deep learning-based approach for banana leaf diseases classification.** Datenbanksysteme für Business, Technologie und Web (BTW 2017)-Workshopband.

[11] Priyadharshini, R. A., Arivazhagan, S., Arun, M., & Mirnalini, A. (2019). **Maize leaf disease classification using deep convolutional neural networks.** Neural Computing and Applications, 31(12), 8887-8895.

[12] Atila, Ü., Uçar, M., Akyol, K., & Uçar, E. (2021). **Plant leaf disease classification using efficientnet deep learning model.** Ecological Informatics, 61, 101182.

[13] Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). **Deep neural networks based recognition of plant diseases by leaf image classification.** Computational intelligence and neuroscience, 2016.

[14] Tm, P., Pranathi, A., SaiAshritha, K., Chittaragi, N. B., & Koolagudi, S. G. (2018, August). **Tomato leaf disease detection using convolutional neural networks.** In 2018 Eleventh International Conference on Contemporary Computing (IC3) (pp. 1-5). IEEE.

[15] Barbedo, J. G. A. (2018). **Impact of dataset size and variety on the effectiveness of deep learning and transfer learning for plant disease classification.** Computers and electronics in agriculture, 153, 46-53.

[16] Makerere University, AI Lab. (2020, November). **Cassava Leaf Disease Classification**, Version 1. Retrieved March 15, 2021 from https://www.kaggle.com/c/cassava-leaf-disease-classification/

[17] Tan, M., & Le, Q. (2019, May). **Efficientnet: Rethinking model scaling for convolutional neural networks**. In International Conference on Machine Learning (pp. 6105-6114). PMLR.

[18] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). **Mobilenetv2: Inverted residuals and linear bottlenecks**. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 4510-4520).

[19] **Streamlit** • The fastest way to build and share data apps. (n.d.). Streamlit. Retrieved May 2, 2021, from https://streamlit.io/