

**A
Minor Project-I Report
On
“Detection of Cotton Leaves Diseases”**

**Submitted in partial fulfillment of
The requirements for the 5th Semester Sessional Examination of
BACHELOR OF TECHNOLOGY**

IN
COMPUTER SCIENCE & ENGINEERING

**By
NAME
Ravjot Singh Rayat(20CSE167)
Rahul Kumar Kuila(20CSE283)
E.Kavya(20CSE145)**

REGISTRATION NO.

20UG010297

20UG010410

20UG010275

**Under the able Supervision of
Dr. Premansu Sekhar Rath**

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING



GIET UNIVERSITY, GUNUPUR

2022 - 23



GIET UNIVERSITY, GUNUPUR

Dist. - Rayagada, Odisha-765022, Contact:- +91 7735745535, 06857-250170,172, Visit us:- www.giet.edu

Department of Computer Science & Engineering

CERTIFICATE

This is to certify that the project work entitled "Detection of Cotton Leaves Diseases" is done by Name- Ravjot Singh Rayat, E.Kavya and Rahul Kumar Kuila Regd. No.- 20UG010297, 20UG010275 and 20UG010410 in partial fulfillment of the requirements for the 5th Semester Sessional Examination of Bachelor of Technology in Computer Science and Engineering during the academic year 2022-23. This work is submitted to the department as a part of evaluation of 5th Semester Minor Project-I.

Project Supervisor

Class Teacher

Project Coordinator, 3rd Year

HoD, CSE, 3rd Year

ACKNOWLEDGEMENT

I would like to thank my supervisor sir , Dr. Premansu Sekhar Rath for their invaluable guidance and support throughout the development of this project . Their expertise and knowledge were instrumental in helping us to achieve our goals.

Secondly I would like to thank my class teacher, Dr. Bidush Kumar Sahoo for their assistance with FastApi . Their contribution was greatly appreciated.

Thirdly I would like to thanks our Project Coordinator Sir for their support and encouragement which was essential to the success of this project.

A special gratitude to our HOD ma'am , Dr. Bandita Sahu for giving us this great opportunity to work on this project and learn new things.

Ravjot Singh Rayat(20UG010297)
Rahul Kumar Kuila(20UG010410)
E.Kavya(20UG010275)

CONTENT

Abstract	1
Introduction	2
System Analysis	3
Works Done	4
System Design Analysis	11
Coding	14
Testing	27
Conclusion and Limitations	20
Bibliography	21

1.ABSTRACT

Cotton is a major agricultural crop that is grown worldwide and in Gunupur for its fibers, which are used to make textiles. However, cotton plants are susceptible to a variety of diseases, which can greatly reduce yields and cause significant economic losses for farmers. In this project, we developed a computer vision system to automatically detect and classify different cotton leaf diseases.

Our system uses image processing techniques to extract features from images of cotton leaves, which are then fed into a machine learning model to identify the presence and type of disease. We evaluated our system on a dataset of images of cotton leaves with different diseases and found that it achieved high accuracy in detecting and classifying the diseases.

The project focuses on developing a computer vision system to automatically detect and classify different cotton leaf diseases. Cotton plants are susceptible to a variety of diseases, such as Curl_Virus , Bacterial Blight and Fussarium_wilt , which can cause significant damage to the leaves and reduce yields. Early detection and diagnosis of these diseases is crucial for farmers to take timely and appropriate action to prevent the spread of diseases and minimize losses.

To develop our system, we collected a dataset of images of cotton leaves with different diseases and healthy leaves too. We used image processing techniques to extract features from these images, such as color, texture, and shape, which were then fed into a machine learning model to train the model to recognize the presence and type of disease. We evaluated our system on a test dataset and found that it achieved high accuracy in detecting and classifying the diseases.

In addition to its potential use in the field, our system also has applications in research and education, as it can provide a convenient and efficient tool for studying cotton leaf diseases and their effects on the plants.

Overall, our project demonstrates the potential of using computer vision and machine learning techniques to detect and diagnose cotton leaf diseases, which can help farmers to prevent the spread of diseases and maximize yields. Our system shows great potential for use in detecting and diagnosing cotton leaf diseases in the field, which can help farmers to take timely and appropriate action to prevent the spread of diseases and minimize losses.

2.INTRODUCTION

2.1 Purpose:

Cotton is a major agricultural crop that is grown worldwide for its fibers, which are used to make textiles. However, cotton plants are susceptible to a variety of diseases, which can greatly reduce yields and cause significant economic losses for farmers. Early detection and diagnosis of these diseases is crucial for farmers to take timely and appropriate action to prevent the spread of diseases and minimize losses.

2.2 Project Scope:

- Developing a system that can accurately identify different diseases that affect cotton leaves.
- Testing the system on a large dataset of images of diseased and healthy leaves to evaluate its confidence.
- Implementing the system in a user-friendly tool that can be used by farmers and other stakeholders to quickly and easily diagnose cotton leaf diseases.
- Evaluating the impact of the system on the cotton industry, including its potential to improve crop yields and reduce the use of pesticides.

2.3 Product Features:

In this project, we aim to develop a computer vision system to automatically detect and classify different cotton leaf diseases. Our system uses image processing techniques to extract features from images of cotton leaves, which are then fed into a machine learning model to identify the presence and type of disease. We evaluate our system on a dataset of images of cotton leaves with different diseases and assess its performance in detecting and classifying the diseases.

Overall, our project aims to demonstrate the potential of using computer vision and machine learning techniques to detect and diagnose cotton leaf diseases, which can help farmers to prevent the spread of diseases and maximize yields

3.SYSTEM ANALYSIS

3.1 Hardware Requirements:

- Windows 10 or 11
- 8GB Ram
- i5 Processor
- Graphic Card - Intel UHD 620

3.2 Software Requirements:

- **3.2.1 Jupyter Notebook:** Jupyter Notebook is an open-source web application that allows users to create and share documents that contain live code, equations, visualizations, and narrative text. Jupyter Notebook supports a wide range of programming languages, including Python, R, Julia, and Scala. It is often used for data analysis, scientific computing, and machine learning.
- **3.2.2 FastApi:** FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints. FastAPI comes with all the features you need to build production-ready APIs, including interactive documentation, authentication, and deployment to production. It's also designed to be fast and easy to use, with performance that is up to 100 times faster than other frameworks. Additionally, FastAPI integrates with popular libraries and frameworks, such as Django.
- **3.2.3 Visual Studio Code:** Visual Studio Code is a free, open-source code editor developed by Microsoft. It is a popular choice for developers and is available on Windows, Linux, and macOS. It features support for debugging, embedded Git control, syntax highlighting, code completion, and more. Overall, Visual Studio Code is a versatile and powerful code editor that is well-suited for a wide range of development tasks
- **3.2.4 React JS:** React is a popular JavaScript library for building user interfaces. It is developed and maintained by Facebook, and is used to build single-page applications and user interfaces for mobile applications. React allows developers to create large web applications that can change data, without reloading the page. It aims to provide a declarative, efficient, and flexible way to build user interfaces.

4.WORKS DONE:

The image processing technique is employed for detecting diseases on cotton leaves early and accurately. The processing scheme consists of image acquisition through camera or web, image pre-processing includes image enhancement and image segmentation where the affected and useful areas are segmented, feature extraction and classification. Finally the presence of diseases on the plant leaf are going to be identified. For feature extraction, they are using the K-mean clustering algorithm method for classification and Neural-network as recognizer. The step-by-step procedure is shown as below. 1) RGB image acquisition 2) Pre-processing of image using histogram equalization 3) Resize the image 4) CNN Algorithm for image segmentation 5) Computing features extraction 6) Classification & Recognition using neural networks 7) Statistical analysis. Study of diseases on the cotton leaf studied by using the image processing toolbox and also the diagnosis

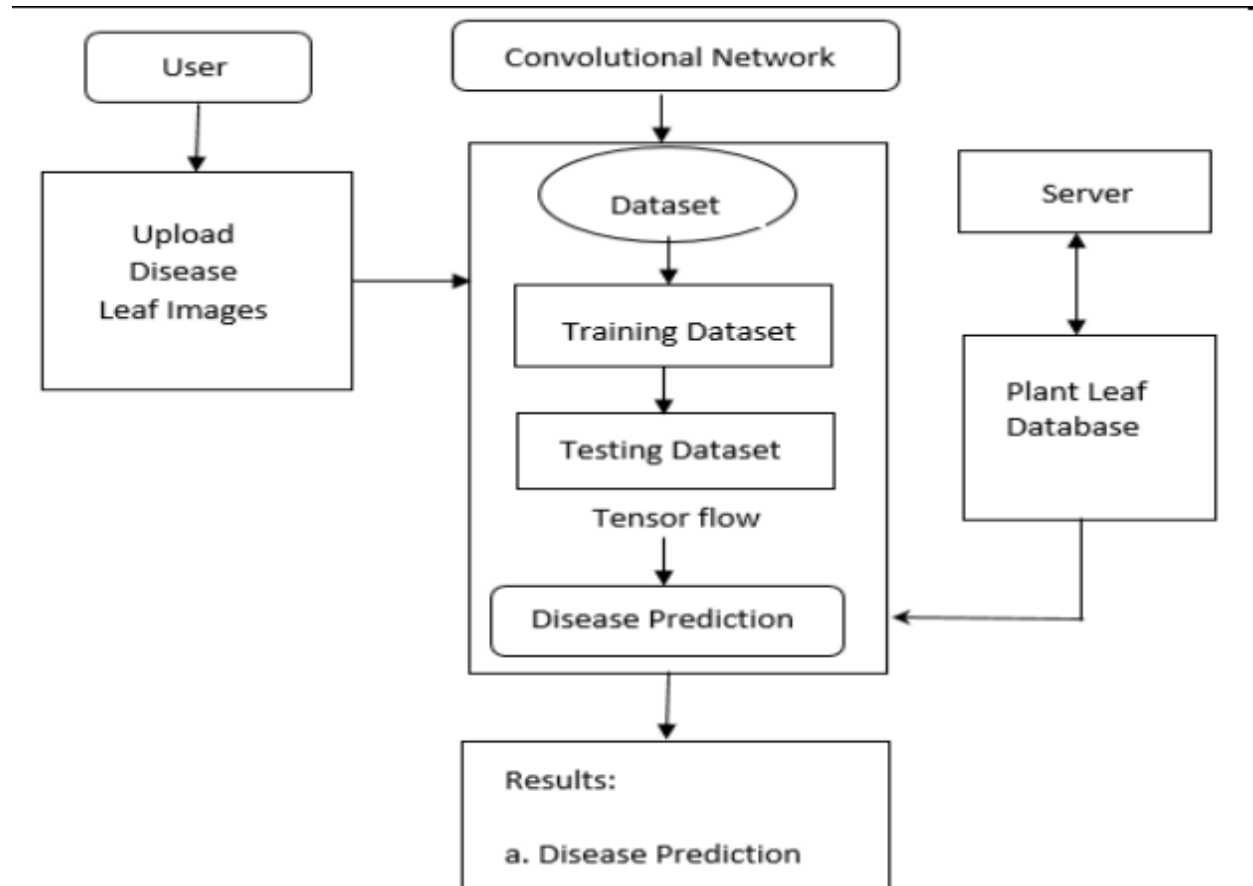
Using MATLAB helps to suggest necessary remedies for that disease arising on the leaf of a cotton plant. The accurate recognition for using CNN. Clustering method the Euclidean distance is 89.56% and the execution time for CNN Clustering method using Euclidean distance is 436.95 second and also thresholding is done by a dynamically range $[0,1]$ depending on color intensity from leaves image. Hence that disease detection using CNN Clustering method using Euclidean distance is the excellent methods to disease detection on cotton leaves.

This system consists of two parts: 1) digital image assessment and feature extraction of sample cotton leaf 2) to implement the back propagation artificial neural network in machine learning. This process has the following five steps- Image acquisition, Image Pre-processing, Image enhancement, Image segmentation and Feature extraction. To classify the quality of cotton leaf diseases, the Convolutional Neural Network tool of MATLAB is used. This quality identification is done based on the RGB and HSV components of the image. This CNN tool works on neurons, neurons are further connected to hidden layers of neurons. The CNN tool also has a back-propagation process. The prediction of the outcome is taken randomly by the neural network process. The advantage of this method is it can predict the data correctly with minimized error. This method was able to detect cotton leaf with or without defects from the image.

In a purpose system, we use a real-time dataset that contains various images of cotton disease like bacterial blight, bronze wilt, curly lift, fouler fungal disease. We have given some images for training and some are testing. Initially we take the images from the real time dataset and give it to the model for identifying the cotton disease. As, we give the images to our system, it shows the result in the form of probability. The methodology of cotton disease detection using image processing has the following steps:

4.1 Image pre-processing: In these phases, we require better resolution images and with better quality. All these images are resized with specific manner and resolution. These images we remove noise content and rotate the images using a data augmentation process.

4.2 Image segmentation: Image segmentation it's a process of dividing a digital image into various sections. Its use to remove the region of the pixel in infected leaf and easily identified the model which part is infected.



4.3 Feature extraction: In this feature extraction process, extract some of the important features of the defected leaf. It can create colored structure and convert the color value from RGB components of defected parts of cotton leaf image. The feature we can use to train our neural network. When all processes are done then we give the train and test data to the model and apply the CNN algorithm. The Following Flowchart you can see how the model is working.

4.4 Data Augmentation There are 4 classes of diseases for which the images were collected. The images were less in number. As deep learning based approaches need more images, we have performed data augmentation using MATLAB. This is one time process which gives 10 rotated versions of original image. Therefore lot of data is generated which is helpful to training model. We have used various data science related libraries like keras, tensorflow, fatsapi, opencv, matplotlib, numpy etc. For the purpose of building keras model we have used sequential modelling technique. The architecture of model consists of two conv2D layers. Each conv2D

layer is followed by activation layer named 'relu' and maxpooling layer. Maxpolling is that the highest value they can catch and declare. Once the data is available at final maxpooling layer, it is subjected to set of fully connected neuron as they are in CNN. For this purpose flattening is done and dense layers are added. Flatten is convert output to in one dimensional array that work is done by flatter. This creates the architecture of deep learning model which will be trained using the data which was uploaded earlier. The data is available on the colab server. Path variable will read the images from the path one by one. Each image is read using opencv library.

Subsequently, the images are resized with dyadic image processing. The paths of the images also tell about the Class of each image which is extracted and stored in a variable called label. The data and label lists are converted into numpy arrays for the purpose of training the model. The data train: test split ratio is 75:25. Runtime data augmentation during training is also made available to the optimizer. For building the model, the classifier is notified that the image dimensions are 256*256*3 along with 4 classes.

We have used categorical cross-entropy for measuring the losses during training process which are monitored continuously. We have used Adam optimizer which is latest optimizer that SGD. The training process gives a trained model which can further be used for testing purpose. For testing, we have uploaded the .zip file that contained test images. The performance is checked for images from this .zip file after unzipping it.

Algorithm Used Traditional feature learning methods rely on semantic labels of images as supervision. They usually assume that the tags are evenly exclusive and thus do not point out towards the complication of labels. The learned features endow explicit semantic relations with words. CNN itself is a technique of classifying images as a part of deep learning. In which we apply single neural network to the full image.

I. Accepts a volume of size $W1 \times H1 \times D1$

II. Requires four hyper parameters: Number of filters K Their spatial extent F The stride S The amount of zero padding P.

III. Produces a volume of size $W2 \times H2 \times D2$ where: 1) $W2 = (W1 - F + 2P) / S + 1$ 2) $H2 = (H1 - F + 2P) / S + 1$ (i.e. width and height are Computed equally by symmetry) 3) $D2 = K$

IV. With parameter sharing, it introduces $F * F * D1$ weights per filter, for a total of $(F * F * D1) * K$ weights and K biases. In the output volume, the dth depth slice (of size $W2 * H2$) is the result of performing a valid convolution of the dth filter over the input volume with a stride of S, and then offset by dth bias.

V. A common setting of the hyper parameters is $F=3$, $S=1$, $P=1$ However, there are common conventions and rules of thumb that motivate these hyper parameters.

CNN uses the layers for image processing. If the image having the more than one objects then the CNN recognizes the edges and classify the image accordingly. Pixel is the smallest portion of the image. One single image contains number of pixels. These pixels group together it makes entire image. CNN uses feature detector. Feature detector used to detect significant features of image data in order to provide detection. It is the smallest matrix of weights. To reduce the bigger images into smaller images strides are used. Stride is the number of pixel by which we slide our filter matrix over the input matrix. This process is called convolution. By this the shape of the input image is modified feature detection there by detecting the particular feature from the input image and to get the information about that feature. This is called the feature map. Large images takes lot of time. It is easier to process small images in faster manner.

For the image identification we use tensor flow. Keras layer like input layer, dense layer, convolution 2D layer, Maxpool2D layer, activation layer and Flatten. The keras modelling are different techniques but we use sequential modelling. In these sequential modelling we tell that execute the step by step layer and the all network are made. We use LeNet class to classify the width, height and depth of the image. Softmax classifier can give us probability to the result whether these image are powdery disease or foliar disease or other disease and out of 100% what the probability of the disease they can give us if the probability is 95% so model is good. We can split the path and set the string through labels. The raw pixel intensities to the range to $[0, 1]$ and in grey scale image maximum value are 255 and minimum value is 0. So we divide 255 to 255 minimum value is 1 so it can be normalized. We take the 20% data to test and 80% data to train.

Deep CNN with ReLUs trains several times faster. This method is used for the extraction of the entire convolutional and fully connected layer. Apart from the output, standard installation is not required; is used after ReLU disconnection after the first and second convolutional layers because it lowers the top-1 and top-5 values. At CNN, neurons inside the latent layer are separated into “feature maps.”

The neurons within the feature map share the same weight and bias. Neurons within a map element search for the same feature. These neurons are different because they are connected to different neurons in the lower layer. So in the first hidden layer, the neurons within the feature map will be linked to different regions of the input image. The hidden layer is separated by feature maps where each neuron in the feature map looks at the same element but at different image capture locations. Basically, a feature map is the result of applying convolution to an entire image.

The features of each layer are displayed in a separate block, where visibility represents the strongest performance of a given feature map, from the first convolutional layer, where the elements from individual pixels to the simple lines, in the fifth convolutional layer where layered features and parts of the leaves are displayed. Another important layer of CNN is the integration layer, which is a type of offline reduction. The functionality of the pools provides a kind of

translation flexibility; it works independently across the input depth slide and makes its size geographically.

Excessive blending is used to benefit the reduction of excessive excess. And in favor of reducing overeating, the quit layer is applied to the first two layers that are fully connected. But the failure to drop out of school is that it increases training time by 2-3 times compared to a normal neural network of direct construction. Bazesian performance tests have also proven that ReLUs and school dropouts have the effects of collaboration, which means that they are beneficial when used together. CNN's advancement refers to their ability to study medium image representation with contradictory and low-level features designed for alternative image classification methods.

4.5.1 Testcase 1

Train model with 10 epoch In test case 1 we perform operation on 10 times of single image the training accuracy of 1st epoch is 0.4815 and valid accuracy of 1 st epoch is 0.6024 so our model is not that much good but when we process done in till 10th epoch the training accuracy was 0.8815 and valid accuracy is 0.9824 the result is model was perform good and its give the accuracy of 98% accurate.

4.5.2 Testcase 2

Check the model and we give sample image Bronze wilt disease. The model gives the probability which disease this is they can compare all the disease in the dataset and give the percentage the output is Bronze wilt disease - 0.94, Curly Leaf -0.77, Foliar fungal-0.14, Powdery Mildew-0.22, Fresh Leaf 0.61 so the model gives these result and percentage of Bronze wilt Disease is 0.94 so model is working properly.

The conclusion of these test cases is when we do more operation on image the model can give you accurate output and when we provide original images the model is working well with proper accuracy. We test more on that but the hardware requirement was not fulfill for these operation.

We took image data set of cotton leaf from the field After giving input leaf image it shows the disease percentage occur on the input leave it will be very beneficial for the farmers to the applied pesticides solution on the particular portion of infected leaf. The dataset of cotton crop was not available on the internet so we went to the cotton field ourselves and took photos of the cotton leaves and made a dataset from it. Images and the same dataset we have used in our system. Solution in the form of pesticides.

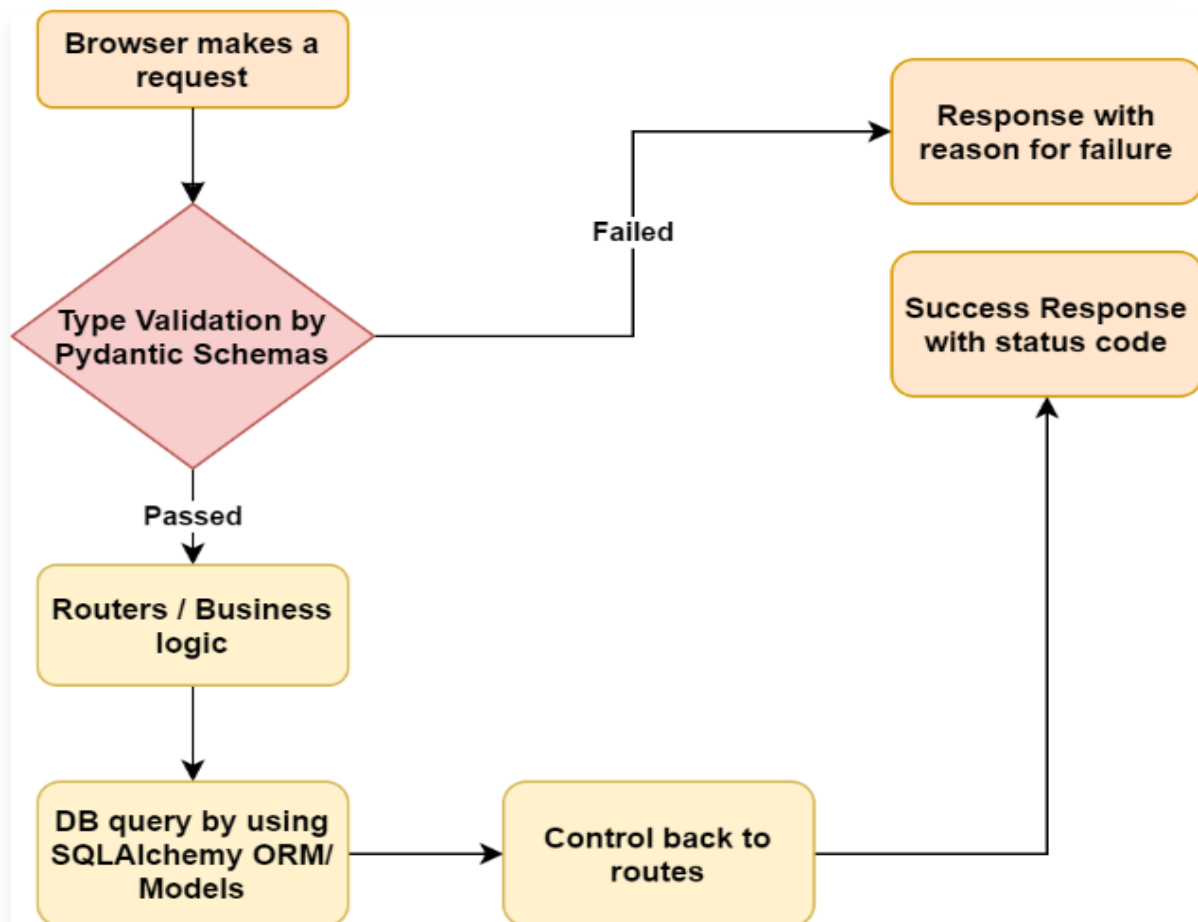
It will only show the result of cotton leaf disease if we provide other input images rather than cotton leave it will show the result between our define disease classes and this is the only drawback of our system at what time we can provide limited set of images and if we try to provide five or more than five images at one time then system may be hang. There are number of diseases on the cotton plant like seedling diseases bacterial diseases boll roots but our system

only predict the disease which are occur on the leaf of cotton like foliar fungal, bronze wilt, mildew, curly leaf etc. Our system can input a lot of photos at once so it helps to save the farmer's time. And if more than five photos are given for input in our system, our system may hang or the user may have to face some system problems.

The Algorithm will help the end user to segregate the infected crop based on percentage of infection to take preventive measures at as early stage as possible. The algorithm will help in minimizing the use of pesticides thereby improving the environment and ecological balance. The proposed work has vast applications to help the Indian farmers in early identification of cotton crop diseases.

In future we can expand disease dataset for detect more disease. Work on accuracy that can improve user result and give the correct pesticides or insecticides to the user also we built for mobile friendly app so process done on cloud. That one thing can benefit of user because any device they have the process can work.

4.6 Deployment: FastAPI is a modern, fast (high-performance), web framework for building APIs with Python 3.6+ based on standard Python type hints. It is built on top of Starlette for the web parts and Pydantic for the data parts. FastAPI comes with all the features you need to build production-ready APIs, including interactive documentation, authentication, and deployment to production. It's also designed to be fast and easy to use, with performance that is up to 100 times faster than other frameworks. Additionally, FastAPI integrates with popular libraries and frameworks, such as Django and SQL Alchemy, to make it easy to build sophisticated applications quickly



5.SYSTEM DESIGN AND ANALYSIS

5.1 High Level Design(HLD)

5.1.1 Flowchart

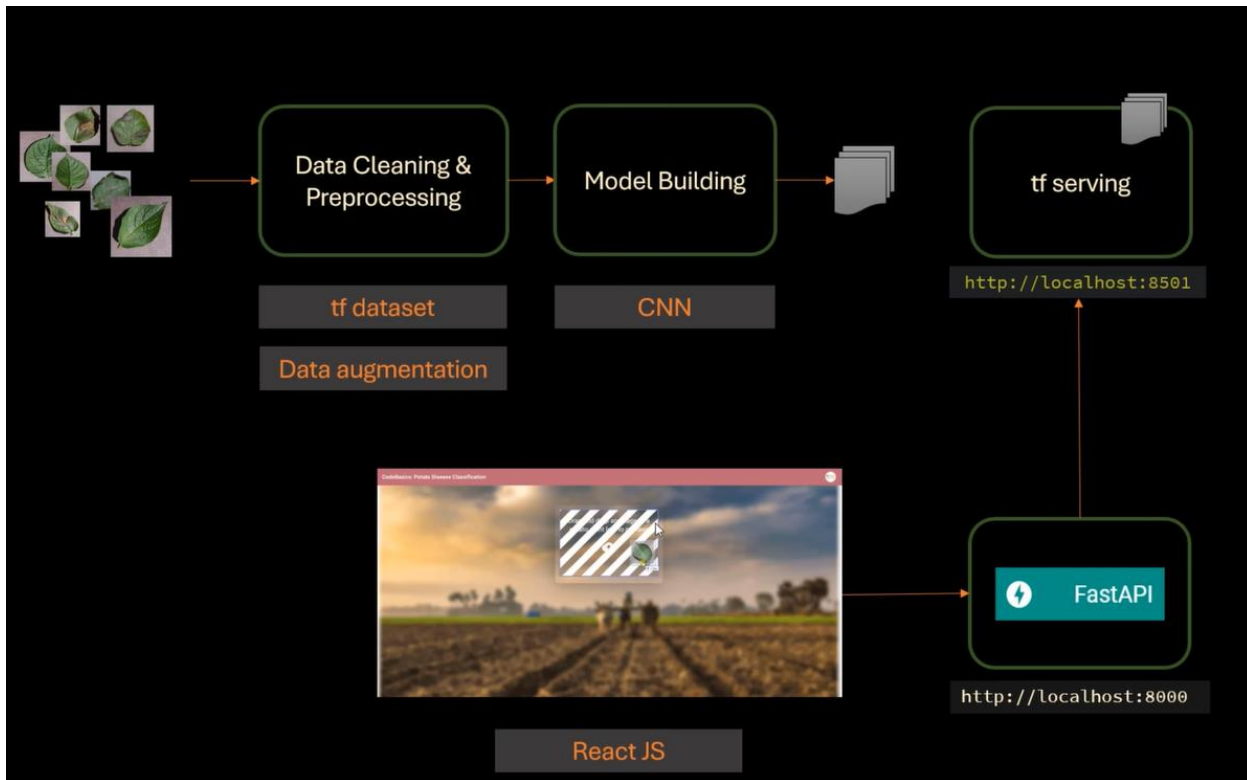


Fig 5.1.1

5.1.2 Algorithm Flowchart

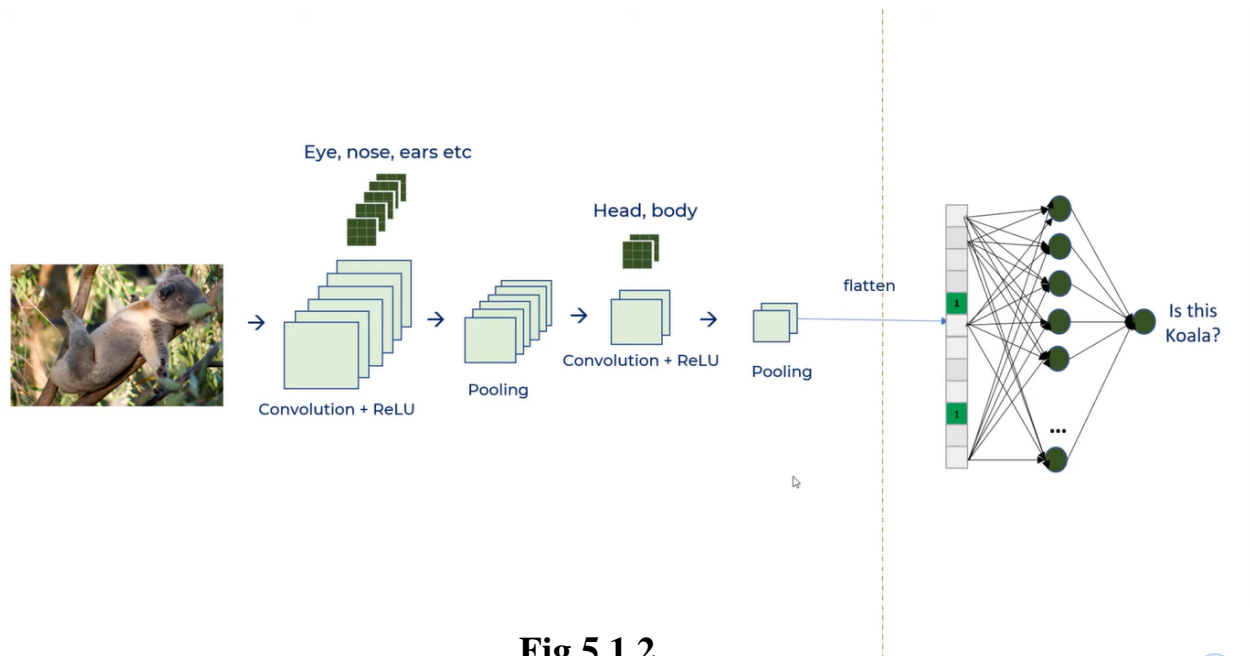


Fig 5.1.2

5.2 Low Level Diagram (LLD)

5.2.1 Process Specification

Step 1: Importing the dataset through Keras. Dataset contains .

1500 images of diseases leaves and healthy leaves.

Step 2: Splitting the dataset into training and testing dataset.

Step 3: Caching,shuffle and prefetching the dataset.It is done to save the time.

Step 4: Scaling the images by dividing it by 255 and performing the data augmentation to increase the amount of data by generating new data points from existing data.

Step 5: Extracting the features from the image by Max pooling and Convolution2D.

Step 6: Compiling the algorithm for 55 epochs.

Step 7: Saving the model by OS.

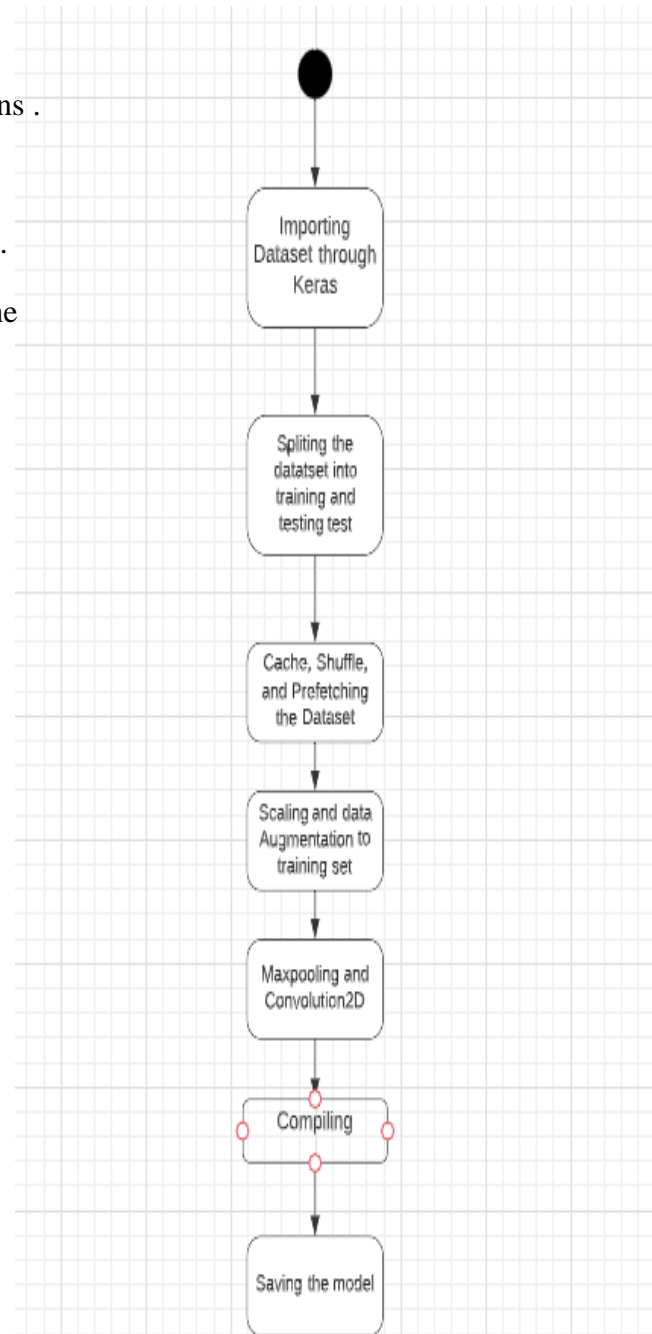
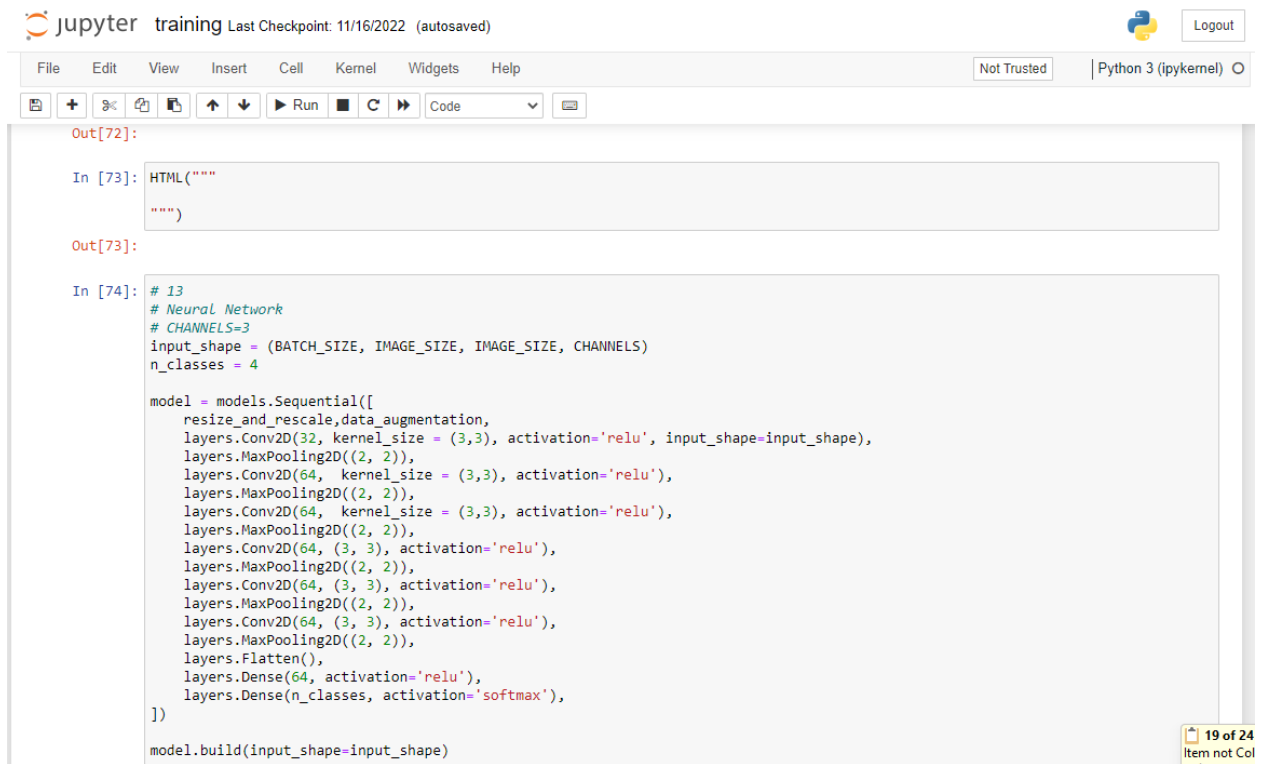


Fig 5.2.1

6.CODING



The image shows a Jupyter Notebook interface with the title "training" and a subtitle "Last Checkpoint: 11/16/2022 (autosaved)". The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help), a toolbar with icons for file operations and execution, and a status bar indicating "Not Trusted" and "Python 3 (ipykernel)".

The notebook contains two code cells. The first cell, labeled "In [73]:", contains an HTML comment: `HTML("""` followed by a blank line and `""")`. The second cell, labeled "In [74]:", contains a Python code snippet for a neural network model. The code defines the input shape and number of classes, then builds a sequential model with multiple layers of convolution, max pooling, and dense layers, ending with a softmax layer.

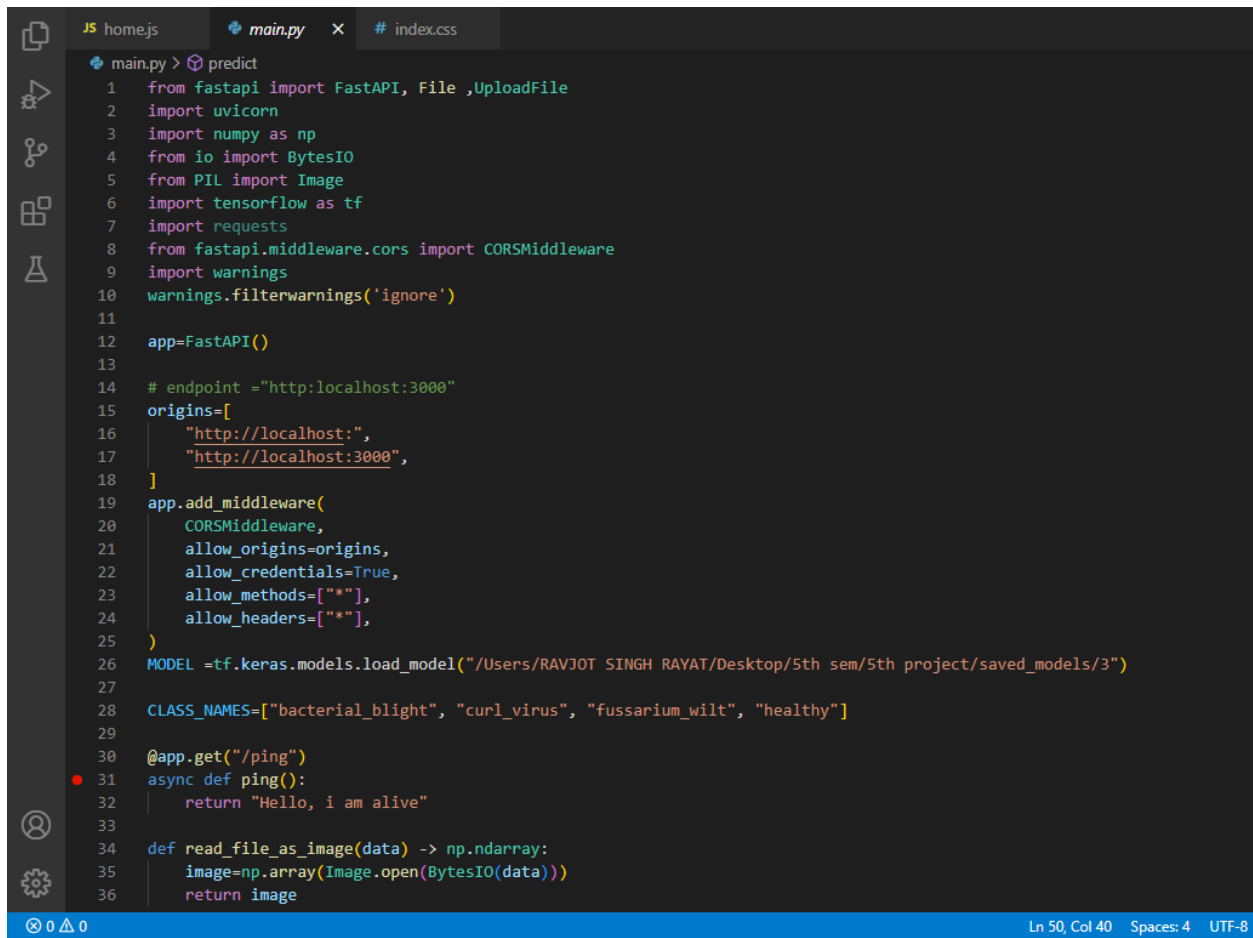
```
# 13
# Neural Network
# CHANNELS=3
input_shape = (BATCH_SIZE, IMAGE_SIZE, IMAGE_SIZE, CHANNELS)
n_classes = 4

model = models.Sequential([
    resize_and_rescale,data_augmentation,
    layers.Conv2D(32, kernel_size = (3,3), activation='relu', input_shape=input_shape),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, kernel_size = (3,3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Conv2D(64, (3, 3), activation='relu'),
    layers.MaxPooling2D((2, 2)),
    layers.Flatten(),
    layers.Dense(64, activation='relu'),
    layers.Dense(n_classes, activation='softmax'),
])

model.build(input_shape=input_shape)
```

The status bar at the bottom right indicates "19 of 24" items and "Item not Col".

Fig 6.1

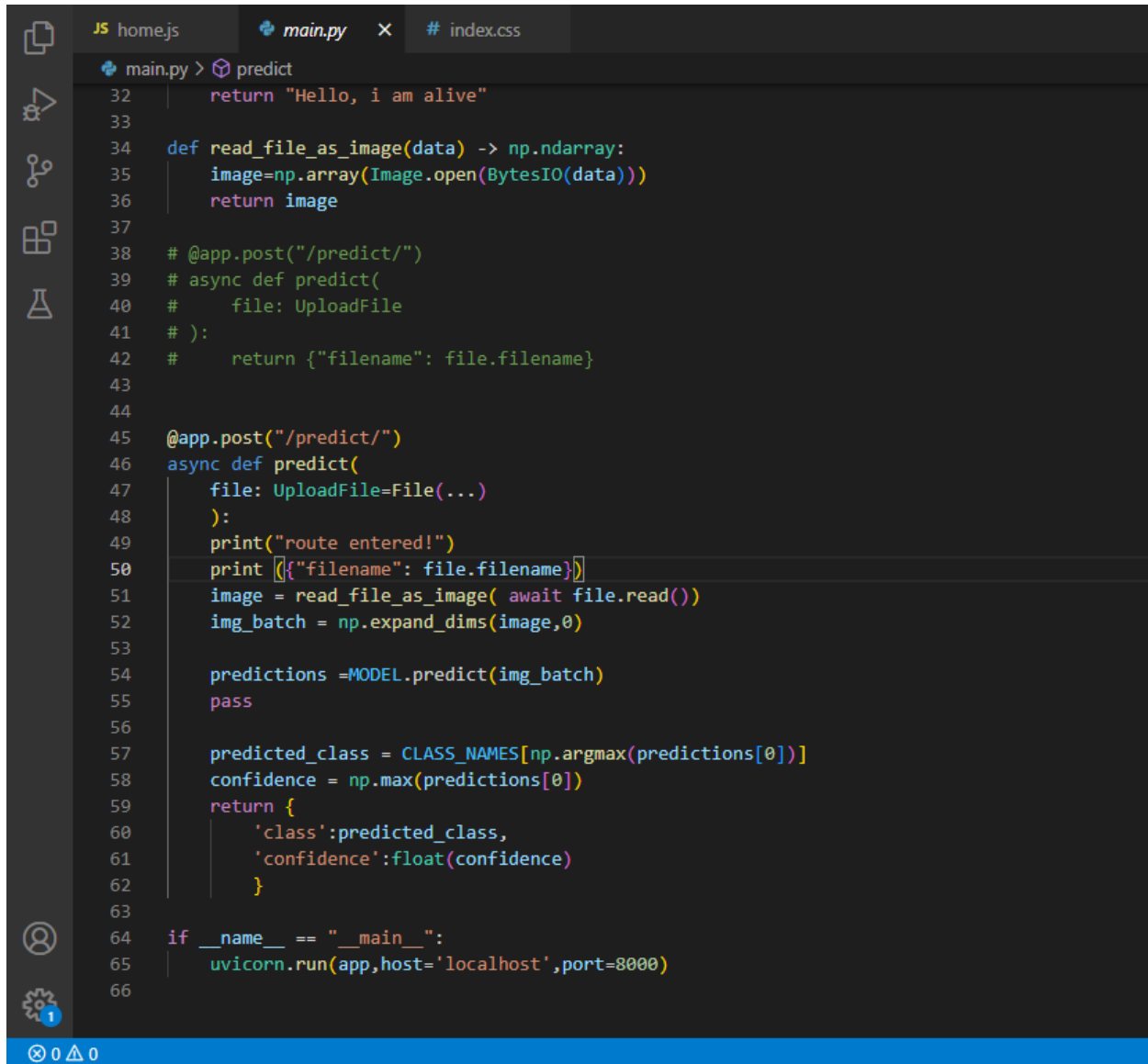


The image shows a Visual Studio Code editor window with a dark theme. The top bar shows three tabs: 'home.js', 'main.py' (active), and 'index.css'. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, and Testing. The main editor area displays a Python file named 'main.py' with the following code:

```
main.py > predict
1 from fastapi import FastAPI, File ,UploadFile
2 import uvicorn
3 import numpy as np
4 from io import BytesIO
5 from PIL import Image
6 import tensorflow as tf
7 import requests
8 from fastapi.middleware.cors import CORSMiddleware
9 import warnings
10 warnings.filterwarnings('ignore')
11
12 app=FastAPI()
13
14 # endpoint ="http://localhost:3000"
15 origins=[
16     "http://localhost:",
17     "http://localhost:3000",
18 ]
19 app.add_middleware(
20     CORSMiddleware,
21     allow_origins=origins,
22     allow_credentials=True,
23     allow_methods=["*"],
24     allow_headers=["*"],
25 )
26 MODEL =tf.keras.models.load_model("/Users/RAVJOT SINGH RAYAT/Desktop/5th sem/5th project/saved_models/3")
27
28 CLASS_NAMES=["bacterial_blight", "curl_virus", "fussarium_wilt", "healthy"]
29
30 @app.get("/ping")
31 async def ping():
32     return "Hello, i am alive"
33
34 def read_file_as_image(data) -> np.ndarray:
35     image=np.array(Image.open(BytesIO(data)))
36     return image
```

The status bar at the bottom indicates 'Ln 50, Col 40', 'Spaces: 4', and 'UTF-8'.

Fig 6.2



The image shows a code editor with three tabs: `home.js`, `main.py`, and `# index.css`. The `main.py` tab is active, showing a Python script. The script includes a `predict` function that returns a string, a `read_file_as_image` function that reads an image file, and a Flask `async def predict` endpoint that processes an uploaded image file, predicts a class, and returns the predicted class and confidence. The script also includes a `__main__` block that runs the application on `localhost` at port `8000`.

```
main.py > predict
32 |     return "Hello, i am alive"
33 |
34 | def read_file_as_image(data) -> np.ndarray:
35 |     image=np.array(Image.open(BytesIO(data)))
36 |     return image
37 |
38 | # @app.post("/predict/")
39 | # async def predict(
40 | #     file: UploadFile
41 | # ):
42 | #     return {"filename": file.filename}
43 |
44 |
45 | @app.post("/predict/")
46 | async def predict(
47 |     file: UploadFile=File(...)
48 | ):
49 |     print("route entered!")
50 |     print ({"filename": file.filename})
51 |     image = read_file_as_image( await file.read())
52 |     img_batch = np.expand_dims(image,0)
53 |
54 |     predictions =MODEL.predict(img_batch)
55 |     pass
56 |
57 |     predicted_class = CLASS_NAMES[np.argmax(predictions[0])]
58 |     confidence = np.max(predictions[0])
59 |     return {
60 |         'class':predicted_class,
61 |         'confidence':float(confidence)
62 |     }
63 |
64 | if __name__ == "__main__":
65 |     uvicorn.run(app,host='localhost',port=8000)
66 |
```

Fig 6.3

7.TESTING

7.1 Input Image



Fig 7.1

7.2 Expected Output : Healthy Image

7.3 Input Image

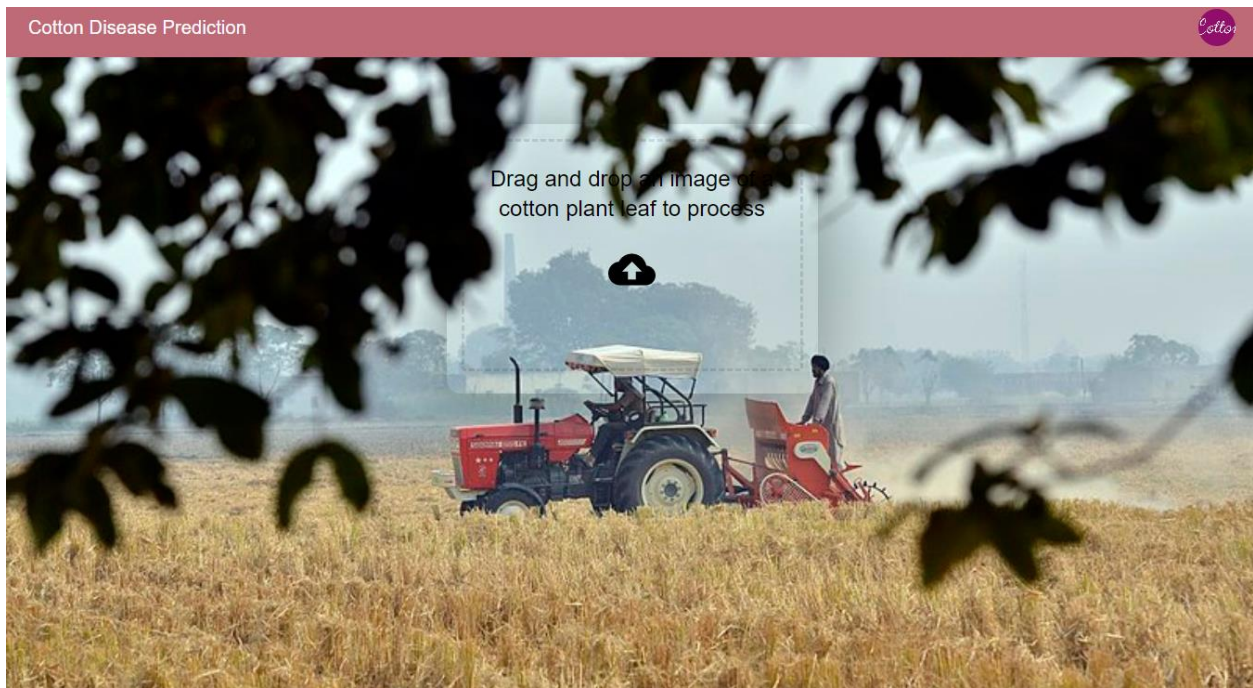


Fig 7.3

7.4 Output

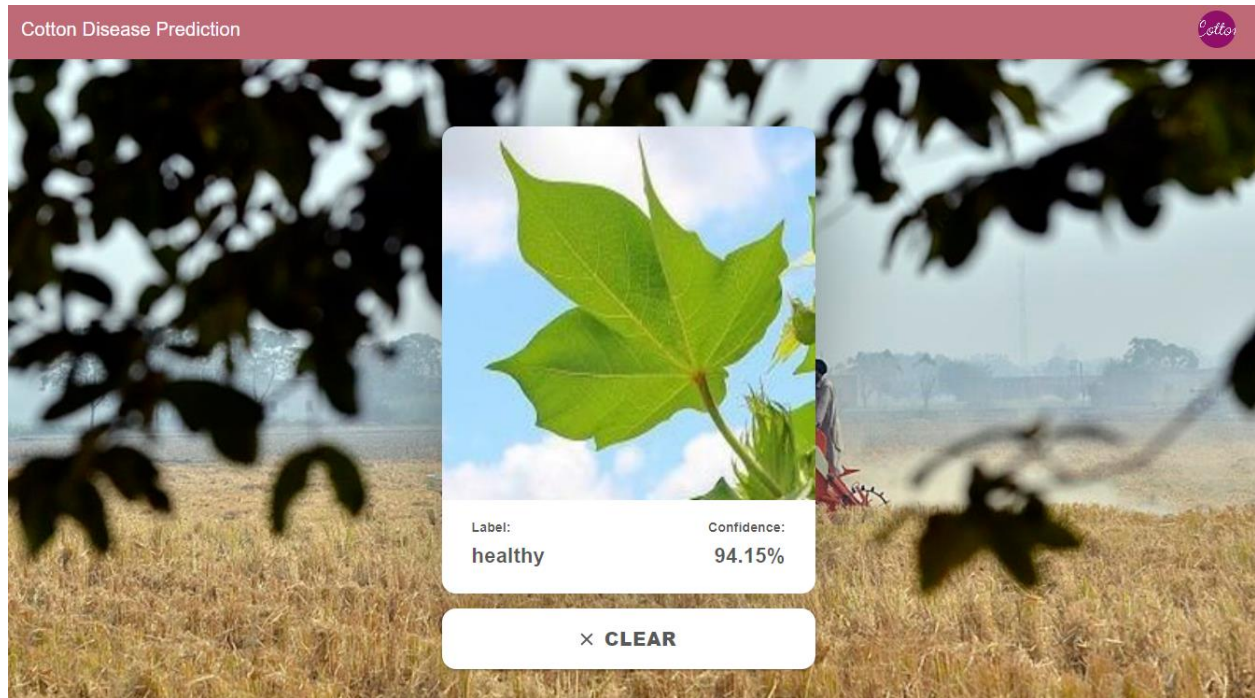


Fig 7.4

8.CONCLUSION AND LIMITATIONS

One potential limitation of this project is that it may only be effective in detecting diseases in cotton leaves with a resolution of 256x256 pixels. This may limit the accuracy of the predictions, as images with a higher or lower resolution may not provide the same level of detail to the CNN. Additionally, the effectiveness of the CNN in detecting diseases in cotton leaves may be limited by the quality and diversity of the training data used to train the model. If the training data is not representative of the types of diseases and conditions that the model will encounter in real-world scenarios, the model may not be able to accurately predict the presence of diseases in new images.

This model can only predict the image of size 256x256 size and if any other image is given as input then it don't predict whether it is the leaves of cotton or not. It says the confidence % is less.

8.BIBLIOGRAPHY

References:

FastApi: <https://fastapi.tiangolo.com/>

Machine Learning: Youtube channel codebasis .

React Js: <https://reactjs.org/tutorial/tutorial.html>

Papers referred:

<https://ijcrt.org/papers/IJCRT2106528.pdf>

https://www.irjmets.com/uploadedfiles/paper/volume3/issue_1_january_2021/5926/1628083243.pdf