SRM INSTITUTE OF SCIENCE AND TECHNOLOGY SCHOOL OF COMPUTING

DEPARTMENT OF DATASCIENCE AND BUSINESS SYSTEMS

18CSE484T - DEEP LEARNING

MINI PROJECT REPORT
PLANT LEAF DISEASE CLASSIFICATION

Name: Harsh Kumar Sharma

Register Number: RA1911027010082

Mail ID: hs7919@srmist.edu.in

Department: Data Science and Business Systems (DSBS)

Specialization: Big data Analytics

Semester: 7

Team Members

Name: Anurag Pancholi	**Registration Number: RA1911027010075**
Name: Harsh Kumar Sharma	**Registration Number: RA1911027010082**

CONTENT PAGE

ABSTRACT

Agriculture plays a crucial role in the Indian economy. Early detection of plant diseases is very much essential to prevent crop loss and further spread of diseases. Most plants such as apple, tomato, cherry, grapes show visible symptoms of the disease on the leaf. These visible patterns can be identified to correctly predict the disease and take early actions to prevent it. The conventional method is the farmers or plant anthologists manually observe the plant leaf and identify the type of disease. In this project, a deep learning model is trained to classify the different plant diseases. The convolutional neural network (CNN) model is used due to its massive success in image-based classification. The deep learning model provides faster and more accurate predictions than manual observation of the plant leaf. In this work, the CNN model and pre-trained model such as ResNet is trained using the dataset.

INTRODUCTION AND MOTIVATION

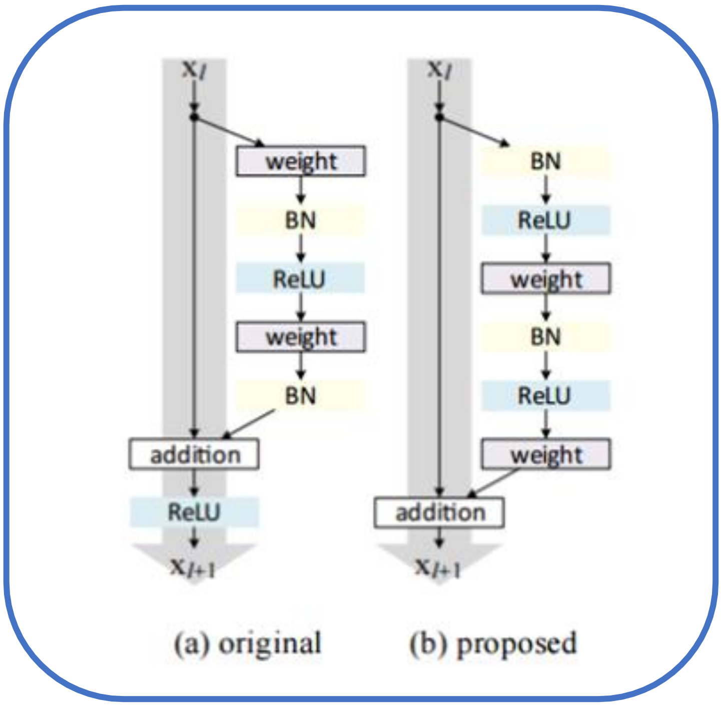
Plant disease detection is a significant challenge in the agriculture sector. Some of the plants show visible symptoms on the plant leaf. These leaf patterns can be used to identify different diseases and take immediate action to prevent the spread. Most of these plant diseases are difficult to detect through naked eyes, and even experienced persons end up wrong. The accuracy of the manual prediction depends upon the experience and knowledge of the person. There are much research works done for plant leaf disease identification using machine learning and deep learning models. This study proposes a deep learning model to classify different plant diseases. In standard machine learning models, the features are extracted manually, and the algorithm learns from that data. Thus it is a two-step process. The deep learning model uses an artificial neural network that can learn features from an input image and make intelligent decisions on its own. Thus deep learning models are far more capable than the standard machine learning models for image-based classification. The working model uses convolutional neural networks and transfer learning to classify different plant leaf diseases. CNN is a type of deep learning neural network and has good success in image-based classification. The proposed system is faster and more accurate than the conventional way of manual observation of each plant leaf. Deploying such a model into a mobile application can help farmers detect different plant diseases using mobile cameras and take necessary actions to avoid disease spread.

KEY METHOLOGIES USED

1. **Deep Learning:** Deep learning, a category of machine learning algorithms which uses various layers to do the extraction of higher level from the raw input. Deep learning is a machine learning method that instructs a computer to do filtration of inputs across the layers Deep learning illustrates the way human brain does the filtration of information. Many deep learning techniques utilizes the neural network architectures. The term “deep” cite to the various hidden layers present inside neural network. In contrast to this conventional neural network that consists of 2-3 hidden layers, the deep neural networks can have as much as one hundred and fifty.

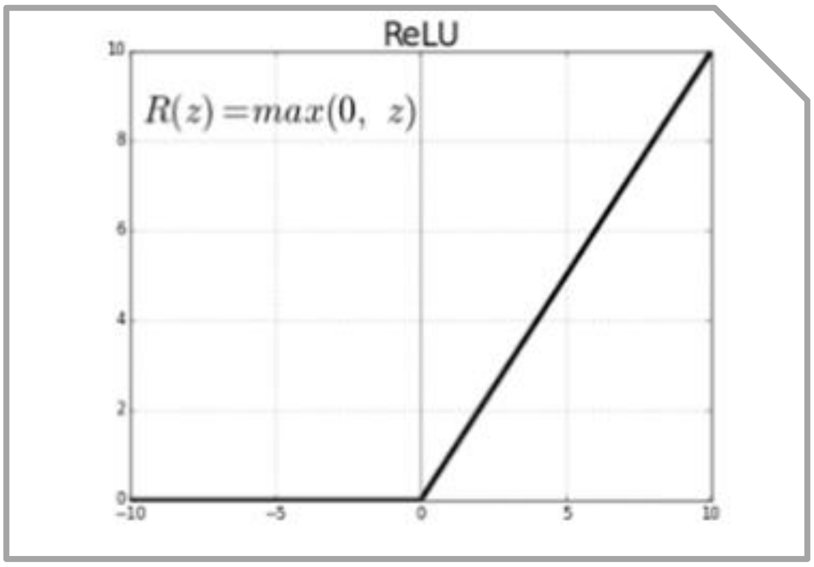
2. **Convolutional Neural Network:** One variant of deep neural networks is called as convolutional neural networks (CNN). A CNN combines well-read features with input data, and then it uses 2D convolutional layers, and hence makes this architecture more suitable for processing 2D data, like images. CNNs abolish the demand for manual feature removal and extraction for the classification of the images. The CNN model of its own extracts features straight from images. The features that are extracted aren’t pre-trained; they are well-read while the network is trained on few groups of images. The Convolutional Neural Network (CNN) model has numerous of layers which execute the processing of image in convolutional layers include- Input layer, Output Layer, Convo Layer, , Fully, Soft-max layer, Connected layer, Pooling Layer.

3. **ResNet**: Researchers at Microsoft research in 2015 propose the Resnet model. They have introduced a new architecture called Residual Network. After the AlexNet model, the winner of the ImageNet 2012 competition, all other proposed architecture uses more layers to reduce the error rate. One of the main problems faced is the vanishing/exploding gradient. To solve this problem, the residual network has been introduced. In this architecture, few layers are skipped from training and are connected directly to the output class. It helps to reduce the vanishing/exploding gradient problem. ResNet is one of the monster architectures which truly define how deep a deep learning architecture can be. Residual Networks consists of multiple subsequent residual modules, which are the basic building block of ResNet architecture.

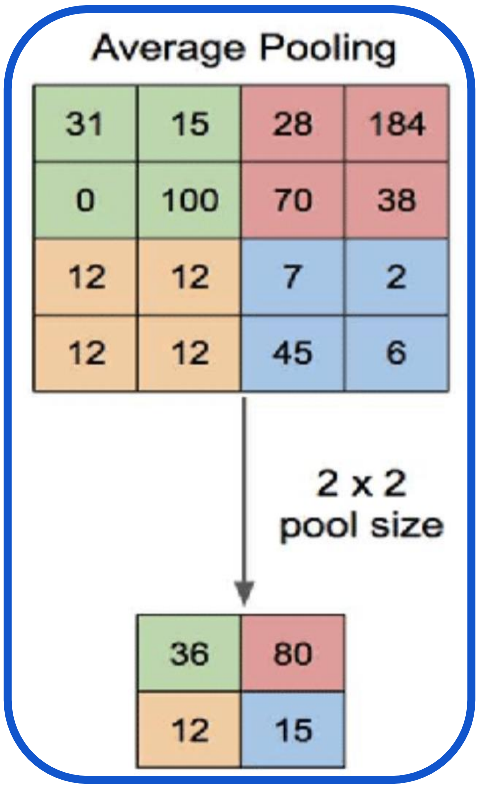


4. **ReLU**: ReLU (Rectified Linear Unit) Activation Function is the most used activation function in the world right now. Since, it is used in almost all the convolutional neural networks or deep learning. The ReLU is half rectified (from bottom). f(z) is zero when z is less than zero and f(z) is equal to z when z is above or equal to zero. Range: [0 to infinity). The

function and its derivative both are monotonic. But the issue is that all the negative values become zero immediately which decreases the ability of the model to fit or train from the data properly. That means any negative input given to the ReLU activation function turns the value into zero immediately in the graph, which in turns affects the resulting graph by not mapping the negative values appropriately.

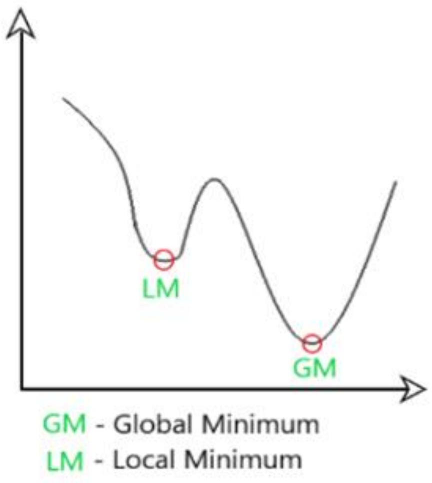


5. **Average Pooling**: A limitation of the feature map output of convolutional layers is that they record the precise position of features in the input. Even small movements in the position of the feature in the input image will result in a different feature map. This can be solved by using down sampling: lower resolution version of an input signal is created that still contains the large or important structural elements, without the fine detail that may not be as useful to the task. Down sampling can be achieved with convolutional layers by changing the stride of the convolution across the image. This is the task of pooling layer. Average pooling involves calculating the average for each patch of the feature map. This means that each 2×2 square of the feature map is down sampled to the average value in the square.

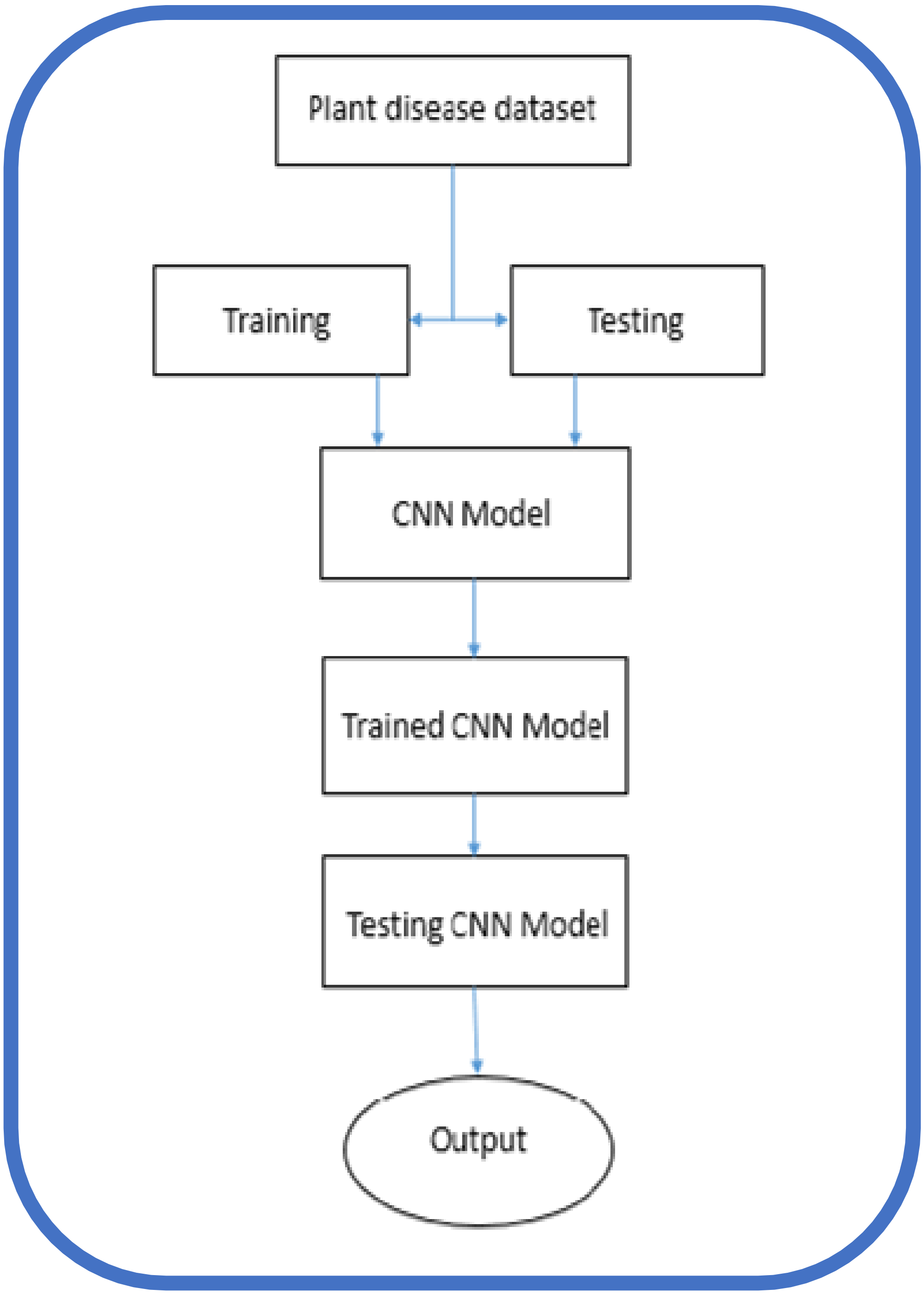


6. **ADAM**: Adaptive Moment (ADAM) Estimation is an algorithm for optimization technique for gradient descent. The method is really efficient when working with large problem involving a lot of data or parameters. It requires less memory and is efficient. Combination of the ‘gradient descent with momentum’ + ‘RMSP’ algorithm. Momentum: This algorithm is used to accelerate the gradient descent algorithm by taking into consideration the ‘exponentially weighted average’ of the gradients. Using averages makes the algorithm converge towards the minima in a faster pace. Root Mean Square Propagation (RMSP): Root mean square prop or RMSprop is an adaptive learning algorithm that tries to improve AdaGrad. Instead of taking the cumulative sum of squared gradients like in AdaGrad, it takes the ‘exponential moving average’. This maintains per-parameter learning rates that are adapted based on the average of recent magnitudes of the gradients for the weight. This means the algorithm does well on online and non-stationary problems. Adam

Optimizer inherits the strengths or the positive attributes of the above two methods and builds upon them to give a more optimized gradient descent. We control the rate of gradient descent in such a way that there is minimum oscillation when it reaches the global minimum while taking big enough steps (step-size) so as to pass the local minima hurdles along the way. Hence, combining the features of the above methods to reach the global minimum efficiently.



FLOW DIAGRAM



IMPLEMENTATION REQUIRED

1) Initial Packages – Pandas, NumPy, Matplotlib, Seaborn – for basic statistical analysis and mathematical insights.

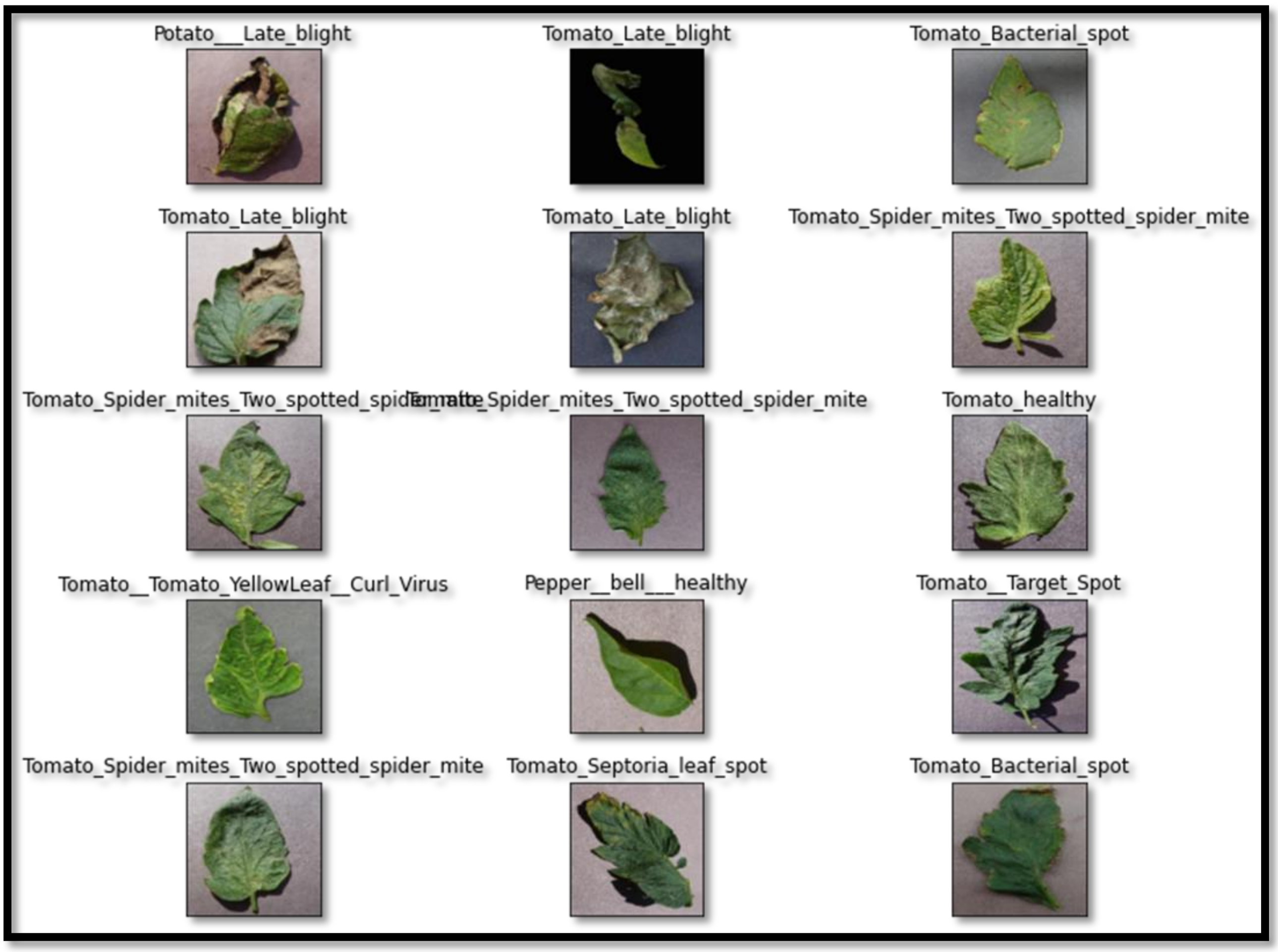
2) Keras - Keras is a high-level, deep learning API developed by Google for implementing neural networks. It is written in Python and is used to make the implementation of neural networks easy. It also supports multiple backend neural network computation.

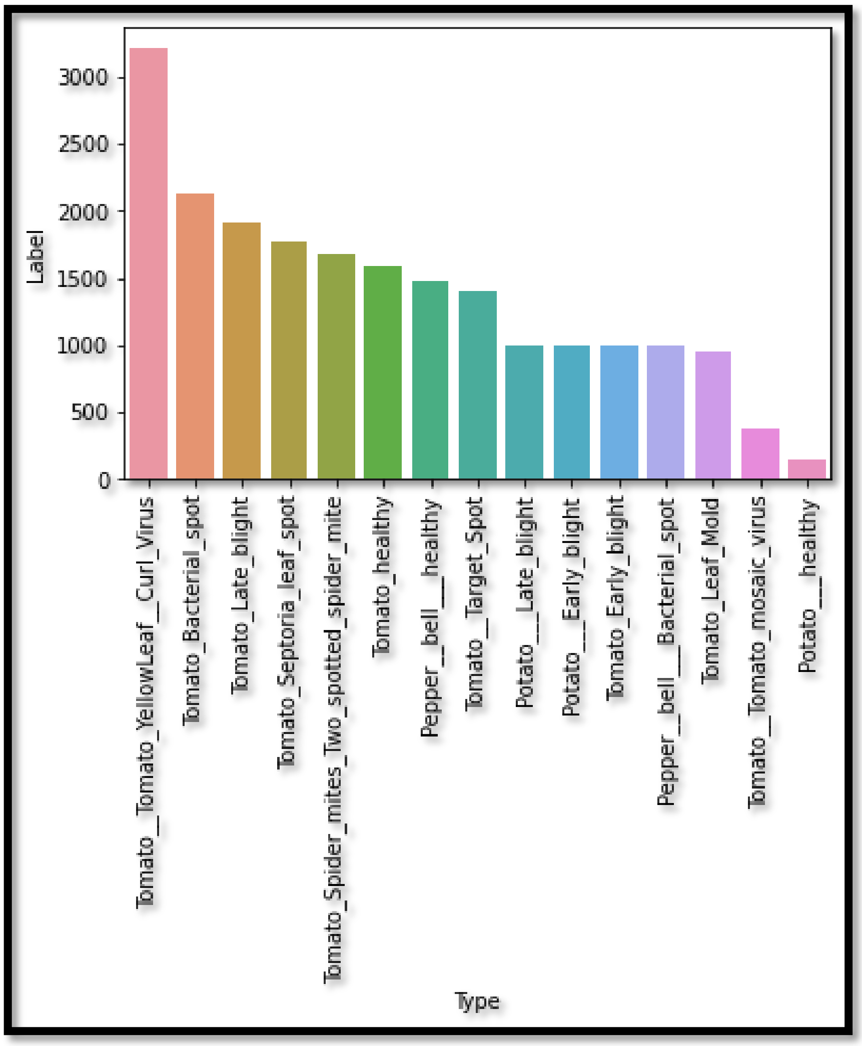
3) **Scikit-Learn** - Scikit-learn is a free software machine learning library for the Python programming language.

4) **Python** – Python based programming language interface in order to run and execute the application.

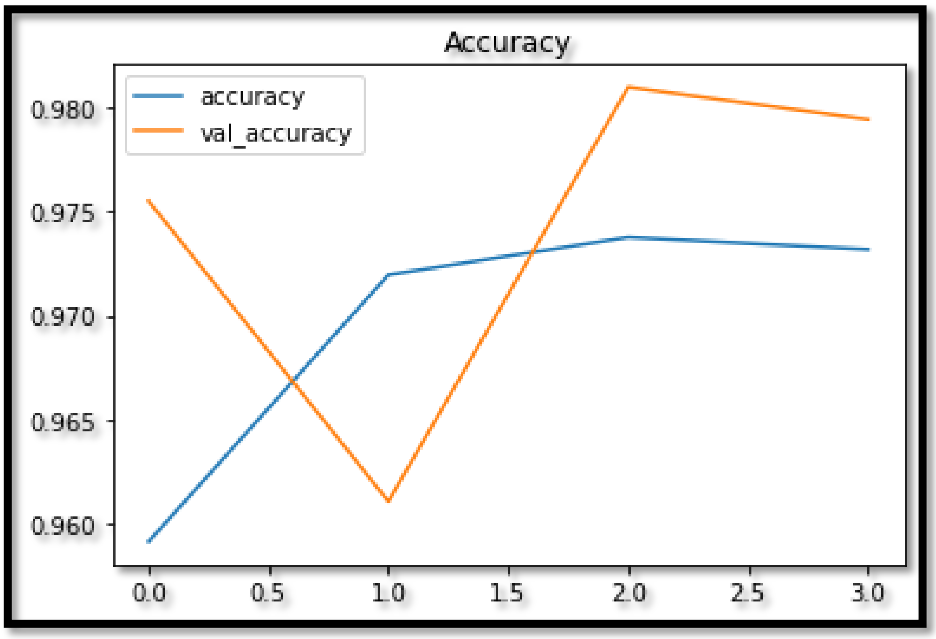
5) **Google Colab** - Colab is a free Jupyter notebook environment that runs entirely in the cloud – cloud based instance which helps to set up a virtual python based environments and run machine learning or deep learning models.

OUTPUT SCREENSHOTS

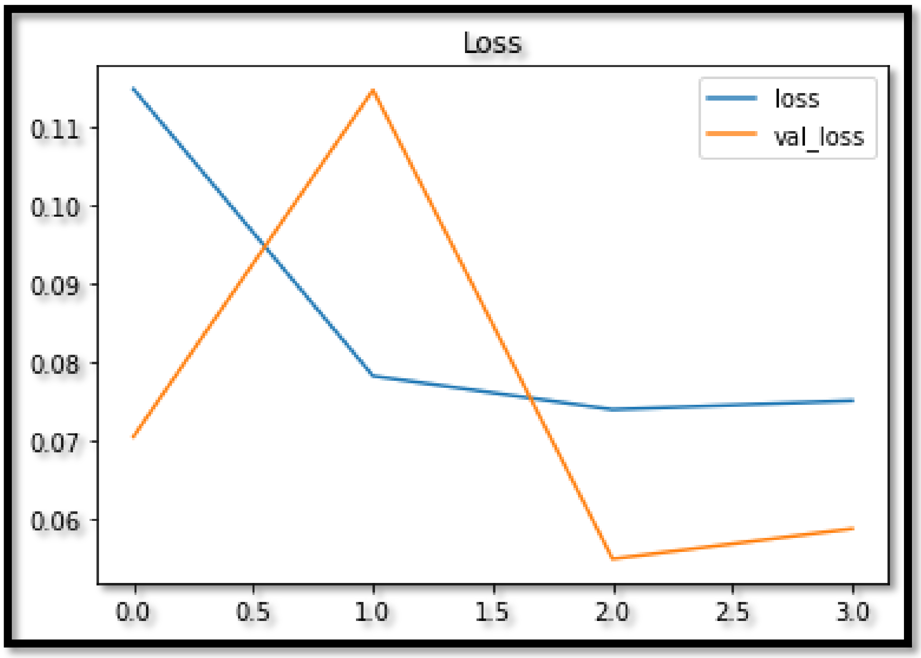

Potato___Late_blight
Tomato_Late_blight
Tomato_Bacterial_spot
Tomato_Late_blight
Tomato_Late_blight
Tomato_Spider_mites_Two_spotted_spider_mite
Tomato_Spider_mites_Two_spotted_spider_mite
Tomato_Spider_mites_Two_spotted_spider_mite
Tomato_healthy
Tomato__Tomato_YellowLeaf__Curl_Virus
Pepper__bell___healthy
Tomato__Target_Spot
Tomato_Spider_mites_Two_spotted_spider_mite
Tomato_Septoria_leaf_spot
Tomato_Bacterial_spot

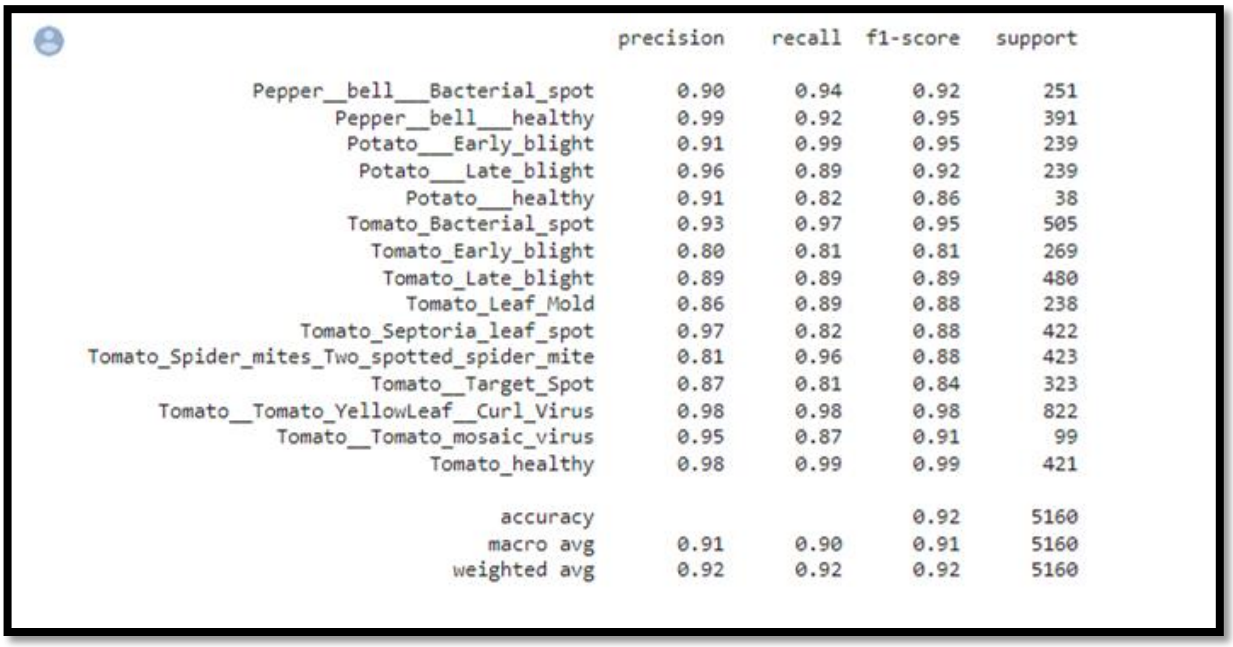


3000
2500
2000
1500
1000
500
0
Label
Tomato__Tomato_YellowLeaf__Curl_Virus
Tomato_Bacterial_spot
Tomato_Late_blight
Tomato_Septoria_leaf_spot
Tomato_Spider_mites_Two_spotted_spider_mite
Tomato_healthy
Pepper__bell___healthy
Tomato__Target_Spot
Potato___Late_blight
Potato___Early_blight
Tomato_Early_blight
Pepper__bell___Bacterial_spot
Tomato_Leaf_Mold
Tomato__Tomato_mosaic_virus
Potato___healthy
Type



Accuracy
accuracy
val_accuracy
0.980
0.975
0.970
0.965
0.960
0.0
0.5
1.0
1.5
2.0
2.5
3.0

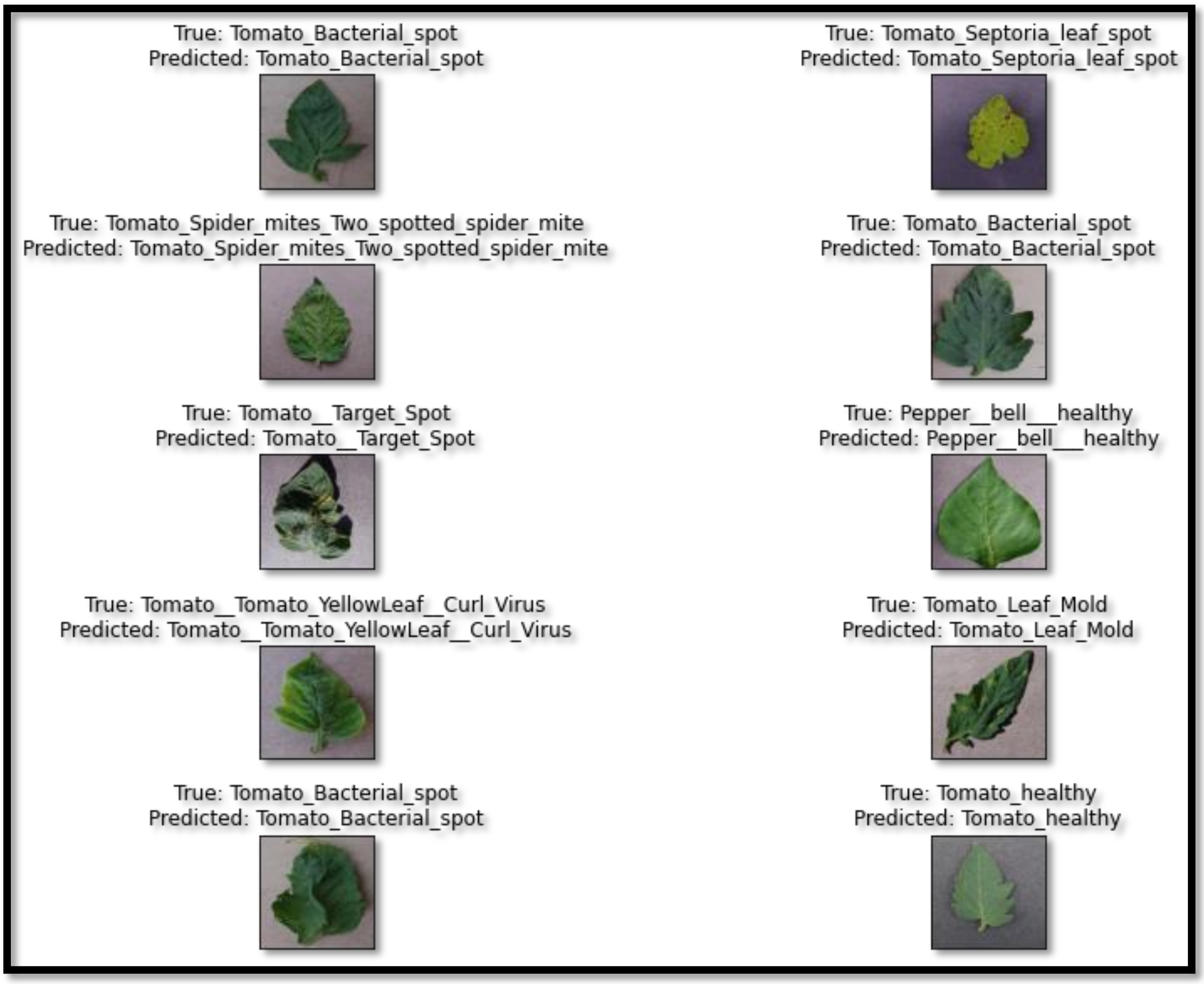




```
                                         precision    recall  f1-score   support

             Pepper__bell___Bacterial_spot     0.90      0.94      0.92       251
                       Pepper__bell___healthy     0.99      0.92      0.95       391
                        Potato___Early_blight     0.91      0.99      0.95       239
                         Potato___Late_blight     0.96      0.89      0.92       239
                             Potato___healthy     0.91      0.82      0.86        38
                        Tomato_Bacterial_spot     0.93      0.97      0.95       505
                          Tomato_Early_blight     0.80      0.81      0.81       269
                           Tomato_Late_blight     0.89      0.89      0.89       480
                             Tomato_Leaf_Mold     0.86      0.89      0.88       238
                    Tomato_Septoria_leaf_spot     0.97      0.82      0.88       422
  Tomato_Spider_mites_Two_spotted_spider_mite     0.81      0.96      0.88       423
                          Tomato__Target_Spot     0.87      0.81      0.84       323
        Tomato__Tomato_YellowLeaf__Curl_Virus     0.98      0.98      0.98       822
                  Tomato__Tomato_mosaic_virus     0.95      0.87      0.91        99
                               Tomato_healthy     0.98      0.99      0.99       421

                                     accuracy                           0.92      5160
                                    macro avg     0.91      0.90      0.91      5160
                                 weighted avg     0.92      0.92      0.92      5160
```


True: Tomato_Bacterial_spot
Predicted: Tomato_Bacterial_spot
True: Tomato_Septoria_leaf_spot
Predicted: Tomato_Septoria_leaf_spot
True: Tomato_Spider_mites_Two_spotted_spider_mite
Predicted: Tomato_Spider_mites_Two_spotted_spider_mite
True: Tomato_Bacterial_spot
Predicted: Tomato_Bacterial_spot
True: Tomato__Target_Spot
Predicted: Tomato__Target_Spot
True: Pepper__bell___healthy
Predicted: Pepper__bell___healthy
True: Tomato__Tomato_YellowLeaf__Curl_Virus
Predicted: Tomato__Tomato_YellowLeaf__Curl_Virus
True: Tomato_Leaf_Mold
Predicted: Tomato_Leaf_Mold
True: Tomato_Bacterial_spot
Predicted: Tomato_Bacterial_spot
True: Tomato_healthy
Predicted: Tomato_healthy

CONCLUSION

A large part of the Indian population relies on agriculture, hence it becomes very essential to detect and recognize the leaf diseases that results in losses, since agriculture is critical to the growth of the economy. This project based on deep learning approach called CNN is utilized to build 13 different plant leaf disease identification, detection and recognition system. This approach utilized a minimum set of layers to identify the diseases of seven classes. The neural network is trained with Plant Village dataset. A Graphical User Interface is designed for this system. This GUI permits the user to choose the images from the dataset. User can select any image from the dataset and the image gets loaded, following which the prediction of the disease will be shown on the User Interface. Convolutional neural network, trained for identifying and recognizing the plant leaf disease, could classify and predict the diseases correctly for almost all the images with few anomalies thus and obtained 94.8% accuracy.

REFERENCES

- https://www.simplilearn.com/tutorials/deep-learning-tutorial/what-is-keras

- https://www.ijrte.org/wp-content/uploads/papers/v10i3/C64580910321.pdf

- https://www.mathworks.com/discovery/deep-learning.html

- https://colab.research.google.com/drive/14upt1XrivEK4TSy3IV4SHn3GZAINME3j?authuser=1

- https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53

SOURCE CODE

```
import os, glob

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

import pandas as pd


from sklearn.model_selection import train_test_split

from keras.preprocessing.image import ImageDataGenerator

from keras.models import Model

from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense,
Dropout

from keras.callbacks import Callback,EarlyStopping

from keras.applications import ResNet50

from keras.applications.resnet import preprocess_input

from sklearn.metrics import  classification_report

name_class = os.listdir(file_path)

name_class

labels = list(map(lambda x: os.path.split(os.path.split(x)[0])[1], filepaths))

filepath = pd.Series(filepaths, name='Filepath').astype(str)
```

```
labels = pd.Series(labels, name='Label')

data = pd.concat([filepath, labels], axis=1)

data = data.sample(frac=1).reset_index(drop=True)

data.head(5)

train, test = train_test_split(data, test_size=0.25, random_state=42)

fig, axes = plt.subplots(nrows=5, ncols=3, figsize=(10,8),
subplot_kw={'xticks':[],'yticks':[]})

for i, ax in enumerate(axes.flat):

    ax.imshow(plt.imread(data.Filepath[i]))

    ax.set_title(data.Label[i])

plt.tight_layout()

plt.show()

train_datagen =
ImageDataGenerator(preprocessing_function=preprocess_input,

                                    validation_split=0.2)

test_datagen =
ImageDataGenerator(preprocessing_function=preprocess_input)

train_gen = train_datagen.flow_from_dataframe(

    dataframe=train,

    x_col='Filepath',

    y_col='Label',

    target_size=(100,100),
    class_mode='categorical',
    batch_size=32,
    shuffle=True,
    seed=42
)
…test_gen = test_datagen.flow_from_dataframe(
    dataframe=test,
    x_col='Filepath',
    y_col='Label',
    target_size=(100,100),
    class_mode='categorical',
    batch_size=32,
    shuffle=False
)
pretrained_model = ResNet50(
    input_shape=(100,100, 3),
    include_top=False,
    weights='imagenet',
    pooling='avg'
)
```

```
pretrained_model.trainable = False
inputs = pretrained_model.input

x = Dense(128, activation='relu')(pretrained_model.output)
x = Dense(128, activation='relu')(x)

outputs = Dense(15, activation='softmax')(x)

model = Model(inputs=inputs, outputs=outputs)
model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)
my_callbacks  = [EarlyStopping(monitor='val_accuracy',
                              min_delta=0,
                              patience=2,
                              mode='auto')]
history = model.fit(
    train_gen,
    validation_data=valid_gen,
    epochs=4,

    callbacks=my_callbacks

)

results = model.evaluate(test_gen, verbose=0)


print("    Test Loss: {:.5f}".format(results[0]))

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

# Predict the label of the test_gen

pred = model.predict(test_gen)

pred = np.argmax(pred,axis=1)


# Map the label

labels = (train_gen.class_indices)

labels = dict((v,k) for k,v in labels.items())

pred = [labels[k] for k in pred]

y_test = list(test.Label)

print(classification_report(y_test, pred))

fig, axes = plt.subplots(nrows=5, ncols=2, figsize=(12, 8),

                        subplot_kw={'xticks': [], 'yticks': []})


for i, ax in enumerate(axes.flat):
    ax.imshow(plt.imread(test.Filepath.iloc[i]))

    ax.set_title(f"True: {test.Label.iloc[i]}\nPredicted: {pred[i]}")

plt.tight_layout()

plt.show()
```

GITHUB PROJECT LINK

https://github.com/Anuragpancholi07/Plant-Disease-Classification