PLANT DISEASE IDENTIFICATION

A Online Summer Internship project report submitted to the JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD in partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted By

D. SHASHIKANTH (18071A0572)

CH. SHREYA REDDY(18071A0569)

P. SAMHITHA(18071A05A2)

R. UMESH CHANDRA (18071A05P6)

Under the Guidance Of

Dr. C KIRANMAI

COMPUTER SCIENCE AND ENGINEERING DEPARTMENT

VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute, NAAC Accredited With 'A++' Grade, NBA Accredited, Approved by AICTE, New Delhi, Affiliated to JNTUH)

VALLURUPALLI NAGESWARA RAO VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY

(An Autonomous Institute)

Hyderabad-500090

CERTIFICATE

This is to certify that **D. Shashikanth (18071A0572), CH. Shreya Reddy (18071A0569), P. Samhitha(18071A05A2) , R. Umesh Chandra(18071A05P6)** have successfully completed their project work at CSE Department of VNR VJIET, Hyderabad entitled **"PLANT DISEASE IDENTIFICATION"** in partial fulfillment of the requirements for the award of B. Tech degree during the academic year 2020-2021.

Project Guide	Head of the Department
Dr. C Kiranmayi	**Mrs. B. V. Kiranmayee**
Professor	Associate Professor &
Department of CSE	Head of the Department CSE
VNR VJIET	VNR VJIET

DECLARATION

This is to certify that the project work entitled **"PLANT DISEASE IDENTIFICATION "** submitted in VNR Vignana Jyothi Institute of Engineering & Technology in partial fulfillment of requirement for the award of Bachelor of Technology in Computer Science and Engineering is a bonafide report of the work carried out by us under the guidance and supervision of Dr. C Kiranmayi (Professor), Department of CSE, VNRVJIET. To the best of our knowledge, this report has not been submitted in any form to any university or institution for the award of any degree or diploma.

D. SHASHIKANTH	**CH. SHREYA REDDY**	**P. SAMHITHA**	**R. UMESH CHANDRA**
(18071A0572)	(18071A0569)	(18071A05A2)	(18071A05P6)
III B.Tech-CSE,	III B.Tech-CSE,	III B.Tech-CSE,	III B.Tech-CSE,
VNR VJIET	VNR VJIET	VNR VJIET	VNRVJIET

ACKNOWLEDGEMENT

Behind every achievement lies an unfathomable sea of gratitude to those who activated it, without it would ever never have come into existence. To them we lay the words of gratitude imprinting within us.

We are indebted to our venerable principal **Dr. C. D. Naidu** for this unflinching devotion, which led us to complete this project. The support, encouragement given by him and his motivation lead us to complete this project.

We express our thanks to internal guide **Dr. C Kiranmayi** and also Head of the department **Mrs. B. V. Kiranmayee** for having provided us a lot of facilities to undertake the project work and guide us to complete the project.

We express our sincere thanks to our faculty of the department of **Computer Science and Engineering** and the remaining members of our college **VNR VIGNANA JYOTHI INSTITUTE OF ENGINEERING AND TECHNOLOGY** who extended their valuable support in helping us to complete the project in time.

D Shashikanth (18071A0572)

CH Shreya Reddy (18071A0569)

P Samhitha (18071A05A2)

R Umesh Chandra (18071A05P6)

Problem Statement :

Rice plant disease identification using Convolutional Neural Networks and Keras .

Abstract:

Now-a-days we can see various kinds of plant diseases which apparently causes high risk to farming and farmers however their quick distinguishing proof stays troublesome in numerous parts of the world. The field of leaf-based image classification is found to be very useful in emerging accurate techniques to solve this problem. In our project we used Machine learning and Convolutional Neural Networks to distinguish between the two diseases namely Rice Blast and Leaf streak (seen in paddy) from the datasets created. Our project includes various phases of implementation namely dataset creation, feature extraction, training the classifier and classification. We trained our dataset using CNNs. We also used Keras and Tensorflow to implement our model in CNN. Overall, using machine learning and CNNs to train the data sets available publicly gives us a clear way to distinguish between the two diseases present in paddy.

INDEX

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction to Machine Learning

Machine Learning is a tender which offers frameworks with capability to perfunctorily take in and enhance from information it secures without really being modified to do as such. This idea accentuates on the advancement of projects that can consequently breakdown information and utilize it to absorb for themselves to improve decisions. This idea is critical to the area of Artificial Intelligence.

Learning initiates with data and impression of this data, for instance, coordinate finding, or direction, or representations, to search for designs, present regularly in material and at last distinguish between poor and enhanced judgments later with respect to cases that we give. The primary goal is to allow the PCs to learn ordinarily without human interpolation and modify exercises as indicated by the required.

ML enables programming applications to wind up more precise in anticipating results without being expressly modified. The essential introduce of ML is to fabricate calculations that can get input information and utilize measurable going-over to get ahead a yield an incentive inside a satisfactory range.

1.1.1 Some machine learning methods

ML algorithms are categorized as **supervised** or **unsupervised.**

i) **Supervised Learning calculations** can remove those that have been 30 recognized previously to fresh information utilizing named cases to foresee future occasions. Beginning since examination of a preparation data set, the calculation delivers a gathered capacity to make forecasts about the yield esteems. The framework can give efforts to any first-hand contribution after suitable preparing. The learning calculation can likewise contrast its yield and the right, expected income and realize blunders with a specific end objective to adjust the model as needs be.

ii) In differentiate, **unsupervised ML** deviousness are developed when the data used to prepare is neither arranged nor marked. Unsupervised learning envisages frameworks can surmise a capacity to portray a concealed structure from unlabeled information

iii) **Semi-regulated ML calculations** fall some place in the middle of managed and unsupervised learning, since they utilize both unlabeled information as well as marked one for preparing measures of some labeled information and a lot of unlabeled information. The frameworks that utilization this strategy can impressively enhance learning precision. Typically, semi-administered learning is picked when the obtained named information requires gifted and pertinent assets with a specific end goal to prepare it/gain from it.

1.2 Introduction to Convolutional Neural Networks

A Convolutional Neural Network (ConvNet/CNN) is a Deep Learning algorithm which can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The pre-processing required in a ConvNet is much lower as compared to other classification algorithms. While in primitive methods filters are hand-engineered, with enough training, ConvNets have the ability to learn these filters/characteristics.

The architecture of a ConvNet is analogous to that of the connectivity pattern of Neurons in the Human Brain and was inspired by the organization of the Visual Cortex. Individual neurons respond to stimuli only in a restricted region of the visual field known as the Receptive Field. A collection of such fields overlap to cover the entire visual area.

Why ConvNets over Feed-Forward Neural Nets?

An image is nothing but a matrix of pixel values, right? So why not just flatten the image (e.g. 3x3 image matrix into a 9x1 vector) and feed it to a Multi-Level Perceptron for classification purposes? Uh.. not really.

In cases of extremely basic binary images, the method might show an average precision score while performing prediction of classes but would have little to no accuracy when it comes to complex images having pixel dependencies throughout.

.

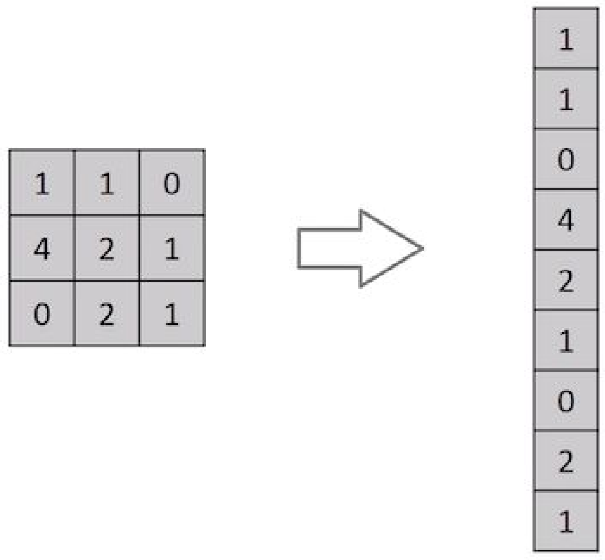


Fig 1.1 :Flattening of a 3x3 image matrix into a 9x1 vector

A ConvNet is able to **successfully capture the Spatial and Temporal dependencies** in an image through the application of relevant filters. The architecture performs a better fitting to the image dataset due to the reduction in the number of parameters involved and reusability of weights. In other words, the network can be trained to understand the sophistication of the image better.

Input Image

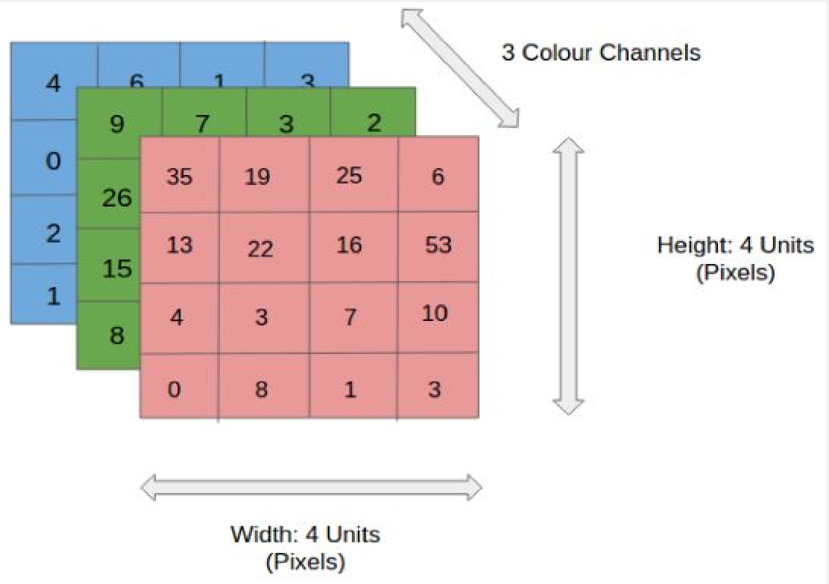


Fig 1.2: 4x4x3 RGB Image

In the figure, we have an RGB image which has been separated by its three color planes — Red, Green, and Blue. There are a number of such color spaces in which images exist — Grayscale, RGB, HSV, CMYK, etc.

You can imagine how computationally intensive things would get once the images reach dimensions, say 8K (7680×4320). The role of the ConvNet is to reduce the images into a form which is easier to process, without losing features which are critical for getting a good prediction. This is important when we are to design an architecture which is not only good at learning features but also is scalable to massive datasets.

1.3 Existing System

Paddy leaf diseases were identified by capturing the image of the disease-affected crop.
1.Using image enhancement and image processing. The image is first captured, processed and enhanced and then converted from RGB color image to gray image and then extracted to histogram.
2. Using multi-level color image thresholding for the disease diagnosis system of Rice Blast disease.
3. Using image acquisition and pre-processing to remove noise and enhance and then converted to a binary image.

1.4 Proposed System

We used Machine learning and Convolutional Neural Networks to distinguish between the two diseases namely Rice Blast and Leaf streak (seen in paddy). We also used Keras and Tensorflow to pre-process, train and classify our datasets.

CHAPTER 2

LITERATURE SURVEY

2.1 Python API's & Libraries required

2.1.1 Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatterplots, etc., with just a few lines of code.

2.1.2 Sklearn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy.

2.1.3 NumPy

NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation etc

2.1.4 Pandas

Pandas is a Python package providing fast, flexible, and expressive data structures designed to make working with "relational" or "labelled" data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, real world data analysis in Python.

Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language.

2.1.5 Tensor Flow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML powered applications.

TensorFlow offers multiple levels of abstraction so you can choose the right one for your needs. Build and train models by using the high-level Keras API, which makes getting started with TensorFlow and machine learning easy.

If you need more flexibility, eager execution allows for immediate iteration and intuitive debugging. For large ML training tasks, use the Distribution Strategy API for distributed training on different hardware configurations without changing the model definition.

2.1.6 Keras

Keras is an API designed for human beings, not machines. Keras follows best practices for reducing cognitive load: it offers consistent & simple APIs, it minimizes the number of user actions required for common use cases, and it provides clear & actionable error messages. It also has extensive documentation and developer guides.

2.2 Python IDE

Integrated Development Environment (IDE) for Python has been bundled with the default implementation of the language. Its main features are – Multi window text editor with syntax highlighting, autocompletion, smart indent and others, python shell with syntax highlighting, integrated debugger with stepping, persistent breakpoints, and call stack visibility.

CHAPTER 3

ALGORITHM DESCRIPTION

3.1 The Shortcoming of Train-Test Split

Imagine you have a dataset with 5000 rows. The train_test_split function has an argument for test size that you can use to decide how many rows go to the training set and how many go to the test set. The larger the test set, the more reliable your measures of model quality will be. At an extreme, you could imagine having only 1 row of data in the test set. If you compare alternative models, which one makes the best predictions on a single data point will be mostly a matter of luck.

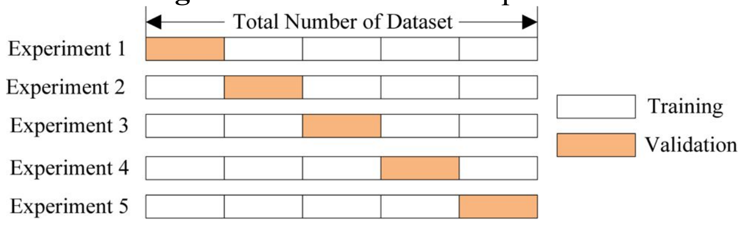
You will typically keep about 20% as a test dataset. But even with 1000 rows in the test set, there's some random chance in determining model scores. A model might do well on one set of 1000 rows, even if it would be inaccurate on a different 1000 rows. The larger the test set, the less randomness (aka "noise") there is in our measure of model quality.

But we can only get a large test set by removing data from our training data, and smaller training datasets mean worse models. In fact, the ideal modelling decisions on a small dataset typically aren't the best modelling decisions on large datasets.

3.2 The Cross-Validation Procedure

In cross-validation, we run our modelling process on different subsets of the data to get multiple measures of model quality. For example, we could have 5 **folds** or experiments. We divide the data into 5 pieces, each being 20% of the full dataset.

Fig 3.1: Cross Validation split



We run an experiment called experiment 1 which uses the first fold as a holdout set, and everything else as training data. This gives us a measure of model quality based on a 20% holdout set, much as we got from using the simple train-test split.

We then run a second experiment, where we hold out data from the second fold (using everything except the 2nd fold for training the model.) This gives us a second estimate of model quality. We repeat this process, using every fold once as the holdout. Putting this together, 100% of the data is used as a holdout at some point.

Returning to our example above from train-test split, if we have 5000 rows of data, we end up with a measure of model quality based on 5000 rows of holdout (even if we don't use all 5000 rows simultaneously.

3.3 Applications of Cross Validation

The cross-validation technique can be used to compare the performance of different machine learning models on the same data set. To understand this point better, consider the following example. Suppose you want to make a classifier for the MNIST data set, which consists of hand-written numerals from 0 to 9. You are considering using either *K Nearest Neighbours (KNN)* or *Support Vector Machine (SVM)*. To compare the performance of the two machine learning models on the given data set, you can use cross validation.

3.4 Advantages of cross-validation

i) More accurate estimate of out-of-sample accuracy.

ii) More “efficient” use of data as every observation is used for both training and testing.

3.5 Sigmoid Function

A sigmoid function is a type of activation function, and more specifically defined as a squashing function. Squashing functions limit the output to a range between 0 and 1, making these functions useful in the prediction of probabilities.

The name Sigmoidal comes from the Greek letter Sigma, and when graphed, appears as a sloping "S" across the Y-axis. A sigmoidal function is a type of logistic function and purely refers to any function that retains the "S" shape, such as tanh(x). Where a traditional sigmoidal function exists between 0 and 1, tanh(x) follows a similar shape, but exists between 1 and -1. On its own, a sigmoidal function is also differentiable, meaning we can find the slope of the sigmoid curve, at any two points.

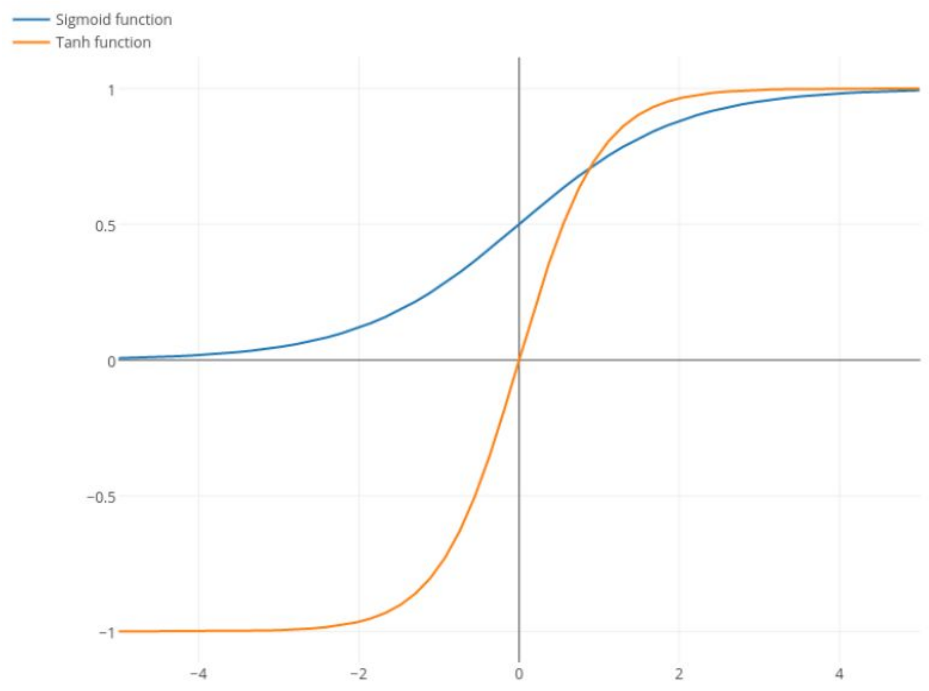


Fig 3.2 : Sigmoid Function

3.6 Sequential Model

Sequential is the easiest way to build a model in Keras. It allows you to build a model layer by layer. Each layer has weights that correspond to the layer the follows it. We use the 'add()' function to add layers to our model. We will add two layers and an output layer.

Sequential model is appropriate for a plain stack of layers where each layer has one input and one output tensor .

A Sequential model is **not appropriate** when:

- Your model has multiple inputs or multiple outputs
- Any of your layers has multiple inputs or multiple outputs
- You need to do layer sharing
- You want non-linear topology (e.g. a residual connection, a multi-branch model)

CHAPTER-4

SYSTEM-ANALYSIS

4.1 System Requirements

4.1.1 Software Requirements

a) Python

Python is a deciphered language Guido van Rossum, Python has a diagram hypothesis that complements code decipherability, and a sentence structure that empowers programming architects to express thoughts in less lines of code noticeably using imperative whitespace. It gives builds up that engage little immense. Incorporates a kind modified organization. Reinforces different perfect models, masterminded, essential, useful and, and a huge and exhaustive.

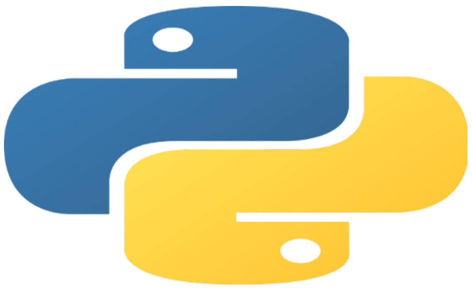


Fig 4.1 Python

b) Jupyter Notebook

The Jupyter Note pad is an open-source web application that allows you to make and offer reports that cover live code, conditions, observations and account content. Utilizations include: information cleaning and change, numerical reproduction, factual demonstrating, information representation, machine learning, and significantly more. The Scratch pad has bolster for more than 40 programming dialects, including Python, R, Julia, and Scala. Note pads can be imparted to others using email, Dropbox, GitHub and the Jupyter Note pad Watcher.

4.1.2 Hardware Requirements

RAM: 4GB and Higher

Processor: Intel i3& above

Hard Disk:10GB Minimum

4.2 System Architecture

a) Importing Dataset

In this we need to Load the dataset (question) which contains the required information.

b) Data preprocessing

Text preprocessing include the following

- Reshaping the Images into required size
- Data Augmentation
- Splitting the Data into Train and validation sets
- Rescaling
- Train the dataset

c) Apply CNN techniques

We apply Convolutional Neural Networks here for image recognition

d)Test dataset

We test the data over trained model it predicts whether the question pairs are duplicate or not.

e) Display results

We plot our confusion matrix. This shows the efficiency of algorithm

CHAPTER 5

MODULES DESCRIPTION

5.1 Data Preprocessing

Data preprocessing is a vital step for analysis of data. It involves conversion of raw data to require or useful format. Data Preprocessing is important because real world data consists of errors and is always inconsistent.

5.2 Feature Extraction

Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy. Many machine learning practitioners believe that properly optimized feature extraction is the key to effective model construction.

5.3 Training the model

The data collected is divided into two sets i.e. training set and testing set. The data to which the model is fitted is known as training data.The process of training a model requires the help of an algorithm which learns from the training data. We have different algorithms in Machine Learning ,Deep Learning and Neural Networks. In our project we are using machine learning algorithm MLP to train the model and then compare the accuracies they provide when test data is provided.

5.4 Testing the model

To test the accuracy of the model we use Testing mechanism. The data collected is divided into training and testing data. Data is tested by using testing data set and new data sets and comparing the result in form of accuracy, precision and recall.Testing the model in our project involves giving the test data, extracting the features of test data.

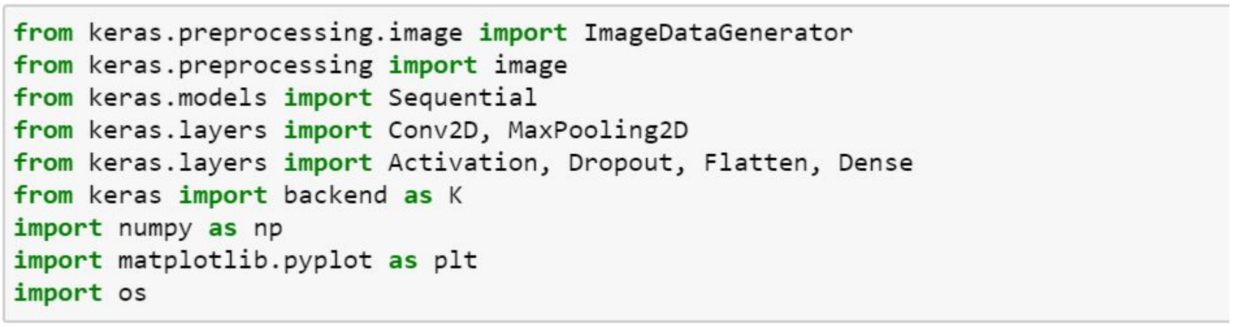
CHAPTER 6

IMPLEMENTATION

6.1 Code for importing modules and loading dataset

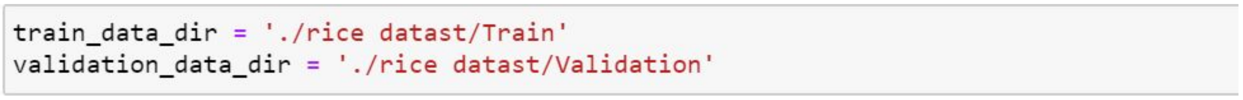
6.1.1 Import modules code

Importing the required modules and libraries to split the dataset, to apply different classification techniques and displaying the result. We here have different libraries to import and use .



```
from keras.preprocessing.image import ImageDataGenerator
from keras.preprocessing import image
from keras.models import Sequential
from keras.layers import Conv2D, MaxPooling2D
from keras.layers import Activation, Dropout, Flatten, Dense
from keras import backend as K
import numpy as np
import matplotlib.pyplot as plt
import os
```

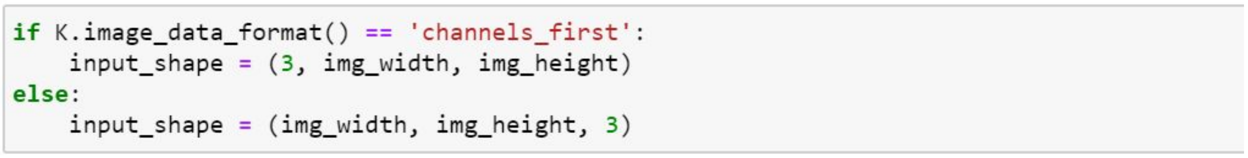
6.1.2 Code for loading dataset



```
train_data_dir = './rice datast/Train'
validation_data_dir = './rice datast/Validation'
```

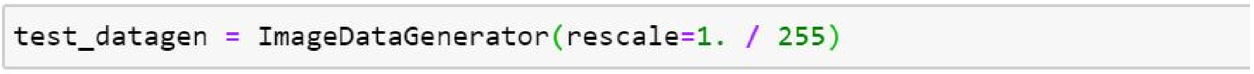
6.2 Data preprocessing and Data augmentation

In this block of code we will set the naming convention of the images



```
if K.image_data_format() == 'channels_first':
    input_shape = (3, img_width, img_height)
else:
    input_shape = (img_width, img_height, 3)
```

In this block we will rescale the images in order to reduce the computational costs



```
test_datagen = ImageDataGenerator(rescale=1. / 255)
```

As we have a smaller dataset , we will use Data augmentation method to increase the size of the data by a number of random transformations, so that our model would never see twice the exact same picture. This helps prevent overfitting and helps the model generalize better.

```
train_datagen = ImageDataGenerator(
    rescale=1. / 255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)
```

6.3 Data Preparation

We use .flow_from_directory() to generate batches of image data (and their labels) directly from our jpgs in their respective folders.

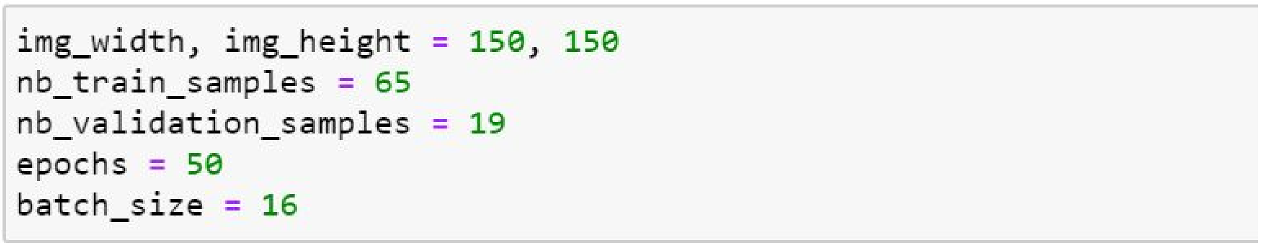
```
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')

validation_generator = test_datagen.flow_from_directory(
    validation_data_dir,
    target_size=(img_width, img_height),
    batch_size=batch_size,
    class_mode='binary')
```

Output:

```
Found 65 images belonging to 2 classes.
Found 19 images belonging to 2 classes.
```

This means we have two sets of data . One is training set and other is Validation set . Each set contain two classes . Training set contain 65 images and Validation set contain 19 images .

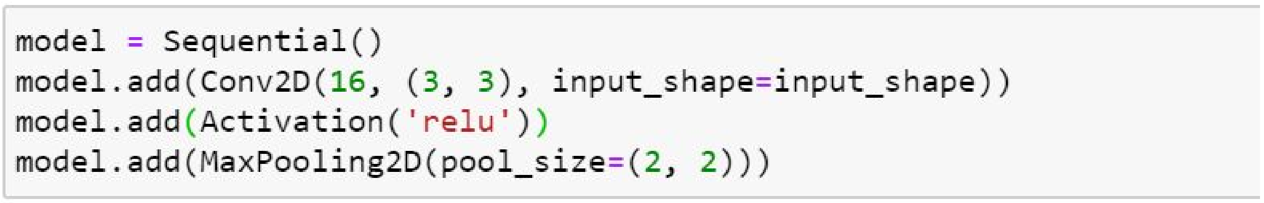
We will set the height and width of image to 150 and number of epochs to 50.Now we will use this data and assign them to variables to use it later .



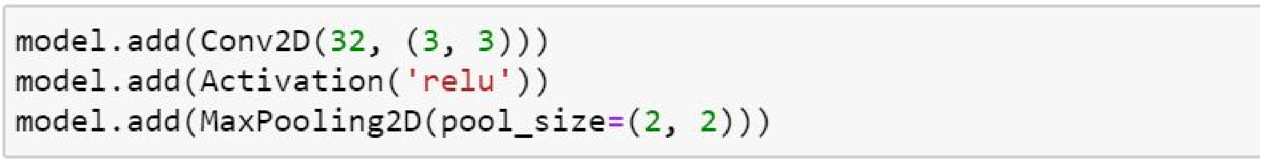
```
img_width, img_height = 150, 150
nb_train_samples = 65
nb_validation_samples = 19
epochs = 50
batch_size = 16
```

6.4 Building the Convolutional neural network with Keras

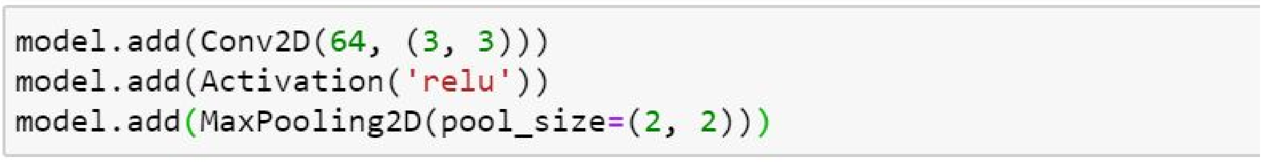
the below block of code will build the first layer of convolutional network with 16 image filters in it . Then as the layer progress we will increase the number of filters in it .



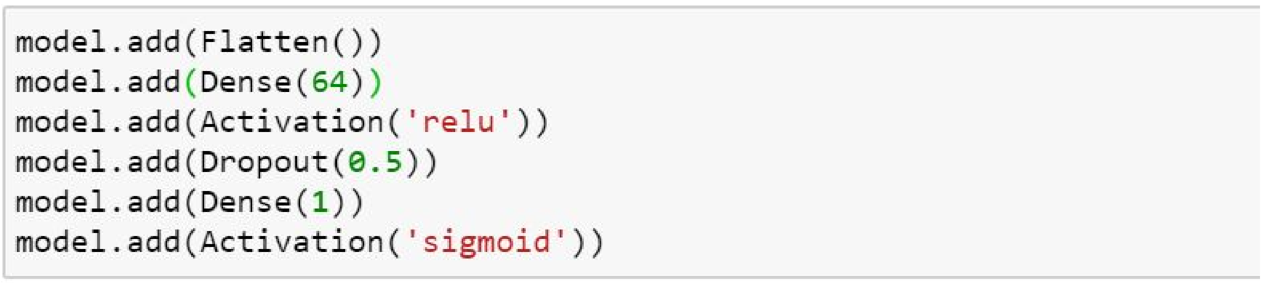
```
model = Sequential()
model.add(Conv2D(16, (3, 3), input_shape=input_shape))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```



```
model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```



```
model.add(Conv2D(64, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
```



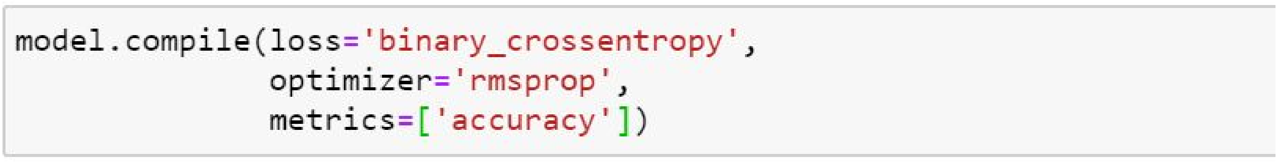
```
model.add(Flatten())
model.add(Dense(64))
model.add(Activation('relu'))
model.add(Dropout(0.5))
model.add(Dense(1))
model.add(Activation('sigmoid'))
```

We end the model with a single unit and a sigmoid activation, which is perfect for a binary classification. To go with it we will also use the binary_crossentropy loss to train our model.

6.5 Running and Fitting the model

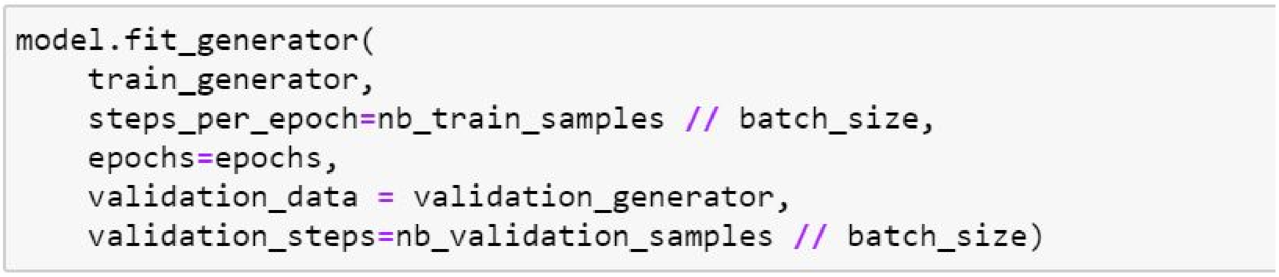
6.5.1 Running the model

We are using **rmsprop** as optimizer for our model .



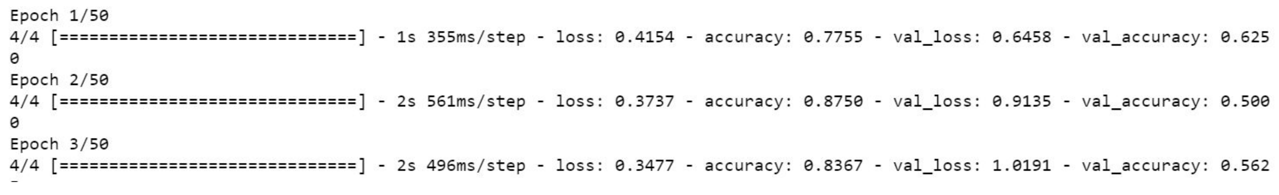
```
model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',
              metrics=['accuracy'])
```

6.5.2 Fitting the model



```
model.fit_generator(
    train_generator,
    steps_per_epoch=nb_train_samples // batch_size,
    epochs=epochs,
    validation_data = validation_generator,
    validation_steps=nb_validation_samples // batch_size)
```

Output :

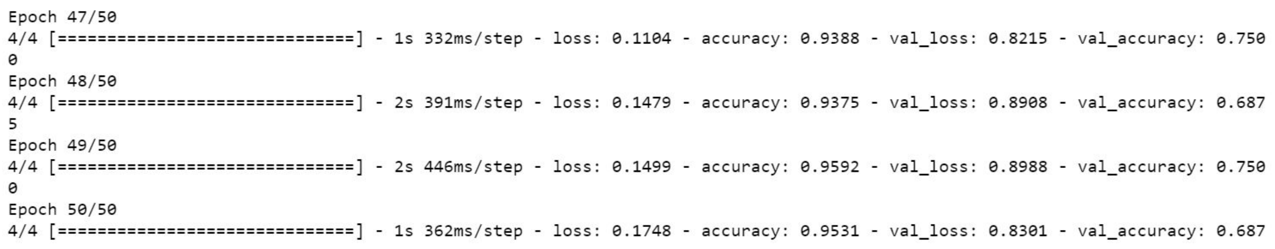


```
Epoch 1/50
4/4 [==============================] - 1s 355ms/step - loss: 0.4154 - accuracy: 0.7755 - val_loss: 0.6458 - val_accuracy: 0.625
0
Epoch 2/50
4/4 [==============================] - 2s 561ms/step - loss: 0.3737 - accuracy: 0.8750 - val_loss: 0.9135 - val_accuracy: 0.500
0
Epoch 3/50
4/4 [==============================] - 2s 496ms/step - loss: 0.3477 - accuracy: 0.8367 - val_loss: 1.0191 - val_accuracy: 0.562
```

•

•

•



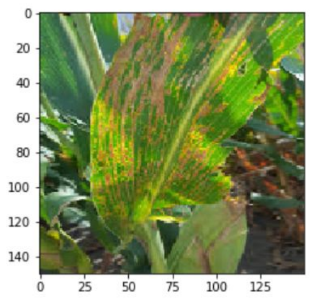
```
Epoch 47/50
4/4 [==============================] - 1s 332ms/step - loss: 0.1104 - accuracy: 0.9388 - val_loss: 0.8215 - val_accuracy: 0.750
0
Epoch 48/50
4/4 [==============================] - 2s 391ms/step - loss: 0.1479 - accuracy: 0.9375 - val_loss: 0.8908 - val_accuracy: 0.687
5
Epoch 49/50
4/4 [==============================] - 2s 446ms/step - loss: 0.1499 - accuracy: 0.9592 - val_loss: 0.8988 - val_accuracy: 0.750
0
Epoch 50/50
4/4 [==============================] - 1s 362ms/step - loss: 0.1748 - accuracy: 0.9531 - val_loss: 0.8301 - val_accuracy: 0.687
```

after 50 epochs the model is giving an testing accuracy of 95.31 % and validation accuracy of 68.7% with loss of 0.17 in testing and 0.83 in validation sets .

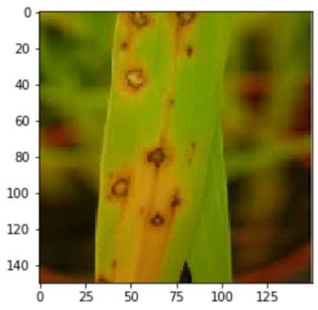
6.6 Visual Representation of the output

```
path = './rice dataset/userset'
for i in os.listdir(path):
    img=image.load_img(path + '/' + i , target_size=(150, 150))
    plt.imshow(img)
    plt.show()


    X = image.img_to_array(img)
    X = X/255
    X = np.expand_dims(X,axis=0)
    images = np.vstack([X])
    classes = model.predict(images ,batch_size = 5)
    print(classes[0])
    if classes[0]>0.5:
     print(i + " has rice blast ")
    else:
     print(i + " has leaf streak")
```

Output :



```
[0.00306916]
1.jpg has leaf streak
```



```
[0.5608861]
rb4.jpg has rice blast
```

CHAPTER 7

TESTING

7.1 Testing Plan

Testing process starts with a test plan. This plan identifies all the testing related activities that must be performed and specifies the schedules, allocates the resources, and specified guidelines for testing. During the testing of the unit the specified test cases are executed and the actual result compared with expected output. The final output of the testing phase is the test report and the error report.

7.1.1 Test Data

Testing process begins with a test design. This arrangement recognizes all the testing related exercises that must be performed like the timetables, assigning the assets, and determining rules for testing. This testing of the unit of the predetermined experiments are executed and the genuine outcome is expected. The last part of the testing stage is the test report and the error report.

7.1.2 Unit testing

Every individual module has been tried against the necessity with some test information.

7.1.3 Test Report

The module is working appropriately given the client must enter data. All information section frames have been tested with indicated test cases and all information passage shapes are working properly.

7.1.4 Error Report

On the off chance that the client does not enter information in determined request, at that point the client will be incited with error messages. Error reduction is done to deal with the normal and sudden mistakes.

CHAPTER - 8

CONCLUSIONS

8.1 Conclusion

Rice production in India is an important part of National Economy. Predicting a disease that attacks the paddy helps the farmer to get suitable remedy rather than being confused what to use. Here we have two major diseases that often attack paddy, which are Leaf Streak and Rice Blast. These two diseases look very similar, but have different attacking mechanisms. So, we help the farmers by predicting between those two diseases. We used the Keras and TensorFlow APIs as a base to develop our model. We used the concept of "Convolutional Neural Network" to train our model. It's a slow process but provides good accuracy levels compared to some other ML algorithms. Every line of our code is self-explanatory and easy to understand. The we trained the model with a handpicked data from internet. The model is tested to provide desired results.

8.2 Future Scope

Now-a-days machines are even taught to cultivate the agricultural fields. So, it is important for a machine to recognize diseases of the crop it's been cultivating unless the whole yield would be worthless. So, our model can be helpful for a lot of machines to learn the type of the diseases and distinguishing them. We further increase the scope of our project and train our model to predict a wide range of diseases over a vast range of crops cultivated. We further try to implement the same project with other APIs which decreases the prediction time and doesn't compromise with the prediction accuracy.

CHAPTER - 9

BIBLIOGRAPHY

References :

[1] https://www.google.com/search?q=rice+blast&sxsrf=ALeKk01mb3fVTY4MCcuxvcy-w3Un1DXvhg:1598456720659&source=lnms&tbm=isch&sa=X&ved=2ahUKEwjz-Zeom7nrAhVEAXIKHcwdCWYQ_AUoAXoECB4QAw&biw=1280&bih=652

[2] https://www.google.com/search?q=leaf+streak&tbm=isch&ved=2ahUKEwjIlKCpm7nrAhWBWSsKHS99B8IQ2-cCegQIABAA&oq=lea&gs_lcp=CgNpbWcQARgAMgQIIxAnMgQIABBDMgQIABBDMgQIABBDMgQIABBDMgQIABBDMgQIABBDMgQIABBDMgQIABBDMgQIABBDUOaWAlihmQJg7KICaABwAHgAgAHqAYgBvgSSAQUwLjIuMZgBAKABAaoBC2d3cy13aXotaW1nwAEB&sclient=img&ei=koNGX8iDNoGzrQGv-p2QDA&bih=652&biw=1280

[3] https://scikit-learn.org/stable/tutorial/index.html

[4] https://jupyter.org/install.html

[5] https://stackoverflow.com/questions/44090420/cnn-architecture

[6] https://keras.io/api/layers/convolution_layers/convolution2d/

[7] https://www.tensorflow.org/tutorials

[8] https://www.coursera.org/learn/convolutional-neural-networks?specialization=deep-learnings