INSTITUTE FOR ADVANCED COMPUTING AND SOFTWARE DEVELOPMENT, AKURDI, PUNE

Plant Disease Detection using Convolution Neural Network

PG-DBDA March 2024

Submitted By:

Group No: G-09

Roll No.	**Name:**
243547	Shagufta Nadaf
243549	Shruti Patil

Mr. Abhijit Nagargoje
Project Guide

Mr. Rohit Puranik
Centre Coordinator

ABSTRACT

The agricultural sector faces significant challenges due to plant diseases, which can severely impact crop yields and food security. There is a pressing need for efficient, accurate, and scalable solutions for early disease detection.

In this project, Convolutional Neural Networks (CNNs) were employed to develop a robust system for detecting and diagnosing plant diseases using images of leaves. The project utilized an extensive open database of 19,721 images representing 15 different [plant, disease] classes, including healthy plants. Various CNN architectures were tested, and the most successful model achieved a 94.8% accuracy rate in identifying the correct [plant, disease] combination or confirming the plant's health.

The applications of this model are extensive. It can be integrated into mobile applications, providing farmers with a tool for instant disease diagnosis and management. The model can also be embedded in automated crop monitoring systems, offering real-time, large-scale disease detection. Moreover, it serves as a valuable component in smart farming systems, contributing to precision agriculture by enabling proactive and informed decision-making. These applications underscore the model's potential to significantly enhance crop management practices and mitigate the impact of plant diseases on agricultural productivity.

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to everyone who has supported me throughout the course of this project. First and foremost, I would like to thank my project guide Mr.Abhijit Nagargoje and Centre Coordinator Mr.Rohit Puranik, for their invaluable guidance, insightful feedback, and constant encouragement. Their expertise and knowledge have been instrumental in the successful completion of this project.

I am also deeply grateful to IACSD Pune for providing me with the resources and platform to conduct this research. My special thanks go to the faculty members and technical staff of the DBDA for their assistance and support.

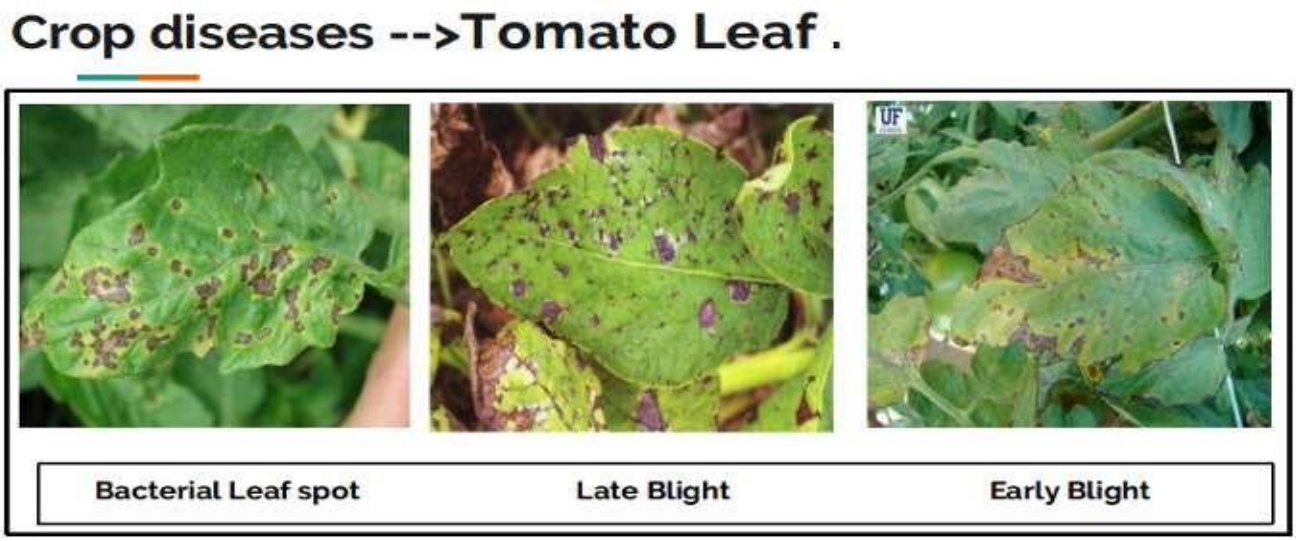
Shagufta Nadaf (240341225046)

Shruti Patil (240341225047)

Table of Contents

1. INTRODUCTION

Since the past days and in the present too, farmers usually detect the crop diseases with their naked eye which makes them take tough decisions on which fertilisers to use. It requires detailed knowledge the types of diseases and lot of experience needed to make sure the actual disease detection. Some of the diseases look almost similar to farmers often leaves them confused. Look at the below image for more understanding.



They look the same and almost similar. In case the farmer makes wrong predictions and uses the wrong fertilizers or more than the normal dose (or) threshold or Limit (every plant has some threshold fertilizers spraying to be followed), it will mess up the whole plant (or) soil and cause enough damage to plant and fields.

To prevent this situation need better and perfect guidance on which fertilizers to use, to make the correct identification of diseases, and the ability to distinguish between two or more similar types of diseases in visuals. This is where Convolution Neural Networks comes handy.

1.1 Purpose

Detection of plant disease through some automatic technique is beneficial as it reduces a large work of monitoring in big farms of crops, and at very early stage itself it detects the symptoms of diseases i.e. when they appear on plant leaves.

Project deals with the real time detection of diseases that affect the plant and the area affected using Convolutional neural network (CNN) Model.

Convolutional neural network models were developed to perform plant disease detection and diagnosis using simple leaves images of healthy and diseased plants, through deep learning methodologies. So that appropriate fertilizers can be used to prevent further damage to plants from pathogenic viruses.

1.2 Scope of the project

Plant diseases cause a major production and economic losses in the agricultural industry. The disease management is a challenging task. Usually the diseases or its symptoms such as coloured spots or streaks are seen on the leaves of a plant. In plants most of the leaf diseases are caused by fungi, bacteria, and viruses. The diseases caused due to these organisms are characterized by different visual symptoms that could be observed in the leaves or stem of a plant. Usually, these symptoms are detected manually.

With the help of CNN, Automatic detection of various diseases can be detected with the help of CNN. CNN plays a crucial role in the detection of plant diseases since it provides best results and reduces the human efforts. The CNN could be used in the field of agriculture for several applications. It includes detection of diseased leaf, stem or fruit, to measure the affected area by disease, to determine the colour of the affected area. Tomato cultivation is one of the most remunerative farming enterprises in India. The naked eye observation by the experts is approach usually taken in identification and detection of plants. This approach is time consuming in huge farms or land areas. The use of CNN techniques in detection and identification of Different plant diseases in the earlier stages and thereby the quality of the product could be increased. These systems monitor the plant such as leaves and stem and any variation observed from its characteristic features, variation will be automatically identified and also will be informed to the user.

2.Software Life Cycle Model

In order to make this Project we are going to use Classic LIFE CYCLE MODEL. Classic life cycle model is also known as WATER FALL MODEL. The life cycle model demands a Systematic sequential approach to software development that begins at the system level and progress through analysis design coding, testing and maintenance.

Fig. Software Development Cycle

The Classic Life Cycle Model

The waterfall model is sequential software development process, in which progress is seen as flowing steadily downwards (like a waterfall) through the phases of conception initiation, Analysis, Design (validation), Implementation. Testing and maintained

The SDLC Life Cycle, or just the SDLC as it's known, is the process of developing software to meet a need or solve a problem

3.Overall Description

Data:

Name	No of Classes	Class Names
Bell Pepper	02	Pepper bell Bacterial spot Pepper bell healthy
Potato	03	Potato Early blight Potato Late blight Potato healthy
Tomato	10	Bacterial spot Early blight Late blight Leaf Mold Septoria leaf spot Spider mites Target Spot Yellow Leaf Curl Virus Mosaic virus Healthy

Imports:

- **Matplotlib:**

  Matplotlib is a collection of command style functions that make matplotlib work like MATLAB. Each pyplot function makes some change to a figure. We have used it to show visualizations of analysis.

- **Numpy:**

  Numpy is used to for mathematical operations. This package provides easy use of mathematical function.

- **TensorFlow:**

  TensorFlow is open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks.

- **Keras:**

  Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, R, Theano, and PlaidML.

4.Requirement Specification

Initial nonfunctional requirement will be:

- Getting the large datasets which can provide developer enough data to train the model.
- Maintain the minimum variance and bias so the model is successfully work.
- Avoid the underfitting and overfitting.

Initial functional requirement will be:

- Selecting the appropriate algorithms.
- Determining the appropriate input format to algorithm.
- Train the model.
- Test the model.

Hardware Requirement:

- Processor: Intel Dual Core and above
- RAM: Minimum 4GB
- OS: Windows, Linux

Software Requirement:

- Anaconda Navigator
- Jupyter NoteBook / Visual Studio Code

5.Plan of Project Execution

Feasible study phase

- Study of different distributed algorithms.
- Different types of plant images are studied and corresponding. After detail study, labelling in done by segregating the images and with different diseases

Requirement analysis

- The database is pre-processed such as Image reshaping, resizing and conversion to an array form.
- There is a huge database so basically the images with better resolution and angle are selected. After selection of images we should have deep knowledge about the different leaves and the disease they have.

Design phase

- Designing algorithm and graphs for getting more accuracy and satisfaction.
- Adjusting the Images in categories so that our model identifies the diseases.

Implementation phase

- CNN has different layers that are Dense, Dropout, Activation, Flatten, Convolution2D and MaxPooling2D these all will implement.

Testing phase

- After the model is trained successfully the software can identify the disease if theplant species is contained in the database.
- After successful training and pre-processing, comparison of the test image and trained model takes place to predict the disease.

Deployment phase

- After follow all above steps then deploy the model on the Internet so that our Famer's can use it for more production in farming growth.

6.System Design

Flowchart of the System:

- The flowchart of the algorithm is represented in Figure

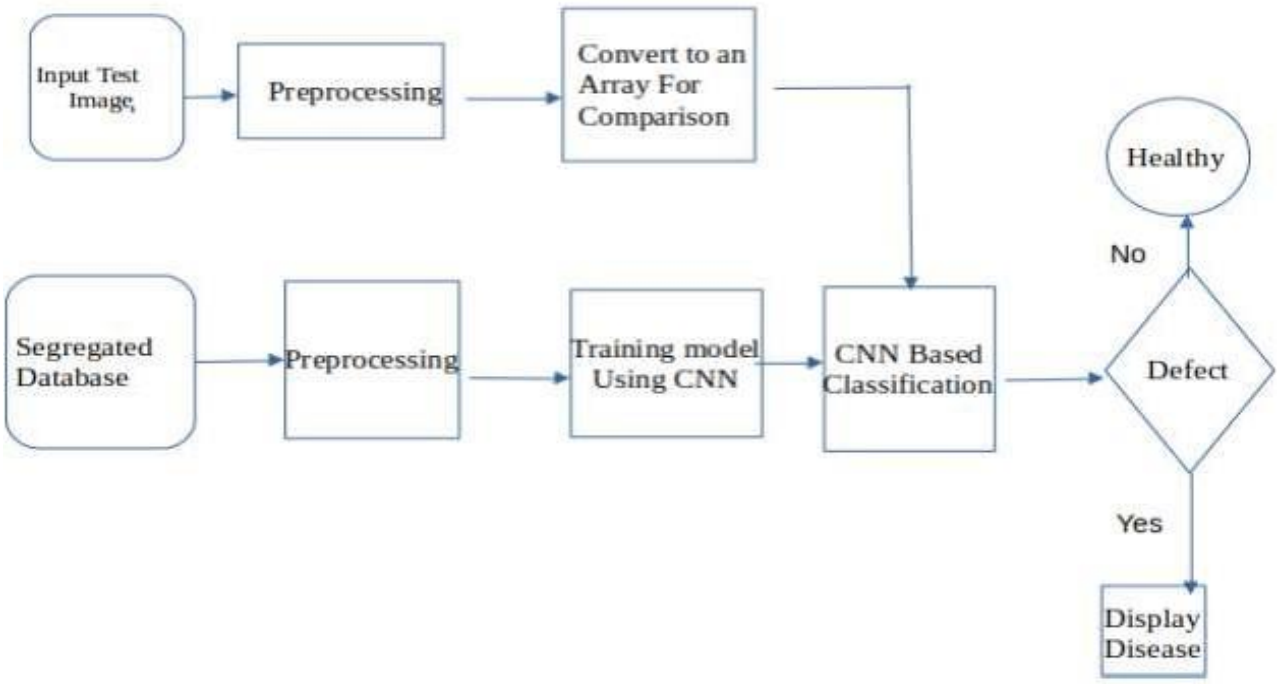


Fig. Flowchart of the System

The above flowchart describes the working flow of the project. First pre-processed the data and selected the model. Then algorithm was later implemented in python and deployed on web. The model was later tested against raw images.

7.Data Pre-processing

- The dataset is divided into 80% for training and 20% for validation.
- First, augmentation settings are applied to the training data.
- set height and width of input image.
- The settings applied include flipping (horizontal), shearing of range (0.2) and zoom (0.2). Rescaling image values between (0 –1) called normalization. All these parameters are stored in the variable “train_datagen” and “test_datagen”.

```
[ ] # Preprocessing data. Data Augmentation
    train_datagen= ImageDataGenerator(rescale=1/255,
                                      shear_range=0.2,
                                      zoom_range=0.2,
                                      validation_split=0.2, # validation split 20%.
                                      horizontal_flip=True)
    test_datagen=ImageDataGenerator(rescale=1/255)
```

8.Model Building

Algorithm Research and Selection:

For this classification problem, following deep learning model was used:

Convolution Neural Network

It is well known for its widely used in applications of image and video recognition and also in recommender systems and Natural Language Processing (NLP). However, **convolutional** is **more** efficient because it reduces the number of parameters which makes different from other deep learning models.

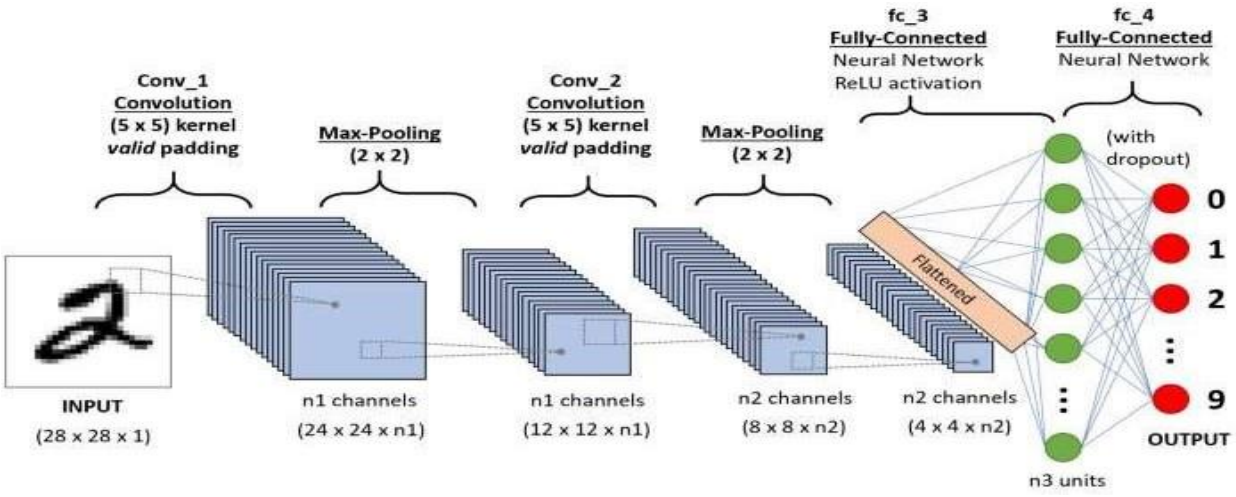


Fig. Convolutional Neural Network

Main Steps to build a CNN (or) Conv. net:

1. Convolution Operation+
2. ReLU Layer (Rectified Linear Unit)
3. Pooling Layer (Max Pooling)
4. Flattening
5. Fully Connected Layer

1. **Convolution Operation:**

   Convolution is the first layer to extract features from the input image and it learns the relationship between features using kernel or filters with input images.

2. **ReLU Layer:**

   ReLU stands for the Rectified Linear Unit for a non-linear operation. The output is $f(x) = max(0,x)$. we use this because to introduce the non-linearity to CNN.

3. **Pooling Layer:**

   It is used to reduce the number of parameters by down sampling and retain only the valuable information to process further. There are types of Pooling:

- Max Pooling.
- Average and Sum pooling.

4. **Flattening:**

   we flatten our entire matrix into a vector like a vertical one. So, that it will be passed to the input layer.

5. **Fully Connected Layer:**

   we pass our flatten vector into input Layer . we combined these features to create a model. Finally, we have an activation function such as softmax or sigmoid to classify the outputs.

Short information about Activation Functions:

These functions are needed to introduce a non-linearity into the network. Activation function is applied and that output is passed to the next layer.

α. **Sigmoid:** The Sigmoid function used for **binary classification** in logistic regression model. While creating artificial neurons sigmoid function used as the **activation function**. In statistics, the **sigmoid function graphs** are common as a cumulative distribution function.

β. **Hyperbolic Tangent:** In neural networks, as an alternative to **sigmoid function**, **hyperbolic tangent function** could be **used** as **activation function**. When you backpropage, derivative of **activation function** would be involved in calculation for error effects on weights.

χ. **ReLU :** ReLU is the max function(x,0) with input x e.g. matrix from a convolved image. ReLU then sets all negative values in the matrix x to zero and all other values are kept constant.

δ. **Softmax :** Used in multiple classification logistic regression model. In building neural networks softmax functions used in different layer level.

9.Algorithm Implementation

To implement the idea of disease detection and to train the machine accordingly requires lot of steps which are mentioned below: -

1. Label Data for input like Training Data, Testing Data and Valid Data in different folders.
2. Import all libraries like NumPy, Pandas, Matplotlib, Tensorflow, Convolution2D, MaxPooling2D, Flatten, Dense, Sequential Model, ImageDataGenerator, image, random etc.
3. Assign paths of the folders where training and testing data are available.
4. Assign Dictionary to write all the diseases name in a sequential way.
5. Add Convolutional Layer with 32 Filters each of filter size 3*3 and apply Relu activation function.
6. Add Maxpooling layer for extracting the features from convolutional layer.
7. Repeat Step5 and Step6.
8. Add Flatten Layer to convert 3D Array to 1D Array.
9. Add Hidden Layers or Dense Layers with 512 neurons and activation function is relu
10. Add Hidden layer or Dropout layer with value 0.25
11. Add Hidden Layers or Dense Layers with 128 neurons and activation function is relu.
12. Add Output Layer with 15 neurons and Activation function Softmax.
13. Apply Compile function to compile all the layers with loss function categorical_crossentropy and optimizer='Adam'.
14. Model Fit Function is used to fit all variables like training set, steps per epoch=516, epochs=15, validation data=test set etc.
15. Start Training, minimum time taken to train data in this dataset will be 3 hours.
16. After the model gets trained, apply path of the folder where valid images are available.
17. Apply Predict function to predict the valid image to get output.

```
# CNN building.
model = Sequential()

#op matrix = n - f+1 = 252 = 256 - 5 +1
model.add(Conv2D(32, (5, 5),input_shape=input_shape,activation='relu'))
#252 / 3 = 84
model.add(MaxPooling2D(pool_size=(3, 3))) # 84

# 82 = 84 - 3+ 1
model.add(Conv2D(32, (3, 3),activation='relu')) # 82
#82/2 = 41
model.add(MaxPooling2D(pool_size=(2, 2))) # 41

#39 = 41 - 3+1
model.add(Conv2D(64, (3, 3),activation='relu'))
#39/2 = 19.5 ~ 19
model.add(MaxPooling2D(pool_size=(2, 2)))#19

model.add(Flatten())
model.add(Dense(512,activation='relu'))
model.add(Dropout(0.25))
model.add(Dense(128,activation='relu'))
model.add(Dense(num_classes,activation='softmax'))
model.summary()
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 252, 252, 32)	2,432
max_pooling2d (MaxPooling2D)	(None, 84, 84, 32)	0
conv2d_1 (Conv2D)	(None, 82, 82, 32)	9,248
max_pooling2d_1 (MaxPooling2D)	(None, 41, 41, 32)	0
conv2d_2 (Conv2D)	(None, 39, 39, 64)	18,496
max_pooling2d_2 (MaxPooling2D)	(None, 19, 19, 64)	0
flatten (Flatten)	(None, 23104)	0
dense (Dense)	(None, 512)	11,829,760
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 128)	65,664
dense_2 (Dense)	(None, 15)	1,935

Total params: 11,927,535 (45.50 MB)
Trainable params: 11,927,535 (45.50 MB)
Non-trainable params: 0 (0.00 B)

Model Flowchart

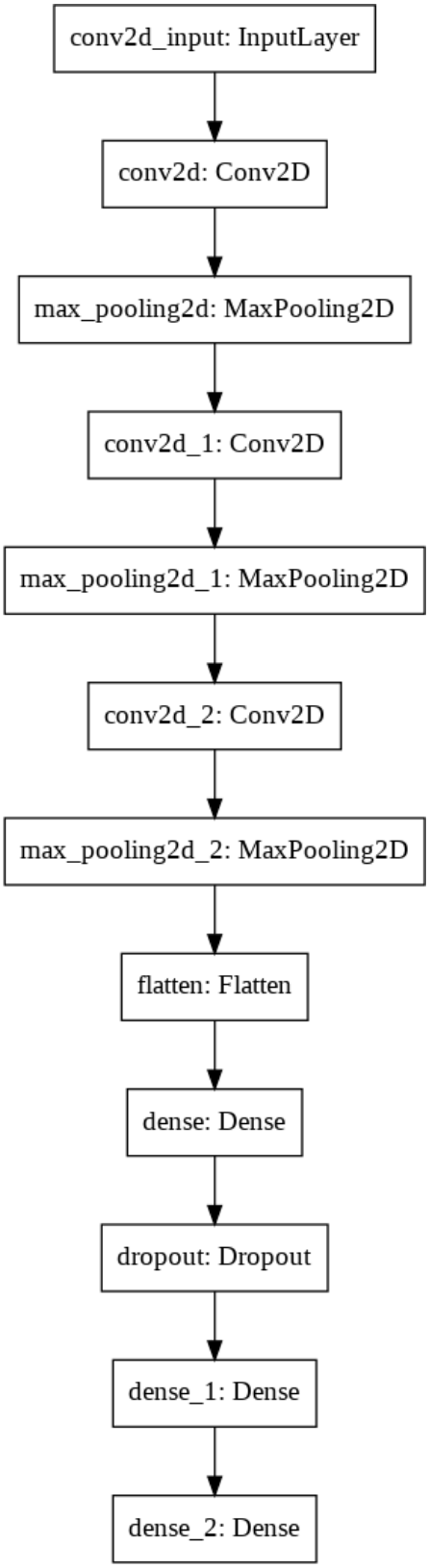


Fig. Flow chart of CNN Model

10.Results

Training Accuracy

```
Epoch 1/10
487/487 [==============================] - 3680s 8s/step - loss: 1.8912 - accuracy: 0.3922 - val_loss: 0.8351 - val_accuracy: 0.7135
Epoch 2/10
487/487 [==============================] - 463s 952ms/step - loss: 0.8104 - accuracy: 0.7304 - val_loss: 0.5578 - val_accuracy: 0.8062
Epoch 3/10
487/487 [==============================] - 459s 944ms/step - loss: 0.6284 - accuracy: 0.7880 - val_loss: 0.4535 - val_accuracy: 0.8438
Epoch 4/10
487/487 [==============================] - 456s 937ms/step - loss: 0.5078 - accuracy: 0.8303 - val_loss: 0.3728 - val_accuracy: 0.8758
Epoch 5/10
487/487 [==============================] - 452s 929ms/step - loss: 0.4347 - accuracy: 0.8569 - val_loss: 0.3246 - val_accuracy: 0.8924
Epoch 6/10
487/487 [==============================] - 448s 920ms/step - loss: 0.3852 - accuracy: 0.8674 - val_loss: 0.2044 - val_accuracy: 0.9329
Epoch 7/10
487/487 [==============================] - 443s 910ms/step - loss: 0.3588 - accuracy: 0.8817 - val_loss: 0.2540 - val_accuracy: 0.9129
Epoch 8/10
487/487 [==============================] - 437s 898ms/step - loss: 0.2951 - accuracy: 0.9020 - val_loss: 0.2062 - val_accuracy: 0.9282
Epoch 9/10
487/487 [==============================] - 434s 891ms/step - loss: 0.2688 - accuracy: 0.9072 - val_loss: 0.1404 - val_accuracy: 0.9536
Epoch 10/10
487/487 [==============================] - 433s 889ms/step - loss: 0.2315 - accuracy: 0.9222 - val_loss: 0.1257 - val_accuracy: 0.9612
```

Applied CNN model on Training data and got 96.12% accuracy for Train Data**.**

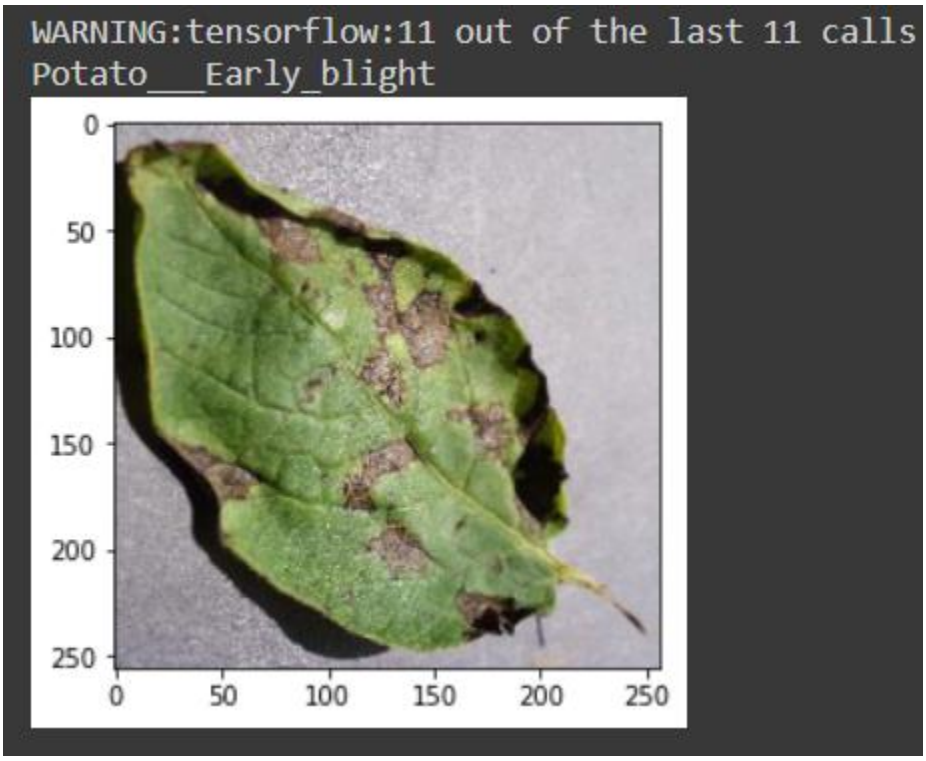
Testing Accuracy

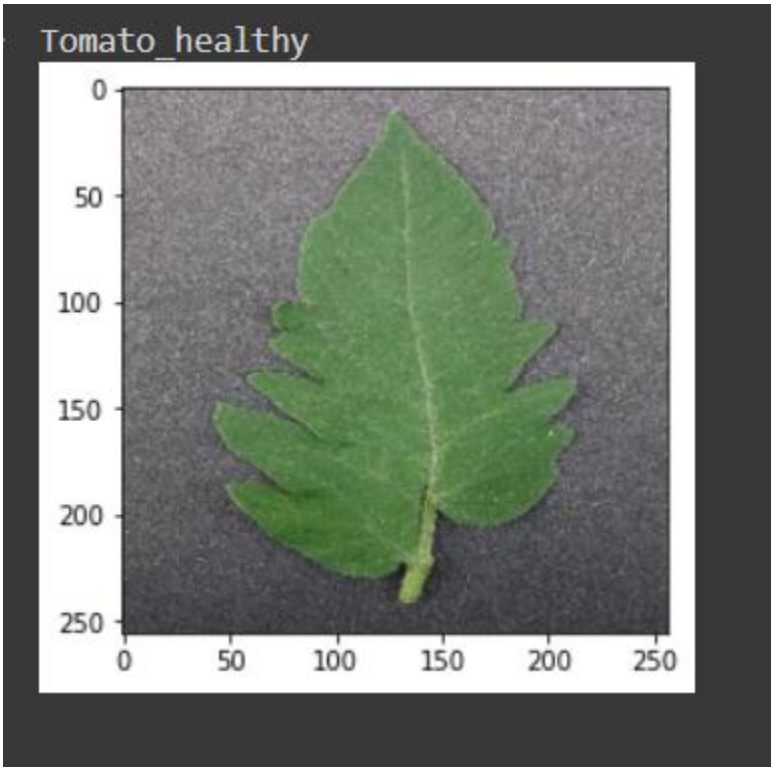
```
score,accuracy =model.evaluate(test_generator,verbose=1)
print("Test score is {}".format(score))
print("Test accuracy is {}".format(accuracy))
```

```
129/129 [==============================] - 714s 6s/step - loss: 0.2459 - accuracy: 0.9150
Test score is 0.2458813637495041
Test accuracy is 0.915027916431427
```

Applied Trained CNN Model on Test data and got accuracy 91.50%.

Training Accuracy	96.12%
Testing Accuracy	91.50%


WARNING:tensorflow:11 out of the last 11 calls
Potato___Early_blight
0
50
100
150
200
250
0
50
100
150
200
250


Tomato_healthy
0
50
100
150
200
250
0
50
100
150
200
250

11.Future Scope

The proposed system can be used for the real time detection of plants using drones on the green fields which will help the farmers to detect the disease. It also helps in ordering fertilizer from ecommerce site. Mobile application can be built for personalized use which will be helpful.

Vision is to Create and use a dataset of multi-spectral images to create the model, and then use multi-spectral cameras on drones for capturing images.

12.Conclusion

This system has utilized deep learning capabilities to achieve automatic plant disease detection system. This system is based on a simple classification mechanism which exploits the feature extraction functionalities of CNN. For prediction finally, the model utilizes the fully connected layers. The system has achieved an overall 94% testing accuracy on publically accessible dataset,

It is concluded from accuracy that CNN is highly suitable for automatic detection and diagnosis of plants. This system can be integrated into mini-drones to live detection of diseases from plants in cultivated areas.

.

Reference

1. Machine Learning with Python Cookbook: Practical Solutions from Pre-processing to Deep Learning
2. Hands-On Machine Learning with Scikit-Learn and Tensor-Flow: Concepts, Tools, and Techniques to Build Intelligent Systems
3. "PlantVillage", Plantvillage.psu.edu, 2020. [Online]. Available: https://plantvillage.psu.edu/. [Accessed: 31- Jan- 2020]
4. S. Mohanty, D. Hughes and M. Salathé, "Using Deep Learning for
5. Image-Based Plant Disease Detection", Frontiers in Plant Science, vol. 7, 2016. Available: 10.3389/fpls.2016.01419
6. https://www.tensorflow.org/tutorials/images/cnn
7. https://www.tensorflow.org/learn
8. Overview of Research on Plant Leaves Disease Detection using Image Processing Techniques: https://rb.gy/aynrzm