DEVELOPMENT OF A WEB AND MOBILE BASED APPLICATION FOR MAIZE LEAF DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK

OSONDU RONALD CHINEDU

(18CK024252)

A PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF ELECTRICAL & INFORMATION ENGINEERING, IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE AWARD OF THE BACHELOR OF ENGINEERING DEGREE IN ELECTRICAL AND ELECTRONICS ENGINEERING

SUPERVISOR: ENGR. DR. KENNEDY O. OKOKPUJIE

JUNE 2023

DECLARATION

I hereby certify that I worked at the Department of Electrical and Information Engineering, Covenant University, on the project detailed in this report, under the supervision of DR. KENNEDY O. OKOKPUJIE. Additionally, to the best of my understanding, no section of this study has ever been submitted elsewhere or here in connection with a previous application for the degree to be granted. All sources of information cited were appropriately recognized.

OSONDU RONALD CHINEDU

(18CK024252)

CERTIFICATION

This is to certify that the project titled "DEVELOPMENT OF A WEB AND MOBILE BASED APPLICATION FOR MAIZE LEAF DISEASE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORK**"** done by OSONDU RONALD CHINEDU meets the requirements and regulations governing the award of the Bachelor of Engineering (Electrical and Electronics Engineering) degree of Covenant University and is approved for its contribution to knowledge and literary presentation.

Supervisor: Sign: ____________________

Name: **Dr Kennedy O. Okokpujie** Date: ______________

Head of Department: Sign: ______________________

Name: **Dr Isaac A. Samuel** Date: __________________

Internal Examiner: Sign: ___________________

Name: ______________ Date: ____________________

DEDICATION

I respectfully dedicate my effort to God in Heaven, whose divine providence has given me the strength, knowledge, and grace to begin and finish this project and providing me with guidance, love, and protection throughout my five years of study at Covenant University. To my supervisor, I would want to offer my heartfelt gratitude for your advice, mentorship, and patience during this study process. Your direction, wisdom, and encouragement have indeed been invaluable to me, and I am glad for the chance to collaborate with you. My parents are also honored by this endeavor, who has served as a major source of support and inspiration for me throughout my years of study at Covenant University.

ACKNOWLEDGEMENT

My heartfelt thanks go to Almighty Father; He kept me alive to oversee this project and granted me wisdom, knowledge and understanding. I want to thank Dr. Kennedy O. Okopujie, my supervisor, for enlightening me on the value of taking part in development projects. I want to thank him for encouraging me to go above and beyond the norm and respectable customs. I want to thank him for his unending encouragement and for giving me the desire to be unique in all I do. I also want to thank Dr Isaac Samuel, Head of the Department, and all of my lecturers for educating me and subjecting me to a variety of assignments that have equipped me for the intrepid completion of this project.

ABSTRACT

Due to maize crop significance in the lives of African farmers and its potential to change African economies, maize is one of six food items that have been selected as a vital food commodity for Africa. Unfortunately, maize plant diseases have adversely impacted farmer yields, which have resulted in a decrease in maize agricultural production. As a result, the purpose of this study is to build a system that can diagnose illnesses of maize leaves using photographs of the leaves. To achieve this goal, pertained convolutional neural network (CNN) models have been chosen because of their prior ability. The transfer learning technique was used to build new models to characterize maize infections based on the curated and pre-processed dataset. The top possible models were determined to be MobileNetV2, InceptionV3, and ResNet50. During training, the accuracy of MobileNetV2, InceptionV3, and ResNet50 were 95%, 91%, and 77%, respectively. The test accuracy of MobileNetV2, InceptionV3 and ResNet50 were 86% , 83% and 72% respectively. Following a performance study, the most accurate model (MobileNetV2) was deployed through a web and mobile application interface. This research will assist farmers determine the sort of infection ailing their maize leaf plants and the right course of action, in addition to furthering SDGs 2 and 12.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER ONE

INTRODUCTION

1.1 BACKGROUND STUDY

Agriculture forms the most integral part of our lives as the food we require is obtained from it. It is a big sector of farming which provides job opportunities for various individuals around the globe. Some of these crops are grown solely for profit. These are cash crops. Food crops are those crops grown for human consumption. One of the most produced food crop nationally and internationally is Maize. Maize is one of the most produced crops in West Africa. It has become the most staple crop in many parts of West Africa replacing traditional grains like sorghum and millet in many regions. A mature maize plant contains 15–18 leaves, yet 5 of those leaves drop off as the plant tassels expand. The leaf offers the surface area where photosynthesis occurs and light is absorbed. Maize is the number one source of carbohydrate in West Africa. Beyond human consumption, it has a variety of uses, including for use in livestock and for production of plastics and ethanol [1]. In Africa, maize makes up about 24 percent of the agricultural area, with an average annual production of roughly 3 tons per hectare [2]. Nigeria is Africa's leading producer, with over 34 million tons produced. According to the data collected from the United States Department of Agriculture, Nigeria now produces ten times more Maize per year than it did when it gained independence in 1960. However unfavorable weather conditions have had a negative impact on maize production. For instance, extreme temperatures, either too hot or too cold can stress the plants and reduce their ability to photosynthesize effectively, leading to decreased yields.

Additionally, the increasing use of land for urbanization and other purposes can decrease the amount of land available for maize farming. Because of how these illnesses affect farmers' production, researchers have created a number of potential remedies for quickly and accurately detecting crop diseases. One approach is to employ machine learning algorithms that may be tuned to distinguish specific illness signs and make accurate predictions about the presence of diseases in crops.

1.2 SIGNIFICANCE OF STUDY

A traditional diagnostic method such as Laboratory testing has been employed by farmers to tackle crop diseases. Farmers collect samples of affected crops and send them to the laboratory for analysis. This involves taking samples of plant tissue, such as leaves or stems and preparing them for testing. The result of the laboratory testing can be used to identify the presence of specific disease and determine the appropriate course of action for mitigating its impact on the crops. However, laboratory testing can be expensive, requiring the use of specialized equipment and trained personnel, which may not be readily available to all farmers. In the context of crop disease detection, a CNN can be trained to recognize the symptoms of specific diseases in crops, such as changes in leaf color or texture. Once the CNN has been trained, it may be used to rapidly and correctly detect the existence of a particular disease in a crop by examining photos of the damaged plants. This can give a rapid and reliable method of detecting and diagnosing agricultural illnesses, allowing farmers to take prompt action to mitigate the disease's impact. For research purposes, the implementation of a CNN-based model could be a valuable tool for studying the spread and progression of diseases in maize crops, providing insight into the underlying mechanisms and potential interventions. Additionally, the use of a CNN in this context could be an effective way to apply machine learning techniques to the study of plant diseases, potentially leading to the development of similar systems for other crops and diseases.

1.3 PROBLEM STATEMENT

Maize is grown on a local, moderate, and large scale across Africa under a variety of conditions related to the environment and climate, adding to food production and commercial crops, but the primary challenge is that maize plants are prone to a variety of ailments. Maize streak viral disease (MSVD), Maize Grey Leaf Spot (MGLS), Northern Corn Leaf Blight (NCLF), and Maize Common Rust (MCR) are the most common epidemic illnesses discovered in maize. Maize plant disease has considerably impacted farmers' yields, which can cause a fall in agricultural production of maize. To tackle these diseases, a web and mobile–based model would be accessible to farmers and other agricultural professionals, making it easy to use and requiring no specialized software or training.

1.4 AIM AND OBJECTIVES OF STUDY

1.4.1 AIM

The study aims to develop a Web and Mobile Based Application for Maize Leaf Disease Classification using Convolutional Neural Network.

1.4.2 OBJECTIVES

The objectives of the project are to:

i. curate maize diseased leaf dataset;

ii. pre-process the curated maize diseased leaf dataset;

iii. develop models based on transfer learning technique.

iv. evaluate the developed model effectiveness using the appropriate metrics, such as recall, accuracy and precision;

v. implement the model as a web and mobile-based application, allowing farmers and other agricultural professionals to access and use it easily.

1.5 METHODOLOGY

The methodology of this project is itemized below:

(i) Data Procurement and Data Collection.

(ii) Identifying suitable CNN architectures to be used to develop the classification model.

(iii) Employing transfer learning from pre-trained models to create the image classification model.

(iv) Perform necessary test and validation checks to ensure the model is not overfitting to the training data.

(v) Provide a comparative analysis of the models tested to determine which is best for maize leaf disease image classification.

(vi) Deploying Trained Model to a Web and Mobile Application.

1.6 SCOPE OF STUDY

For the purpose of model training and evaluation, this project comprises collecting and categorizing photographs of maize leaves as well as dividing the sample set into training, testing, and validation sets. A CNN model will be designed using a deep learning framework, and the model will be trained on the labeled dataset. The model is evaluated to assess its accuracy and analyzing the results to identify areas of improvement. After evaluation, the model is deployed to a web app and mobile app that allows users to upload images and receive classification results.

1.7 LIMITATIONS OF STUDY

i. The model's capacity to classify maize leaf images is restricted to those it has been trained on.
ii. The model may overfit to the training data, resulting in poor performance on new, unseen data.
iii. For accurate classification, the model requires that the picture be in the RGB (red-green-blue) color space.
iv. For categorization, the model requires a high-quality picture.

1.8 PROJECT REPORT ORGANIZATION

Conforming to the guidelines given by the Department of Electrical and Information Engineering, this project will be divided into five chapters detailing the extent of study, the reason for analysis, the basis, work carried out, etc. Chapter one contains the project introduction, which entails the background of study; the significance and aim of the project; as well as the objectives, scope, and the methodology implemented for the project. Chapter Two entails a comprehensive and well-written literature review of past works related to the project topic, focusing on certain areas of interest. Chapter Three discusses the design specifications and methodology used in the creation of the proposed project. Chapter Four focuses on the testing and implementation; challenges faced, suggestions on further work, and also the results obtained from the project. Chapter Five consists of the conclusion and recommendations.

CHAPTER TWO

LITERATURE REVIEW

2.1 PREAMBLE

Agriculture is a major economic sector in Nigeria, and has a big economic impact on the country [3]. In Nigeria, maize is grown in a variety of climates, including the humid and wet conditions found in the southern part of the country, and the dry arid conditions found in the northern part of the country. Maize is also grown at different elevations, from sea level to high elevations in the mountainous areas of the country.

Maize is susceptible to a number of diseases, including leaf diseases that can affect the appearance and health of the plant. Some common maize leaf diseases in Nigeria include Maize streak viral disease (MSVD), Maize Grey Leaf Spot (MGLS), Maize Common Rust (MCR).These diseases can be caused by fungi, bacteria or viruses and can be transmitted through seeds, soil or insects. In order to safeguard their maize harvests, farmers must be conscious of these diseases and take action to manage or avoid them. In response to this, researchers have been exploring various approaches to developing systems that detect these diseases.

2.2 MAIZE LEAF DISEASES

Viruses, bacteria, fungus, and nematodes are just a few of the pathogens that can cause infections in maize leaves. Some of the most common diseases include Northern Corn leaf blight, Gray leaf spot, Common rust, Southern rust and stalk rots. These diseases can cause significant damage to maize crops and lead to yield loss. The fungus Exserohilum turcicum is responsible for northern leaf blight and results in long, narrow, and tan to brown leaf lesions. Gray leaf spot caused by the fungus Cercospora zeae-maydis, symptoms include small, circular, gray leaf spots with dark borders. Common rust caused by the fungus Puccinia sorghi, symptoms include orange or rust-colored pustules on the leaves. Southern rust caused by the fungus Puccinia polysora, symptoms include yellow, orange or red pustules on the leaves. Stalk rots caused by various funguses, symptoms include yellow, orange or red pustules on the leaves. Stalk rots caused by various fungi, symptoms include wilting, lodging and breaking of the stalks. Preventive measures include using disease- free seed, crop rotation and proper sanitation.

2.3 CONVOLUTIONAL NEURAL NETWORK

A common artificial neural network type for image and video identification applications is the convolutional neural network (CNN). CNNs are particularly good at spotting patterns and features in images since they are made to process data with a grid-like architecture, like an image. They have numerous layers which work together to recognize and categorize objects in photos or videos [5]. In especially in radiology and pathology, convolutional neural networks (CNNs) have been employed for illness diagnosis in medical imaging. For instance, CNNs have been trained to recognize skin cancer in dermoscopic images, lung cancer in CT scans, and diabetic retinopathy in fundus images. CNNs can automatically learn features from the images, which are then employed to group the photos into various groups, such as “normal” or “abnormal”. A substantial collection of photos is used to train the CNNs labeled with the corresponding disease or condition, allowing them to detect the disease in new images with high accuracy.

One advantage of employing CNNs for illness diagnosis is their ability to evaluate a lot of pictures rapidly and effectively. The efficacy of the disease detection algorithm can potentially be enhanced by combining CNNs with other image processing methods, such as segmentation. It is worth noting that CNNs for disease detection are still in the research stage and not yet widely used in clinical practice, but the results are promising and the area is under active research.

2.4 ARTIFICIAL NEURAL NETWORK

Artificial neural networks (ANNs) are a type of machine learning model that can be used for disease detection. ANNS can learn from data and make predictions since they are modeled after the inner workings of the human mind. ANNs may be trained on a dataset of medical pictures, such as MRI scans, in the context of illness identification and learn to identify patterns that indicate the presence of a particular disease. Once trained, the ANN can then be used to analyze new images and make predictions about whether or not the disease is present and is used in disease detection tasks, including detecting cancer, lung diseases and heart diseases.

2.4.1 CONVOLUTIONAL NEURAL NETWORK VS ARTIFICIAL NEURAL NETWORK

While CNN (Convolutional Neural Network) and ANN (Artificial Neural Network) are comparable in terms of deep learning algorithms, they have diverse uses and distinct benefits. CNNs are primarily used for image and video analysis, such as object recognition and image classification. They are developed to interpret information having a topology that resembles a grid and are able to learn spatial hierarchies of features through the use of convolutional layers. ANNs, on the other hand, may be used to a number of tasks, such as voice recognition and natural language processing in addition to image and video processing. ANNs are typically composed of fully connected layers and are not as specialized as CNNs for processing grid-like data. In conclusion, whereas

ANNs can be utilized for a variety of tasks, CNNs are a specialized sort of ANNs that are especially well-suited for image and video analysis tasks.

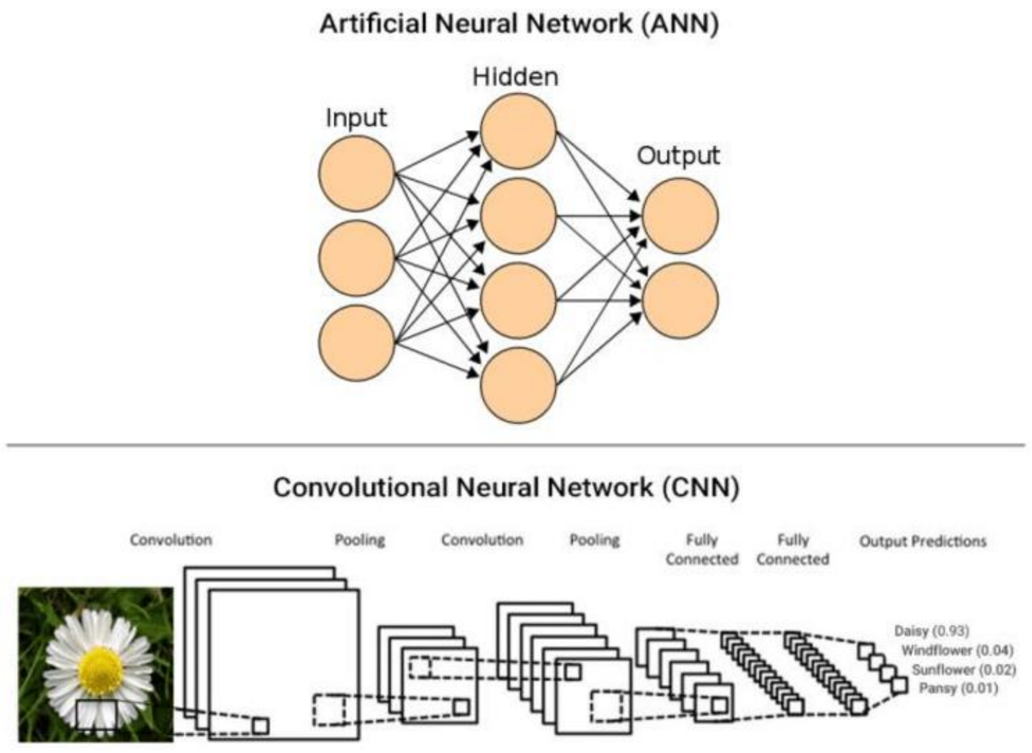


Figure 2: 1 Artificial Neural Network Vs Convolutional Neural Network [4]

2.4.2 TRANSFER LEARNING

Instead of creating a model from zero, the machine learning technique known as transfer learning employs a previously trained model as a reference point for a new task. The pre-trained model may be used to complete a different but related job with less training data because it has previously learned generic characteristics from a big dataset. A smaller dataset tailored to the new task can be used to fine-tune the pre-trained model or its feature extractor can be used to extract the feature from which a new model tailored to the current job is fed [5]. Transfer learning has been shown useful in a number of areas, including voice recognition, processing natural languages, and machine vision..

2.5 REVIEW OF RELATED WORKS

To detect and identify the type of leaf as well as the illness it has been damaged by, researchers have employed Convolution Neural Networks (CNN) and several machine learning approaches. In this context we will investigate what they did and the results of their study.

Malusi Sibiya et al [7] developed a model capable of recognizing three forms of maize leaf disorders (common rust, gray leaf spot and northern corn leaf blight) among healthy leaves. Using the Neuroph Studio framework and a Windows PC, the author created his own 50 layer neural network. 100 photos of each ailment and 100 images of healthy leaves were used to train the network, Captured in a maize farm with a Pixel 3 smartphone. In independent class testing, the model achieved excellent accuracy rates: common rust was 87%, Northern Corn Leaf blight was 99.9%, gray leaf spot was 91%, and healthy maize leaves were 93.5%. Additionally, the model was built on a user-friendly graphical user interface (GUI) platform, making it accessible for non-programmers.

Bin Lui et al [8] Among the 7,669 photos of grape leaves shot with a smartphone camera were powdery mildew, brown spot, mites, black rot, leaf spots, and healthy leaves. We collected information on healthy leaves, mites, downy mildew, and anthracnose. To limit the amount of parameters and avoid overfitting, the deep segmented pooling layer was employed in the model. Due to the various lengths of leaf images disease spots, inception features are added to the model to increase multi-scale picture retrieval capabilities. The suggested CNN-based disease detection method for grape leaves was created on a Tesla

GPU architecture employing the TensorFlow and Keras frameworks. Utilizing the larger dataset, the proposed model was trained to categorize seven different types of grapevine leaves. With a recognition accuracy of 97.22%, the experimental findings suggest that the proposed technique outperforms other well-known transfer learning methods. As compared to typical ResNet and GoogLeNet designs, the proposed DICNN model achieves optimal solution and greater accuracy throughout the training phase. The study's findings show that the proposed strategy completely categorizes illnesses affecting grape leaves, and they give a benchmark for the use of deep learning techniques to agricultural disease classification.

Erik Lucas da Rocha [9] obtained a Plant Village dataset. There are 3852 single-leaf photos of maize leaves, each of which is classified as gray leaf spot, northern corn leaf blight, common rust, or healthy. The experiment's goal is to assess each CNN's ability in identifying maize leaf disease in the dataset. Performance metrics were all utilized when each CNN was evaluated using the top hyperparameters determined through Bayesian optimization. It's worth noting that the three CNNs' accuracy was 97%, indicating how optimization boosted generalization across all models. The gray leaf spot class received somewhat lower production scores than the other 3 classifications when the data from all 4 classifications was combined. This is because the classes differ in size; the gray leaf spot is the smallest compared to other classes.

Jing Chen et al [10] finally, there were 7905 photographs in the database, all of which were taken with a Canon PowerShot G12 camera and show tea leaf illnesses. The number of photos for each class utilized as the illness classification model's training, validation, and testing datasets were gathered. The network architecture discussed here, known as LeafNet, is advancement over the traditional AlexNet model. The LeafNet architecture specifically includes five fully connected layers, two fully linked layers, and the last layer acting as the classification layer. Moreover, the first, second, and fifth convolutional layers are constructed using a filter count that is intended to be equivalent to 50% of the filters used in AlexNet. Moreover, the completely linked layer has 500, 100, and seven neurons, respectively, instead of the typical AlexNet architecture's use of 100. The accuracy was assessed in this study for identifying disease conditions in tea leaves from pictures. The LeafNet system successfully identified tea leaf disorders more consistently than the SVM and MLP methods (60.62% and 70.77%respectively), with a mean accuracy of 90.16 percentage. In terms of recognizing tea leaf ailments, it is reasonable to claim that the LeafNet system clearly beat the MLP and Svm classifiers. The LeafNet might thus be used in the future to boost the effectiveness and accuracy of detection of diseases in tea plant leaves.

Pooja Wadnere et al [11] conducted a thorough examination of recent advancements in the use of machine learning algorithms to detect leaf diseases in plant. When supplied with sufficient data for training, deep learning techniques demonstrate exceptional accuracy in plant leaf disease recognition. Our review highlights the significance of compiling vast and varied datasets, implementing data augmentation and transfer learning, and utilizing CNN activation maps to enhance classification accuracy. Additionally, we discuss the importance of detecting plant leaf diseases with limited samples and the potential of hyper –spectral imaging for detecting plant diseases at an early stage. Although most deep learning frameworks presented in the literature perform well on their respective datasets, they lack robustness when applied to other datasets. As such, there is pressing need for the development of more robust deep learning models capable of handling diverse disease datasets.

Abdel-Aziz et al [12] Comprehensive convolutional neural models and transfer learning methods were used to distinguish crops and weeds. Specifically, the Residual Network 101 architecture was used and achieved impressive results, with an accuracy of 96.04% for the test set and 98.47% for the validation set, respectively. The dataset included roughly 5539 photos of plants from 12 species at various growth phases. The method required using a number of sequentially applied approaches, such as converting dataset photos to the jpeg format to speed up processing without compromising accuracy and employing data augmentation methods including rotation, magnification, and flipping to produce a more balanced dataset. Lastly, features were extracted using a pre-trained ResNet 101 model. This model has the potential to significantly benefit farmers by maximizing crop yield and minimizing losses.

Ashraf Darwish et al [13]explains how to use a group of trained CNNs to divide photos of maize into four categories, three of which indicate diseased leaves and one which represents healthy leaves. The study's pre-trained CNNs were trained with EDLR on a section of a publicly available data including approximately 174,000 photos of damaged and healthy leaves from diverse plants. The improved VGG16 and VGG19, as well as their ensemble all obtained accuracy levels of 97.9%, 97.7%, and 98.2%, respectively. The outcomes demonstrated that the ensemble model performed better than each of the previously trained models.

Bin Liu et al [14] image processing techniques such disturbance, light disturbance, and PCA jittering were used to provide sufficient pathological images of apples, yielding a total of 13,689 images. The AlexNet framework will serve as the foundation for the creation of a new deep convolutional neural network architecture that accurately detects apple leaf diseases, partial fully connected layers were removed, pooling layers were added, the GoogleNet inception structure was implemented, and network parameters were optimized using the NAG algorithm. The recommended model outperformed other models after being trained utilizing the dataset of sick leaves and the Caffe module on the Graphics card architecture. In terms of recognition accuracy, it achieved a recognition rate of 97.62%. The outcomes were thought to be satisfactory.

Yang Lu et al [15] a database of rice disease images was created, comprising a total of 500 images. A number of sources in Heilongjiang China were used to create the photos. This photography was done with a digital color camera. The CNN model outperformed the BP approach, the Swarm Intelligence Optimisation (SIO) technique and the Maximal Margin Classifiers (MMC) method in terms of training performance, convergence rate, and recognition accuracy.

Lucas G. Nachtigall et al [16] The approach involved creating a collection of annotated images of five common diseases affecting human orchards, which was then randomly separated into train, validation, and test subsets. A Convolutional Deep Learning Model and a Feed Forward Neural Network, with the FNN acting as the baseline, were trained and modified using the validation and training subsets. The CNN had the greatest accuracy, at 97.3%, next by the voting system at 96.0%, the top expertise at 93.3%, and the MLP at 77.3%, according to the data. It was demonstrated that the baseline

FNN performed significantly worse than the CNN design, outperforming any single expert and coming close to matching the performance of a group of experts.

Chrystler T. Obrien et al. [17] 2080 leaf photos were used to build the convolutional neural network, and it was effective in obtaining a decent classification rate. In real-world cultivation circumstances where lighting, illumination, and background might be complex, this study has demonstrated that a convolutional neural network model can identify diseases in mango leaves. Red rust, dark mold, Anthracnose, and healthy leaves were the four types of carabao mango leaf illnesses that the CNN was able to recognize, with an accuracy rate of 81.01% and the greatest performance recorded with a learning rate of 0.0001. For future research, it would be beneficial for researchers to collect larger datasets to increase accuracy rates and make the system more applicable to real-world scenarios. The trained system will be integrated into a mobile smartphone for identification of carabao mango leaf diseases, and the collected datasets will be made available on a public domain repository to support related studies by other researchers. Hsing-Chung Chen et al [18]The findings of this study demonstrate that for accurate object picture classification, the CNN algorithm, utilizing a modified AlexNet architecture, as well as the suggested preprocessing strategy and classification method, may be depended upon. The system achieved 98% precision, 97% F-Measure, 96% accuracy, and 95% Recall. Due to their limited memory capacity, Android-based devices can apply this algorithm because they are well-suited for mobile-based platforms. The results of this study are anticipated to help tomato growers or planters since users will be able to swiftly detect the different plant illnesses that are harming their plants by submitting images of the affected leaves using Android devices. This makes it possible to quickly apply early illness management techniques.

D. Oppenheim et al [19]The proposed system aids in identifying leaf diseases, specifically for five major crops- Corn, Rice, Wheat, Sugarcane and Grape which are susceptible to various diseases. Images of crop leaves are collected from various sources, with 200–300 photos per illness, to create a subset. 20% of the subset is used for validation, while 80% is used for training. The system utilizes Tensor flow and Keras libraries for ease of data preprocessing. The model is constructed on Colaboratory, achieving an accuracy of 97.33%. Additionally, the system suggests appropriate pesticides based on the identified disease. This system can be beneficial for farmers and planters to identify leaf disease and manage them in a timely manner.

Ramar Ahila Priyadharshini et al [20] the study presents a modified LeNet architecture based on powerful convolutional neural networks with deep layers (CNN), which makes it able to accurately classifying maize leaf diseases. The proposed method focuses on combining local and global characteristics to improve classification performance. The study also investigates the potential efficacy of the LeNet architecture for classifying plant leaf diseases by investigating various characteristics, such as kernel size and depth. The authors suggest a 3 x 3 kernel size for categorizing maize leaf disease based on the findings. Additionally, the proposed CNN design can be used to categorize various plant leaf diseases.

Alex Krizhevsky et al [21] the ImageNet LSVRC-2010 Competition, 1.2 million high resolution photos needed to be correctly categorized, hence a substantial, deep convolutional neural network was built and trained. With this network, each image can be correctly classified into one of the 1000 categories. The conventional accuracy was 37.5%   when we tested the network. Comparatively speaking, the design is considerably better today. To speed up training, the model uses rectifier linear units (RLU) and a powerful Nvidia convolution algorithm. Also, we employed a recently created regularization technique termed "dropout," This effectively dealt with the problem of overfitting in the completely linked layers. Our model exceeded the previous winner's test error rate of 26.2% in the ILSVRC-2012 competition, with a test error rate of 15.3%.

Gnanavel Sakkarvarthi et al [22] this paper covers the construction of a dataset made up of two different types of crop disease leaves using the deep convolution network-based VGG16 framework. The dataset was then put through training, testing, dataset pre-processing, and feature extraction. The goal was to improve the model's efficiency and evaluate it against other accessible datasets and methodologies. In comparison to other studies, our suggested method had greater accuracy rates for both grape (98.40%) and tomato (95.71%) infections. The suggested model has outperformed the trained InceptionV3, ResNet 152, and VGG19 models, according to the results of the trial. A training accuracy of 98% and a test accuracy of 88.71% were attained by our proposed CNN model. The improvement in agriculture and an increase in food production were made feasible on this accomplishment. A crucial step in agricultural progress, our work has focused on offering a more precise and effective classification and analysis of in-field crops, leaf pictures, and illnesses.

Omneya Attallah et al [23] the pipeline for automatically identifying and diagnosing tomato leaf disease was suggested in this work. The suggested pipeline uses three compact CNNs with various designs that incorporate the capabilities of each design, in contrast to earlier methods that rely on individual CNN models with several deep layers and a bewildering amount of parameters. There are four steps in the pipeline: image of a tomato leaf preparation, concatenation of features, selection and classification, additionally to TL-based feature extraction and compact CNN training. The tomato leaf images underwent an enhancing process after being originally shrunk. Three CNNs are then trained using these previously processed images: MobileNet, Shufflenet and ResNet18 with Transfer learning techniques. Then, using TL, high level features are obtained from each CNN's final FC layer. The photos of tomato leaf diseases are finally classified using six ML classifiers. The K - Classification algorithm and Support Vector Networks (SVN), respectively, attained the greatest accuracy of 99.92% and 99.90%, according to the results. Its competitive performance is evident when the experimental findings of the suggested pipeline are compared to those of earlier studies on the classification of tomato leaf disease.

Manya Afonso et al [24] two rows were captured on camera during an experiment. Images were collected using imaging equipment once a week for six weeks. To avoid picture distortion, a subset of 532 photographs from various dates was selected from this batch of images and then reduced to 224 by 224 pixels while retaining the aspect ratio. The CNN program can preprocess transformations at this resolution. Following that, the selected images were randomly divided into training and testing groups in the proportion 80:20. The set for training consisted of 426 images of 218 healthy and 208 blackleg-infected plants. Using RGB photos of healthy and diseased plants, the study trained two deep convolutional neural networks. ResNet18, one of these networks, attained an accuracy of 95% and a recall rate of 91% for the illness class. These findings demonstrate the capability of convolutional neural networks in detecting backleg diseased potato plants.

Umesh Kumar Lihore et al [25] the study uses an improved CNN model dubbed ECNN to demonstrate a more effective method for real-time detection of Cassava leaf disease. The traditional CNN model uses a large feature set and computationally demanding procedures, which leads to a significant computational overload. To address these issues, the proposed ECNN model includes new features and characteristics, as well as spatial layer separation, which minimizes the amount of features and computational cost. The test findings indicate that the suggested model beat the Conventional CNN architecture, which had a training accuracy of 89.754% and a training loss of 36.414%, while our model had a training accuracy of 94.689% and a training loss of 24.547%.

Amanda S. Paymode [26] This research describes the gathering and compilation of a dataset made up of two different types of crop disease leaves, which was later used to train and evaluate a VGG16 model. In comparison to other available datasets and approaches, the model that was utilized was able to attain better assessment metric values, which led to an improvement in accuracy for tomatoes and grapes of 95.71% and 98.40%, respectively. For agriculture to advance and food production to rise, crop imagery and illnesses must be better classified and analyzed. Further authentic datasets were going to be gathered, prepared, and used by the authors to train deep learning models that interpret multiple crop leaf photos. For even more in-depth investigation, the authors plan to use CNN models based on ResNet and Inception V3. The job is important because it supports and promotes farmers, which raises family income and ultimately helps build strong nations.

Vaibhav Tiwari et al [27] to build the fully convolutional network architecture for identifying and categorizing plant illnesses, 27 leaf pictures from various crops were employed. Python is the programming language that was utilized to train the model. The input photographs were 256 256 pixels in size, with a batch size of 32.Using multiple photos from various categories, a dense model was assessed. The model was evaluated using cross-validation before being tested on unseen photographs from the testing set. The proposed system had a mean test accuracy of 99.199% and correctly identified and classified plant illnesses 99.58% of the time. In future the dataset will be expanded with more diverse images to enhance the model's performance in challenging environments.

Zhen- Tao Hu[28] this paper demonstrates a training method based on SA-GAUSS that aims to integrate and optimize both the convolutional layer and full connection layer. By doing so, it aims to address the lack of feature extraction and global optimization in the original algorithm, resulting in improved network training accuracy. This paper demonstrates that the convolutional neural network trained using the integrated optimization algorithm achieves higher recognition rates.

Solemane Couilibaly et al [29] the proposed methodology utilizes feature extraction, specifically using the previously trained CNN model VGG16 on ImageNet. The experiment was conducted using Keras/TensorFlow, a widely available deep learning framework on an Intel core I5 laptop. The test was configured with an 80/20 split, which resulted in better model performance on evaluation measures. The test dataset used for model evaluation consisted of 18 images with diseases and 9 images without disease; the model's validation accuracy was 95.00% after 30 epochs, while its test accuracy was 89.00%.

Geetharamani et al [30] an advanced convolutional neural network model is presented in this study for detecting leaf image diseases. The model is trained using an open sample of 39 categories of plant leaves pictures. The model's performance was improved using six distinct data augmentation methods: picture flipping, noise fusion, color management, color injection, rotations, and cropping. The outcomes show that the model's usefulness may be improved by including more data. Different training epochs, batch sizes, and dropouts were used to develop the recommended model. The recommended approach outperforms well-liked transfer learning methods by utilizing validation data. Large simulations show that the proposed model outperforms conventional machine learning techniques with an accuracy of 96.46%. Additionally, the proposed model’s consistency and reliability were evaluated through testing.

Siti Zulaikha et al [31] the study presents a computer vision method for identifying plant diseases through image analysis of leaves. The proposed approach utilizes a deep learning classifier to achieve a reliable and comprehensive assessment of a variety of leaf presentations. Specifically, the study employs a compact deep learning architecture, Mobile Net V2 which was honed to detect three different tomato illnesses. The performance of the system is then assessed using a set of 4,671 photos from the Plant Village sample. The findings suggest that MobileNet V2 can identify the ailments with greater than 90% accuracy.

Table 2.1 describes the datasets, the CNN model and the evaluation metrics utilized in each of the analyzed journals is highlighted in the review.

Table 2: 1 Review of Related Works

S/N	**Name of Paper**	**Year of publication**	**Source of data**	**Pre-processing techniques**	**Feature extraction**	**Type of CNN used**	**Model development/ Tools used**	**Testing**	**Deployment and Implementation**
1	[7]	2019	The datasets were obtained by taking pictures of a maize field.	After the data was collected, the data were classified into different classes: Healthy Plant leaves and three other significant maize classes' diseases. The data was labelled accordingly.	The features extracted were embedded in the program's Library	For this paper, the authors built their Convolutional neural network.	The Neuroph Framework was used to build model.	The model was tested to see if it would differentiate between diseased maize leaves and healthy leaves.	It was not mentioned.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
2	[8]	2020	With a digital camera, 7669 photos of grape leaves have been gathered.	The diseased leaves in the original dataset were augmented using six different augmentation methods. In addition, randomized PCA is applied to expand the original dataset.	Inception structure is used to extract multiscale disease spots.	The authors built a Dense Inception Convolutional Neural Network.	The frameworks TensorFlow and Keras were employed.	The model was tested to see if it would differentiate between diseased grape leaves and healthy leaves.	It was not mentioned. However the author stated that it could be deployed on a Tesla P100 GPU.
3	[9]	2020	The paper contains 3852 images collected from Plant Village Dataset.	The pictures were shaped to 224 X 224 pixels for adaptation and data augmentation strategies were used.	The features were extracted.	The CNN models used were: AlexNet, ResNet-50 and SqueezeNet.	The tools used were Python and PyTorch.	The model was tested to see if it would differentiate between diseased maize leaves and healthy leaves.	It was not mentioned.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
4	[10]	2019	The images were captured using a Cannon PowerShot G12 Camera.	The images in the paper were resized to 256 x 256 pixels.	The features were extracted via feature extraction filters.	The LeafNet Model was used.	The tool used was MATLAB	The Leaf Net was compared with SVM algorithm and MLP. It showed greater superiority.	It was not stated.
5	[11]	2022	The papers utilized publicly accessible datasets, which included the Plant-Village dataset.	The images were resized using various data augmentation techniques.	The features were extracted.	The models used were ResNet50, VGG16, LeNet-5 and SENet model.	The data were divided into two sets for each of the models employed in these publications, namely: Test Set and Training Set	To assess the models' effectiveness, multiple performance metrics were employed, including the accuracy of classification.	The majority of the models created were implemented via a mobile app and a web application.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
6	[12]	2019	5339 plant images were obtained from the University of Southern Denmark.	JPEG format was used to convert the photos. The dataset is then balanced using data augmentation techniques like rotation, zooming, and flipping.	The pre-trained ResNet 101 model was used for feature extraction.	The system was powered on ResNet 101	The model was developed by transfer learning.	Accuracy, Precision, Fl score and recall were all tested after model was built.	The model was deployed on a web application.
7	[13]	2019	The dataset used in this paper contains 12332 images gotten from the database hosted at Kaggle.	Data enhancement techniques include rotations, height change, width change, shear, magnification, and horizontal flipping. All photographs were resized to attain a resolution of 256 by 256 pixels.	The features were extracted.	VGG16 and VGG19, two pre-trained CNNs, were employed.	To build the structure the Keras Library was used.	Accuracy, Precision and Recall where evaluated in this paper to demonstrate the Model's ability.	It was not mentioned.

S/N	Name of Paper	Year of publi cation	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model developme nt/ Tools used	Testing	Deployment and Implementation
8	[14]	2017	1053 apple leaf diseases were photographed using a digital camera.	Picture rotations, image symmetry, and PCA jittering were among the digital image pre-processing techniques employed.	The features were not extracted.	AlexNet model and GoogleNet Inception structure were the CNNs used.	On the GPU platform, the Conventional Architecture for Fast Feature Embedding was employed.	Various metrics were used to evaluate the model.	It was not mentioned.
9	[15]	2017	A dataset of 500 natural images were captured from rice experimental field.	To minimize the quantity of the training data, rice disease photos are reduced in size from 5760 × 3840 pixels to 512 x 512 pixels.	Sparse auto encoding is used for feature extraction.	AlexNet CNNs architecture is used.	MATLAB was used.	The proposed model was tested to identify 10 common rice diseases.	It was not stated.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
10	[16]	2016	Three types of apple trees were harvested, and images of each leaf taken against a white background were used to create the dataset.	The photos in the input were downsized to 256 × 256 pixels.	No features were extracted.	AlexNet CNNs architecture is used.	Caffe and DIGITS were the tools used in building the CNNs.	The results were evaluated by calculating accuracy of each classifier and confusion matrix analysis.	It was not mentioned in the paper.
11	[17]	2020	A camera was used to acquire 2080 leaf photos.	The photos in the input were downsized to 256 × 256 pixels.	No features were extracted.	The authors built their own CNN model.	Tensor Flow was used in building the CNNs.	Both the accuracy of the training and validation data was evaluated.	The trained system was installed into a mobile smartphone.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
12	18]	2022	The dataset may be accessed via the API and was collected via Kaggle.	The input images were resized to 64 x 64 RGB pixels.	No features were extracted.	The authors built their own CNN model.	Python, Google Colab, and a number of libraries, including NumPy, Scikit-learn, and Pandas.	Evaluation metrics like: Accuracy, Precison, Recall and F-measure value were tested.	The model was implemented on an Android platform to predict tomato leaf diseases.
13	[19]	2020	The images were taken and stored in JPG format.	For pre-processing, the images have been resized to 150 x 150 dimension	The features were extracted.	MobileNet model is proposed in this paper.	Tensor flow/Keras libraries and the Python programming language are used.	Two experiments were tested: Dataset well trained and Dataset that weren't trained.	It was deployed to a mobile app.
14	[20]	2019	The Plant Village collection contained 3852 maize photos in total.	For pre-processing, PCA whitening is used in this work.	The features were extracted.	This study proposes a brand-new LeNet architecture-based technique.	The activation function Rectified linear unit (ReLU) was utilized.	The CNNs model is trained to test for accuracy for four classes of diseases.	It was not mentioned.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
15	[21]	2012	The photographs were gathered from the internet and tagged with the help of a crowd-sourcing program.	The photos were downscaled to 256 by 256 pixels, which is a fixed resolution.	The features were not extracted.	The authors built their own CNN model.	Python was the tool used.	The images were tested in the ImageNet LSVRC-2010 Contest.	It was not mentioned.
16	[22]	2022	The Plant Village Dataset is where the pictures came from.	Although the effectiveness of the CNN model continuously increased with that size of the input, the input image's size was modified to 128 x 128.	To extract features, a padding procedure was used.	Transfer learning models were trained and validated using the tomato dataset.	SoftMax and RLU maximum activation function were the tools used.	Training Accuracy, Validation Accuracy and Confusion matrix was generated.	The model was not deployed.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
17	[23]	2023	The Plant Village Dataset is where the pictures came from.	The images were modified to a new size of 224 x 224 x 3 for the three CNNs.	The features were extracted.	ResNet-18, ShuffleNet, and MobileNet are three miniature CNNs used in this experiment.	Softmax function was used.	Accuracy, sensitivity, F1-score, precision, specificity, and the Mathew correlation coefficient are used to assess the pipeline's efficacy.	It was not mentioned.
18	[24]	2019	The images were acquired using an industrial RGB camera.	The photographs were downsized using aspect ratio retention to 224 × 224 pixels.	No features were extracted.	ResNet18 and ResNet50 were the CNN models used.	The CNN used was PyTorch framework.	The classifiers were calculated using confusion matrices.	The model was not deployed.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
19	[25]	2022	Images of cassava leaves totaling 6256 were taken from the public Kaggle Dataset.	The unprocessed Cassava photos are normalized in the pre-processing stage. The photos are likewise free of an imbalance. Several image-enhancing techniques were applied.	The features were extracted.	A proposed Enhanced CNN model architecture was used.	Python language was the tool used.	Tests are done on a number of factors, including accuracy, recall, F-measure, and precision.	According to the author, the analysis may be conducted in real-time contexts.
20	[26]	2022	A collection of 152 crop solutions were collected from the Plant Village Dataset.	Images are 256 × 256 pixels throughout pre-processing, de-noising, segmentation, and after.	The features were extracted.	The VGG16 model was used.	Python language was the tool used.	Various parameters like precision, recall, measure and accuracy are tested.	The model was not deployed.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
21	[27]	2021	The Plant Village dataset and the iBean leaf picture dataset were used to gather the photographs.	Image input sizes were reduced to 256 × 256 pixels and the batch size was set at 32.	No features were extracted.	The dense Convolutional neural network was used.	Python is the framework, and the libraries utilized include Keras, Tensor Flow, OpenCV, and PyTorch.	The evaluation included the mean scores for accuracy, specificity, recall, precision, and F1 score.	The model was not deployed.
22	[28]	2018	The images were obtained from MINST and CIFAR-10 databases.	The photos were downsized to a 28 by 28 pixel resolution, with a grayscale value of 8.	The features were extracted.	An Improved Convolutional Neural Network was used.	Simulation is carried out in MATLAB interface.	Error rate was tested using Gaussian algorithm.	The model was not deployed.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
23	[29]	2019	The images were downloaded from the internet.	Data augmentation techniques like zoom resizing, rotation and flipping.	The features were extracted.	ImageNet is used to pre-train the VGG16 model.	The deep learning framework known as Keras was used.	The assessment took into account the average values for F1 score, recall, accuracy, specificity, and precision.	The concept, according to the researcher, may be included into smartphones or digital cameras.
24	[30]	2019	The Plant Village Dataset was used to download pictures of healthy and diseased plant leaves.	Picture flipping, color correction, noise infusion, PCA analysis, color enhancement, rotation, and scaling were employed as methods for improving the data.	No features were extracted.	The Deep CNN was trained using transfer learning.	Python was used for training and testing processes.	The average values for accuracy, specificity, recall, precision, and F1 score were included in the evaluation.	It was not deployed.

S/N	Name of Paper	Year of publication	Source of data	Pre-processing techniques	Feature extraction	Type of CNN used	Model development/ Tools used	Testing	Deployment and Implementation
25	[31]	2020	4671 images from Plant Village Dataset.	Images are rescaled to 224 × 224 pixels to suit the input requirement of the MobileNet V2 model.	No features were extracted.	MobileNet V2 was used.	TensorFlow on Python Platform.	The final accuracy, final loss, and batch size were all evaluated.	It was not deployed.

Table 2: 2 A Comparison of Previous Related Studies with the Proposed Created System Based on Major Performance Metrics

Author	**Year**	**Size of Dataaset Used**	**Training Ratio**	**Validation Ratio**	**Testing Ratio**	**Pre-processing technique**	**Machine Learning Algorithm Used**	**Performance Metrics**	**Deployed as web application**	**Deployed as mobile application**
[31]	2020	The dataset consists of 4671 images	80%	0%	20%	Images were resized to 224 x 224 pixels.	The model used was MobilenetV2	The accuracy of this model was 90%.	No	No
[26]	2021	Size of dataset is 54303 images.	80%	10%	10%	Images were rescaled to 224 x 224 pixels.	The model used was VGG16.	The accuracy of the model was 98.40% grapes and 95.71% for tomatoes	No	No
[8]	2020	Size of dataset is 7669 images	60%	20%	20%	Images were resized to 256 x 256 pixels.	The model used was DICNN.	The accuracy of this model was 97.22%.	No	Yes

Author	**Year**	**Size of Dataaset Used**	**Training Ratio**	**Validation Ratio**	**Testing Ratio**	**Pre-processing technique**	**Machine Learning Algorithm Used**	**Performance Metrics**	**Deployed as web application**	**Deployed as mobile application**
[18]	2022	Size of dataset is 22930 images	80%	0%	20%	The images were resized to 64 x 64 RGB pixels.	The model used was the AlexNet Model.	The accuracy of this model was 98%.	No	Yes
[12]	2019	Size of dataset is 5339 images.	60%	20%	20%	The images were rescaled to 256x 256 pixels	The model used was ResNet 101.	The accuracy of this model was 98.47%	Yes	No
[24]	2019	Size of dataset is 1052 images.	80%	0%	20%	The images were rescaled to 224 x 224 pixels	The model used was ResNet18.	The accuracy of this model was 95%.	No	No

Author	**Year**	**Size of Dataaset Used**	**Training Ratio**	**Validation Ratio**	**Testing Ratio**	**Pre-processing technique**	**Machine Learning Algorithm Used**	**Performance Metrics**	**Deployed as web application**	**Deployed as mobile application**
MY SYSTEM	2023	Size of dataset is 7608 images.	80%	10%	10%	The images were rescaled to 224 x 224 pixels	The model deployed was MobileNet.	The accuracy of this model was 94%	Yes	Yes

2.6 GAP IDENTIFIED

After reviewing **[31], [24], [26]**, [21], it was observed that many of the models that were developed had not been put into practical use. Although these models were created and assessed using different evaluation parameters such as confusion matrix, accuracy, F1 score, and achieved high accuracy when evaluated, very few of them were ever implemented in the field. Therefore I would develop a CNN model using transfer learning techniques to detect maize leaf diseases and it will be deployed via web and mobile application.

2.7 CHAPTER SUMMARY

An overview of maize leaf diseases, including Northern Corn Leaf Blight, Gray Leaf Spot, Common rust, Southern rust, and stalk rots, is provided in this chapter. These ailments, which can seriously harm maize crops and reduce output, are brought on by viruses, bacteria, fungus, and nematodes. Utilizing disease-free seed, rotating crops, and maintaining good hygiene are all preventative approaches for these illnesses. Convolutional neural networks (CNNs) and artificial neural networks (ANNs), machine learning models that can be utilized for disease detection, are then covered in the chapter. While ANNs are more versatile and can be utilized for a range of tasks, including voice recognition and natural language processing, CNNs are best suited for image and video analysis. Lastly, an analysis and summary of earlier research projects or studies that looked into the application of Convolutional Neural Networks (CNNs) and other machine learning techniques to detect and classify plant diseases based on leaf photographs. The main findings, techniques, and limitations of this research will be highlighted, and a summary of the available literature will also be given. Numerous papers from different authors and institutions would likely be included in both the individual topic under discussion and the review of related literature. Researchers can expand on current knowledge and pinpoint areas for additional improvement in their own work by reviewing and summarizing prior research.

CHAPTER THREE

METHODOLOGY

3.1 PREAMBLE

This chapter examines the overall approach, procedures and techniques used to plan, execute, develop and deploy the model. The project's many parts and phases are identified, along with their relationships, and their contributions to the system design as a whole are highlighted. In addition, the many methods and instruments used to achieve the project's goals are thoroughly covered in the discussion. The primary purpose of this study is to create a CNN model for categorizing maize leaf disease using a deep neural framework like TensorFlow or Keras. The pre-processed data is used to train the model, and it is then assessed using an evaluation metrics called accuracy. The project also involves developing a web and mobile app that takes input images of maize leaves and provides the disease classification results using the trained CNN model. The app is user-friendly, responsive and compatible with various mobile platforms. Testing and debugging are performed to ensure the app’s reliability and accuracy.

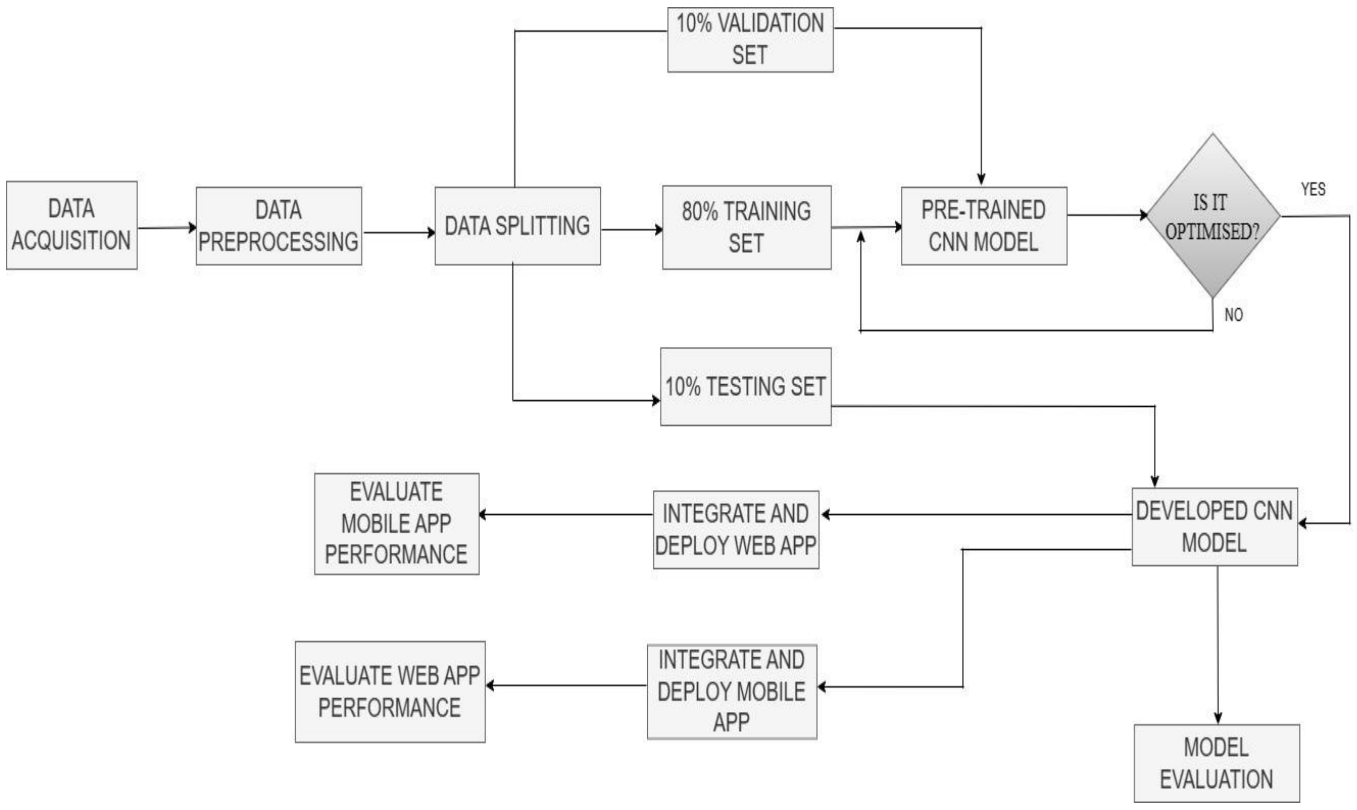


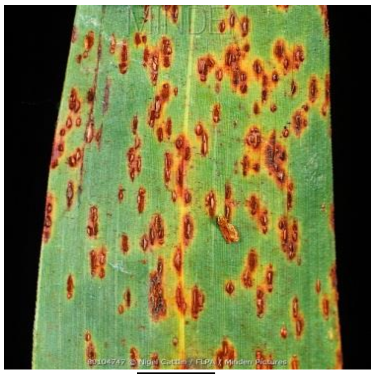
Figure 3: 1 Flowchart of Model Development and Deployment

According to figure 3.1, maize leaf disease dataset were procured. After getting the photos, various preprocessing techniques would be carried out such as image scaling and zooming to improve the model's accuracy and robustness. Thereafter, training, validation, and test sets are created from the dataset. Then, using transfer learning, three CNN models were picked. The 3 CNN models were: MobileNetV2, ResNet50 and InceptionV3 [32]. An Adam optimizer is used to develop the best possible performance of our task. The accuracy of the retrained model is used to assess the model's performance. A web and mobile app is then created that can take input images of maize leaves and provide the disease classification results using the trained CNN model. The app's performance will be evaluated and be deployed on the app store or other relevant platforms.

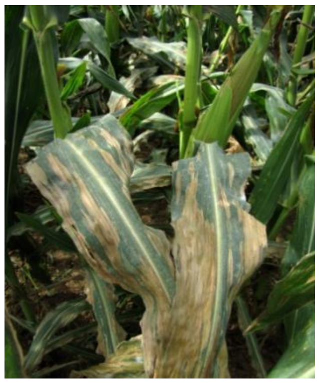
3.2 DATA ACQUISITION

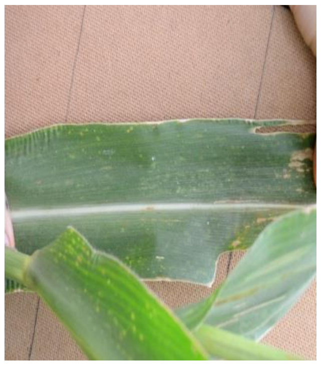
The photographs of healthy and diseased maize plant leaves were utilized in this study's dataset. The University of Pretoria in South Africa provided the data. The images were taken over a variety of years at a variety of locations. Images were taken using Nikon D90SLR camera as well as a variety of smart phones. 7608 images were utilized to create the models in this project. Initially the dataset contained 2355 images of maize leaves; however, some of the images couldn't be identified due to leaf tearing, presence of insects and insect damage. Extra classes were obtained from the Kaggle Dataset to generate a total of five diseased classes and one healthy class. The classes were common rust (CR) (1306 images), gray leaf spot (GLS) (874 images), northern corn leaf blight (NCR) (1146 images), southern rust (SR) (841images), Phaeosphaeria leaf spot (PLS) (857 images), and healthy leaf class (1162 images). Sample images of the various classes are shown below in figure 3.2

(A)


(B)

(C)


(D)


(E)

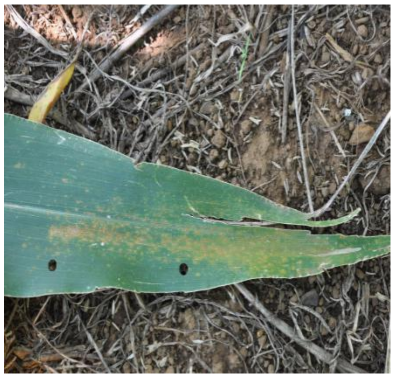

(F)

Figure 3: 2 Each class from the dataset is represented with an image of maize leaves (A)Healthy, (B) Common Rust, (C) Gray Leaf Spot, (D) Northern Corn Leaf Blight,

(E) Southern Rust, (F) Phaeosphaeria Leaf Spot.

3.3 DATA PREPROCESSING

Several photos were difficult to identify during the data pre-processing step because of severe leaf ripping, insect presence, and insect damage. Prior to the photographs being downsized to 256 by 256 pixels, the data was sorted and divided into several labels. I divided the dataset into testing, training, and validation sets in the following proportions: 80: 10: 10. The training set is used to train the models. The validation set is used to adjust the hyperparameters while the testing set is utilized to evaluate the performance of the models.

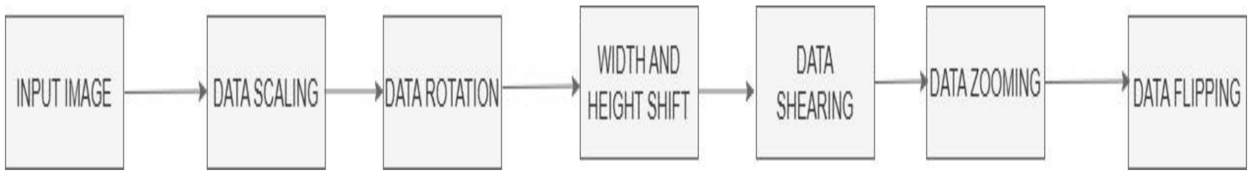


Figure 3: 3 Sequence Diagram of Data Preprocessing

3.3.1 DATA AUGMENTATION

By creating new photos from the old ones using data enhancement methods like rotation, flipping, and zooming [33], the sample set was enlarged. These methods can aid in enhancing the CNN model's generalization and minimizing overfitting. The images will be rotated at angle of 90 degrees to generate new images to increase the variety in the dataset. Python-based Keras library was used to flip the images horizontally and vertically and simulated at different camera angles. These data augmentation techniques can be employed individually or in combination to generate a diverse dataset for training the CNN model.

3.4 MODEL SELECTION

Following the use of data preparation and data augmentation techniques to increase the number of pictures in the validation and training sets, respectively, the transfer learning approach was used to train three selected models. The models were chosen because they had been tested on the ImageNet database, which contains over 1000 different types of pictures, and could predict outcomes with accuracy. Many papers were reviewed in order to choose the top three CNN models for transfer learning tasks. The models employed were MobileNetV2, InceptionV3, and ResNet50.These models were chosen for use in this study because they were shown to be the most effective for transfer learning tasks.

3.4.1 MOBILENETV2

A well-known convolutional neural network architecture called MobileNetV2 was created for portable devices. This is an updated version of the original MobileNet architecture and was unveiled in 2018 by Google researchers. Convolution layers that can be separated in depth form the foundation of the MobileNetV2 architecture ,factorize the standard convolution into two distinct layers: spatial depthwise convolution and 1x1 convolutions [34]. Although spatial depthwise convolution applies a distinct filter to each network interface, 1x1 convolution aggregates the results using a pointwise convolution. [35]. this factorization makes the model more efficient by lowering the amount of parameters and computations needed for each layer. Overall, the MobileNetV2 architecture strikes a compromise between accuracy and efficiency and is a potent tool for computer vision applications on mobile devices.

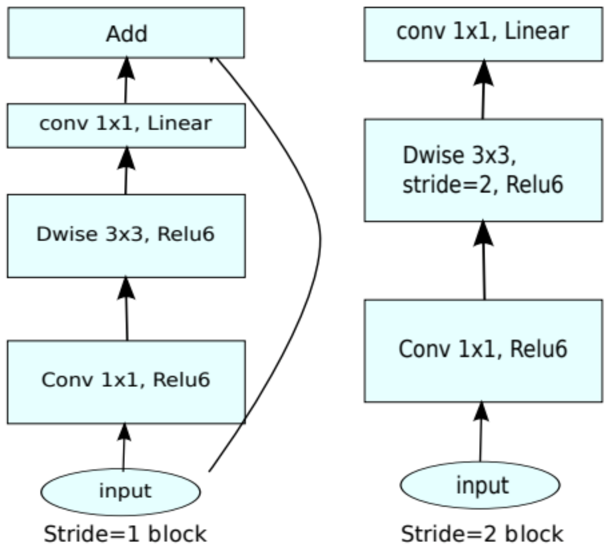


Figure 3: 4 Mathematical Architecture of MobileNetV2 [36]

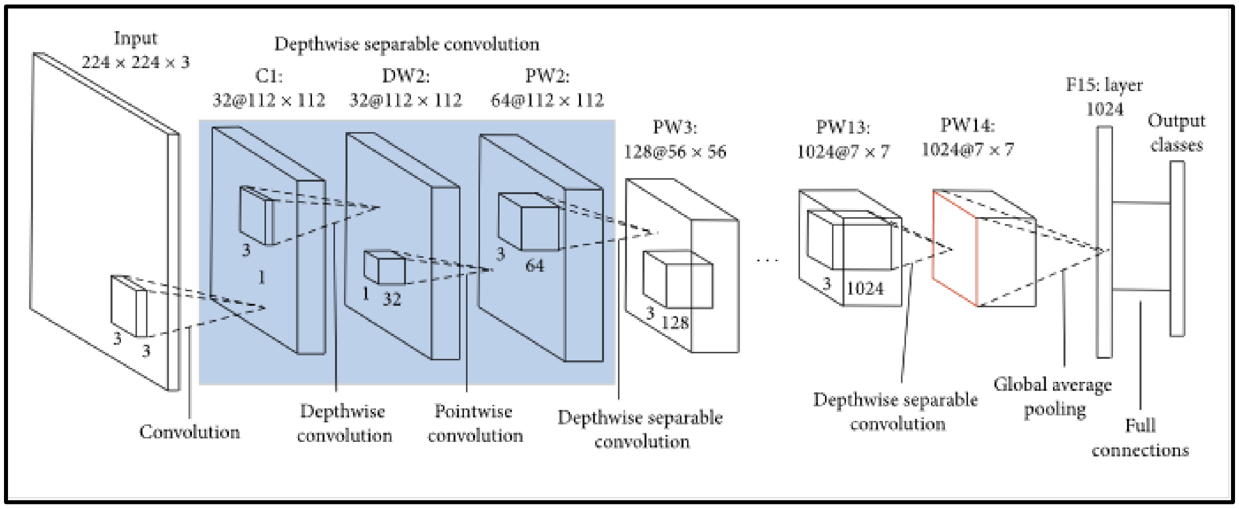


Figure 3: 5 Block Architecture of MobileNetV2 [37]

3.4.2 INCEPTIONV3

InceptionV3 is a 48-layer convolutional neural network. [38]. Convolutional neural networks are used in InceptionV3 to perform tasks like object and picture classification. The sophisticated and effective structure of InceptionV3 allows it to attain great accuracy while requiring fewer parameters than earlier models. The foundation of the InceptionV3 architecture is the idea of using numerous parallel layers of convolutions of various sizes to capture characteristics at various scales. The "Inception modules" that make up InceptionV3 are a collection of modules. Each Inception module combines neural network of varying sizes and pooling methods, which is subsequently supplied to the following layer. Using these Inception modules enables the model to have fewer parameters while yet being more accurate.

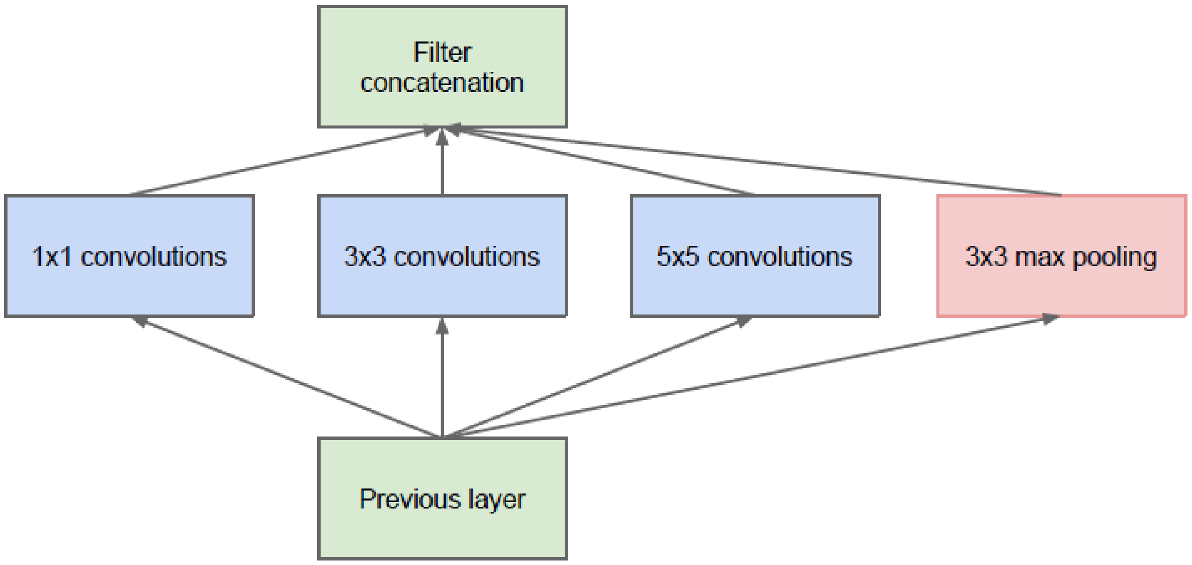


Figure 3: 6 Mathematical Architecture of InceptionV3 model [39]

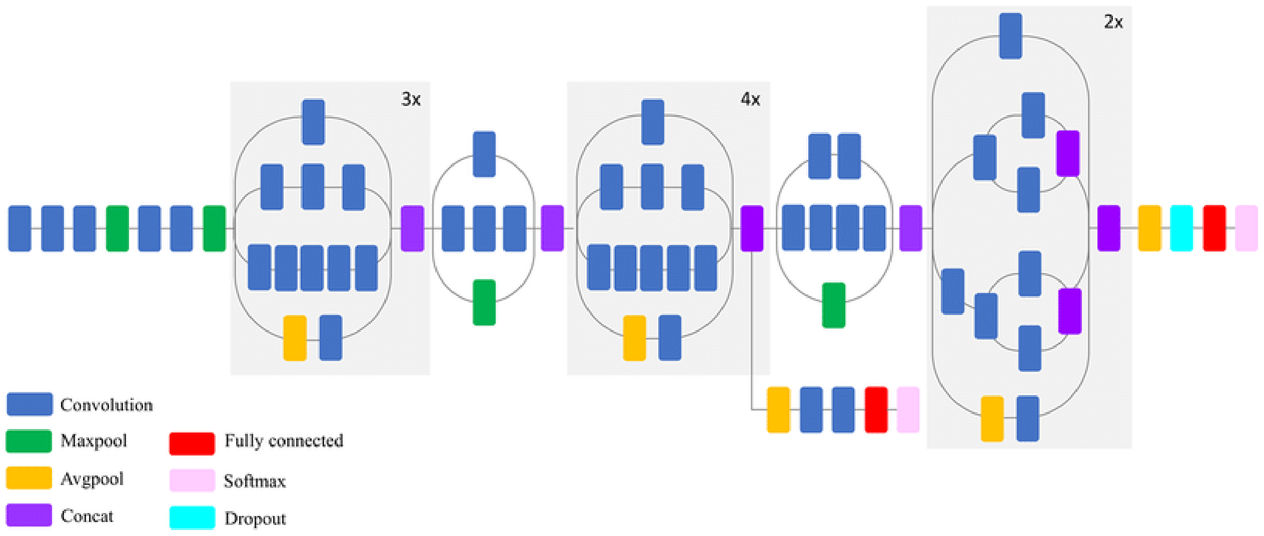


Figure 3: 7 Schematic Diagram of InceptionV3 model [40]

3.4.3 RESNET50

ResNet50 is image classification and object recognition deep convolutional neural network architecture. Microsoft researchers announced it in 2015, and it is based on the residual learning principle, in which the network learns to discover residual connections and use them to increase accuracy. [41]. ResNet50 decreases loss, conserves knowledge gained and increases training efficiency by establishing recurrent connections between layers. 50 layers make up ResNet50, including fully connected, pooling, and convolutional layers. In contrast to earlier neural network architectures, it is intended to be deeper while avoiding the vanishing gradient problem, which can reduce training accuracy. [42].

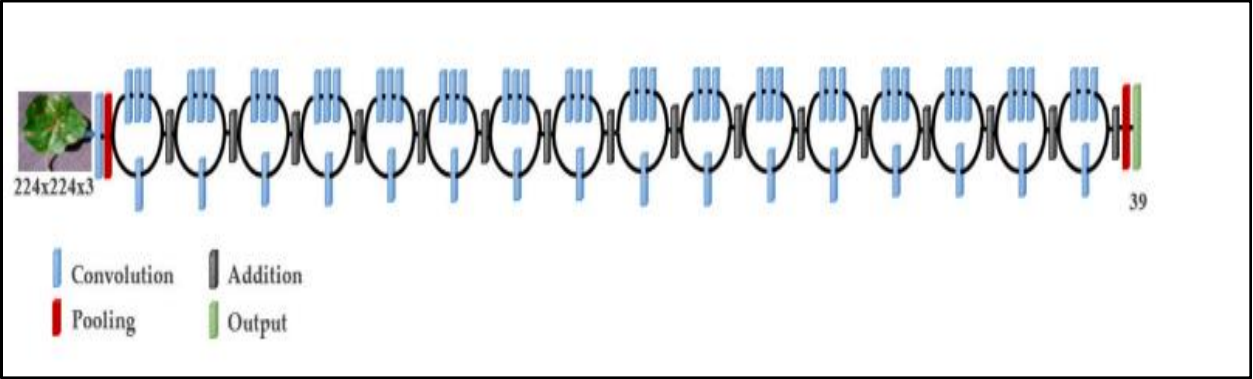


Figure 3: 8 Mathematical Architecture of ResNet50 [43]

3.4.4 SOFTMAX ACTIVATION FUNCTION

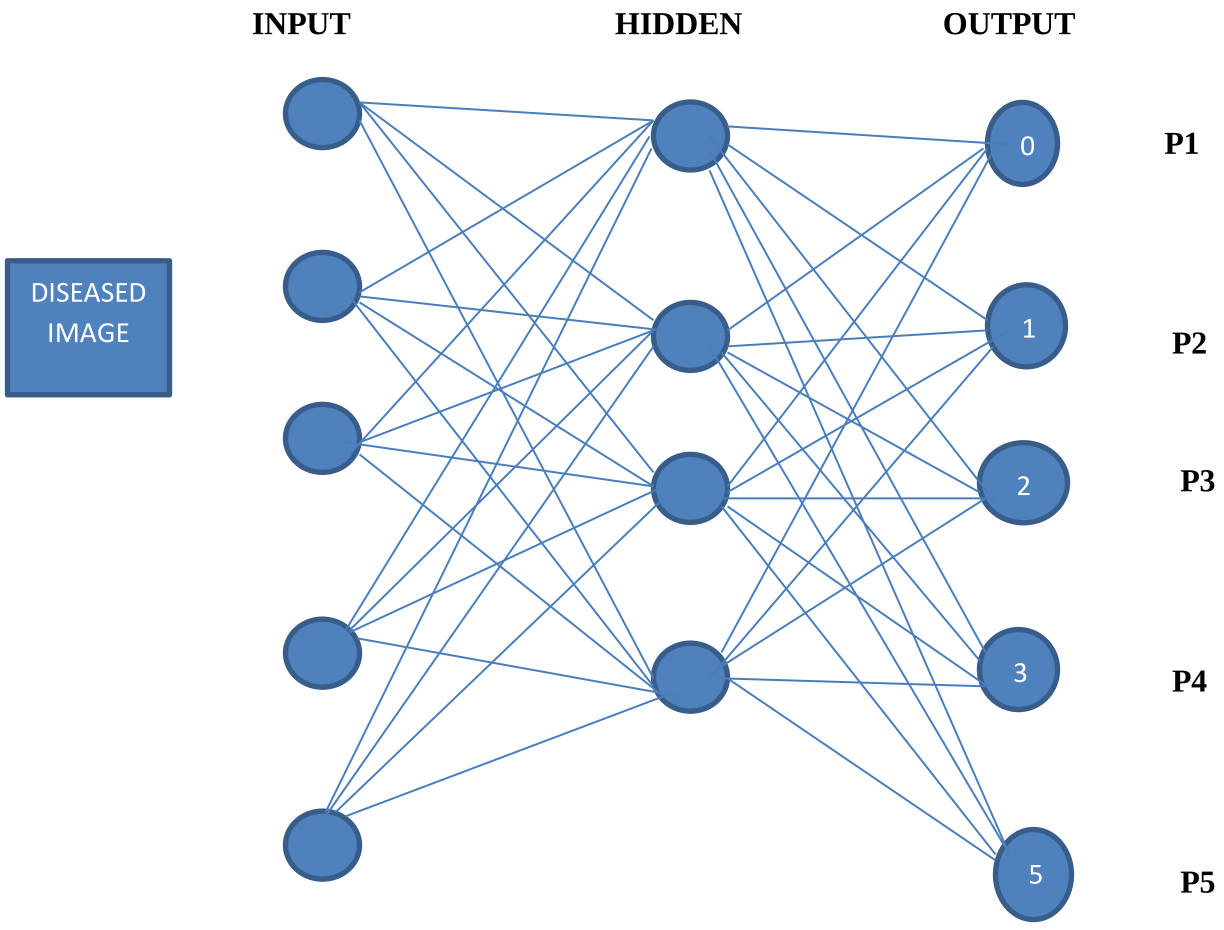


Figure 3: 9 Diagram of Softmax Activation Function

We feed the trained image into the network. The network is trained to predict the input among these 5 classes. The softmax function calculates probability for each class. It is used as an output layer for each neural network.

P1 + P2 +P3 +P4 +P5= 1

The softmax equation is as follows:

$$\boldsymbol{f}(\,\boldsymbol{y_i}\,) = \frac{\boldsymbol{e}^{\,\boldsymbol{yi}}}{\Sigma_{\boldsymbol{k}\ \boldsymbol{e}^{\boldsymbol{yk}}}}$$

Equation 1: Arithmetical interpretation of the softmax function [44]

Where $\boldsymbol{e}^{\,\boldsymbol{yi}}$ = exponential of particular class

And $\Sigma_{\boldsymbol{k}\ \boldsymbol{e}^{\boldsymbol{yk}}}$ = sum of exponential of each class

$$\Sigma_{\boldsymbol{k}\ \boldsymbol{e}^{\boldsymbol{yk}}} = e^0 \ + e1 \ + e2 \ + \ e3 \ + e5$$

= 179.606

For class 1: $\frac{e^0}{179.606} = 0.0056$

For class 2: $\frac{e^1}{179.606} = 0.0015$

For class 3: $\frac{e^2}{179.606} = 0.0411$

For class 4: $\frac{e^3}{179.606} = 0.1118$

For class 5: $\frac{e^5}{179.606} = 0.8263$

From the results below we can see class 5 has the highest probability, so we can see the trained image falls under class 5.

3.4.5 RELU (RECTIFIED LINEAR UNIT)

The Rectified Linear Unit (ReLU) activation function is a popular choice in machine learning, particularly in deep neural networks. First, the input images are preprocessed using an ImageDataGenerator that rescales pixel values and applies various data augmentation techniques. The layers of a pre-trained model are then loaded and frozen to stop them from being trained. The pre-trained model is then enhanced with unique dense layers for categorization. In order to create a 1D vector, the output of the basic model is fed into a flatten layer, which is then passed through a dense layer with 128 units and a ReLU activation function. The output probabilities for each of the six classes are then generated by adding a dense layer with num_classes units and a softmax activation function. The dense layer is given non-linearity using the ReLU activation function, enabling the model to learn more intricate representations of the input data. The ReLU function is defined as follows:

$$ReLU(x) = max\ (0, x)$$

Equation 2: Mathematical Function of the RELU function

This means that if the input value is negative, it is set to zero, otherwise it is passed through as is. This can be expressed mathematically as:

$$f(x) = 0\ if\ x < 0$$

Equation 3: RELU activation function for negative values

$$f(x) = x\ if\ x >= 0$$

Equation 4: RELU activation function for positive values

In the program, the ReLU function is applied element-wise to the output of the dense layer using the following equation: output = max (0, input * weights + biases) Here, the input is the flattened output from the previous layer, weights and biases are the learnable parameters of the dense layer, and the max function applies the ReLU activation function element-wise to the output. This produces the output of the dense layer with ReLU activation, which is then passed on to the next layer for further processing.

3.5 SYSTEM REQUIREMENTS

3.5.1 HARDWARE SPECIFICATION

The equipment utilized to carry out this job has a processor of Intel(R) Core(TM) i7 -7500 CPU at 2.70GHz, with an installed RAM of 8.00GB, 64-bit operating system and a Windows 10 installed.

3.5.2 SOFTWARE SPECIFICATION

The models for this study were created with Python 3.9 in Jupyter Notebook.

The software tools used include:

i. Tensor Flow 2.11.0
ii. Keras 2.9.0
iii. NumPy
iv. Pandas
v. Matplotlib
vi. Flask framework

vii. Jupyter Notebook: You may use and modify notebook files with the server-client web service known as the Jupyter Notebook. It is frequently used as the IDE for constructing deep learning models by data scientists.

viii. Flutter

ix. Spyder

x. Dart programming language

xi. Visual Studio Code

3.6 TRAINING PHASE

Transfer learning was employed in the models' development. The dataset's training portion was eighty percent, validation portion was ten percent, and testing portion was ten percent. Each model was imported from Keras into my JupterNotebook where the model was created (the environment). The model's layers and parameters were entered into my notebook when I was training previously trained CNN models to be capable of carrying out certain tasks. After the loading the model, the base model layers were frozen. By freezing the base model layers, I can minimize the number of training parameters and focus on optimizing the new classification layer for the task at hand. I then added a custom classification layer which consists of: an output layer, a dense layer with all connections, and a flat layer. The fully connected output layer has the same number of units as the new task's classes, and the output layer has a softmax activation layer on top of it. The input layer has 128 units and ReLU activation. To protect the model's pre-trained features, the redundant variables were made to be untrainable while the additional layers were configured to be trainable. Also, the model was created using the ADAM optimiser, and also the learning rate was tuned to 0.0001. 50 epochs were used to train the model. The model was assessed using the cross-entropy loss function, with accuracy serving as the evaluation metric.

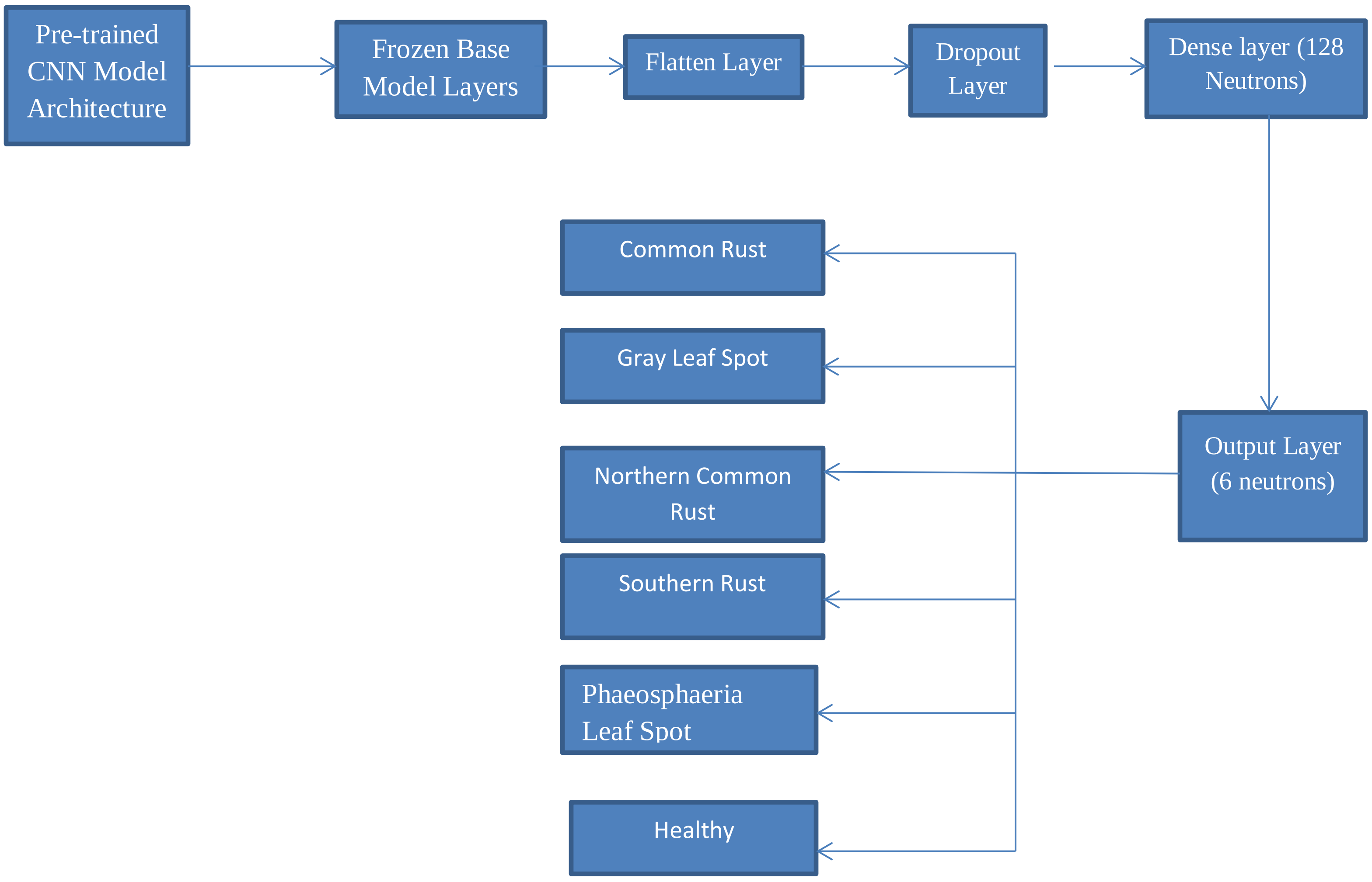


Figure 3: 10 Flowchart Depicting the Transfer Learning Process

3.7 TESTING PHASE

The Convolutional Neural Network (CNN) model's testing step entailed utilizing the trained model to make predictions on fresh, unforeseen data. The CNN inputs a test set of images (or other forms of data) and generates predictions for each image during testing. The model's accuracy was then assessed by contrasting the test set's actual labels with those predicted by the model. A classification matrix was used to evaluate the models performance. A graph of training accuracy versus validation accuracy, training loss versus validation loss, and the confusion matrix were displayed.

3.8 DEPLOYMENT

Through the use of a smartphone app that was developed and put into use, farmers can use the model to classify maize leaf diseases in the field. Computer software created specifically for mobile platforms like cellphones, tablet devices, and smartwatches is known as a mobile app. Flutter is a mobile app framework created by Google that will be used to deploy my model. It will now be released to the app store for user and farmer input. In addition, the model will be deployed to the web. Flask is a web app framework that will be used to deploy my model . The web app is kept on a remote server and is accessible to anybody with an internet connection, in contrast to software that runs natively on a computer's system.

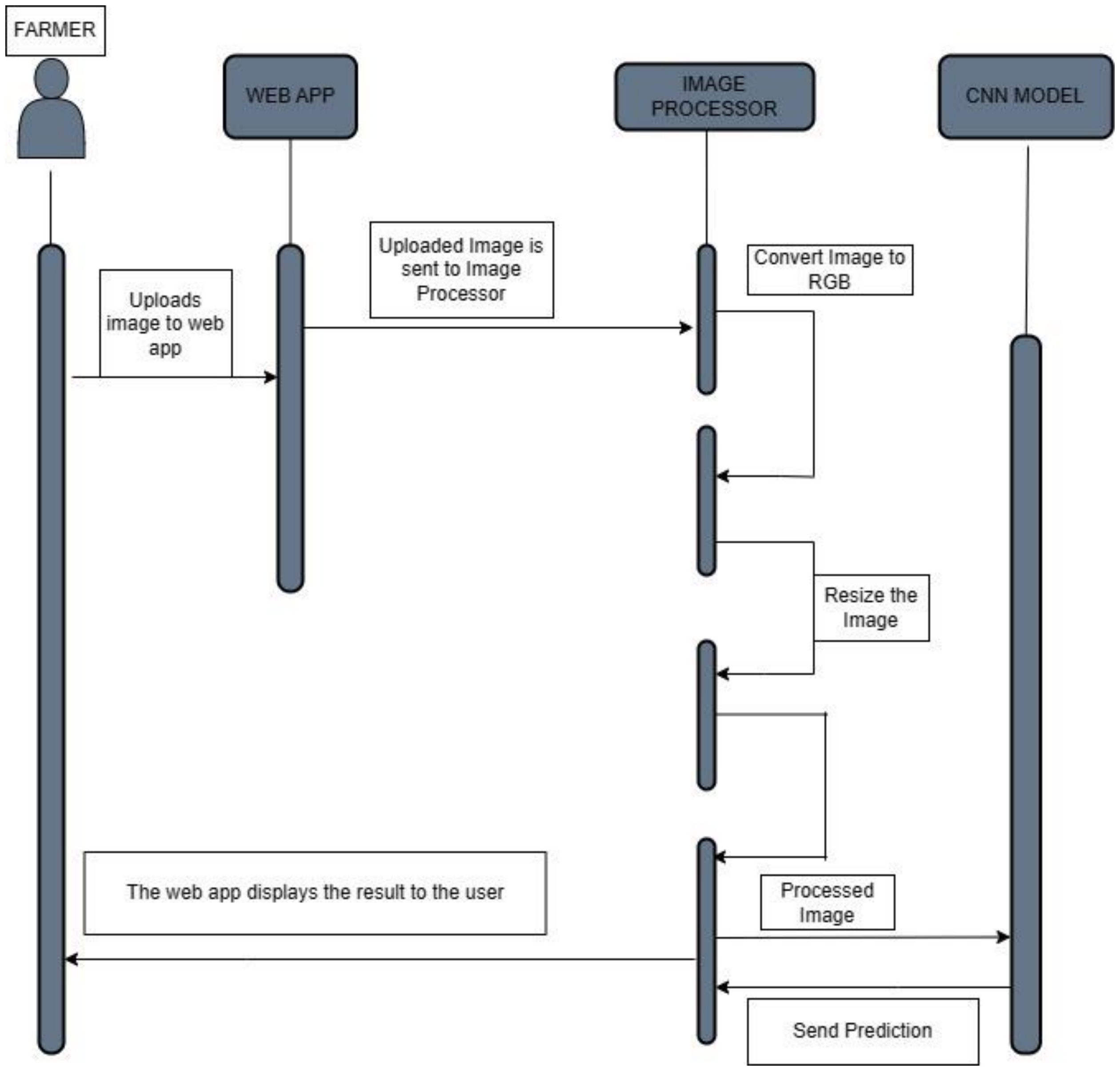


Figure 3: 11 Sequence Diagram of Web App

3.9 CONCLUSION

In this chapter, the model was used and the steps to model development were examined. The steps to model development were also examined and the result of each model was assessed and contrasted to determine which model was the superior. After that, the most effective model was turned into a web and mobile application.

CHAPTER FOUR

RESULT IMPLEMENTATION AND DISCUSSION

4.1 PREAMBLE

Testing a project's validity, efficacy, efficiency, and other aspects is necessary to assess whether or not it is successful after implementation. The application of this methodology would be advantageous to farmers since it would enable and assist farmers in promptly classifying the illness affecting their maize plant. This would enable them to effectively treat the illness. As a result, this chapter explains the implementation of each stage and component of the project, which is then tested to make sure it, is operating flawlessly and that it satisfies the specifications for which it was created.

4.2 IMPLEMENTATION

4.2.1 JUPYTER NOTEBOOK

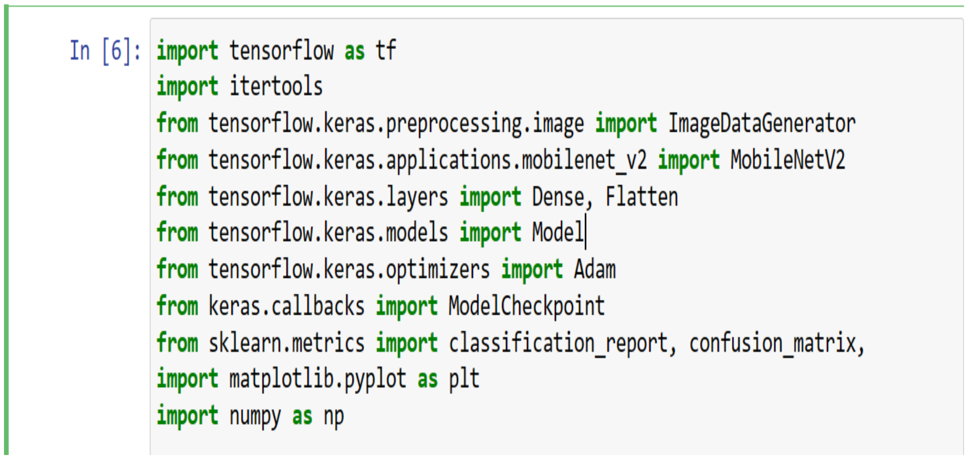
This is a free and open-source internet dynamic software platform that lets you develop and share information that includes calculations, visuals, and data. Users can write and run code in real-time inside the notebook because it is compatible with over 30 computer languages, including Ruby, Julia and Python. Cells make up the notebook interface and can hold markdown text, code, or other forms of content. This makes it simple to communicate ideas, show data, and document code all in the same page.

4.2.2 LIBRARIES

The Tensorflow tool was the first to be downloaded. It is a well-known free software framework used to create and train algorithms for machine learning. The framework allows users to define and optimize complex mathematical computations and model architectures efficiently. The second package downloaded was Numpy. NumPy is a powerful Python library for numerical computing that provides efficient and convenient array manipulation and mathematical operations on arrays. The third package downloaded was Matplotlib. This library is used for producing professional data visualizations like line charts, scatter diagrams, bar plots, and more.

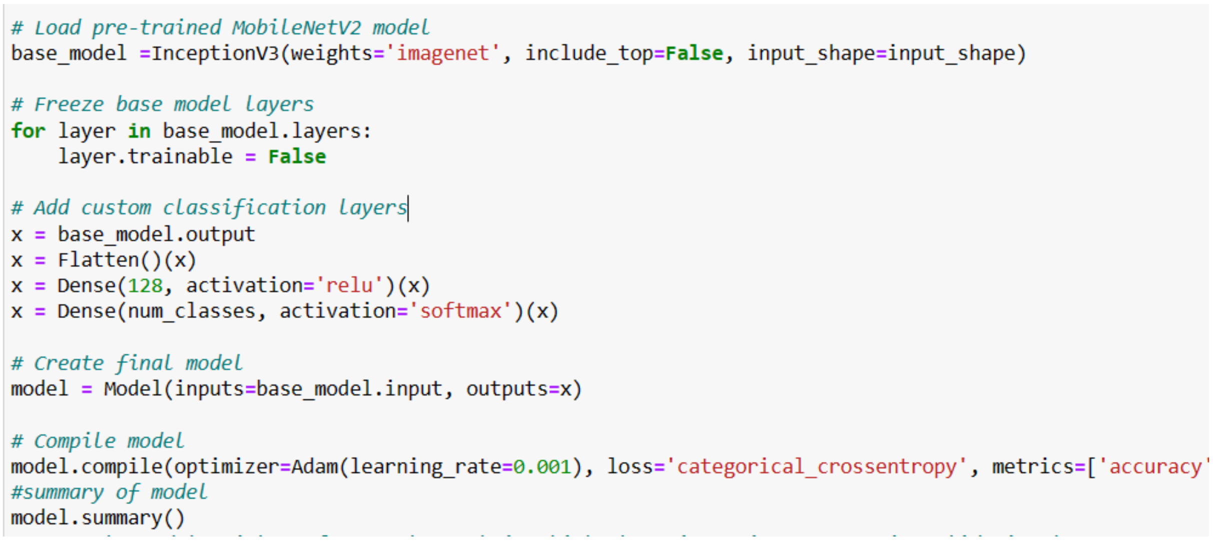
4.2.3 MODEL PREPARATION

The sample was divided into three parts: train, validate, and test, in the proportions 80:10:10. The disease that each image in the dataset is meant to depict has been labeled. I first prepared my dataset and then imported the relevant libraries and packages to build my model. I used transfer learning to retrain my base CNN model, which I imported. I then loaded the training, testing and validation data using the ImageDataGenerator class, which generates batches of augmented image data.

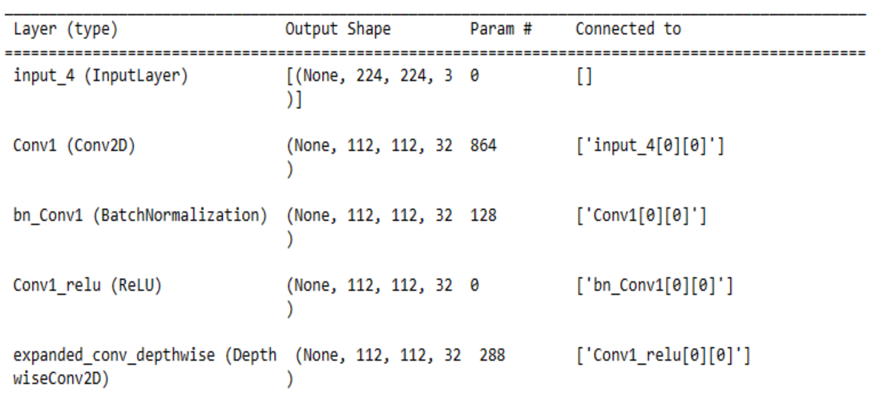


```
In [6]: import tensorflow as tf
        import itertools
        from tensorflow.keras.preprocessing.image import ImageDataGenerator
        from tensorflow.keras.applications.mobilenet_v2 import MobileNetV2
        from tensorflow.keras.layers import Dense, Flatten
        from tensorflow.keras.models import Model
        from tensorflow.keras.optimizers import Adam
        from keras.callbacks import ModelCheckpoint
        from sklearn.metrics import classification_report, confusion_matrix,
        import matplotlib.pyplot as plt
        import numpy as np
```

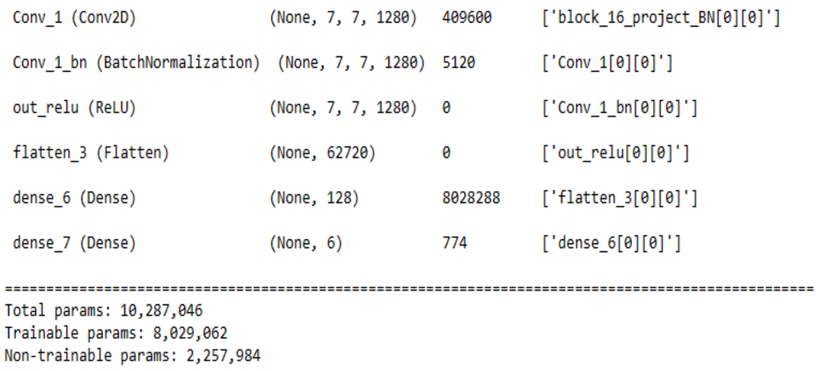
Figure 4: 1 Importing Libraries



```
# Load pre-trained MobileNetV2 model
base_model =InceptionV3(weights='imagenet', include_top=False, input_shape=input_shape)

# Freeze base model layers
for layer in base_model.layers:
    layer.trainable = False

# Add custom classification layers
x = base_model.output
x = Flatten()(x)
x = Dense(128, activation='relu')(x)
x = Dense(num_classes, activation='softmax')(x)

# Create final model
model = Model(inputs=base_model.input, outputs=x)

# Compile model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'
#summary of model
model.summary()
```



```
_____________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
=============================================================================================
 input_4 (InputLayer)           [(None, 224, 224, 3  0           []
                                )]

 Conv1 (Conv2D)                 (None, 112, 112, 32  864         ['input_4[0][0]']
                                )

 bn_Conv1 (BatchNormalization)  (None, 112, 112, 32  128         ['Conv1[0][0]']
                                )

 Conv1_relu (ReLU)              (None, 112, 112, 32  0           ['bn_Conv1[0][0]']
                                )

 expanded_conv_depthwise (Depth  (None, 112, 112, 32  288        ['Conv1_relu[0][0]']
 wiseConv2D)                    )
```



```
 Conv_1 (Conv2D)                (None, 7, 7, 1280)   409600      ['block_16_project_BN[0][0]']

 Conv_1_bn (BatchNormalization)  (None, 7, 7, 1280)  5120        ['Conv_1[0][0]']

 out_relu (ReLU)                (None, 7, 7, 1280)   0           ['Conv_1_bn[0][0]']

 flatten_3 (Flatten)            (None, 62720)        0           ['out_relu[0][0]']

 dense_6 (Dense)                (None, 128)          8028288     ['flatten_3[0][0]']

 dense_7 (Dense)                (None, 6)            774         ['dense_6[0][0]']

=============================================================================================
Total params: 10,287,046
Trainable params: 8,029,062
Non-trainable params: 2,257,984
```

Figure 4: 2 Model Preparation

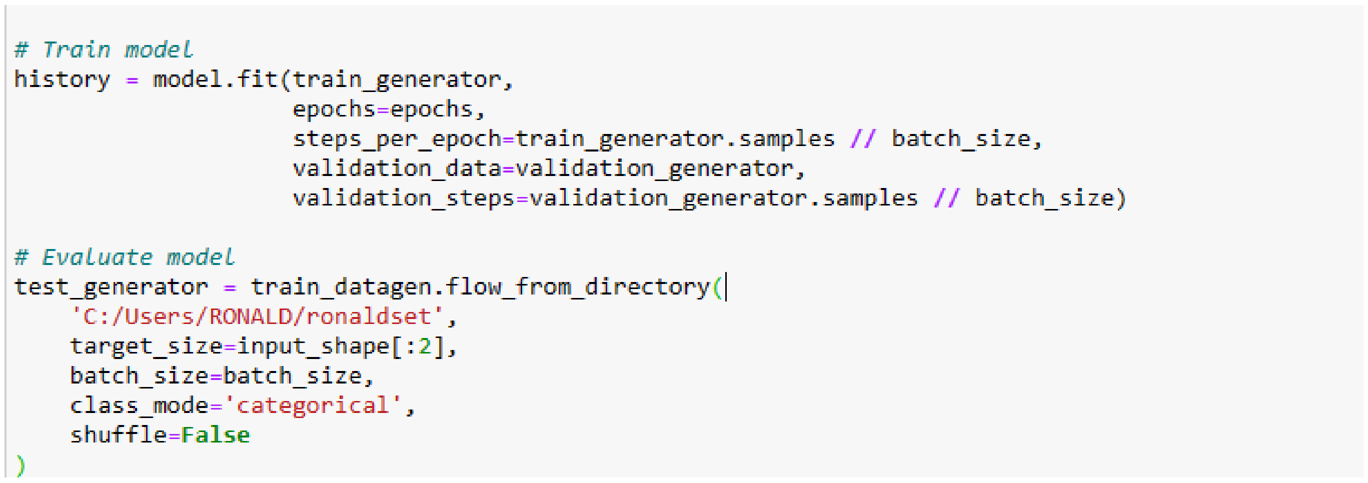
From the figure above, the base models selected were: MobileNet V2, InceptionV3 and ResNet50. These models were imported from the Keras Library and the base layers were frozen. I then added my custom classification layers which include: a thick layer, a flatten layer, and an output layer based on my dataset. I set the no of parameters to be trained and the ones not trained were set to non-trainable.

4.2.4 PARAMETER FINE-TUNING

The next stage was to fine tune the model utilizing the various parameters after constructing the model and the data that would be utilized for the training process. The model was fine-tuned with a learning rate of 0.0001. As soon as the weights are adjusted to take into account the projected. The learning rate is a parameter that controls the number of the steps used to optimize the gradient descent process. Adam is a popular deep learning optimization method that uses adaptive learning rates and a blend of momentum approaches to successfully adjust the network parameters during the course of training. Adam is relatively simple to configure, with the default settings taking care of the vast majority of issues.

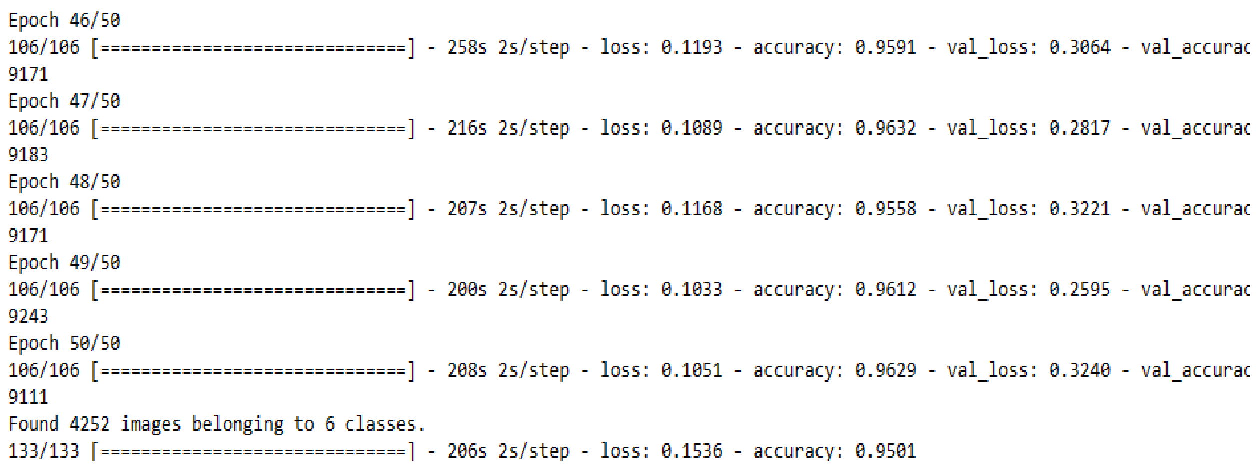
4.2.5 TRAINING METHOD

The next step was to train my model using the selected parameters after fine-tuning my parameters and establishing the necessary functions for my model's evaluation. To lessen overfitting and save the best version of my model after each epoch, other model checkpoints were also introduced. Epoch represents how many times the model would train using the given dataset. The validation dataset was used to track improvements and fine-tune the model to achieve better outcomes. The training dataset was utilized for training.



```
# Train model
history = model.fit(train_generator,
                    epochs=epochs,
                    steps_per_epoch=train_generator.samples // batch_size,
                    validation_data=validation_generator,
                    validation_steps=validation_generator.samples // batch_size)

# Evaluate model
test_generator = train_datagen.flow_from_directory(
    'C:/Users/RONALD/ronaldset',
    target_size=input_shape[:2],
    batch_size=batch_size,
    class_mode='categorical',
    shuffle=False
)
```

Figure 4: 3 Training Model



```
Epoch 46/50
106/106 [==============================] - 258s 2s/step - loss: 0.1193 - accuracy: 0.9591 - val_loss: 0.3064 - val_accurac
9171
Epoch 47/50
106/106 [==============================] - 216s 2s/step - loss: 0.1089 - accuracy: 0.9632 - val_loss: 0.2817 - val_accurac
9183
Epoch 48/50
106/106 [==============================] - 207s 2s/step - loss: 0.1168 - accuracy: 0.9558 - val_loss: 0.3221 - val_accurac
9171
Epoch 49/50
106/106 [==============================] - 200s 2s/step - loss: 0.1033 - accuracy: 0.9612 - val_loss: 0.2595 - val_accurac
9243
Epoch 50/50
106/106 [==============================] - 208s 2s/step - loss: 0.1051 - accuracy: 0.9629 - val_loss: 0.3240 - val_accurac
9111
Found 4252 images belonging to 6 classes.
133/133 [==============================] - 206s 2s/step - loss: 0.1536 - accuracy: 0.9501
```

Figure 4: 4 Training the Model over 50 epochs

4.3 RESULT AND TESTING

Each of the selected models was trained before the result was displayed and its efficacy was evaluated. The accuracy and overall performance of each model were evaluated using Classification Matrix. The graph of Training Accuracy against Validation Accuracy for MobileNetV2 was plotted to display the model training outcome. In order to demonstrate how the model performed while it was being trained, a graph was made that depicts the link between the training loss and the validation loss. A training accuracy of 96% and a training loss of 0.1 were realized after the model had been trained.

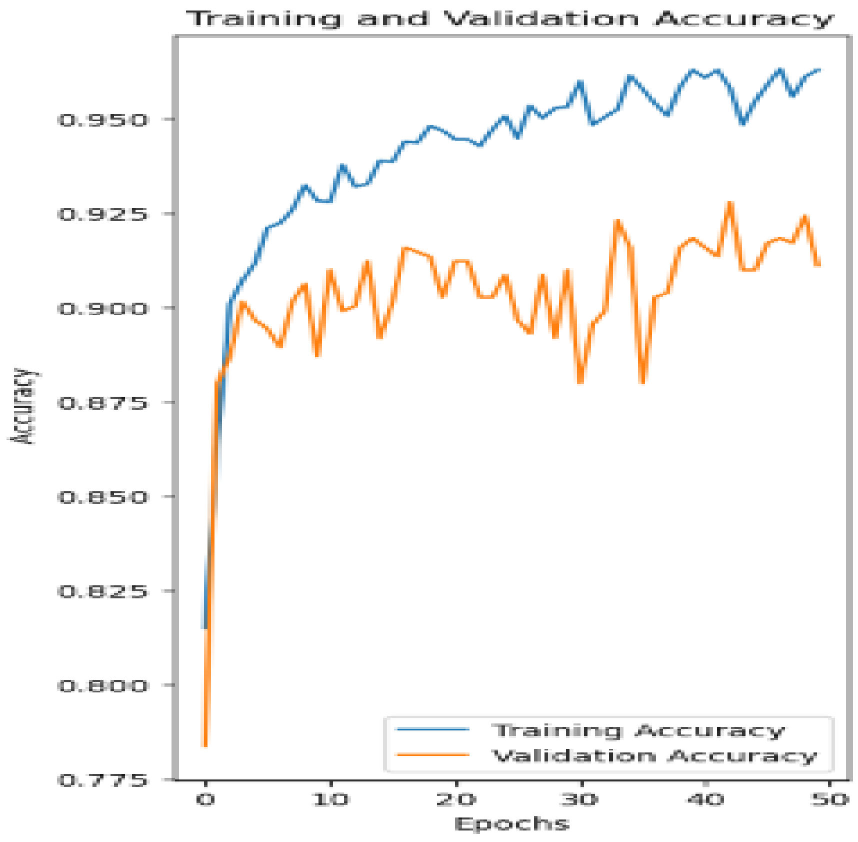


Figure 4: 5 Training Accuracy versus Validation Accuracy Graph for MobileNetV2

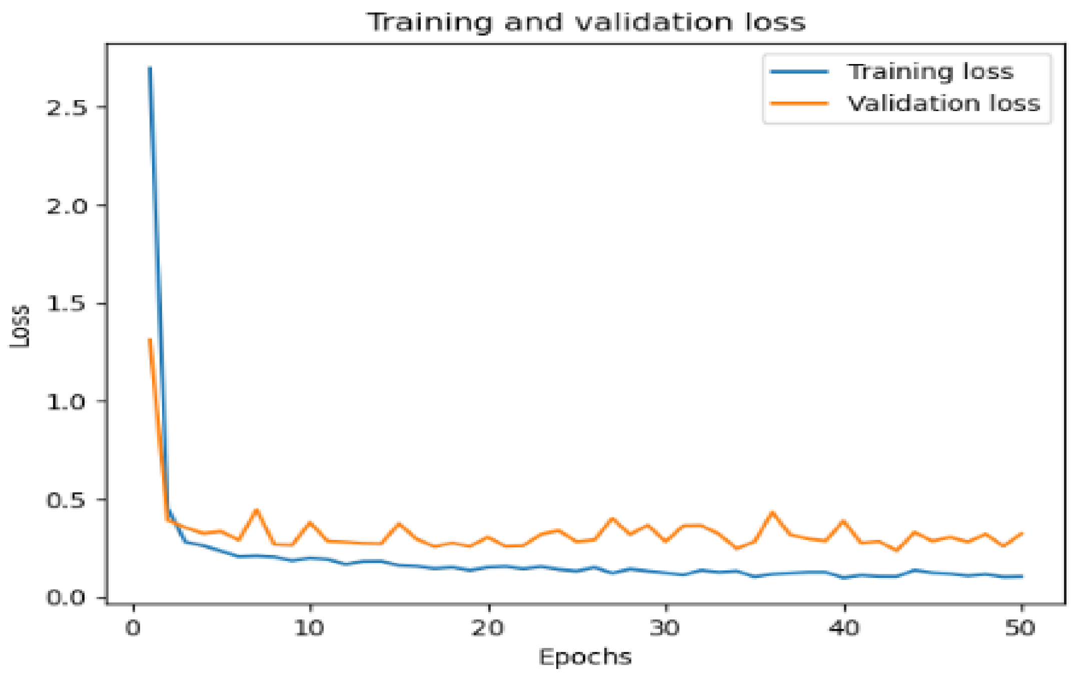
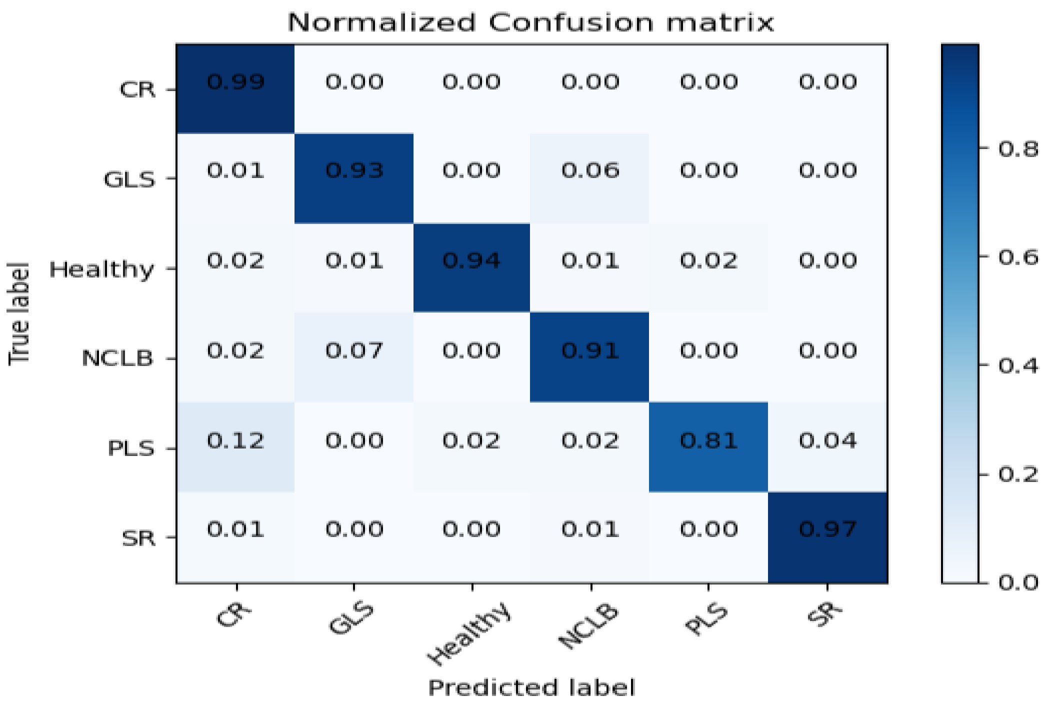


Figure 4: 6 Training Loss versus Validation Loss Graph for MobileNetV2



	CR	GLS	Healthy	NCLB	PLS	SR
CR	0.99	0.00	0.00	0.00	0.00	0.00
GLS	0.01	0.93	0.00	0.06	0.00	0.00
Healthy	0.02	0.01	0.94	0.01	0.02	0.00
NCLB	0.02	0.07	0.00	0.91	0.00	0.00
PLS	0.12	0.00	0.02	0.02	0.81	0.04
SR	0.01	0.00	0.00	0.01	0.00	0.97

Figure 4: 7 Confusion Matrix for MobileNetV2

When the model had been trained using the training dataset, its test dataset was used to evaluate its test accuracy and test loss. The test accuracy is determined as follows:

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{True Positive} + \text{True Negative} + \text{False Positive} + \text{False Negative}}$$

Equation 5: Test Accuracy

The test accuracy for MobileNetV2 when evaluated with the test dataset was 95% and the test loss when evaluated with the test dataset was 0.15.

The graph of Training Accuracy against Validation Accuracy for InceptionV3 was plotted to display the model training outcome. In order to demonstrate how the model performed while it was being trained, a graph was made that depicts the link between the training loss and the validation loss. A training accuracy of 91% and a training loss of 0.4 were realized after the model had been trained.

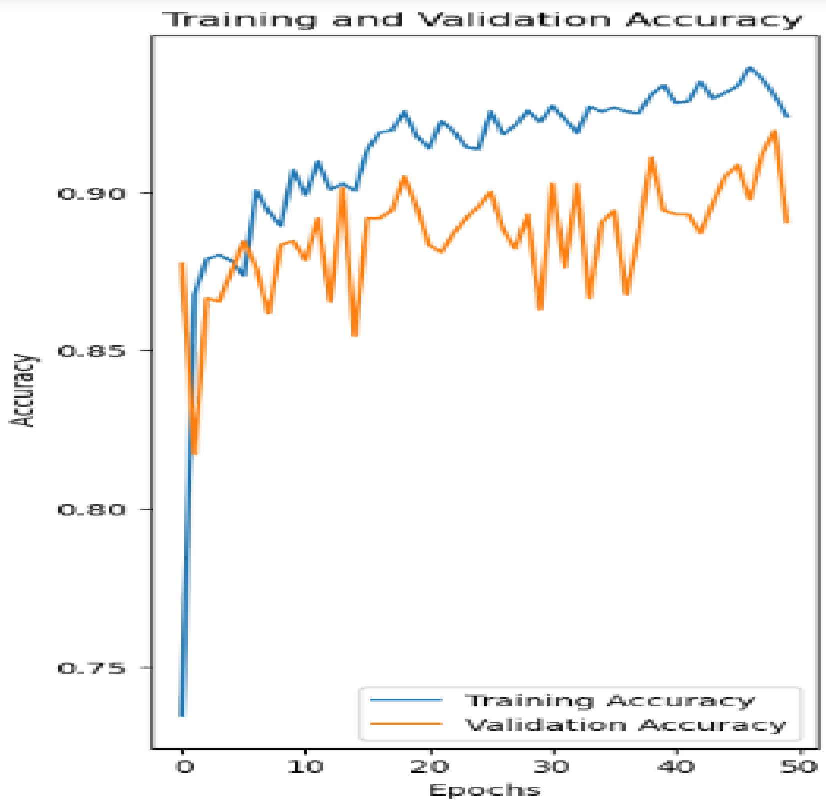


Figure 4: 8 Training Accuracy versus Validation Accuracy Graph for InceptionV3

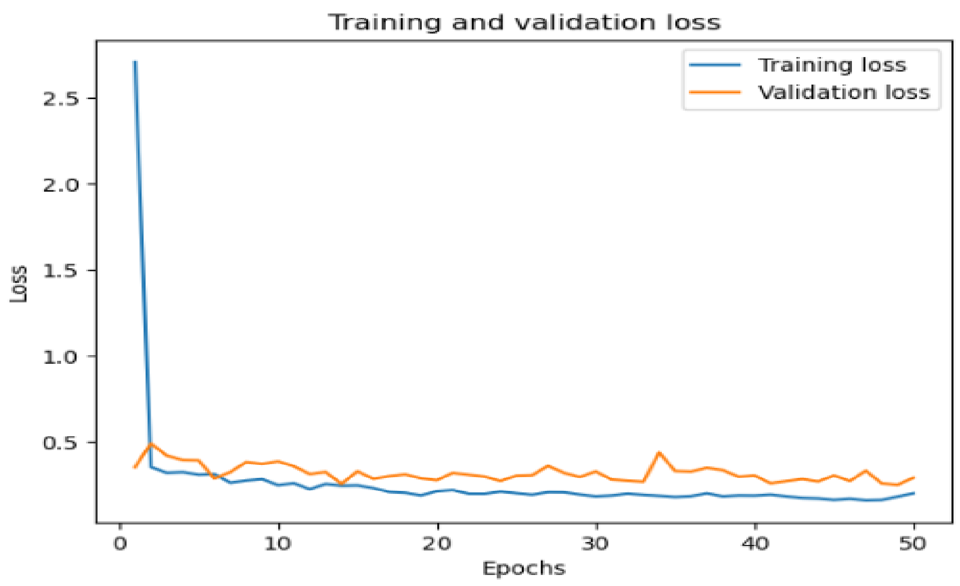
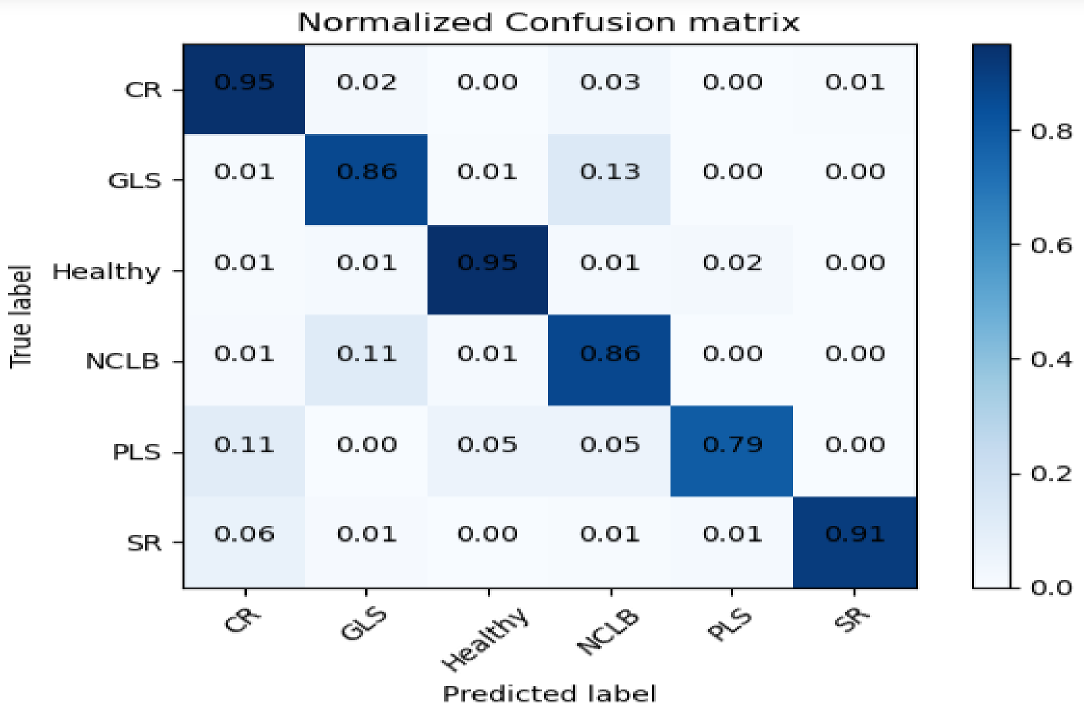


Figure 4: 9 Training Loss versus Validation Loss Graph for InceptionV3



True label \ Predicted label	CR	GLS	Healthy	NCLB	PLS	SR
CR	0.95	0.02	0.00	0.03	0.00	0.01
GLS	0.01	0.86	0.01	0.13	0.00	0.00
Healthy	0.01	0.01	0.95	0.01	0.02	0.00
NCLB	0.01	0.11	0.01	0.86	0.00	0.00
PLS	0.11	0.00	0.05	0.05	0.79	0.00
SR	0.06	0.01	0.00	0.01	0.01	0.91

Figure 4: 10 Confusion Matrix for InceptionV3

The graph of Training Accuracy against Validation Accuracy for ResNet50 was plotted to display the model training outcome. In order to demonstrate how the model performed while it was being trained, a graph was made that depicts the link between the training loss and the validation loss. A training accuracy of 77% and a training loss of 0.6 were realized after the model had been trained.

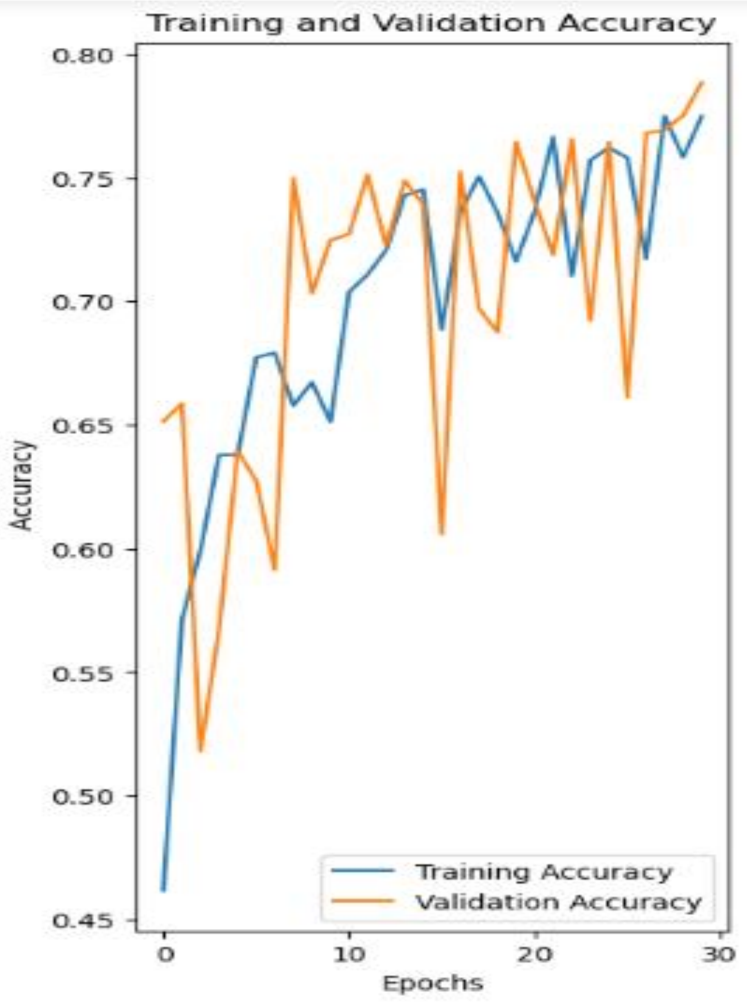


Figure 4: 11 Training Accuracy versus Validation Accuracy Graph for ResNet50

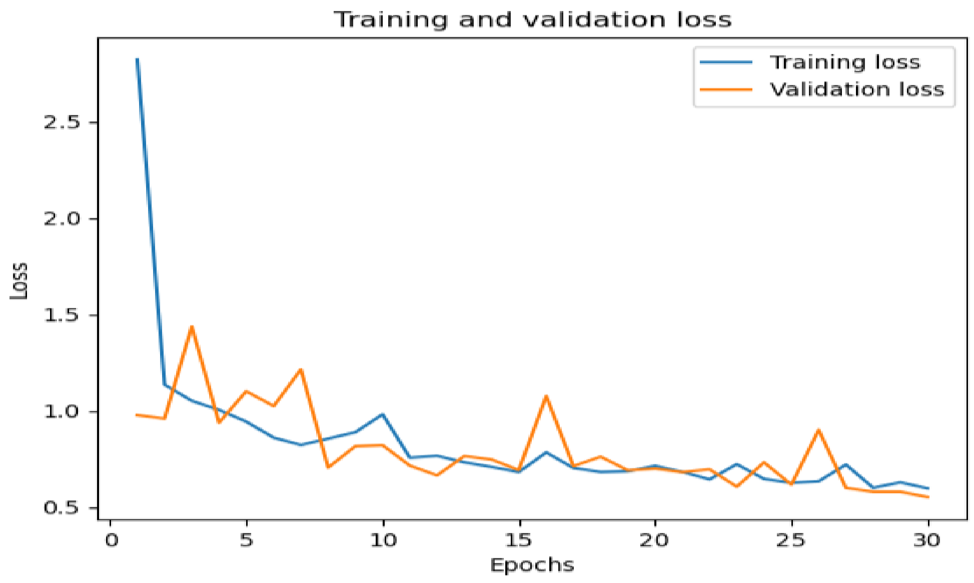


Figure 4: 12 Training Loss versus Validation Loss Graph for ResNet50

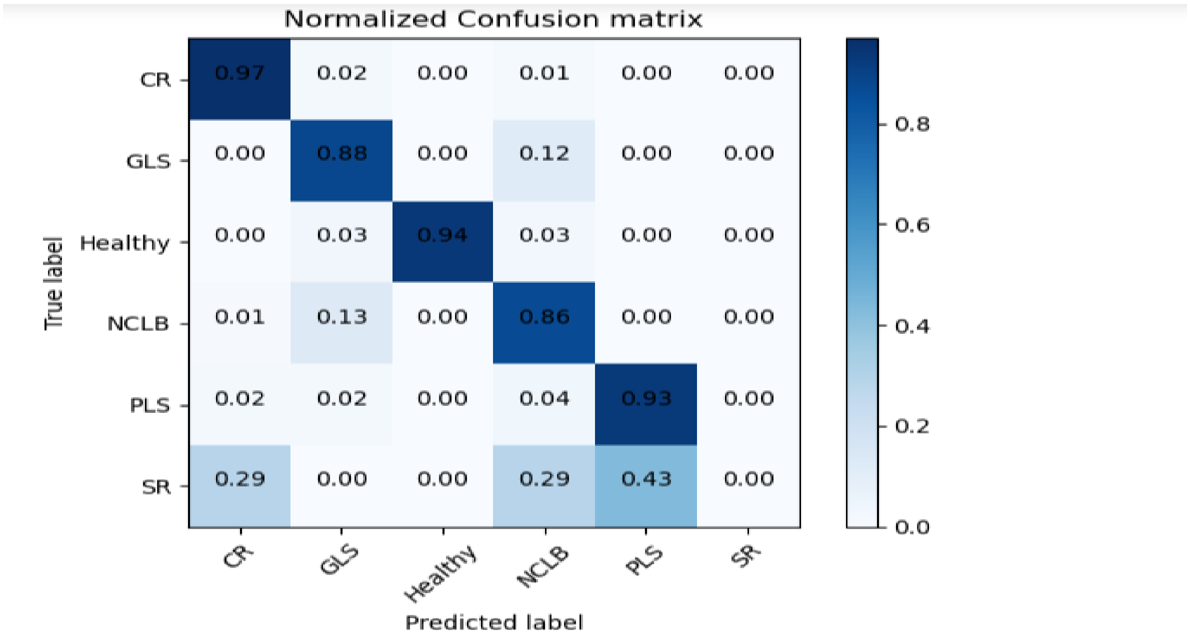


Figure 4: 13 Confusion Matrix for ResNet50

Table 3 shows the training accuracy, training loss, validation loss and validation accuracy for the several models that were chosen:

Table 4: 1 Comparison of CNN result

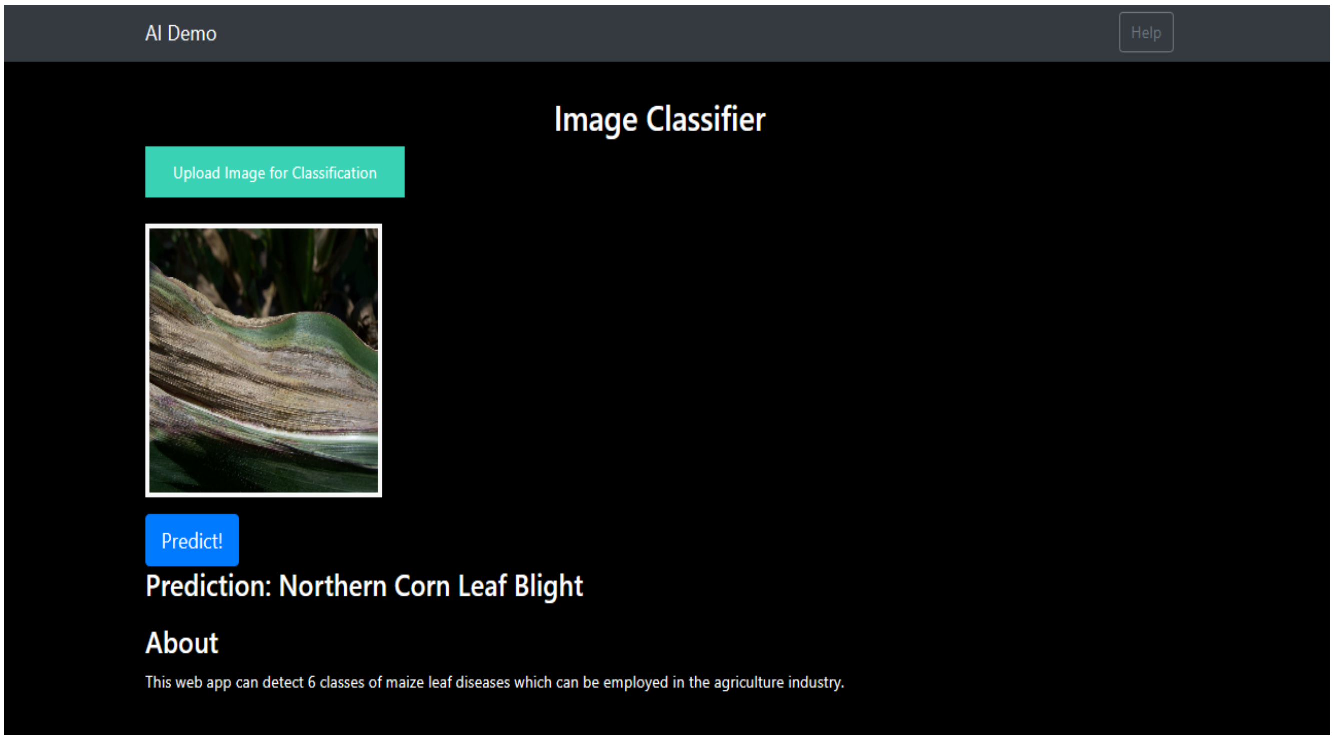
Model Name	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss	Test Accuracy	Test Loss
MobileNetV2	95%	0.15	90%	0.26	86%	0.14
InceptionV3	91%	0.20	84%	0.29	83%	0.18
ResNet50	77%	0.60	74%	0.55	72%	0.54

4.4 DISCUSSION

As can be seen from the table above, MobileNetV2 outperforms both InceptionV3 and ResNet50, making it the model of choice for deployment.

4.5 WEB APP DEPLOYMENT

The model had been established and the next step was to implement it. Visual Studio Code was used for this deployment. Using Visual Studio Code, an online server capable of classifying leaf diseases was constructed. Flask framework was used to develop my web application.



Prediction: Northern Corn Leaf Blight

About

This web app can detect 6 classes of maize leaf diseases which can be employed in the agriculture industry.

Figure 4: 14 Image of Maize Leaf Disease Being Classified with Web Application

The online app is very interactive and simple to use. All you have to do is upload the web app with the sick leaf picture. Once the image is submitted, the web app will scale it to the suitable input size before sending it to the model. The model would categorize the picture and deliver an output based on one of the six classes it had been trained with. The anticipated illness is subsequently presented on the screen to the user.

4.6 PERFORMANCE EVALUATION FOR WEB APPLICATION

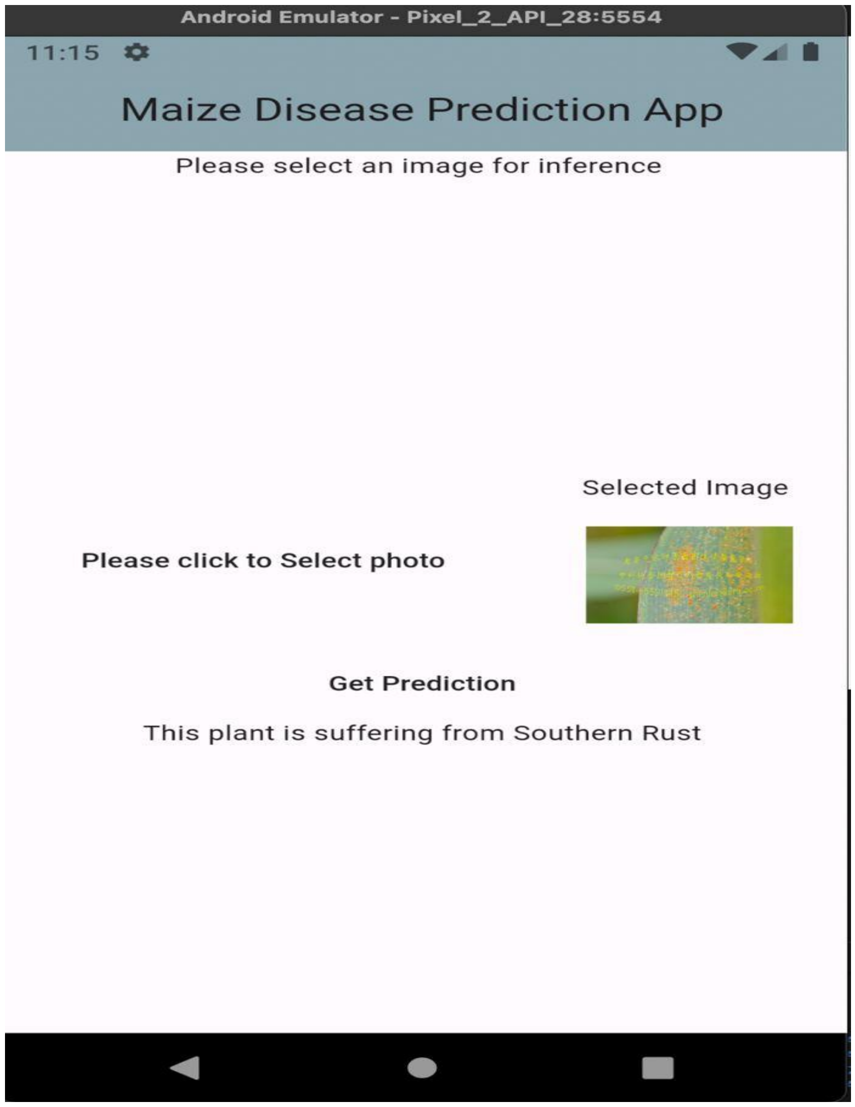
To assess the model's performance, 20 additional photos from each of the dataset's classes were utilized to test the web application and determine the accuracy. When the photos were transmitted for classification, the result was recorded as either true or false depending on whether the image was successfully classified. The result is represented in the table below:

Table 4: 2 Classification Performance of Web Application

Class Name	Total Number of True classifications	Total Number of False classifications	Accuracy
Common Rust	17	3	85%
Gray Leaf Spot	18	2	90%
Healthy	16	4	80%
Northern Corn Leaf Blight	18	2	90%
Phaeosphaeria Leaf Spot	15	5	75%
Southern Rust	16	4	80%

4.7 MOBILE APP DEPLOYMENT

The evolving area of mobile app development influences how we engage with technology. It blends user experience, design, and code to produce apps that are simple to use and effective. To create cutting-edge mobile solutions, developers use a variety of frameworks and tools across iOS and Android. In order to provide seamless user experiences in response to the rising demand for mobile apps, developers must remain current on the newest trends and technology. Mobile app development continues to redefine how we access information and interact with the digital world, whether for work or for fun. The Flutter mobile app framework was used to create the maize leaf disease detection app. Google's Flutter is an open-source framework for developing mobile apps. It has a number of benefits that make it a popular choice for mobile app development. Flutter's cross-platform development potential is one of its primary advantages. Developers may use Flutter to create code once and deliver it across numerous platforms, including iOS and Android. This eliminates the need to maintain distinct codebases for multiple platforms, saving time and effort during development. 20 additional photos from each of the dataset's classes were utilized to test the mobile application and determine the accuracy.


Android Emulator - Pixel_2_API_28:5554

11:15

Maize Disease Prediction App

Please select an image for inference

Selected Image

Please click to Select photo

Get Prediction

This plant is suffering from Southern Rust

Figure 4: 15 Image of Maize Leaf Disease Being Classified With Mobile Application

Table 4: 3 Classification Performance of Mobile Application

Class Name	Total Number of True classifications	Total Number of False classifications	Accuracy
Common Rust	18	2	90%
Gray Leaf Spot	18	2	90%
Healthy	19	1	95%
Northern Corn Leaf Blight	16	4	80%
Phaeosphaeria Leaf Spot	15	5	75%
Southern Rust	19	1	95%

4.8 CHAPTER SUMMARY

This chapter looked at my model's implementation procedure from the model-preparation stage through the training stage. After the model had been trained, its output was assessed to determine how well it would perform on a test dataset. Various metrics were used to evaluate the models and compare the models to which performed the best. The best performing model was chosen and deployed as a web application and mobile application.

4.9 CHAPTER CONCLUSION

The implementation method of my model was investigated in this chapter, from model development to training. After training, the model's output was tested to determine how well it would perform on the test dataset. Several criteria were used to evaluate the models and compare which models performed better. The best-performing model was chosen and implemented as a mobile application and a website.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATION

5.1 PREAMBLE

This chapter provides a summary of all of the work completed in earlier chapters of this report, as well as a review of contributions and accomplishments, as well as the problems experienced in this project. A few recommendations are also made for the project's future development.

5.2. ACHEIVEMENTS

A dataset pertaining to the most frequent maize ailments was obtained for this study. Following the acquisition of the dataset, it was pre-processed and enhanced to make it acceptable for the model that would be utilized. Following pre-processing, multiple models were chosen and re-trained using the transfer learning approach. After successfully training the models, the models were reviewed, and the top performing model was deployed as a web-based app and mobile app.

5.3 CHALLENGES ENCOUNTERED

Several difficulties were experienced during the project's development. They are as follows: choosing a network framework, debugging codes, categorizing a dataset, and overfitting a trained model.

5.4 CONCLUSION

In a nutshell this project was a big success, with all of the goals and objectives met.

5.5. RECOMMENDATIONS

To improve the project's progress, the model must be periodically updated with fresh data in order to react to changing illness patterns and increase classification accuracy.

Furthermore, researchers must keep up with advances in CNNs and image classification techniques in order to include the most recent innovations into your system.

REFERENCES

[1] Harold.M, Director General of Africa Rice,"Cereal Crops: Rice, Maize, Millet,Sorghum, Wheat,"vol.1, pp.2-7,2015

[2] Iken,J.E. and Amusa,N.A, "Maize research and production in Nigeria", African Journal of Biotechnology Vol.3(6), pp.302-307, June 2004.

[3]Tolulope Odetola and Chinonso Etumnu,"Contribution of Agriculture to Economic Growth in Nigeria", The 18th Annual Conference of the African Econometric Society (AES) Accra, Ghana at the session organized by the Association for the Advancement of African Women Economists (AAAWE), 22nd and 23rd July 2013.

[4] I. Gogul, V.Sathiesh Kumar,"Flower Species Recognition System using Convolution Neural Networks and Transfer Learning", 2017 4th International Conference on Signal Processing, Communications and Networking (ICSCN -2017), March 16 – 18, 2017, Chennai, INDIA .

[5] Jonas Teuwen and Nikita Moriakov,"Convolutional neural networks," 2020.

[6] Oriel Kiss," Bayesian Optimization for Machine Learning Algorithms in the Context of Higgs Searches at the CMS Experiment",2019.

[7]Malusi Sibiya and Mbuyu Sumbwanyambe,"A Computational Procedure for the Recognition and Classification of Maize Leaf Diseases Out of Healthy Leaves using Convolutional Neural Netwroks",2019.

[8]Bin Lui, Zefeng Ding, Liangliang Tian, Dongjian He, Shuqin Li and Hongyan Wang, "Grape Leaf Disease Identification using Improved Deep Convolutional Neural Networks", 2020.

[9]Erik Lucas da Rocha, Larissa Ferreira Rodrigues, Joao Fernando Mari,”Maize leaf disease classification using convolutional neural networks and hyperparameter optimization”,2020.

[10]Jing Chen, Qi Liu and Lingwang Gao,” Visual Tea Leaf Disease Recognition Using a Convolutional Neural Network Model”, 2019.

[11]Pooja Wadnere, Dr.P.L. Ramteke,”Detection and Classification of Plant Disease with Deep Learning”, Volume 10 Issue IV Apr 2022.

[12]Abdel-Aziz, Binguitcha-Fare and Prince Sharma,”Crops and weeds classification using Convolutional Neural Networks via optimization of transfer learning parameters”, Volume-8 Issue-5, June 2019.

[13] Ashraf Darwish, Dalia Ezzat and Aboul Ella Hassanien,” An Optimized Model based on Convolutional Neural Networks and Orthogonal Learning Particle Swarm Optimization Algorithm for Plant Diseases Diagnosis”,2019.

[14]Bin Liu, Yun Zhang, DongJian He and Yuxiang Li,”Identification of Apple Leaf Diseases Based on Deep Convolutional Neural Networks”,2017.

[15] Yang Lu, Shujuan Yi, Nianyin Zeng, Yurong Liu, Yong Zhang,”Identification of rice diseases using deep convolutional neural networks”, 2017.

[16]Lucas G. Nachtigall and Ricardo M.Araujo,”Classification of Apple Tree Disorders Using Convolutional Neural Networks”, IEEE 28th International Conference on Tools with Artificial Intelligence, 2016.

[17] Chrystler T. Orbien, Chris Jordan G.Aliac, Elmer A. Maravillas,” Identification of Carabao Mango Leaf Disease using Convolutional Neural Network”, Vol.12,01-Special Issue,2020.

[18] Hsing-Chung Chen, Agung Mulyo Widodo, Andika Wisnujati, Mosiur Rahaman, Jerry Chun-Wei Lin, Liukui Chen and Chien-Erh Weng," AlexNet Convolutional Neural Network for Disease Detection and Classification of Tomato Leaf", Electronics 2022.

[19]Shivani Machha, Nikita Jadhav, Himali Kasar, Prof. Sumita Chandak,"Crop Leaf Disease Diagnosis Using Convolutional Neural Network", Volume 7, Issue 02, Feb 2020.

[20]Ramar Ahila Priyadharshini, Selvaraj Arivazhagan, Madakannu Arun, Annamalai Mirnalini,"Maize leaf disease classification using deep convolutional neural networks",2019.

[21] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks", 2012.

[22]Gnanavel Sakkarvarthi, Godfrey Winster Sathianesan, Vetri Selvan Murugan, Avulapalli Jayaram Reddy, Prabhu Jayagopal and Mahmoud Elsisi,"Detection and Classification of Tomato Crop Disease Using Convolutional Neural Network",2022.

[23]Omneya Attallah, "Tomato Leaf Disease Classification via Compact Convolutional Neural Networks with Transfer Learning and Feature Selection".2023.

[24]Manya Afonso, Pieter M.Blok, Gerrit Polder, Jan M.van der Wolf, Jan Kamp,"Blackleg Detection in Potato Plants using Convolutional Neural Networks", IFAC PapersOnLine(6-11),2019.

[25] Umesh Kumar Lilhore, Agbotiname Lucky Imoize, Cheng-Chi Lee, Sarita Simaiya, Subhendu Kumar Pani, Nitin Goyal, Arun Kumar and Chun-Ta Li, "Enhanced Convolutional Neural Network Model for Cassava Leaf Disease Identification and Classification", 2022.

[26] Ananda S. Paymode, Vandana B.Malode,” Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG”, 2022.

[27]Vaibhav Tiwari, Rakesh Chandra Joshi, Malay Kishore Dutta, ”Dense convolutional neural networks based multiclass plant disease detection and classification using leaf images”, 2021.

[28]Zhen-tao Hu, Lin Zhou, Bing Jin, Hai-jiang Liu,” Applying Improved Convolutional Neural Network in Image Classification”, 2018.

[29] Solemane Coulibaly, Bernard Kamsu-Foguem, Dantouma Kamissoko, Daouda Traore,”Deep neural netwroks with transfer learning in millet crop images”,2019.

[30]Geetharamani G., Arun Pandian J. “Identification of plant leaf diseases using a nine-layer deep convolutional neural network”, 2019.

[31] Siti Zulaikha Muhammad Zaki, Mohd Asyraf Zulkifley, Marzuraikah Mohd Stofa, Nor Azwan, Mohammed Kamri, Nur Ayuni Mohamed, ”Classification of tomato leaf diseases using MobileNet V2”, Vol9, No.2,June 2020,pp.290-296.

[32] Peng Zhao, Chen Li, Md Mamunur Rahaman, Hao Xu, Hechen Yang, Hongzan Sun, Tao Jiang and Marcin Grzegorzek.”A Comparative Study of Deep Learning Classification Methods on a Small Environmental Microorganism Image Dataset (EMDS-6): from Convolutional Neural Networks to Visual Transformers, 2022.

[33]Khaled Alomar, Halil Ibrahim Aysel and Xiaohao Cai,” Data Augmentation in Classification and Segmentation: A Survey and New Strategies”, 2023.

[34] Mark Sandler,Andrew Howard, Menglong Zhu, Andrey Zhmoginov, Liang-Chiech Chen.”MobileNetV2: Inverted Residuals and Linear Bottlenecks, 2019.

[35] Andrew G.Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko,"MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications, 2017.

[36] Stefano Feraco, Angelo Bonfitto, Nicola Amati, Andrea Tonoli." Redundant Multi-Object Detection for Autonomous Vehicles in Structured Environments".2021.

[37] Karen Simonyan and Andrew Zisserman" Very Deep Convolutional Networks for Large-Scale Image Recognition", 2015.

[38]Qing Guan, Xiaochun Wan, Hongtao Lu, Bo Ping, Duanshu Li, Li Wang, Yongxue Zhu, Yunjun Wang, Jun Xiang" Deep convolutional neural network Inception-v3 model for differential diagnosing of lymph node in cytological images: a pilot study", 2019.

[39] Luqman Ali, Fady Alnajjar, Hamad Al Jassmi, Munkhjargal Gocho, Wasif Khan and M.Adel Serhani" Performance Evaluation of Deep CNN-Based Crack Detection and Localization Techniques for Concrete Structures", Sensors 2021.

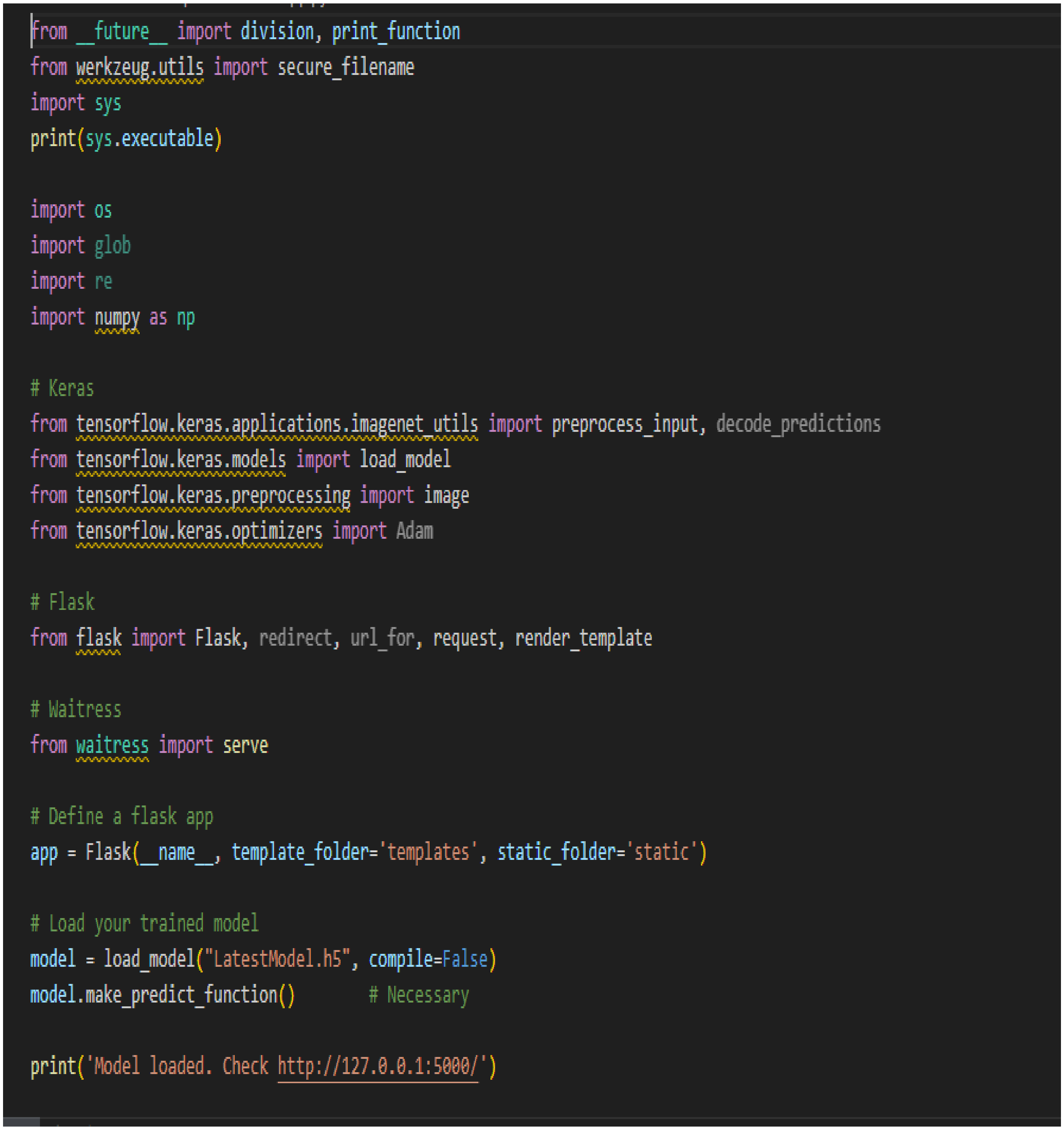
[40] Mohamed Arbane, Rachid Benlamri, Youcef Brik and Mohamed Djerioui "Transfer Learning for Automatic Brain Tumor Classification Using MRI Images". 2020.

[41] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun,"Deep Residual Learning for Image Recognition", 2015.

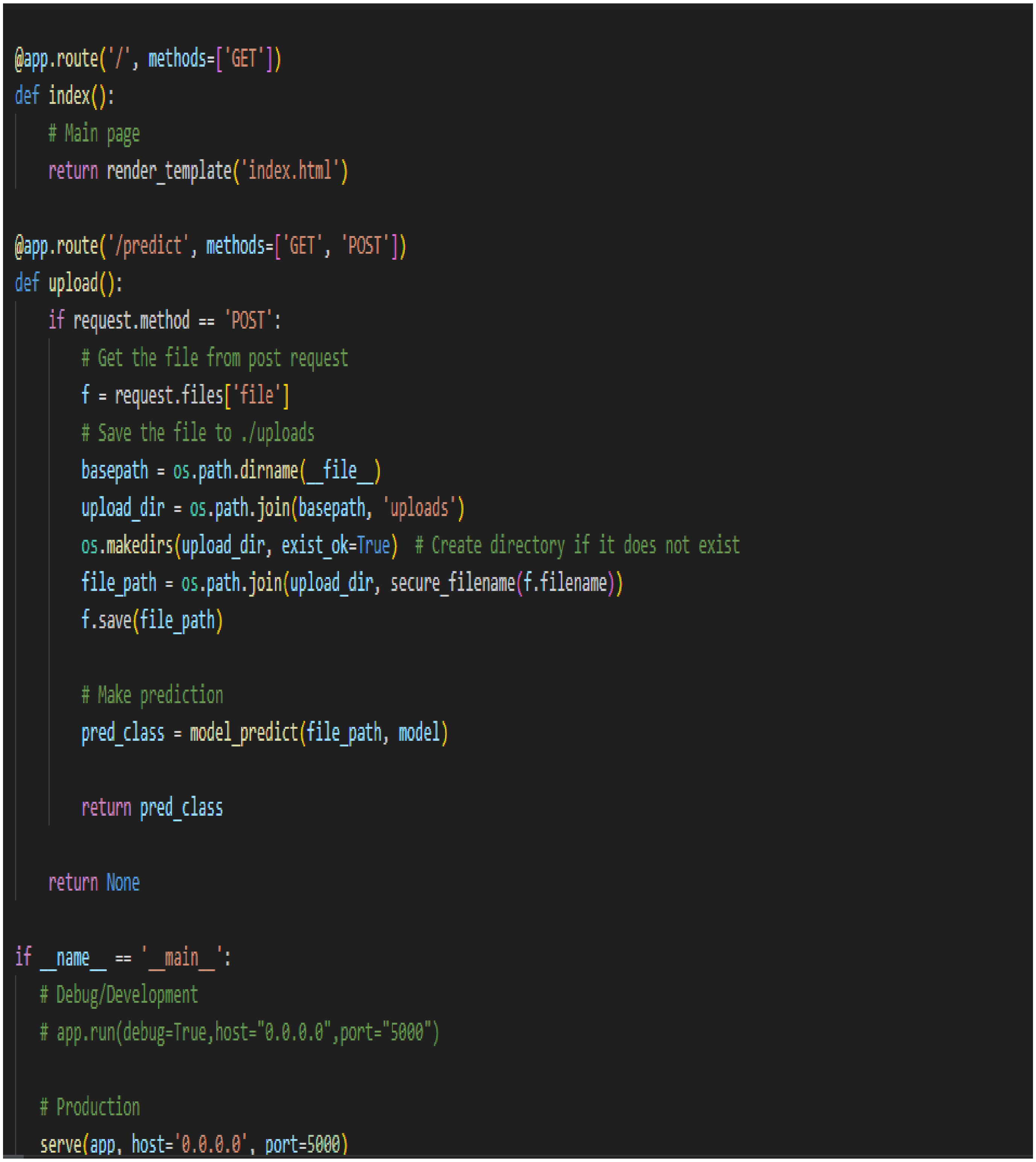
[42] FuTao Ni, Jian Zhang, ZhiQiang Chen,"Pixel-level crack delineation in images with convolutional feature fusion", 2018.

[43] Ricki Chandra Hidayatullah, Sriyani Violina," Convolutional Neural Network Architecture and Data Augmentation for Pneumonia Classification from Chest X-Rays Images", Volume5, Issue 2, February-2020.

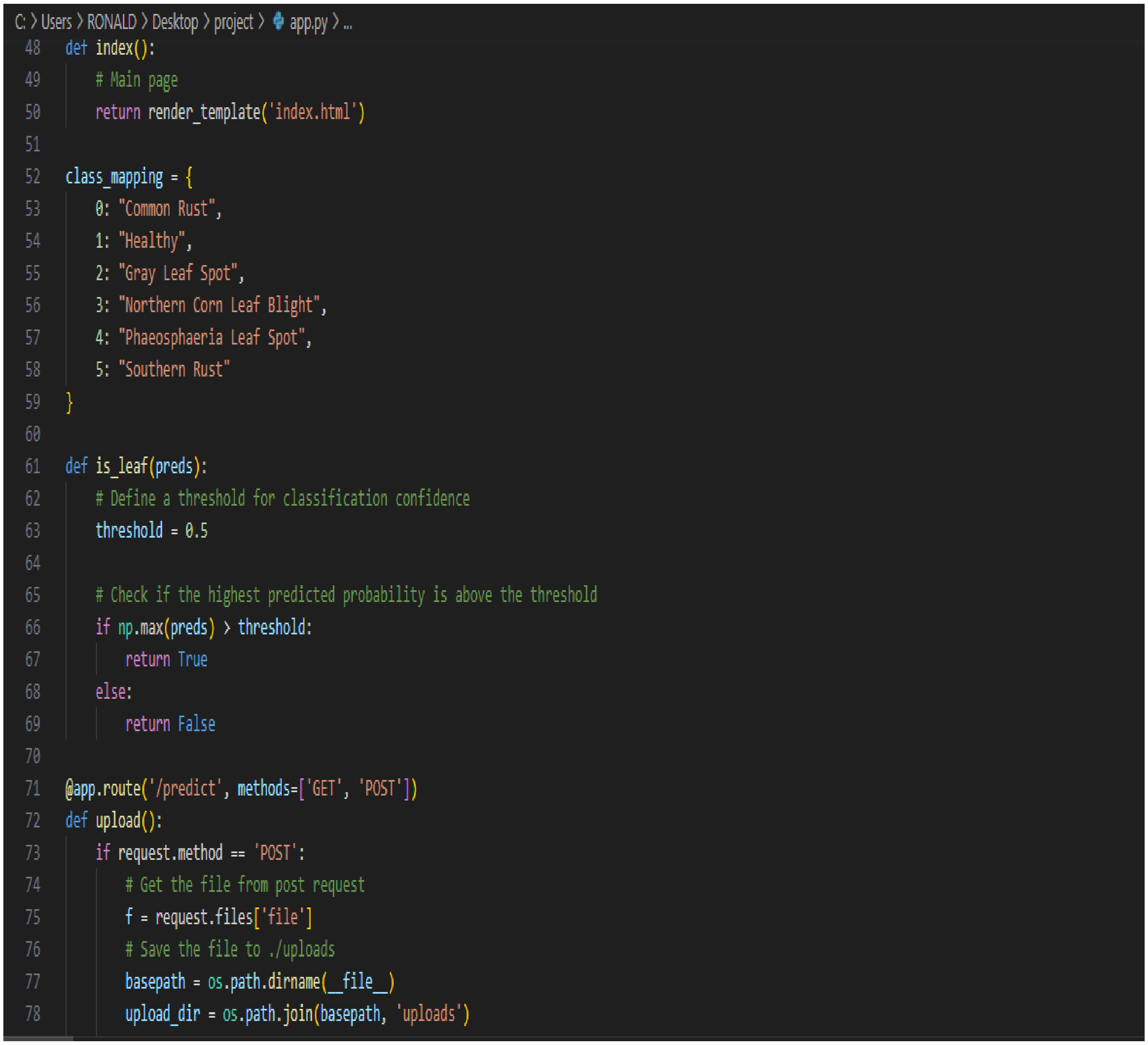
APPENDIX



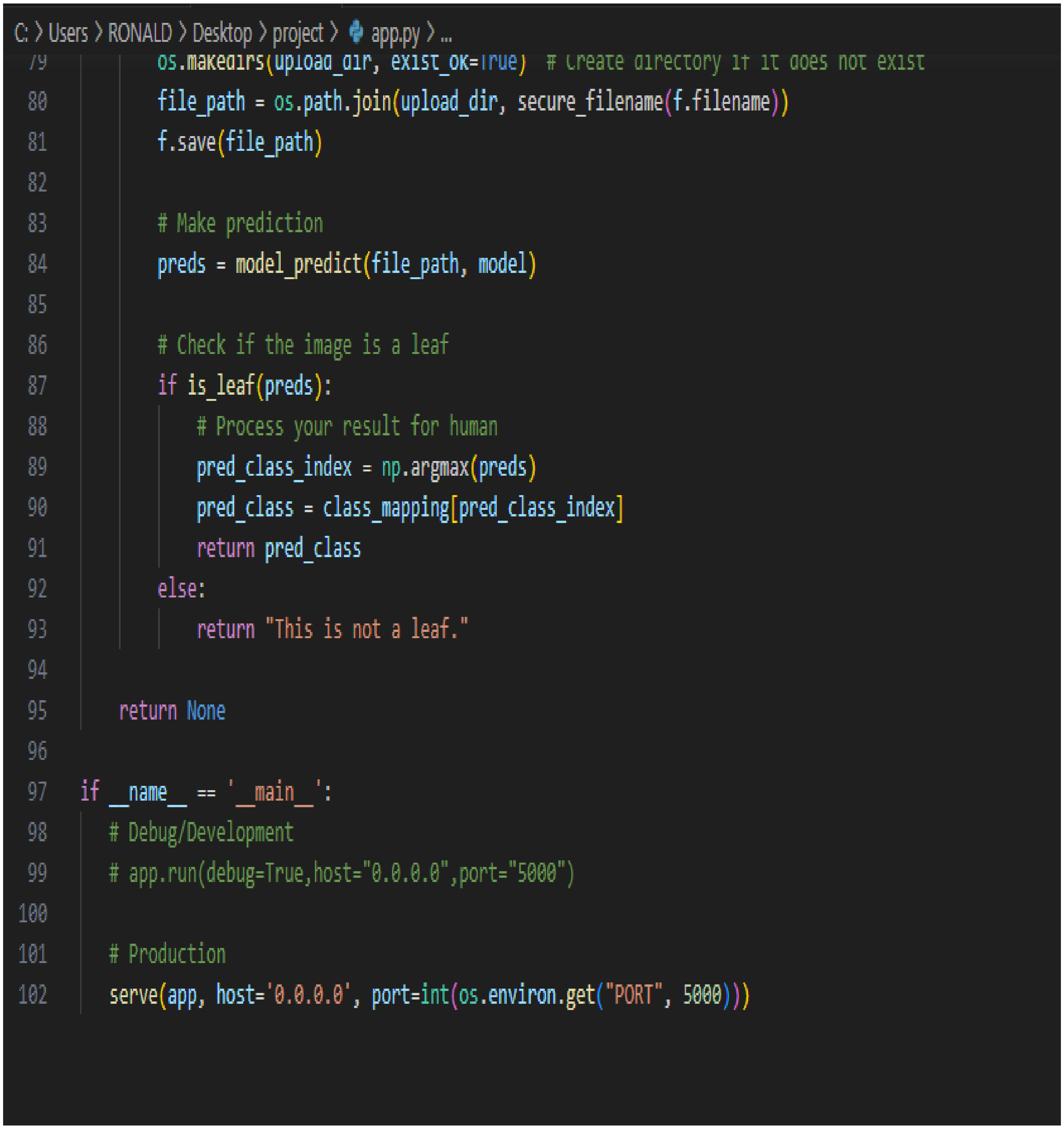
```
from __future__ import division, print_function
from werkzeug.utils import secure_filename
import sys
print(sys.executable)

import os
import glob
import re
import numpy as np

# Keras
from tensorflow.keras.applications.imagenet_utils import preprocess_input, decode_predictions
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
from tensorflow.keras.optimizers import Adam

# Flask
from flask import Flask, redirect, url_for, request, render_template

# Waitress
from waitress import serve

# Define a flask app
app = Flask(__name__, template_folder='templates', static_folder='static')

# Load your trained model
model = load_model("LatestModel.h5", compile=False)
model.make_predict_function()        # Necessary

print('Model loaded. Check http://127.0.0.1:5000/')
```



```
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        upload_dir = os.path.join(basepath, 'uploads')
        os.makedirs(upload_dir, exist_ok=True)  # Create directory if it does not exist
        file_path = os.path.join(upload_dir, secure_filename(f.filename))
        f.save(file_path)

        # Make prediction
        pred_class = model_predict(file_path, model)

        return pred_class

    return None

if __name__ == '__main__':
   # Debug/Development
   # app.run(debug=True,host="0.0.0.0",port="5000")

   # Production
   serve(app, host='0.0.0.0', port=5000)
```



```
C: > Users > RONALD > Desktop > project > app.py > ...
def index():
    # Main page
    return render_template('index.html')

class_mapping = {
    0: "Common Rust",
    1: "Healthy",
    2: "Gray Leaf Spot",
    3: "Northern Corn Leaf Blight",
    4: "Phaeosphaeria Leaf Spot",
    5: "Southern Rust"
}

def is_leaf(preds):
    # Define a threshold for classification confidence
    threshold = 0.5

    # Check if the highest predicted probability is above the threshold
    if np.max(preds) > threshold:
        return True
    else:
        return False

@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        upload_dir = os.path.join(basepath, 'uploads')
```



C: > Users > RONALD > Desktop > project > app.py > ...

```python
            os.makedirs(upload_dir, exist_ok=True)  # Create directory if it does not exist
            file_path = os.path.join(upload_dir, secure_filename(f.filename))
            f.save(file_path)

            # Make prediction
            preds = model_predict(file_path, model)

            # Check if the image is a leaf
            if is_leaf(preds):
                # Process your result for human
                pred_class_index = np.argmax(preds)
                pred_class = class_mapping[pred_class_index]
                return pred_class
            else:
                return "This is not a leaf."

        return None

if __name__ == '__main__':
    # Debug/Development
    # app.run(debug=True,host="0.0.0.0",port="5000")

    # Production
    serve(app, host='0.0.0.0', port=int(os.environ.get("PORT", 5000)))
```