Department of Computer Science and Engineering,
ANNA UNIVERSITY CHENNAI - 600025

CS6301 – Machine Learning Laboratory

Soil classification and plant diseases identification using CNN

NAME	REG.NO	AADHAR NO	E-MAIL	MOBILE NO
AAKASH K	2018103502	816439587887	aakashkumar25112000@gmail.com	6379953528
ASHWATH NARAYAN K S	2018103517	698130597482	ashwath.sharavanan@gmail.com	9443228419

INSTRUCTOR & MENTOR: **Dr AROCKIA XAVIER ANNIE R**

Contents

1. ABSTRACT

Soil is an important ingredient of agriculture. There are all kinds of soil. Each type of soil can have different kinds of features and different kinds of crops grow on different types of soils. We need to know the features and characteristics of various soil types to understand which crops grow better in certain soil types and can yield better .When we can assess a type of soil and plant relevant or suitable seeds accordingly . One can maximize the yield. This model can help smallholder farmers to classify the type of soil and plant seeds according to it.

" Identify the soil type and plant only relevant seeds in it ".

Plant disease is an ongoing challenge for smallholder farmers , which threatens income and food security.The identification of plant disease is an imperative part of crop monitoring systems. In classifying crop diseases ,the traditional method of human analysis by visual inspection is no longer feasible because it's not easy to classify them .Farmers don't get the exact remedies that can be used to stop those diseases. This machine learning model can identify diseases and also suggest the potential remedies to the user

"Identify any diseases early on and take corrective and preventive measures " .

2. INTRODUCTION

By 2050, global crop production must increase by at least 50% to support the predicted demand . The majority of production currently occurs in Africa and Asia, where 83% of farmers are family run with little to no horticultural expertise . Due to this, yield losses of greater than 50%. Introduction of data mining in the agricultural field has made benefits in the research field.

In this paper, we have proposed a working model that can predict soil series with land type and according to prediction it can suggest suitable crops, Furthermore identify plant diseases and provide us with preventive measures . Thus , enabling us to grow crops with better yield. Machine learning is still an emerging and challenging research field in agricultural data analysis

Computer vision and deep learning (DL) techniques have been proven to be state-of-the-art to address various agricultural problems. The recent revolution in smartphone penetration and computer vision models has created an opportunity for image classification in agriculture. Convolutional Neural Networks (CNNs) are considered state-of-the-art in image recognition and offer the ability to provide a prompt and definite diagnosis.

Machine learning techniques can be helpful in this case. The field of machine learning provides well-suited techniques to learn the links between the hyperspectral data and soil texture The crops can be predicted based on a very suitable set of features included in the dataset used , we consider The Soil dataset , It has images for alluvial, clay, red, and black soils .

We give an overview of the current research in soil classification and plant disease identification. The system design with a split of modules and block diagram is described in chapter 7. The applied machine learning approaches are introduced in chapter 4 with the experimental results . Chapter 9 contains the evaluation of the different approaches and the result analysis is briefly discussed. Finally, we conclude this study and give an outlook of possible future research ideas.

3. LITERATURE SURVEY

1. Image-based plant disease identification by deep learning meta-architectures. Muhammad Hammad Saleem , Sapna Khanchi , Johan Potgieter , Khalid Mahmood Arif. Published - 27 October 2020

   Features : Classifies the leaf image fed by the user using CNN with added layers and identify the disease.

   Advantages : Precise disease identification is obtained from the model.

   Disadvantages : Hyperspectral data used in this paper which makes it complexed and less accurate , it can be further enhanced .

2. Soil texture classification with 1d convolutional neural networks based on hyperspectral data F. M. Riese , S. Keller Published - 14 June 2019

   Features : Takes in the soil image from the user and classifies soil texture.

   Advantages : Accurate Soil texture Classification.

   Disadvantages : Soil image with more features are required as input from user.

3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pp. 1097-1105.

Features : Takes in the image from the user and classify the soil according to the features extracted from the dataset.

Advantages : Precise classification of the image is obtained.

Disadvantages : The CNN model has no optimiser and CNN layers could have been more enhanced for better results.

Our innovation : The models described here are less efficient. There fore an better modified CNN model with added extra features along with added layers will make the model more precise. This will lead to a optimised model with high accuracy. The CNN layers and details will be described briefly in upcoming chapters.

4. PROBLEM STATEMENT

Soil is an important ingredient of agriculture. There are several kinds of soil. Each type of soil can have different kinds of features and different kinds of crops grow on different types of soils. We need to know the features and characteristics of various soil types to understand which crops grow better in certain soil types.

Plant disease is an ongoing challenge for smallholder farmers, which threatens income and food security. The identification of plant disease is an imperative part of crop monitoring systems. In classifying crop diseases, the traditional method of human analysis by visual inspection is no longer feasible

5. PROBLEM SOLUTION

5.1 DATA SET:-

SOIL DATASET :

This data-set is created for Soil Type Classification from Image. There are 903 RGB images . The main classifications of the data-set are : "Alluvial Soil", "Red Soil", "Clay Soil" and "Black Soil". The dataset has 2 folders : Test and Train . Train and test , both have 4 main folders representing each classification . Under train , each soil has 180 images and under train each class has 48 images.

PLANT DISEASE DATASET :

This dataset is recreated using offline augmentation from the original dataset. This dataset consists of about 87K rgb images of healthy and diseased crop leaves which is categorized into 38 different classes. The total dataset is divided into 80/20 ratio of training and validation set preserving the directory structure. A new directory containing 33 test images is created later for prediction purposes.The dataset has 2 folders : Test and Train. Train and test , both have 4 main folders representing each classification

5.2 INPUT :-

The user is supposed to give an image either soil image which he wants to classify or a image of the leaf which he wants to find the disease. The input image is fed into the respective model and output (i.e Classification of soil type or classification of disease) is obtained.

5.3 APPROACH:-

Machine learning techniques can be helpful in this case. The field of machine learning provides well-suited techniques to learn the links between the hyperspectral data and soil texture . The crops can be predicted based on a very suitable set of features included in the dataset used , we consider The Soil dataset , It has images for alluvial, clay, red, and black soils .

Computer vision and deep learning (DL) techniques have been proven to be state-of-the-art to address various agricultural problems. The recent revolution in smartphone penetration and computer vision models has created an opportunity for image classification in agriculture. Convolutional Neural Networks (CNNs) are considered state-of-the-art in image recognition and offer the ability to provide a prompt and definite diagnosis.

6. NOVELTY

In this section, we briefly review the published research which is related to the presented classification of soil texture and plant disease identification based on hyperspectral data. A first review of geological remote sensing is given by Cloutis (1996). Traditional approaches like nearest mean, nearest neighbor, maximum likelihood, hidden Markov models and spectral angle matching for the classification of soil texture show

acceptable results (Zhang et al., 2003, 2005; Shrestha et al., 2005).The increasing popularity of deep learning approaches in many research disciplines has also reached the field of remote sensing. Among different deep learning techniques, the deep convolutional neural network (CNN) has been used mostly for image classification (Krizhevsky et al., 2012; Lu et al., 2017). The CNN model provides a relationship between layers and spatial information of the image and hence it is convenient for the classification of images (Arel et al., 2010). Along this line, there are limited works on plant disease classification using CNN. Lu et al., 2017, investigated the ability of deep CNN technique for classification of different rice diseases. A total of 500 images belonging to 10 categories have been considered and used CNN model with three convolution layers, three stochastic pooling layers and softmax layer at the end. The classification accuracy of 90.48% has been reported.

In our project we have combined the ideas of both the soil classification and plant disease identification and have built a CNN which at one time can predict a soil type and predict suitable recommendations and also identify plant diseases and suggest suitable remedies. The project creates a CNN model , The deep convolutional neural network (CNN) is the most popular and extensively used for image recognition (Lu et al., 2017). It mainly comprises convolutional, pooling and fully-connected layers. The following explains our methodology in this project .

Data pre-processing is crucially important to a model's performance . In this first module the dataset is split into training and testing data and then preprocessing is done i.e.., any background noise or disturbances are removed by augmentation and normalization and the dataset is loaded. After that the important features are extracted which is to be used to train the model. We define the parameters to be used in the CNN model with a series of Convolutional and Max pooling layers with dropouts in between . Now the features extracted from the Extraction module are used to form a model which will be trained using convolutional neural networks to gain knowledge about the dataset, we check for small tweaks in hyper parameters and validate the best suitable accuracy. The dataset will be tested to classify the soil type of the soil based on the information gained from the testing set / will be able to identify diseases if any . Here we build a user interface (application) , such that a common farmer or end user can use the model with ease, In classification , the testing dataset will be able to classify the soil type and suggest suitable crops and provide us about information about the diseases if any present in the plant of the speech. We use a sequential API for model building and create a CNN with layers like convolutional, RelU, Max Pooling, Dropout.We also use softmax function to output a vector. For training the CNN Model, the images in the training set and the testing set are fitted to the sequential model we built using Keras library.

7. ARCHITECTURE

7.1 DETAILED ARCHITECTURE

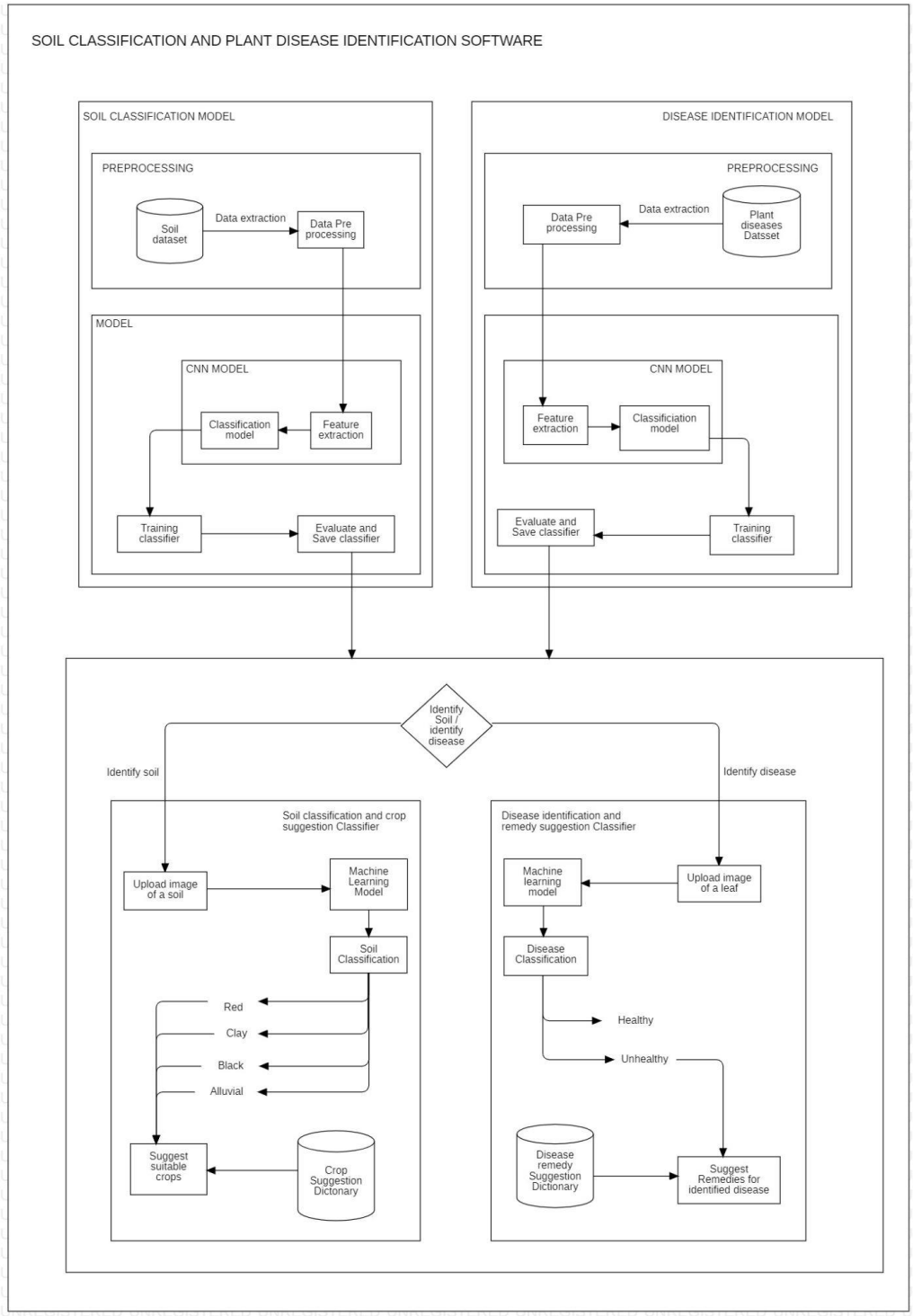

Fig 1 : Architecture diagram for the project

7.2 EXPLANATION OF THE ARCHITECTURE

UPPER TWO BLOCKS

The Soil classification and plant disease identification software has two independent models. One is Soil model and other is Plant model. Soil classification can be performed keeping focus on different features of soil, such as soil particle size, texture, colour or combination of features. Digital image processing and computer vision approaches can be employed on soil images for classification. Colour is an extensive measure of soil, so a substantial measure of soil statistics can be productively acquired by analysing the soil colour. Soil texture in another property which has been used in various approaches to determine the soil type, mainly clay, alluvial , black and red soil. On that account, the process of soil classification and qualitative distinction based on colour texture of the soil is done.

PROCESS INVLOVED

After the feature extraction, CNN algorithms are used to train models. The trained model is evaluated, tested and saved. The convolution Neural networks is used to train and save the plant model. We prefer CNN as it includes properties like Depth(Filters), Stride (number of pixels by which we slide our filter matrix over the input matrix.)Zero padding(it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix.) and Non-Linearity(ReLU).The models are saved and finally it’s time to go for general uses.

FINAL BLOCK

The final block uses both models. When Identify soil is to be done, a soil image is uploaded, the saved soil model are loaded and the classification is made. When Identify disease is to be done ,a leaf image is uploaded, the saved leaf model is loaded and the identification is done. User can use any one of the model to find the classification.

7.3 DATA FLOW DIAGRAM (SOIL MODEL)

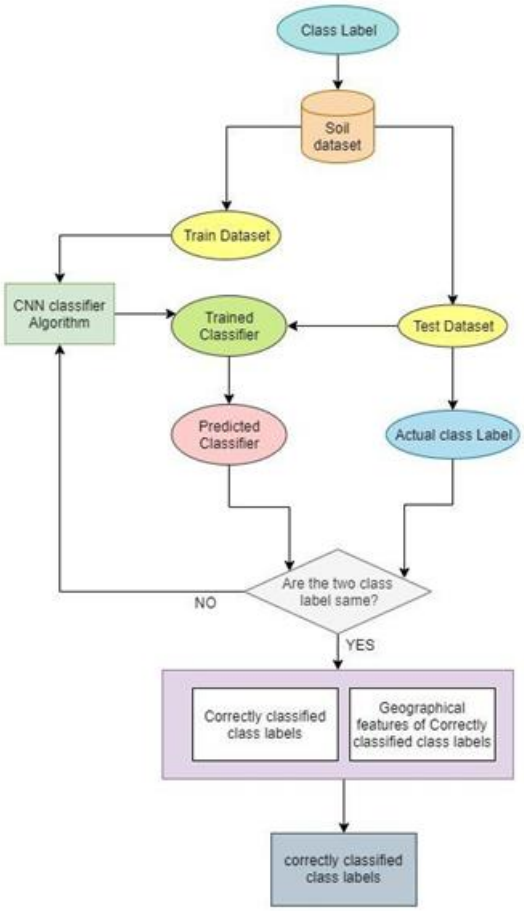

Fig 2 : data flow diagram for soil classification

7.4 EXPLANATION OF THE DATA FLOW DIAGRAM (SOIL MODEL)

The flow of data in the soil classification model is shown in figure 2. The Soil dataset has different class labels for different images. These images are preprocessed and the data is used to train the model using CNN algorithm. In the test dataset, it is preprocessed and send to the trained model for classification. The classified result gives the class label. The label is then used to fetch the suggested crop from the crop suggestion dictionary.

7.5 DATA FLOW DIAGRAM (PLANT MODULE)

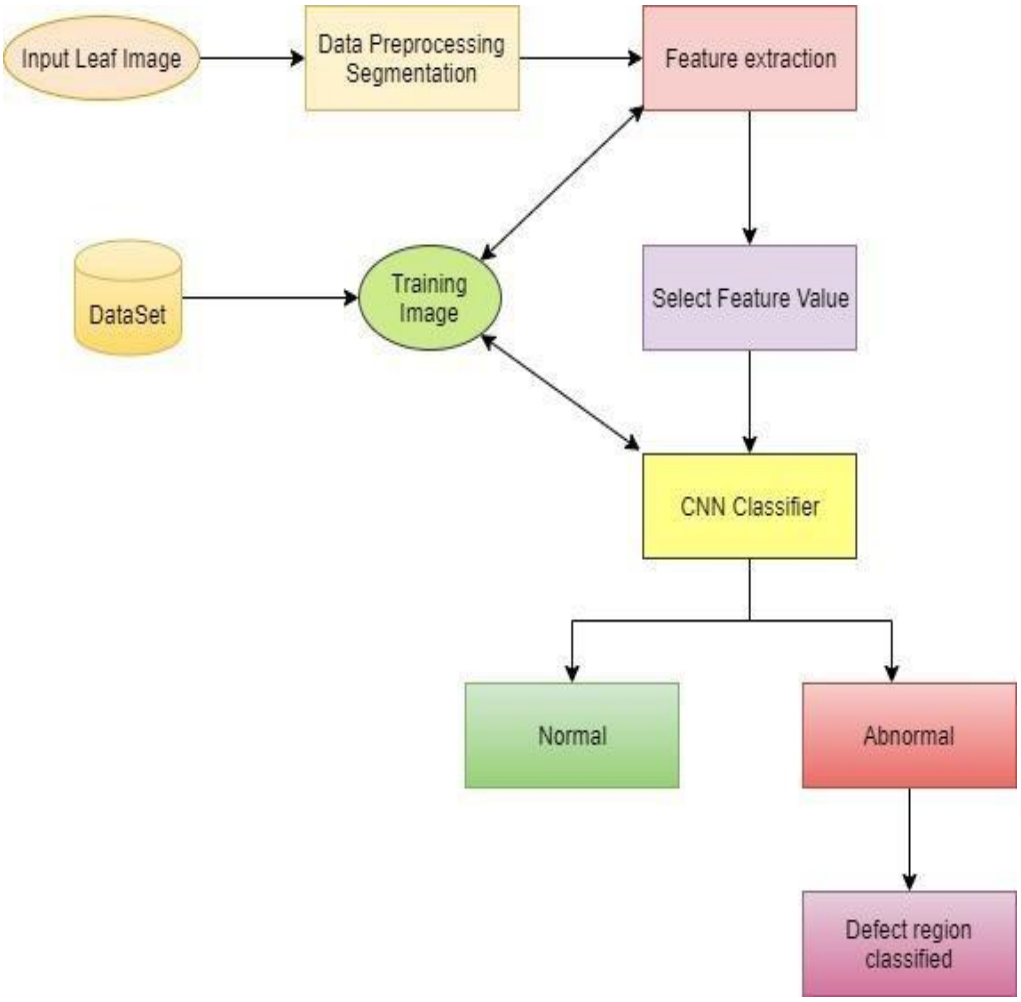

Fig 3 : Data Flow diagram for plant disease classification

7.6 EXPLANATION OF THE DATA FLOW DIAGRAM (PLANT MODEL)

Fig 3 shows the flow of the developed model . Using the dataset the CNN model is trained . The input is an image of a leaf for which disease has to be identified . The input image is preprocessed and the features are extracted , which will be sent to the CNN classifier . The model classifies the input as normal / abnormal and the results are shown. If abnormal the remedies that are identified are displayed.

8. DETAILED MODULE DESIGN

This project has 8 modules, first 4 is related with classification of soil and last 4 is related with plant disease identification.

1 Soil Data set Pre-processing

2 Building CNN MODEL for Soil classification

3 Train ,Test ,Evaluate and save Model

4 SOIL Classification

5 Plant disease Data set Pre-processing

6 Building CNN MODEL for Plant disease classification

7 Train ,Test ,Evaluate and save Model

8 Plant disease identification

Detailed description along with pseudo code for each modules with their respective module block diagram are described below.

8.1 Soil Data set Pre-processing

In this module, We have divided images into 4 folders so that each folder represents a different soil type. The shape of data is (n,x,y,z) which means that there are n images of size x*y pixels and z means the data contains colored images. With the split folders package, we use the ratio() function for splitting data arrays into subsets (for training data and for validating data and using ImageDataGenerator() method We then start to augment and normalize the images to reduce noise , resize and grey scale them

INPUT : Image files from the dataset.

OUTPUT : Data generated normalised images split into training and testing dataset.

PSEUDO CODE :

PHASE 1 (DATA SET SPLIT):

```
Import necessary modules
Path = Directory containing the dataset folder
For each class in the path:
    Count the number of images in the folder
    Set train percent as 80 and validation percent as 20.
    Set range in number of images using percent given above.
    Copy first range in train folder of class
Copy second range in validation folder of class.
```

PHASE 2 (DATA PRE PROCESSING):

```
Import necessary modules
Initialize 2 empty lists images[] and labels[]
For each train,validation and testing:
For each category in dataset:
    Read the image
    Resclae images to 1/255
    For each pixel in image:
        Red=pixel scale of red
        Blue=pixel scale of blue
        Green=pixel scale of green
        Grey=red+blue+green/3
    Convert image to greyscale
    resize the greyscaled image into size 244x244 in order to keep size of the
images consistent
    Handle the exception in case any error occurs
```

ARCHITECTURE :

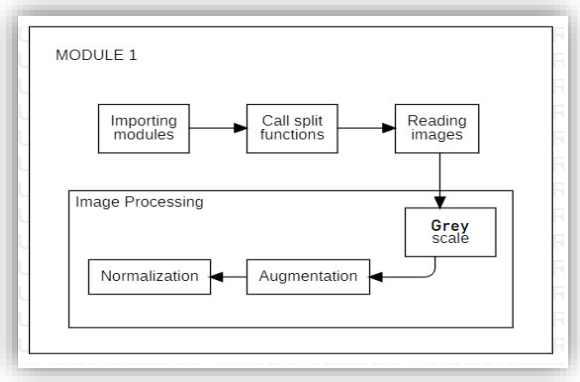

Fig 4. Module 1 block diagram

8.2 Building CNN MODEL for Soil classification

To classify the Soil, we build a CNN (Convolutional Neural Networks) model We use a sequential model. So that, the layers in the network will be added in sequence. We'll use a feed forward network with 6 convolutional layers followed by a fully connected hidden layer. We'll also use dropout layers in between. Dropout regularizes the networks, i.e. it prevents the network from overfitting. All our layers will have relu activations except the output layer. Output layer uses softmax activation as it has to output the probability for each of the classes.Sequential is a keras container for linear stack of layers. Each of the layers in the model needs to know the input shape it should expect, but it is enough to specify input_shape for the first layer of the Sequential model. Rest of the layers do automatic shape inference.To attach a fully connected layer (aka dense layer) to a convolutional layer, we will have to reshape/flatten the output of the conv layer. This is achieved by Flatten layer

INPUT : Parameters for the CNN model

OUTPUT : CNN model.

PESUDO CODE :

```
Import necessary keras libraries
Define model parameters:
    No of classes
    Batch size
    Use sequential API for model building
Create first and second layer groups containing the layers:      Convolutional
    RelU
    Maxpooling
    Flatten and Dropout Layer to stack the output convolutions above as well as cater
overfitting
    Apply softmax classifier
Import adam optimizer from keras library
Use model.compile function from keras library and set the following parameters:
    loss = 'categorical_crossentropy'
    optimizer = Adam,
     metrics = ['accuracy']
```

ARCHITECTURE :

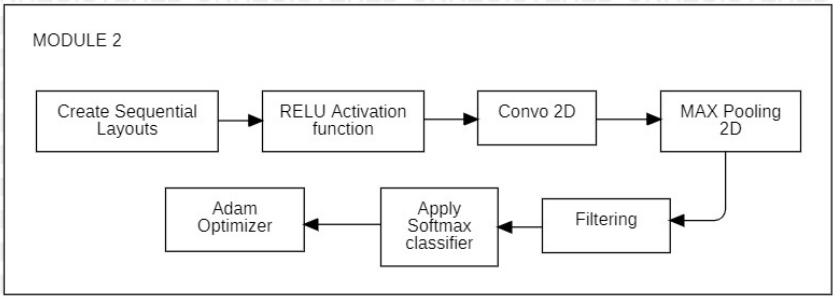

Fig 5. Module 2 block diagram

8.3 Train ,Test ,Evaluate and save Model

During the training, our model will iterate over batches of training sets, each of size batch_size. For each batch, gradients will be computed and updates will be made to the weights of the network automatically. One iteration over all the training set is referred to as an epoch. After building the model architecture, we then train the model using model.fit(). The dataset contains a test folder, it has the details related to the image path and their respective class labels. From there, we extract the image path and labels using pandas. Using the confusion matrix, we get the best model with high accuracy.

INPUT : CNN model, Testing and Training data.

OUTPUT : Soil Classifier.

PESUDO CODE :

```
To train the model:
Fit the model with train data using model.fit function provided by keras
set the following parameters:
    Batch size = 32
    epoch = 100,
    Call back parameter which exits training if particular patience in  loss is met.
Test model with test data using model.evaluate to get loss and accuracy over test data.
Evaluate model with confusion matrix using values of precision,recall,accuracy.
Save model.
```

ARCHITECTURE :

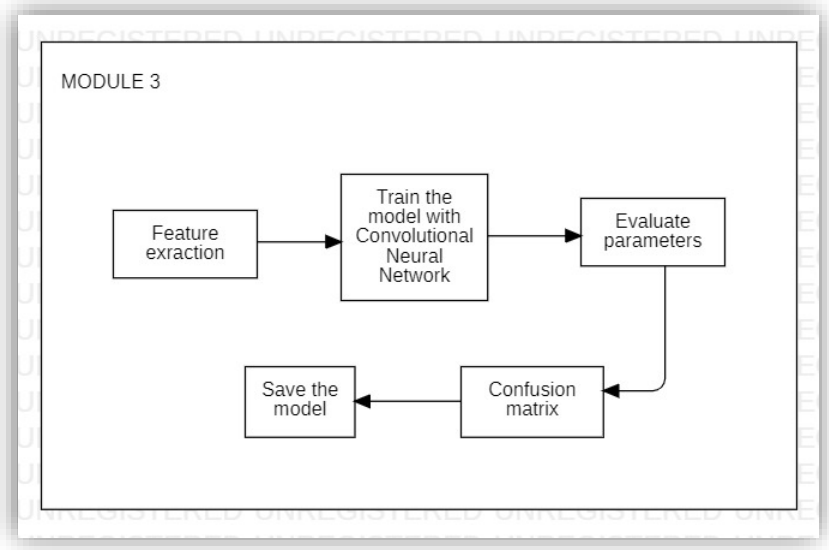

Fig 6. Module 3 block diagram

8.4 SOIL Classification

In module 4 We import necessary modules and label the 4 main soil classification. The h5 extension model which we saved previously is loaded here. The input image of SOIL is given through path to the model which is loaded and the output is obtained.

INPUT : Uploaded image of soil.

OUTPUT : Predicted soil type.

PESUDO CODE :

```
Import necessary modules
Load saved model for soil classification.
Upload image that is to be classified.
Save the image
Pass location of the saved image to the model and predict.
```

ARCHITECTURE :

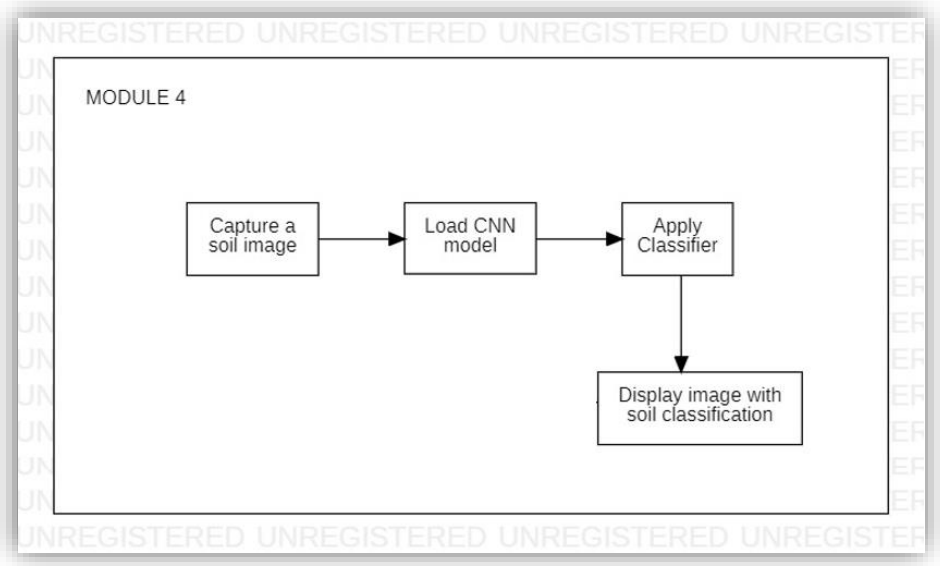

Fig 7. Module 4 block diagram

8.5 Plant disease Data set Pre-processing

Data pre-processing is crucially important to a model's performance . We use RGB data which produces clear, noise free images. The dataset is divided into 80% for training and 20% for validation. First, augmentation settings are applied to the training data. Augmenting training images can not only reduce overfitting but can improve a model's overall performance. Finally, all images are resized and normalized. Resizing is carried out using a compress function.

INPUT : Image files from the dataset.

OUTPUT : Data generated normalised images split into training and testing dataset.

PSEUDO CODE :

PHASE 1 (DATA SET SPLIT):

```
Import necessary modules
Path = Directory containing the dataset folder
For each class in the path:
    Count the number of images in the folder
    Set train percent as 80 and validation percent as 20.
    Set range in number of images using percent given above.
    Copy first range in train folder of class
Copy second range in validation folder of class.
```

PHASE 2 (DATA PRE PROCESSING):

```
Import necessary modules
Initialize 2 empty lists images[] and labels[]
For each train,validation and testing:
For each category in dataset:
    Read the image
    Resclae images to 1/255
    For each pixel in image:
        Red=pixel scale of red
        Blue=pixel scale of blue
        Green=pixel scale of green
        Grey=red+blue+green/3
    Convert image to greyscale
    resize the greyscaled image into size 244x244 in order to keep size of the
images consistent
    Handle the exception in case any error occurs
```

ARCHITECTURE :

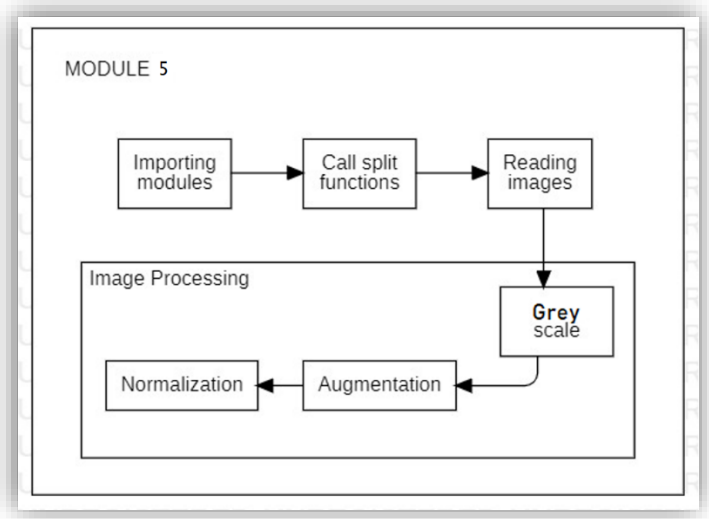

Fig 8. Module 5 block diagram

8.6 Building CNN MODEL for Plant disease classification

To classify the Soil, we build a CNN (Convolutional Neural Networks) model We use a sequential model. So that, the layers in the network will be added in sequence. We'll use a feed forward network with 6 convolutional layers followed by a fully connected hidden layer. We'll also use dropout layers in between. Dropout regularizes the networks, i.e. it prevents the network from overfitting. All our layers will have relu activations except the output layer. Output layer uses softmax activation as it has to output the probability for each of the classes. Sequential is a keras container for linear stack of layers. Each of the layers in the model needs to know the input shape it should expect, but it is enough to specify input_shape for the first layer of the Sequential model. Rest of the layers do automatic shape inference .To attach a fully connected layer (aka dense layer) to a convolutional layer, we will have to reshape/flatten the output of the conv layer. This is achieved by Flatten layer

INPUT : Parameters for the CNN model

OUTPUT : CNN model.

PESUDO CODE :

```
Import necessary keras libraries
Define model parameters:
    No of classes
```

```
    Batch size
    Use sequential API for model building
Create first and second layer groups containing the layers:      Convolutional
    RelU
    Maxpooling
    Flatten and Dropout Layer to stack the output convolutions above as well as cater
overfitting
    Apply softmax classifier
Import adam optimizer from keras library
Use model.compile function from keras library and set the following parameters:
    loss = 'categorical_crossentropy'
    optimizer = Adam,
     metrics = ['accuracy']
```

ARCHITECTURE :

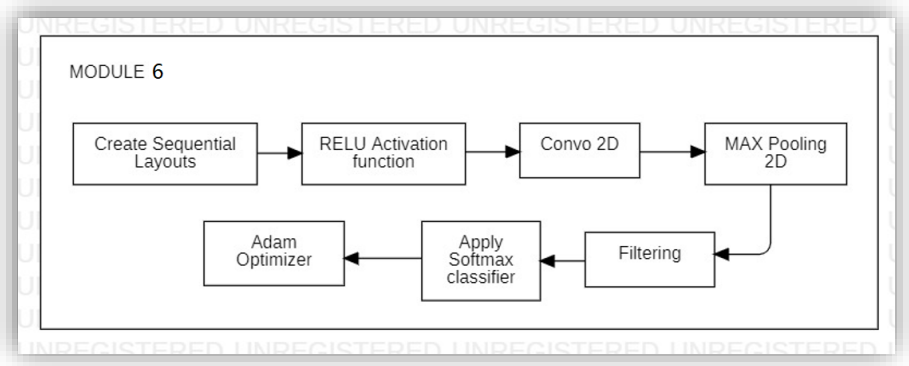

Fig 9. Module 6 block diagram

8.7 Train ,Test ,Evaluate and save Model

During the training, our model will iterate over batches of training sets, each of size batch_size. For each batch, gradients will be computed and updates will be made to the weights of the network automatically. One iteration over all the training set is referred to as an epoch. After building the model architecture, we then train the model using model.fit(). The dataset contains a test folder, it has the details related to the image path and their respective class labels. From there, we extract the image path and labels using pandas. Using the confusion matrix, we get the best model with high accuracy.

INPUT : CNN model, Testing and Training data.

OUTPUT : Soil Classifier.

PESUDO CODE :

To train the model:
Fit the model with train data using model.fit function provided by keras
set the following parameters:
    Batch size = 32
    epoch = 100,
    Call back parameter which exits training if particular patience in loss is met.
Test model with test data using model.evaluate to get loss and accuracy over test data.
Evaluate model with confusion matrix using values of precision,recall,accuracy.
Save model.

ARCHITECTURE :

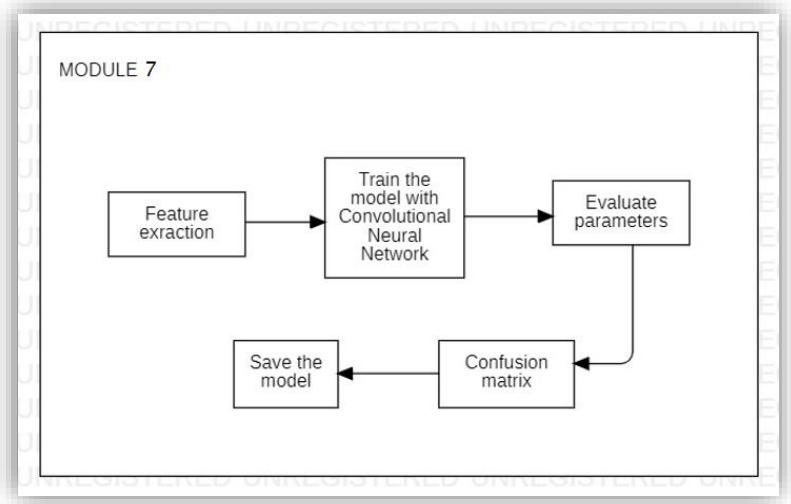

Fig 10. Module 7 block diagram

8.8 Plant disease identification

In our final module , We import necessary modules and label the 38 classification. The h5 extension model which we saved previously is loaded here. The input image of LEAF is given through path to the model which is loaded and the output is obtained.

INPUT : Uploaded image of Leaf.

OUTPUT : Predicted disease type.

PESUDO CODE :

Import necessary modules
Load saved model for plant disease identification.
Upload image in which disease is to be identified.
Save the image
Pass location of the saved image to the model and predict the disease.

ARCHITECTURE :

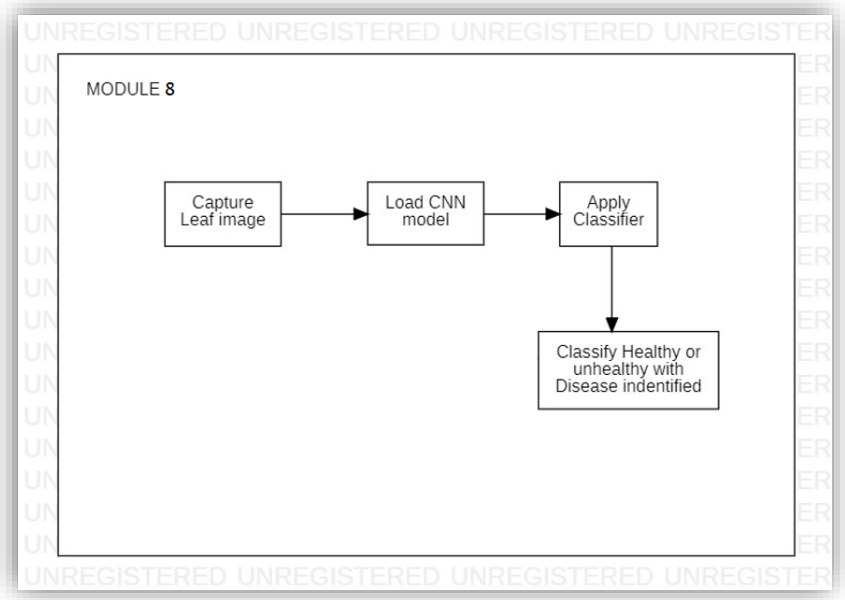

Fig 11. Module 8 block diagram

9. IMPLEMENTATION

9.1 INITIAL SETUP :

- Jupyter notebook is installed.
- A virtual python environment is create.
- TensorFlow module should be installed.
- Jupyter notebook is opened through cmd prompt inside the python environment.

9.2 CODE SNIPPETS

9.2.1 Soil Data set Pre-processing

SPLIT THE DATA SET

In [1]:
```
import splitfolders
```

In [2]:
```
splitfolders.ratio("Soil_Dataset/Train", output="Soil_Dataset/data/", seed=1337, ratio=(.8, .2), group_prefix=None)
```

In [3]:
```
from tensorflow import keras
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

IMPORT REQUIRED PACKAGES

In [4]:
```
from keras import layers
from tensorflow.keras.applications.xception import preprocess_input
from tensorflow.keras.preprocessing.image import load_img , img_to_array , ImageDataGenerator
```

In [5]:
```
SoilType = ['Alluvial_Soil', 'Black_Soil', 'Clay_Soil', 'Red_Soil']

DATA_PATH = 'Soil_Dataset/'
```

RESCALING AND RESIZING DATASET IMAGES

In [6]:
```
#import train data
train_datagen = ImageDataGenerator(rescale=1/255,
                                   shear_range = 0.3,
                                   zoom_range = 0.3,horizontal_flip = True,
                                   vertical_flip =  True ,
                                   rotation_range=60)


train_data = train_datagen.flow_from_directory(DATA_PATH+'train',
                                               target_size = (244, 244),
                                               class_mode='sparse',
                                               shuffle=True,seed=1)
```

```
Found 715 images belonging to 4 classes.
```

In [7]:
```
#import val data

val_datagen = ImageDataGenerator(rescale = 1/255)
val_data = val_datagen.flow_from_directory(DATA_PATH+'data/val',
                                                       target_size=(244,244),
                                                       class_mode='sparse',
                                                       shuffle=True,seed=1)
```

```
Found 144 images belonging to 4 classes.
```

In [8]:
```
# import test data


test_datagen = ImageDataGenerator(rescale = 1/255)
test_data = test_datagen.flow_from_directory(DATA_PATH+'Test',
                                             target_size=(244,244),
                                             class_mode='sparse',
                                             shuffle=False,seed=1)
```

```
Found 188 images belonging to 4 classes.
```

9.2.2 Building CNN MODEL for Soil classification

BULIDING THE CNN MODEL

In [9]:
```
# Defining Cnn
model = tf.keras.models.Sequential([
  layers.Conv2D(32, 3, activation='relu',input_shape=(244,244,3)),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.3),
  layers.Conv2D(128, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(256, activation='relu'),
  layers.Dropout(0.15),
  layers.Dense(128, activation='relu'),
  layers.Dropout(0.1),
  layers.Dense(4, activation= 'softmax')
])
```

In [10]:
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

9.2.3 Train ,Test ,Evaluate and save Model

In [11]:
```
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
```

In [12]:
```
history = model.fit(train_data, validation_data= val_data, batch_size=32, epochs = 100, callbacks=[early])
```

In [13]:
```
model.evaluate(test_data)
```

Out[13]:
```
[0.23725448548793793, 0.914893627166748]
```

In [14]:
```
y_pred =  model.predict(test_data)
y_pred =  np.argmax(y_pred,axis=1)
len(test_data)
test_data.classes
y_pred
```

In [15]:
```
from sklearn.metrics import confusion_matrix, classification_report, roc_curve
```

CONFUSION MATRIX

In [16]:
```
def plot_confusion_matrix (cm, classes,normalize=False,title='Confusion matrix',cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [17]:
```
import itertools
cm = confusion_matrix(y_true = test_data.classes, y_pred = y_pred)
plot_confusion_matrix(cm, SoilType, title= 'confusion matrix')
```

CLASSIFICATION REPORT

In [18]:
```
print(classification_report(test_data.classes, y_pred))
```

ACCURACY vs EPOCH - GRAPH

In [20]:
```
def plot_hist(hist):
    plt.plot(hist.history["accuracy"])
    plt.plot(hist.history["val_accuracy"])
    plt.title("model accuracy")
    plt.ylabel("accuracy")
    plt.xlabel("epoch")
    plt.legend(["train", "validation"], loc="upper left")
    plt.show()
```

In [21]:
```
plot_hist(history)
```

LOSS vs EPOCH - GRAPH

In [22]:
```
def plot_hist_loss(hist):
    plt.plot(hist.history["loss"])
    plt.plot(hist.history["val_loss"])
    plt.title("model loss")
    plt.ylabel("loss")
    plt.xlabel("epoch")
    plt.legend(["train", "validation"], loc="upper left")
    plt.show()
```

In [23]:
```
plot_hist_loss(history)
```

9.2.4 SOIL Classification

MODULE 4

IMPORTING REQUIRED PACKAGE

In [1]:
```
import splitfolders
splitfolders.ratio("Soil_Dataset/Train", output="Soil_Dataset/data/", seed=1337, ratio=(.8, .2), group_prefix=None)
```

In [2]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import sys
sys.setrecursionlimit(10000)
```

LABEL THE CLASSIFICATION TYPES

In [3]:
```
SoilType = ['Alluvial_Soil', 'Black_Soil', 'Clay_Soil', 'Red_Soil']
```

LOADING THE SAVED MODEL

The model which we already saved in .h5 extension is loaded

In [5]:
```
soil_model = load_model('models/soil_model2.h5')
```

IMAGE AS INPUT AND CLASSIFICATION

In [10]:
```
image_path = "t_case-1.jpg"

image = load_img(image_path,target_size=(224,224))
image = img_to_array(image)
image = image/255
image = np.expand_dims(image,axis=0)

result = np.argmax(soil_model.predict(image))
print("Classification is :", SoilType[result])
```

9.2.5 Plant disease Data set Pre-processing

MODULE 5

In [1]:
```
import splitfolders
```

In [2]:
```
splitfolders.ratio("leaf/train", output="leaf/data/", seed=1337, ratio=(.8, .2), group_prefix=None)
```

In [2]:
```
from tensorflow import keras
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
from keras import layers
from tensorflow.keras.applications.xception import preprocess_input
from tensorflow.keras.preprocessing.image import load_img , img_to_array , ImageDataGenerator
import sys
sys.setrecursionlimit(10000)
```

In [4]:
```
healthType = ['Apple___Apple_scab',
 'Apple___Black_rot',
 'Apple___Cedar_apple_rust',
 'Apple___healthy',
 'Blueberry___healthy',
 'Cherry_(including_sour)___Powdery_mildew',
 'Cherry_(including_sour)___healthy',
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
 'Corn_(maize)___Common_rust_',
 'Corn_(maize)___Northern_Leaf_Blight',
 'Corn_(maize)___healthy',
 'Grape___Black_rot',
 'Grape___Esca_(Black_Measles)',
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
 'Grape___healthy',
 'Orange___Haunglongbing_(Citrus_greening)',
 'Peach___Bacterial_spot',
 'Peach___healthy',
 'Pepper,_bell___Bacterial_spot',
 'Pepper,_bell___healthy',
 'Potato___Early_blight',
 'Potato___Late_blight',
 'Potato___healthy',
 'Raspberry___healthy',
 'Soybean___healthy',
 'Squash___Powdery_mildew',
 'Strawberry___Leaf_scorch',
 'Strawberry___healthy',
 'Tomato___Bacterial_spot',
 'Tomato___Early_blight',
 'Tomato___Late_blight',
 'Tomato___Leaf_Mold',
 'Tomato___Septoria_leaf_spot',
 'Tomato___Spider_mites Two-spotted_spider_mite',
 'Tomato___Target_Spot',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
 'Tomato___Tomato_mosaic_virus',
 'Tomato___healthy']
```

In [5]:
```
DATA_PATH = 'leaf/'
```

In [6]:
```
#import train data
train_datagen = ImageDataGenerator(rescale=1/255,
                                   shear_range = 0.3,
                                   zoom_range = 0.3,horizontal_flip = True,
                                   vertical_flip =  True ,
                                   rotation_range=60)


train_data = train_datagen.flow_from_directory(DATA_PATH+'train',
                                               target_size = (244, 244),
                                               class_mode='sparse',
                                               shuffle=True,seed=1)
```

```
Found 70295 images belonging to 38 classes.
```

In [7]:
```
#import val data

val_datagen = ImageDataGenerator(rescale = 1/255)
val_data = val_datagen.flow_from_directory(DATA_PATH+'data/val',
                                                       target_size=(244,244),
                                                       class_mode='sparse',
                                                       shuffle=True,seed=1)
```

```
Found 14076 images belonging to 38 classes.
```

In [8]:
```
# import test data


test_datagen = ImageDataGenerator(rescale = 1/255)
test_data = test_datagen.flow_from_directory(DATA_PATH+'test_1',
                                                       target_size=(244,244),
                                                       class_mode='sparse',
                                                       shuffle=False,seed=1)
```

```
Found 56219 images belonging to 38 classes.
```

9.2.6 Building CNN MODEL for Plant disease classification

MODULE 6

In [9]:
```
# Defining Cnn
model = tf.keras.models.Sequential([
  layers.Conv2D(32, 3, activation='relu',input_shape=(244,244,3)),
  layers.MaxPooling2D(),
  layers.Conv2D(64, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.3),
  layers.Conv2D(128, 3, activation='relu'),
  layers.MaxPooling2D(),
  layers.Dropout(0.2),
  layers.Flatten(),
  layers.Dense(256, activation='relu'),
  layers.Dropout(0.15),
  layers.Dense(128, activation='relu'),
  layers.Dropout(0.1),
  layers.Dense(38, activation= 'softmax')
])

print(model.input.shape)
```

```
(None, 244, 244, 3)
```

In [10]:
```
model.compile(optimizer='adam',loss='sparse_categorical_crossentropy', metrics=['accuracy'])
```

9.2.7 Train ,Test ,Evaluate and save Model

MODULE 7

In [11]:
```
early = tf.keras.callbacks.EarlyStopping(monitor='val_loss',patience=5)
```

In [12]:
```
history = model.fit(train_data, validation_data= val_data, batch_size=32, epochs = 100, callbacks=[early])
```

In [16]:
```
y_pred =  modell.predict(test_data)
```

In [15]:
```
len(test_data)
test_data.classes
```

Out[15]:
```
array([ 0,  0,  0, ..., 37, 37, 37])
```

In [17]:
```
#y_pred =  modell.predict(test_data)
y_pred =  np.argmax(y_pred,axis=1)
len(test_data)
test_data.classes
y_pred
```

Out[17]:
```
array([ 0,  0,  0, ..., 37, 33, 37], dtype=int64)
```

In [18]:
```
from sklearn.metrics import confusion_matrix, classification_report, roc_curve
```

In [24]:
```
def plot_confusion_matrix (cm, classes,normalize=False,title='Confusion matrix',cmap=plt.cm.Blues):
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

    print(cm)

    thresh = cm.max() / 2.
    for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
        plt.text(j, i, cm[i, j],
            horizontalalignment="center",
            color="white" if cm[i, j] > thresh else "black")

    #plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
```

In [25]:
```
import itertools
cm = confusion_matrix(y_true = test_data.classes, y_pred = y_pred)
plot_confusion_matrix(cm, healthType, title= 'confusion matrix')
```

In [21]:
```
print(classification_report(test_data.classes, y_pred))
```

9.2.8 Plant disease identification

MODULE 8

In [1]:
```
import splitfolders

splitfolders.ratio("leaf/train", output="leaf/data/", seed=1337, ratio=(.8, .2), group_prefix=None) # default values
```

IMPORTING MODULES

In [2]:
```
import numpy as np
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing.image import load_img, img_to_array
import sys
sys.setrecursionlimit(10000)
```

LABEL THE CLASSIFICATION

In [3]:
```
healthType = ['Apple___Apple_scab',
 'Apple___Black_rot',
 'Apple___Cedar_apple_rust',
 'Apple___healthy',
 'Blueberry___healthy',
 'Cherry_(including_sour)___Powdery_mildew',
 'Cherry_(including_sour)___healthy',
 'Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot',
 'Corn_(maize)___Common_rust_',
 'Corn_(maize)___Northern_Leaf_Blight',
 'Corn_(maize)___healthy',
 'Grape___Black_rot',
 'Grape___Esca_(Black_Measles)',
 'Grape___Leaf_blight_(Isariopsis_Leaf_Spot)',
 'Grape___healthy',
 'Orange___Haunglongbing_(Citrus_greening)',
 'Peach___Bacterial_spot',
 'Peach___healthy',
 'Pepper,_bell___Bacterial_spot',
 'Pepper,_bell___healthy',
 'Potato___Early_blight',
 'Potato___Late_blight',
 'Potato___healthy',
 'Raspberry___healthy',
 'Soybean___healthy',
 'Squash___Powdery_mildew',
 'Strawberry___Leaf_scorch',
 'Strawberry___healthy',
 'Tomato___Bacterial_spot',
 'Tomato___Early_blight',
 'Tomato___Late_blight',
 'Tomato___Leaf_Mold',
 'Tomato___Septoria_leaf_spot',
 'Tomato___Spider_mites Two-spotted_spider_mite',
 'Tomato___Target_Spot',
 'Tomato___Tomato_Yellow_Leaf_Curl_Virus',
 'Tomato___Tomato_mosaic_virus',
 'Tomato___healthy']
```

LOAD THE SAVED MODEL

In [4]:
```
leaf_model = load_model('models/leaf-model.h5')
```

TEST CASES

INPUT : leaf image

OUTPUT : disease classification

In [5]:
```
image_path = "leaf/dataset/test/Apple___Apple_scab/test1.JPG"

image = load_img(image_path,target_size=(224,224))
image = img_to_array(image)
image = image/255
image = np.expand_dims(image,axis=0)

result = np.argmax(leaf_model.predict(image))
print("Classification is :", healthType[result])
```

10. RESULTS AND COMPARISON

10.1 ACCURACY VS EPOCH

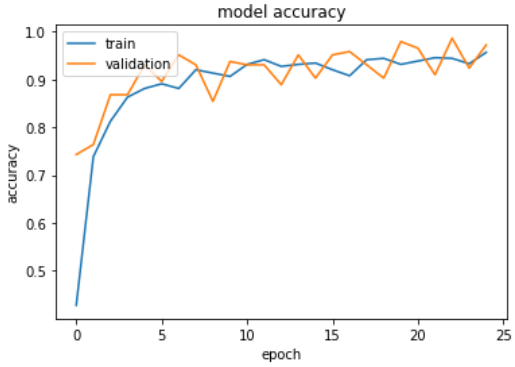

Fig. 12 : Accuracy vs Epoch graph for the model

Fig 12 shows , A graph between accuracy and epoch is plotted. The accuracy of the model raises and remains almost unchanged after a particular epoch. The accuracy of the model during training and validating comes out to be 95% and 97%.

10.2 LOSS VS EPOCH

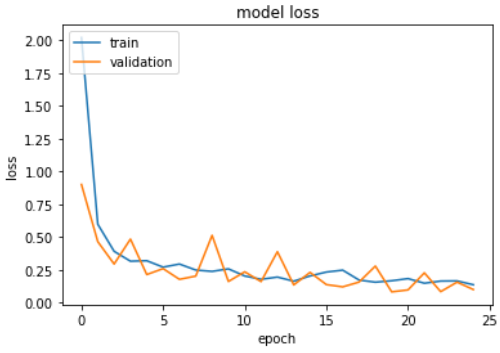

Fig. 13 : Loss vs Epoch graph for the model

A graph between Loss and epoch is plotted in fig 13. The Loss in the model lowers and remains almost unchanged after a particular epoch. The Loss in the model is lowered and an exceptional loss less than 10% is achieved.

10.3 TABLE OF INFERENCE

	Epoch 25	**Epoch 50**	**Epoch 75**	**Early Callback**	**Dropout Layer – 0.4**
Train Accuracy	93%	95.8%	97%	89%	85%
Train Loss	11.9%	12.8%	4%	26%	36%
Precision	91%	91%	94%	87%	85%
Recall	90%	90%	94%	86%	82%
F1 score	91%	90%	94%	86%	83%
Result	Better than callback	Improvement with computation time	Overloaded with more computation	Decent result	More Loss

	Added layer	**Regularized**	**Low train data**	**Existing Model**	**Proposed model**
Train Accuracy	91%	82%	93%		92%
Train Loss	21%	26%	27%		11%
Precision	87%	88%	86%	87%	91%
Recall	86%	87%	84%	84%	90%
F1 score	85%	87%	84%		91%
Result	Train accuracy abruptly increases.	Underfitting	Overfitting		

10.4 COMPARING MODELS WITH ACCURACY AND LOSS

- **Epoch 25**

Though accuracy and loss reached a sideway trend after the 13th epoch in *Fig 14* it was moving up and down in the 14th and 15th epoch giving low accuracy in 14 and high accuracy in 15.

GRAPH

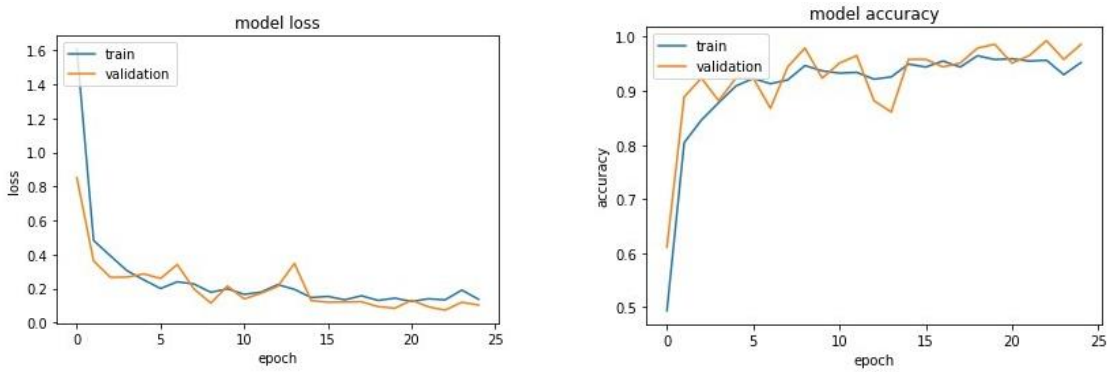

Fig. 14 : Accuracy vs Epoch and Loss vs Epoch graph for EPOCH 25

- **Epoch 50**

Fig 15 shows , When 50 epoch was set after a threshold point, the accuracy and loss was moving in a range bound sideways trend. However, the accuracy had small improvement after consuming a large computational time.

GRAPH

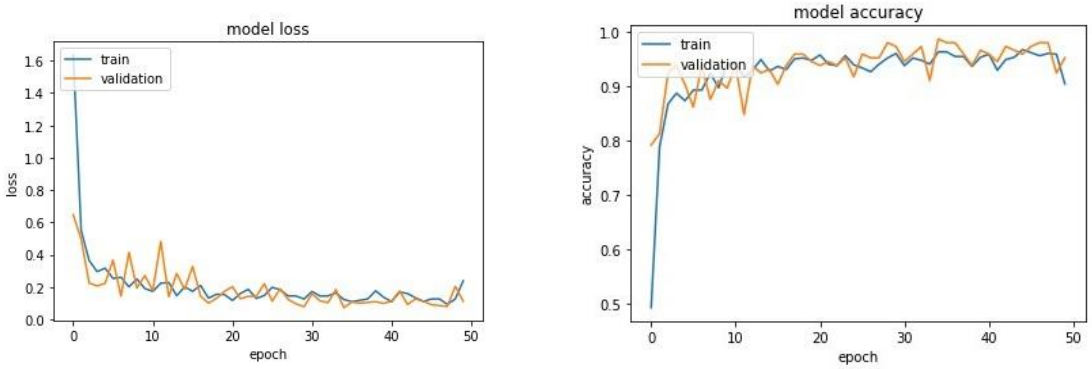

Fig. 15 : Accuracy vs Epoch and Loss vs Epoch graph for EPOCH 50

- **Epoch 75**

As seen in epoch 50, the same thing occurs here.But as more computational time is consumed to train data, train accuracy was far more greater than precision which is a sign of overfitting , which is seen in *fig 16*

GRAPH

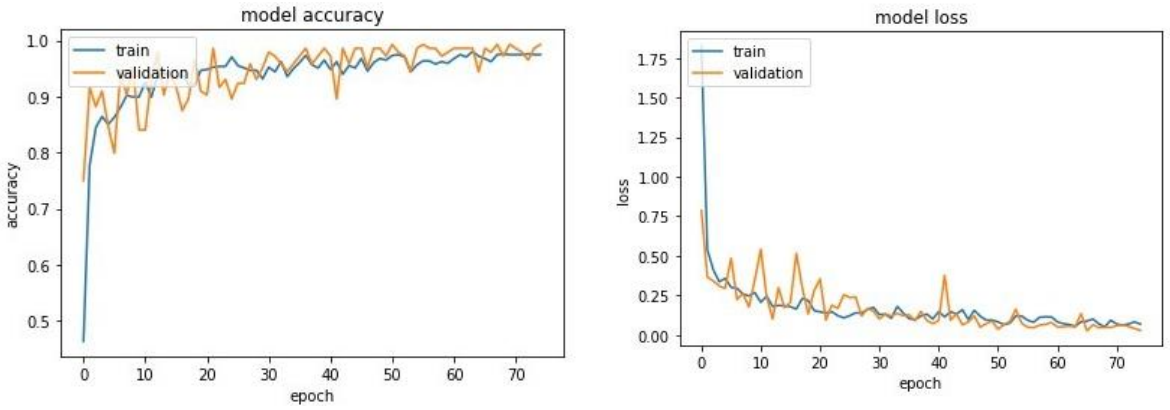

Fig. 16 : Accuracy vs Epoch and Loss vs Epoch graph for EPOCH 75

- **Early Call back**

This is a condition where epoch is terminated after the initiation of the first side wave--*fig 17*. In this, decent accuracy with less computational time is obtained which is way more better than higher epochs. In this, dropout regularizes the model to the very hard extent and prevents it from over fitting engulfing some amount of accuracy.

GRAPH

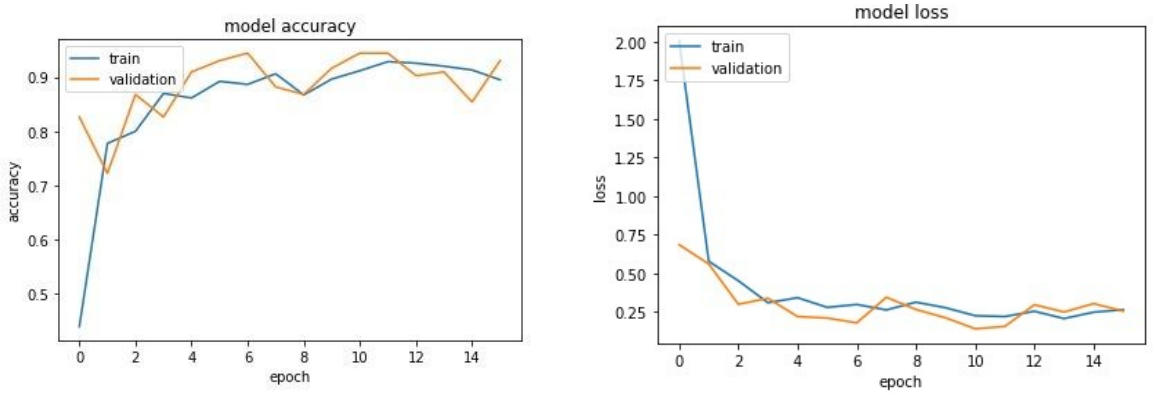

Fig. 17 : Accuracy vs Epoch and Loss vs Epoch graph for EARLY CALL BACK

- **Dropout Layer – 0.4**

In this, dropout regularizes the model to the very hard extend and prevent it from over fitting engulfing some amount of accuracy--*fig 18*

GRAPH

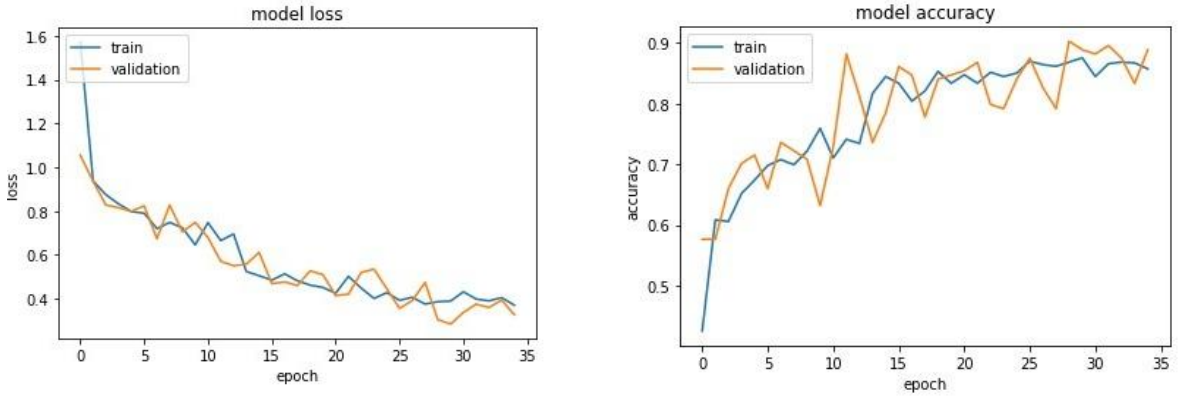

Fig. 18 : Accuracy vs Epoch and Loss vs Epoch graph for DROPOUT LAYER

- **Added layer**

If deeper layers are used, train accuracy will be higher than test accuracy as training in different levels of layers provide excellent learning experience --*Fig 19*. This makes precision poor. So Our model must be simple in order to balance bias and variance curves.

GRAPH

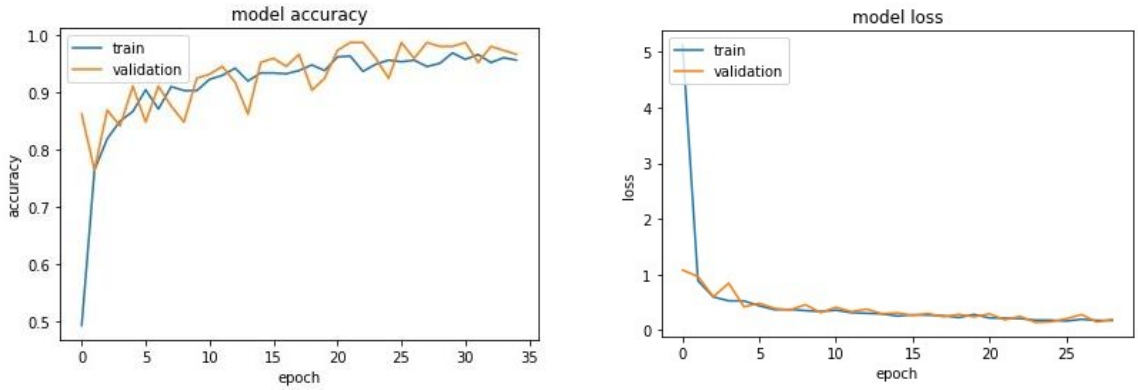

Fig. 19 : Accuracy vs Epoch and Loss vs Epoch graph for ADDED LAYER

- **Regularized**

As the model is so simple, the test accuracy goes above train accuracy in *Fig 20*. We must have balance layers in order to train a balanced model.

GRAPH

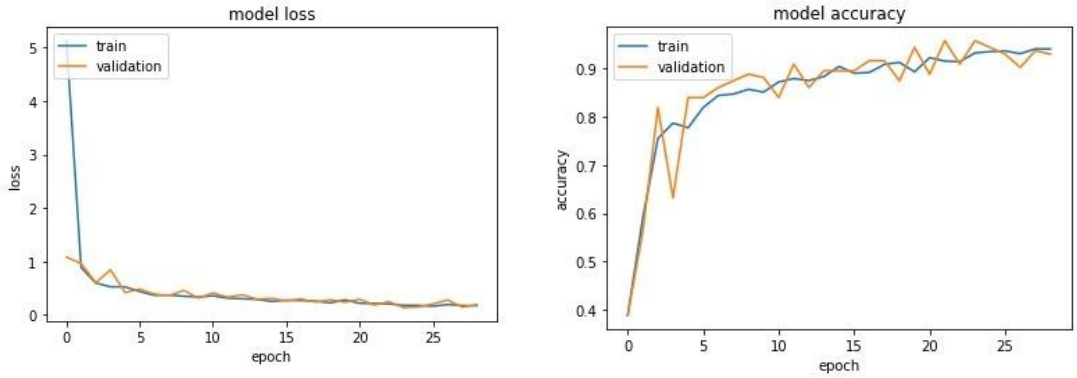

Fig. 20 : Accuracy vs Epoch and Loss vs Epoch graph for REGULARIZED

- **Low train data**

This is an example of overfitting as the train data is not enough to train the model and train accuracy goes up in *fig 21*. So, we must give more data for training.

GRAPH

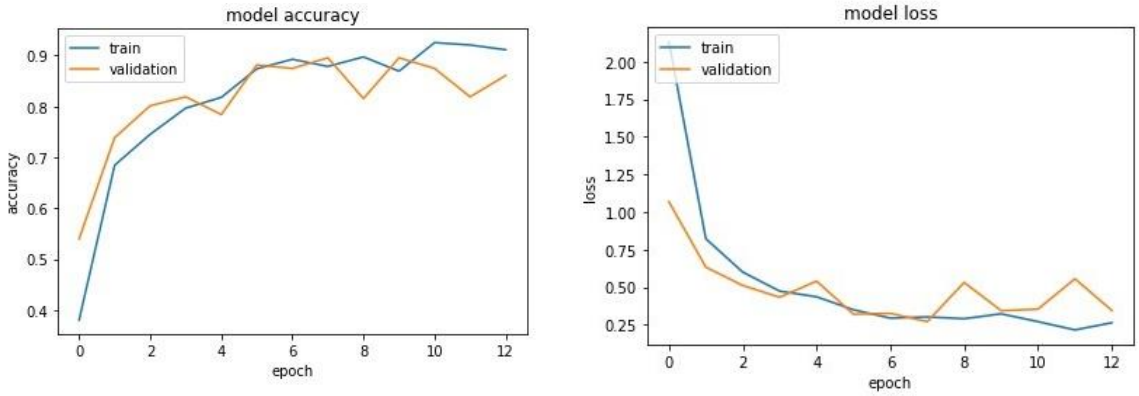

Fig. 21 : Accuracy vs Epoch and Loss vs Epoch graph for LOW TRAIN DATA

11. CONCLUSION

The project aims at creating a model that efficiently classifies the soil instances with higher accuracies. We have also incorporated the model to predict plant diseases according to the leaf image as input and disease is identified.

Soil prediction involves types of crop classifications and geographical attributes. It also aims at creating a system that processes the real-time soil data to predict the crops with higher accuracy. The proposed system was developed taking in mind the benefits of the farmers and agricultural sector .The developed system can detect disease in plants and also provide the remedy that can be taken against the disease. By proper knowledge of the disease and the remedy can be taken for improving the health of the plant .

This project has shown the application of an Artificial Neural Network for soil classification and disease identification using a program developed for this purpose. Using actual soil data, it provided an accurate way to classify soils according to series with an accuracy level of 93% and using plant disease dataset , we have achieved an accuracy of 95%. Such high accuracy together with the program developed may provide an important tool for farmers and government agriculture personnel for real world applications .

.

12. LIST OF REFERENCES

- Image-based plant disease identification by deep learning meta-architectures. Muhammad Hammad Saleem , Sapna Khanchi , Johan Potgieter , Khalid Mahmood Arif. Published - 27 October 2020

- Soil texture classification with 1d convolutional neural networks based on hyperspectral data F. M. Riese , S. Keller Published - 14 June 2019

- Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems, pp. 1097-1105.

- https://jupyter.org/documentation

- https://www.tensorflow.org/guide

- Meiqin Zhang, Shanqin Wang*, Shuo Li, Jing Yi,Peng Fu, 2019. Prediction and Map-making of SoilOrganic Matter of Soil Profile Based on Imaging Spectroscopy

- Cloutis, E. A., 1996. Review article hyperspectral geological remote sensing: evaluation of analytical techniques. International Journal of Remote Sensing 17(12), pp. 2215–2242.

13. APPENDIX A:

The undersigned acknowledge they have completed implementing the project “Soil classification and plant diseases identification using CNN” and agree with the approach it presents.

Signature: _____ **Date**: 17/05/2021

Name: Aakash K

Signature: _____ **Date**: 17/05/2021

Name: ASHWATH NARAYAN K S

14. APPENDIX B: REFERENCES

The following table summarizes the research papers referenced in this document.

Document Name and Version	Description	Location
Image-based plant disease identification by deep learning meta-architectures	Classifies the leaf image fed by the user using CNN with added layers and identify the disease.	https://www.mdpi.com/2223-7747/9/11/1451
Soil texture classification with 1d convolutional neural networks based on hyperspectral data	Takes in the soil image from the user and classifies soil texture	https://arxiv.org/abs/1901.04846
Imagenet classification with deep convolutional neural networks. In Advances in Neural Information Processing Systems	Takes in the image from the user and classify the soil according to the features extracted from the dataset.	https://kr.nvidia.com/content/tesla/pdf/machine-learning/imagenet-classification-with-deep-convolutional-nn.pdf

15. APPENDIX C: KEY TERMS

The following table provides definitions for terms relevant to this document.

Term	Definition
CNN	CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean **fully connected networks**, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks make them **prone to overfitting data**.
Softmax	**softmax** function, also known as **softargmax** or normalized exponential function, is a generalization of the logistic function to **multiple dimensions**. It is used in multinomial logistic regression and is often used as the last **activation function of a neural network** to normalize the output of a network to a probability distribution over predicted output classes, based on Luce's choice axiom.
F1 score	The F1 Score is the **2*((precision*recall)/(precision+recall))**. It is also called the F Score or the **F Measure**. Put another way, the F1 score conveys the **balance** between the **precision and the recall**.