The Cotton Guard AI

Cotton Disease Detection Using Deep Learning Methods

By

Capt Shehroz Butt (Group Leader)

Maj Muhammad Sohaib

Capt Mehroz Qasim

Capt Moeez Ahmed Farooq

Supervised by:

Asst Prof Dr. Muhammd Sohail

Submitted to faculty of Department of Computer Software Engineering,
Military College of Signals, National University of Sciences and Technology, Islamabad,
in partial fulfillment for the requirements of B.E Degree in Software Engineering.

JUNE 2024

In the name of ALLAH, the Most benevolent, the Most Courteous

CORRECTNESS AND APPROVAL CERTIFICATE

This is a formal declaration that the thesis work included in this report has been

"The Cotton Guard AI"

is carried out by

Capt Shehroz Butt

Maj Muhammad Sohaib

Capt Mehroz Qasim

Capt Moeez Ahmed Farooq

Under my direction and in my opinion, the Military College of Signals at the National University of Sciences and Technology (NUST), Islamabad, offers a Bachelor of Software Engineering degree that is completely adequate in breadth and excellence.

Approved by

Supervisor

Asst Prof Dr Muhammad Sohail

Date: _____

DECLARATION OF ORIGINALITY

We thus certify that no part of the work included in this thesis has been submitted to this institute or any other place in support of any other award or qualification.

ACKNOWLEDGEMENTS

Allah Subhan'Wa'Tala is the sole guidance in all domains.

with your help, our parents, coworkers, and most importantly, our supervisor, **Asst Prof Dr Muhammad Sohail**.

The member of the institution preserved in their job despite all obstacles.

Certificate of Plagiarism (Turnitin Report)

____ is the similarity index for this thesis. The supervisor's approved Turnitin report is attached.

_____

Capt Shehroz Butt

359396

_____

Maj Muhammad Sohaib

359376

_____

Capt Mehroz Qasim

359391

_____

Capt Moeez Ahmed Farooq

359398

_____

Signature of Supervisor

ABSTRACT

An early detection of crop diseases is important as it helps in minimizing the losses which would otherwise be incurred and ensuring food security for the agricultural sectors worldwide including Pakistan Army's agriculture-based initiatives. This specific project aims to diagnose cotton diseases through a deep learning approach— more precisely Convolutional Neural Networks (CNNs). The system proposed based on CNN endeavors to detect different types of diseases by studying pictures of cotton plants that are taken in the field— this can lead to an immediate implementation of control measures. Despite its simplicity, this project plays a major role in improving sustainability and productivity among the large scale of cotton farming undertaken by the Pakistan Army as it covers thousands acres with agricultural lands. This study highlights the fusion of cutting-edge deep learning algorithms with pragmatic agricultural goals— an epitome of where technology meets agriculture. This could resonate with various other agricultural development projects in the locality, hence having a broader reach for the impact.

Contents

List of Diagrams

Chapter 1: Introduction

Agriculture is both a major money spinning venture and an essential element in feeding the increasing population of the globe. However, this industry has several challenges associated with the increased demand for food and crop diseases. The danger from such illnesses is that they have the potential to dramatically reduce agricultural output, resulting in vast crop losses and poor quality food, and therefore food scarcity. Cotton is a prime example as it is an economically, socially, and ecologically important crop as a key cash crop for millions of small farmers, and of global importance as a clothing fiber. Notwithstanding, cultivating cotton has certain challenges, as infections provide the primary threat to crop quality and productivity. Therefore, in order to preserve the sustainability of cotton production, it is imperative that these disorders be identified and treated.

In order to curb the incidence or rather effectively reduce the final forecasted impact of crop diseases on yield, it is crucial that they are detected in good time and with high precision; effective management strategies are also critically needed. This is where using technology tools and data driven applications can be very critical

1.1 Overview

This study provides critical new evidence of whether deep learning methods are suitable for the detection of cotton diseases, which is vital to maintain agricultural health. Additionally, its effectiveness in addressing Convolution Neural Networks (CNNs) was investigated. In most cases, transfer learning and tuning were applied to adjust CNN models. Due to its deep learning-based approach, the proposed system provides an efficient way for recognizing as well subclassifying illnesses present in cotton plants. Seven criteria are used by the model to classify cotton diseases.

1.2 Problem Statement

With the considerable effect of cotton disease on agriculture production, it is necessary to have an efficient and effective identification process. Traditional detection techniques can be used, but they have certain limitations with respect to labor-intensive and time-consuming methods, as well as insufficient accurate expression. To solve this problem, we study the automatic diagnosis of cotton illness based on DL models particularly Convolution Neural networks (CNN) to achieve better output possible. This benchmark is to evaluate how well our model can differentiate the damage degree of whiteflies on cotton leaves. Overall, the aim is to develop a method for an accurate automated early detection of cotton disease which in turn could play a key role in minimizing crop losses and ensuring timely control.

1.3 Proposed Solution

The proposed solution involves the use of a machine learning model in particular Convolutional Neural Networks which have been fed with a specific dataset after training and validation, for photos showing cotton sickness. Tuning strategies and the use of transfer learning will also be ways to optimize Model for job response. This is remarkable because it not only identifies the diseases, but also sorts out the extent of Whitefly damage on cotton leaves. The proposed method uses pioneering machine learning techniques in an effort to offer a consistent, computerized tool for the early and accurate diagnosis of cotton infection. This will mainly aid with swift disease supervision, which could decrease crop losses and increase agricultural output. The general goal of this study is to modernize the way cotton diseases are recognized and treated, which will have a key impact on the agricultural sector.

1.4 Working Principle

We will be using Convolutional Neural Network Architecture in this project. CNNs have been broadly used in classification of image, detection of object, and segmentation.

- Data Collection
- Preprocessing of Data
- Selection of Model
- Training of Model
- GUI presentation

1.4.1 Data Collection:

Two publicly available datasets were used for the project from kaggle [1]. Both datasets were merged to produce a large dataset having multiple classes of diseases.

1.4.1.1 Dataset for the Model:

Publicly available datasets were used. Dataset had 6 classes of diseases and approximately 2500 images. The classes were Aphids, Armyworm, Bacterial blight, Healthy, Powdery Mildew, Target spot.

```
Aphids: 600
Army worm: 202
Bacterial Blight: 600
Healthy: 230
Powdery Mildew: 600
Target spot: 245
Whitefly: 46
```

Fig 1: Dataset composition

1.4.2 Dataset Preprocessing:

The data pre-processing has five main stages.

1.4.2.1 Resizing:

The images have to be downsized to an even size that fits the model's needed input dimensions. You should resize the photos to the 64x64 resolution since the image_size hyper parameter is set to that value in this instance.

1.4.2.2 Normalization:

To standardize the image pixel values so they fall inside a given range. Re-scaling the pixel values that are in the range from 0 to 1 is typically the technique involved.

1.4.2.3 Data Augmentation:

To improve the variety of training data and, consequently, the capacity of model for generalization, data augmentation techniques are implemented. Frequently used methods for enhancing data include brightness tweaks, rotations, flips at random, and zooming.

1.4.2.4 Converting to Tensor:

Convert the pictures into TensorFlow tensors from Numpy arrays or other image formats. TensorFlow tensors are expected as input by the model.

1.4.2.5 Batching:

Before providing the pictures to the model, arrange them into batches. Usually, this is done to increase training-related computational efficiency.

1.4.3 Model Selection:

The CNNs are made especially to take meaningful elements out of pictures. CNNs are a good fit for our project because it entails using image analysis to identify diseases in cotton plants. CNNs immediately pick up on an image's hierarchical feature representations. This implies that the network's ability to recognize tiny patterns suggestive of diseases in cotton plants increases as it processes the image through successive layers, allowing it to capture ever-more abstract and complicated data. Spatial invariance is the ability of CNNs to identify patterns independent of where they are in the image. This is important for cotton crop disease identification since the tormented areas may show up in different places on the image.

1.4.4 Model Training:

This preprocessed and supplemented information is used to train the CNN, making it easier to identify complex patterns linked to both healthy and unhealthy cotton plants. By utilizing pre trained weights transfer learning with CNN speeds up convergence and may enhance the models overall performance.

1.4.5 Model Architecture:

Deep Learning Specialized Architecture that is Convolutional Neural Networks (CNNs) are made to comprehend visual data. A CNN is composed of three primary layers that are pooling, fully connected, and convolutional. The convolutional layers employs convolutional process to extract significant characteristics, such as edges or textures by putting the filter to input. The acquired features are subsequently down sampled using pooling layers to minimize their dimensionality preserving the most important information, these layers are often stacked with many convolutional and cooling layers, arranged one

after another, to detect more complex patterns. By deciphering these high-level patterns, the fully linked layers ultimately complete the classification or regression task. Regularization methods such as dropout are widely applied in academic research and practical applications to minimize overfitting and make the generalization performance of machine learning models better. CNNs are frequently used for photo classification, object recognition, and many other computer vision problems. They effectively train hierarchical representations of visual data.

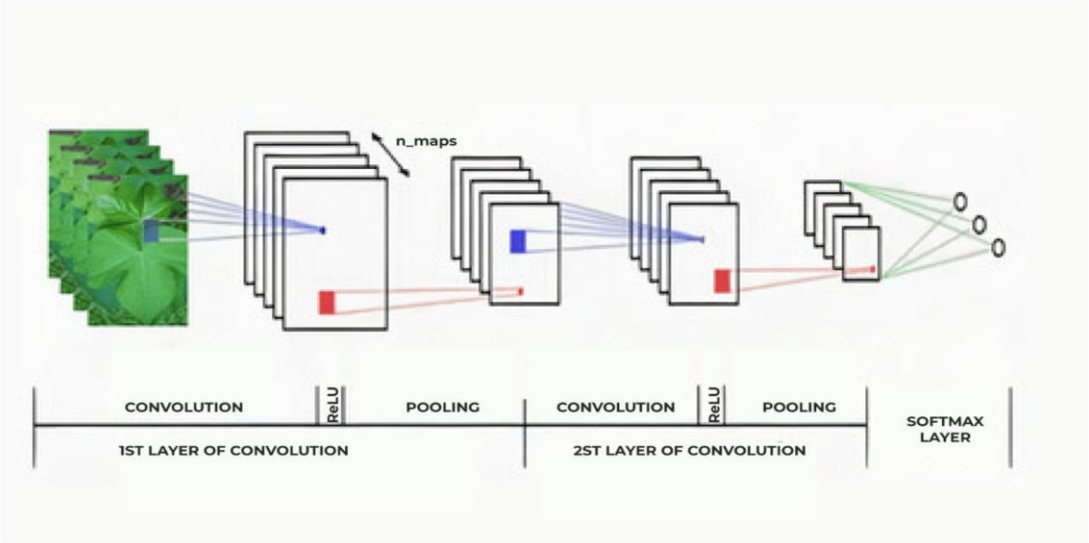


Fig 2: Architecture of CNN

1.4.6 Hyper Parameters:

Variable	Value
Learning Rate	0.001
Loss Function	Categorical Cross-Entropy
Batch Size	64
Epoch	100

1.4.7 GUI presentation:

The CNN model may be easily interacted with by stakeholders thanks to the user-friendly interface. For non-technical users, like farmers or agricultural specialists, the system becomes more user-friendly by enabling them to upload photographs and obtain predictions.

1.5 Objectives

1.5.1 Main Objectives:

"Creating a cutting-edge web interface with Django for the front end and CSS and HTML for the back end, as well as a deep learning-based system for the accurate and efficient detection of cotton disease. The research aims to transform the conventional methods of disease detection in cotton farming."

1.5.2 Academic Objectives:

- The creation of a web application that uses deep learning to detect cotton illness.
- To boost cotton yields.
- To create a project that enhances the interests of the agricultural industry.

1.6 Scope

Given its substantial impact on agricultural output, cotton diseases require prompt and accurate detection. Though useful, traditional detection techniques are limited in terms of labor intensity, timeliness, and precision. To solve this specific problem, this study investigates the automated diagnosis of cotton illness using the models of deep learning, more especially CNNs. The ultimate objective is to support the development of an automated, accurate, and early detection method for cotton illnesses. This will help to mitigate these diseases in a timely manner and may even minimize crop los

1.7 Deliverables

1.7.1 Web Based Application

The deliverables include a web based application named as **The CottonGuard AI** and has following features

1.7.1.1 Image uploading

The user can upload image of any size by clicking choose file button.

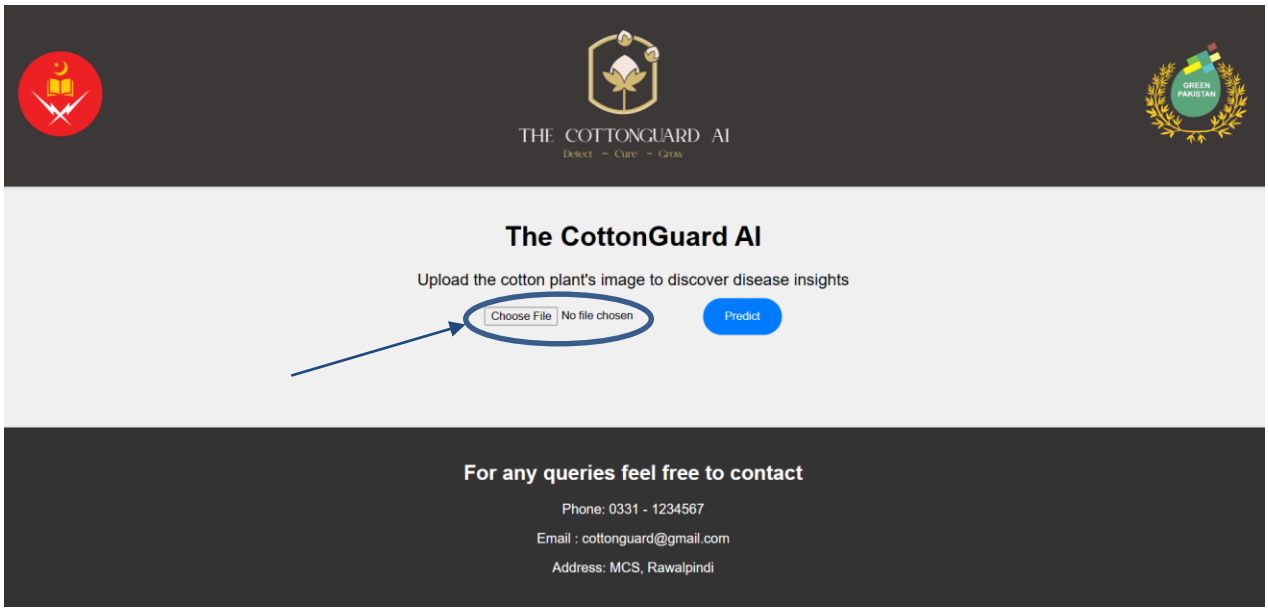


Fig 3: User Interface

1.7.1.2 Disease Detection/ Prediction

After uploading image, the user can view the predicted result by clicking predict button. After clicking the predict button the image will be displayed along with the disease name.

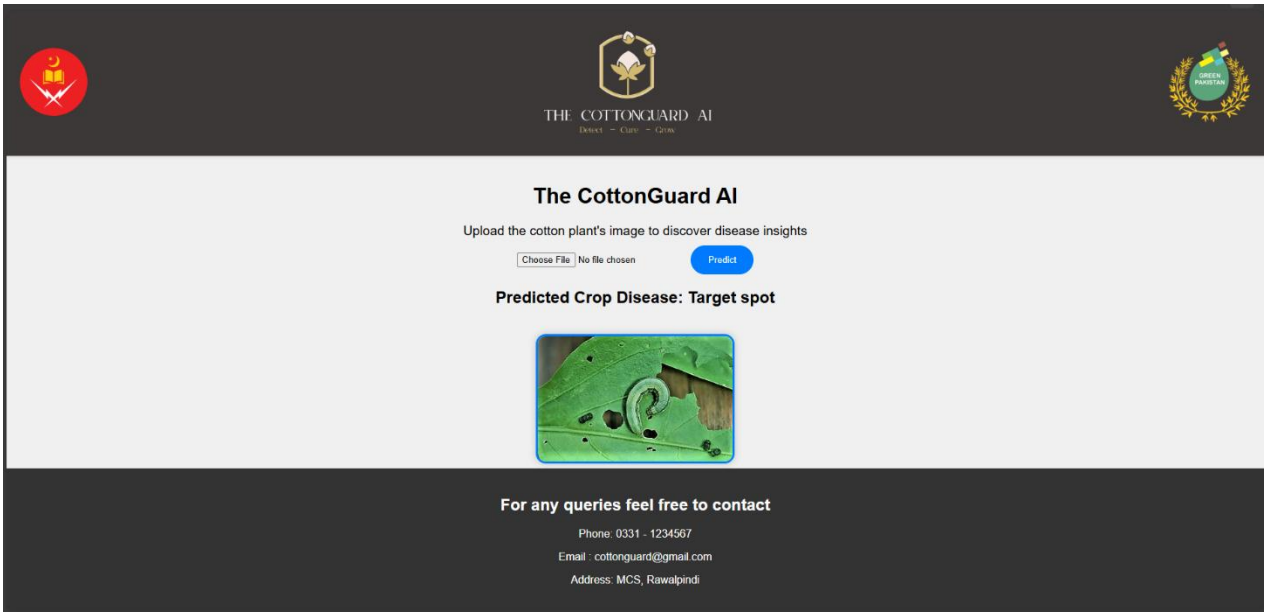


Fig 4: Predicted Disease name

1.8 Relevant Sustainable Development Goals

What is the Locally Relevant Socio-Economic Issue that the Project Addresses?

Justify how particular SDG is related to your FYP?

SDGs Linked to our Project:

Goal no. 3: Good Health and Well-being

Goal no. 12: Responsible Consumption and Production

Pakistan is one of those countries whose economy heavily rely on its agricultural sector. Keeping in mind the importance and role of cotton crop not only in agricultural sector but in economic sector as well, the project addresses the issue of cotton crop disease detection. As explained earlier, timely detection can help the farmers make an informed decision which will ultimately help in increase in production of cotton crop benefitting both farmers (well-being of farmers) and our economic sector as well.

Similarly it also addresses the issue of responsible production by helping and easing out farmers in making timely decision to help them grow their crop in a more efficient and productive way.

1.9 Thesis Structure

Chapter no.2 contains the review of the literature and work done in this field.

Chapter no.3 contains the design and interfacing.

Chapter no.4 contains functional and non-functional requirements

Chapter no.5 contains code analysis and evaluation

Chapter no.6 contains the conclusion of the project.

Chapter no.7 outlines the additional future tasks that can be done in order to commercialize this idea.

Chapter 2: Review of Literature

2.1 Overview

The Deep learning technologies are gaining traction in a number of industries, including manufacturing, transportation, healthcare, and agriculture. But the key emphasis of our research will be on using deep learning methods to identify cotton illnesses in the agriculture sector. Being the most vital cash crop, cotton is crucial to the economies of many nations, including Pakistan. In order to ensure a strong, effective, and timely reaction, this research intends to investigate and build deep learning models specifically suited to the identification and control of major cotton diseases.[2].

2.2 Historical background

With the development of deep learning algorithms, research on cotton disease identification has evolved dramatically. Traditional image processing was used in the beginning, and then machine learning techniques that necessitated intensive feature engineering were used. Using design such as CNN allowed for a more automatic and reliable transition to deep learning, improving accuracy and efficiency.

Nevertheless, the first deep learning model had problem with overfitting and required large amount of labelled information. Through data augmentation, transfer learning, and enhanced training algorithms research later solved these constraints.

The transition from conventional techniques to deep learning in the diagnosis of cotton disease underscores the crucial advancements in agricultural technology, consistently enhancing precision and productivity in this vital domain. [3]

2.3 Existing Work Done

For the agricultural secto to increase crop production capacity by promptly detecting and managing illness in their early stages, a plant disease diagnosis system must be put in place. The suggested method analyzes leaf sample and look for early illness indication using image processing techniques. To precisely identify the areas affected by illness, the research uses thresholding techniques to split the afflicted regions int leaf pictures. In addition the research indicates Gray-Level co-occurance Matrix (GLCM) characteristics that were taken frn the leaf segments affected by disease. The type of illness infecting the cotton plants is classified using these GLCM parameters. Farmers may safeguard their crops and increase output by reducing the spread and impact of specific illnesses by precisely diagnosing them and making educated decisions. [4]

The study highlights how important it is to use GLCM-based analysis and image processing techniques for cotton disease identification because they offer a quick and non-intrusive way to determine whether the diseases are present in cotton plants. Timely actions, including tailored treatments or preventive measures, are made possible by early detection and result in improved crop protection and higher yields. [5]

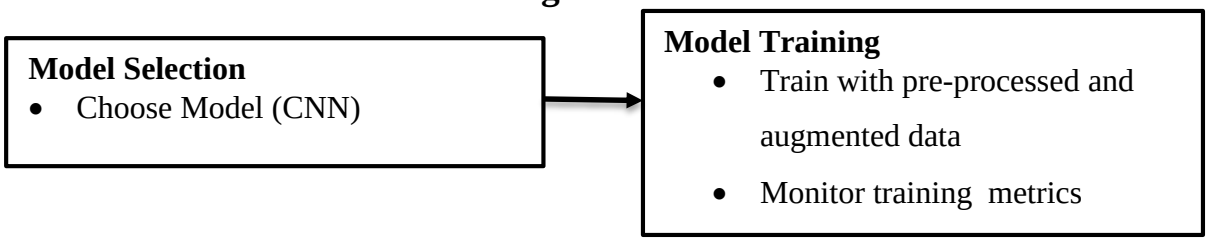
According to the article by M.Zekiwos [6], obtaining field photos with digital cameras and cell phones is a crucial first step in creating this model. First, digital cameras and smart phones are used to take pictures of cotton leaves in the field that may contain pests or diseases. After that, pre-processing methods are used to get the captured images ready for additional examination. Following pre-processing, CNN is provided with images in order to extract the image features. To effectively convey the photographs, CNN takes out the most pertinent information from the pictures. The most suited representations 8 for

every image are then found by applying image analysis algorithms to these extracted features. The project generates training and testing datasets for disease and pest identification based on the retrieved features. New photos are categorized into the appropriate classifications of pests and diseases using a trained knowledge base that has been established. This classification helps farmers control and manage the hazards to their crops by enabling quick and accurate diagnosis as well as prompt response.

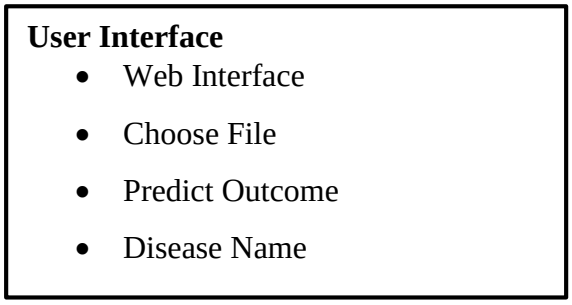
Chapter 3: System Design and Interfacing

3.1 System Decomposition

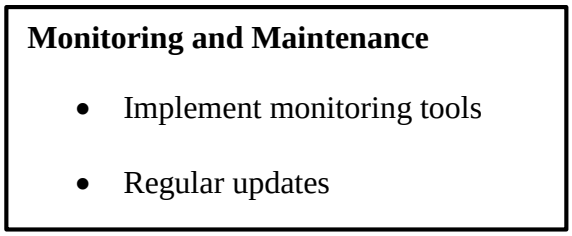
1.5.1 Model Selection and Training



3.1.2 User Interface



3.1.3 Monitoring and Maintenance



3.2 Design Rationale

Convolutional Vision Transformer (CNN) Selection:

- **Rationale:** CNNs are made expressly to extract useful features from pictures. CNNs are a good fit for our project because it entails using image analysis to identify diseases in cotton plants. CNNs immediately pick up on an image's hierarchical feature representations. This

implies that the network's ability to recognize tiny patterns suggestive of diseases in cotton plants increases as it processes the image through successive layers, allowing it to capture ever-more abstract and complicated data. Spatial invariance is the ability of CNNs to identify patterns independent of where they are in the image. This is crucial for cotton plant disease detection since the afflicted patches may show up in different places on the picture.

Data Processing and Augmentation:

Rationale: To ensure that different disease states are represented, data collection entails obtaining a wide set of photographs of cotton plants. Preprocessing brings the data into compliance with CNN specifications by means of resizing and standardization. By increasing the heterogeneity of the dataset, augmentation improves the generalization abilities of the model.

Model Training and Transfer Learning:

Rationale: The pre-processed and supplemented information is used to train the CNN, making it easier to identify complex patterns linked to both healthy and unhealthy cotton plants. By utilizing pre-trained weights, transfer learning with CNN speeds up convergence and may enhance the model's overall performance.

User Interface:

Rationale: Stakeholders can engage with that CNN model on an intuitive platform. The system becomes more user friendly for non-technical users, such as farmers or agricultural specialists, by enabling them to upload photographs and obtain forecast.

Monitoring and Maintenance:

Rationale: Overtime, the efficacy of the CNN model is ensured by the ongoing performance monitoring. Early detection of deviations or performance degradation is facilitated by the

implementation of logging and monitoring technologies. Frequent updates with fresh data improves the model's capacity to adjust to evolving cotton plant disease patterns.

3.3 Design of Data

3.3.1 The Data Flow Diagrams

The Level-0

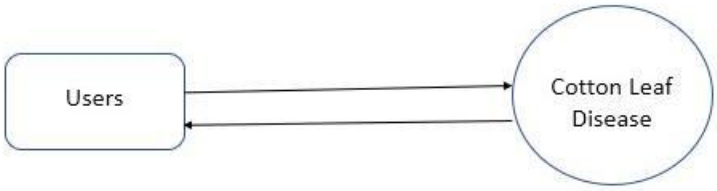


Fig 5: The Level 0 DFD

The Level-1

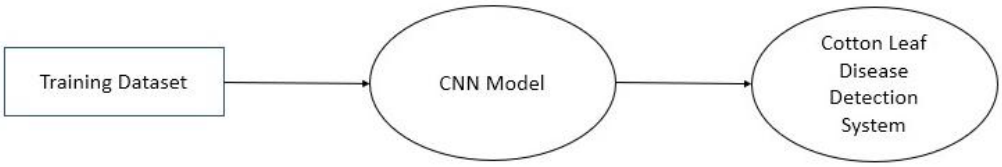


Fig 6: The Level 1 DFD

The Level-2

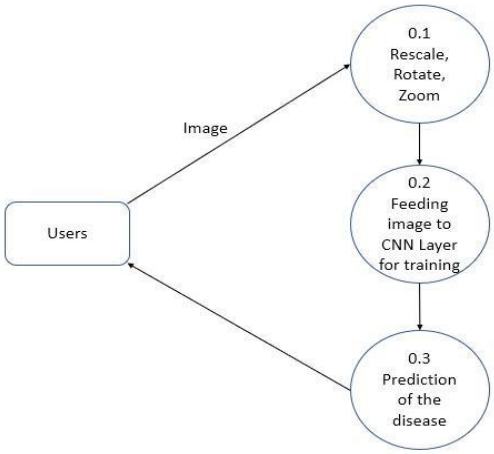


Fig 7: The Level 2 DFD

3.3.2 ERD

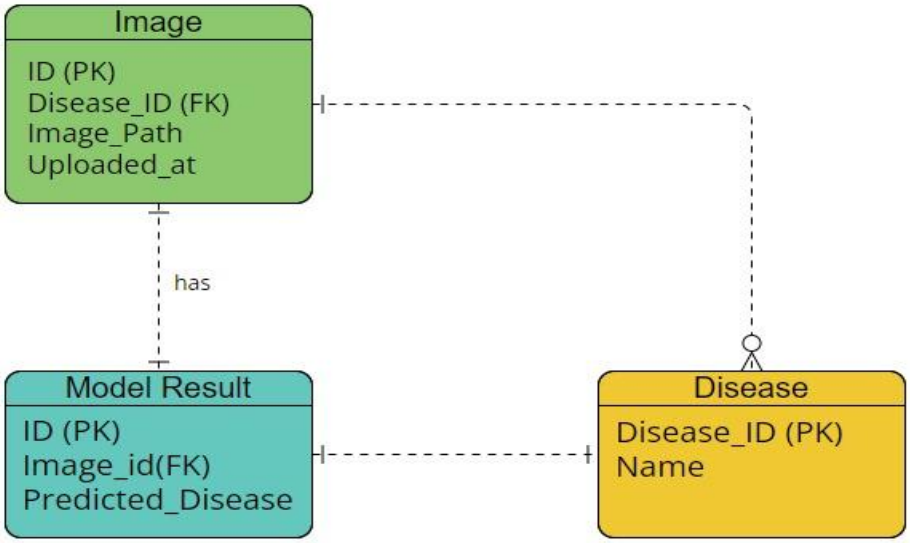


Fig 8: ERD

3.3.3 The Use-Case Diagram

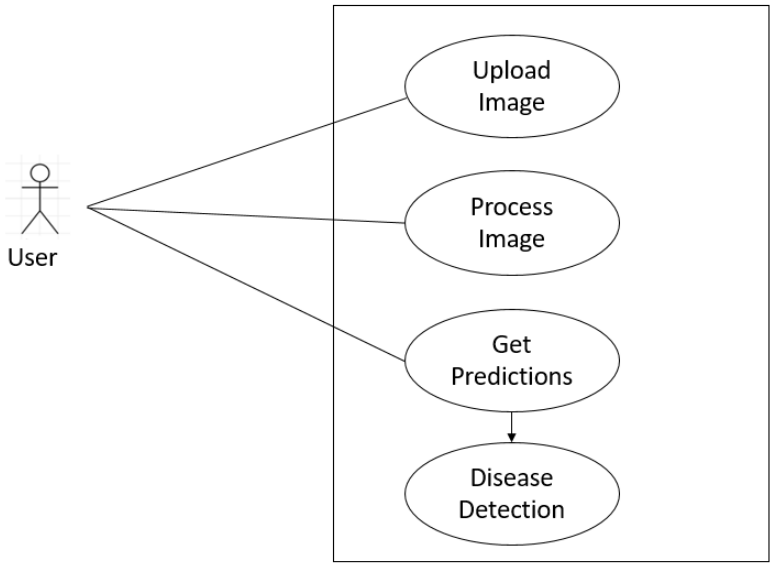


Fig: 9-a Use Case Diagram

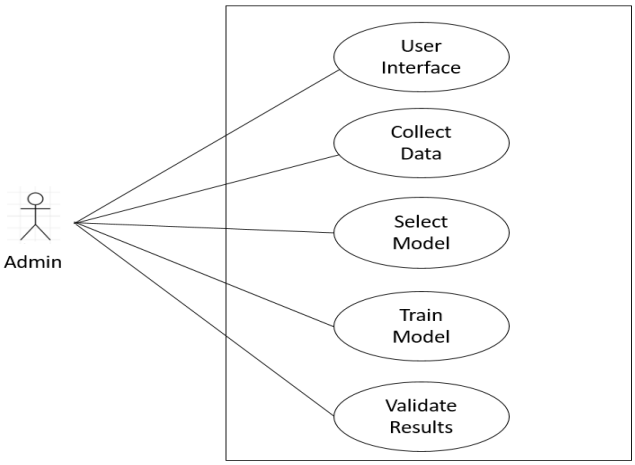


Fig: 9-b Use Case Diagram

3.3.3 The Sequence Diagram

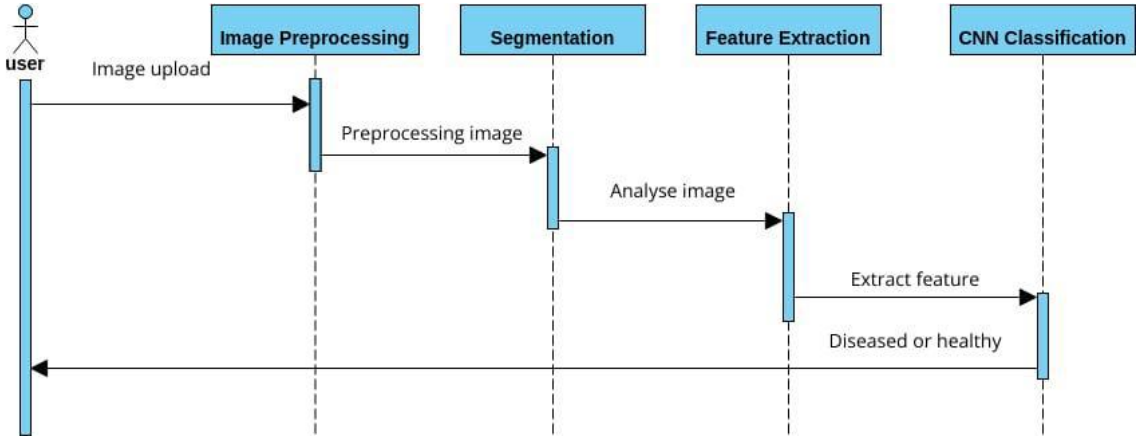


Fig 10: The Sequence Diagram

3.3.4 Key Design Principles

Hybrid Architecture:

Rationale. Convolutional layers advantages for creating creating spatial hierarchies and capturing long range dependencies are combined in CNNs designs. Local and global context awareness are best balanced by this design decision.

Transfer Learning for Efficiency:

Rationale. Using transfer learning with CNN models that have already been trained speeds up training and may improve models efficiency. This is very helpful when handling little labeled data for a given task.

User-Centric Design:

Rationale. The project adheres to a user-centric strategy when it incorporates a user interface. This design decision takes into account the demands of the end user and gives

them a useful way to use the disease identification system without needing extensive technical knowledge.

Adaptability and Continual Improvement:

Rationale. The system's ability to adjust the changing circumstances is guaranteed by its modular design in conjunction with monitoring and maintainance procedures. Consistent enhancement through fresh data enables the CNN model to maintain its applicability and precision over time.

Technology Alignment:

Rationale: CNN was selected in accordance with the most recent developments in deep learning architectures for computer vision applications. This guarantees that the project takes advantage of the most recent methods for classifying images.

3.4 Preparing Dataset

A collection of examples sharing a common characteristic is called a dataset. The dataset shapes a machine learning algorithm going future. The more data that is provided, the more efficient it becomes. To build this model, we used about 2500 images of cotton crop leaves.

Each and every image has a fixed size of 64 by 64 pixels. Sets of the dataset are separated out for training, validation, and testing.

3.3.1 Model Architecture

- We will be using Convolutional Neural Network Architecture in this project. CNNs have been extensively utilized for tasks such as classification of image, detection of object, and segmentation.
- A CNN model is defined via Keras.

- The CNN model comprises of three convolutional layers with growing sizes of filter (32, 64,128).
- Each convolutional layer is trailed by normalization of batch and maximum-pooling.
- The output is sent through the 2 completely connected layers (dense layers) with the ReLU activation.
- The final layer has as many neurons as there are classes, with softmax activation for multi-class classification.

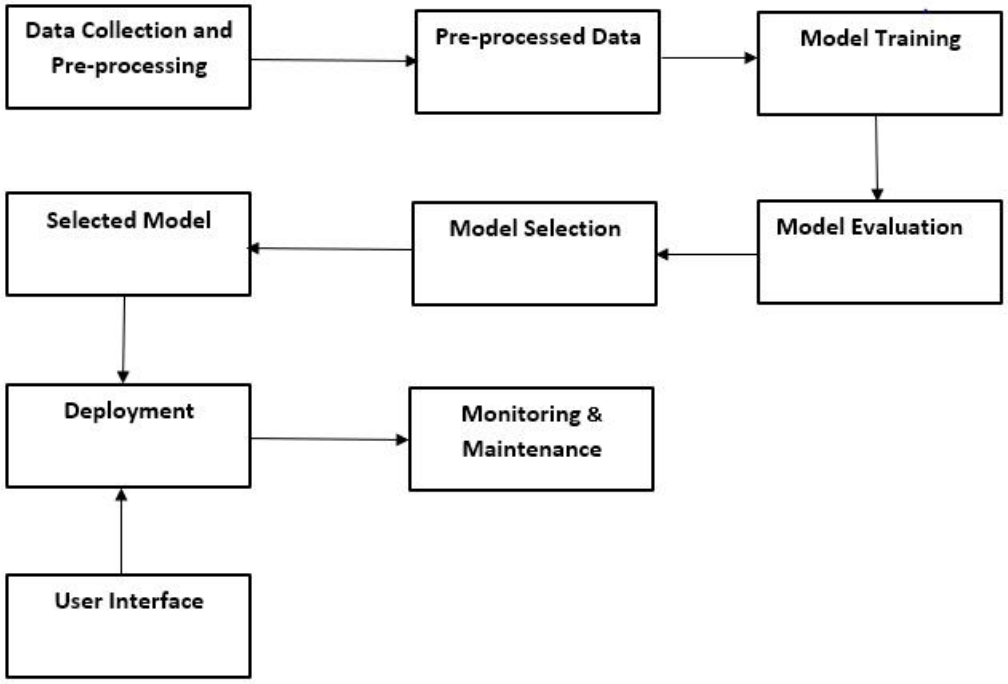


Fig 11: System Architecture

3.1.2 Model Decomposition

3.1.3 Compilation and Training of Model

- The CNN model has been compiled with Adam optimizer and categorical cross-entropy loss.
- It is trained on the training data (X_train and y_train_encoded) for 100 epochs.
- Validation data (X_val and y_val_encoded) used for monitoring performance during training.

3.1.4 Model Evaluation

- The test accuracy is computed using the testing data (X_test and y_test_encoded).
- Accuracy of Training and Validation curves are plotted to show model performance during training.
- Curves of Training and validation loss are also plotted.

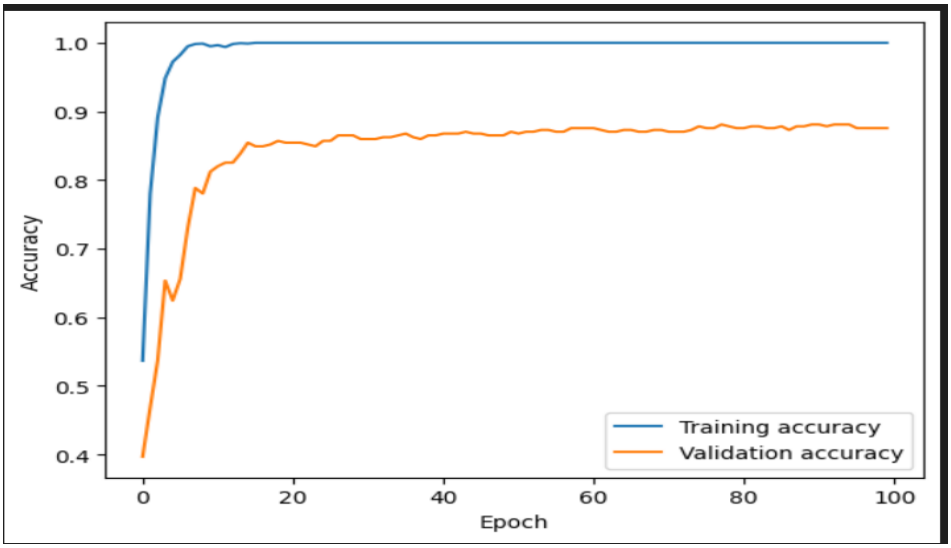


Fig 12: Accuracy of Training and Validation Curves

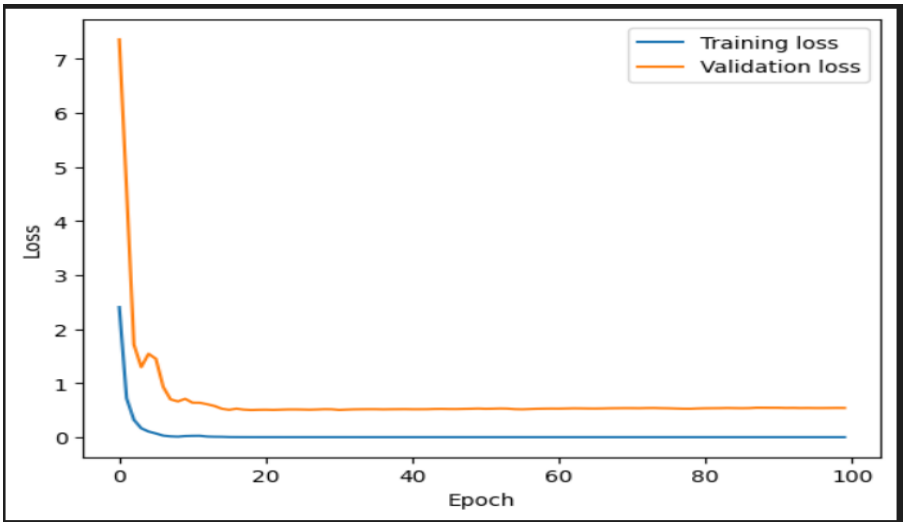


Fig 13: Loss of Training and Validation Curves

3.1.5 Confusion Matrix

- A confusion matrix is computed using the predictions of model on test data.
- The matrix shows how well the model predicts each class.
- It helps evaluate performance of model as per true positives, true negatives, false positives, and false negatives.

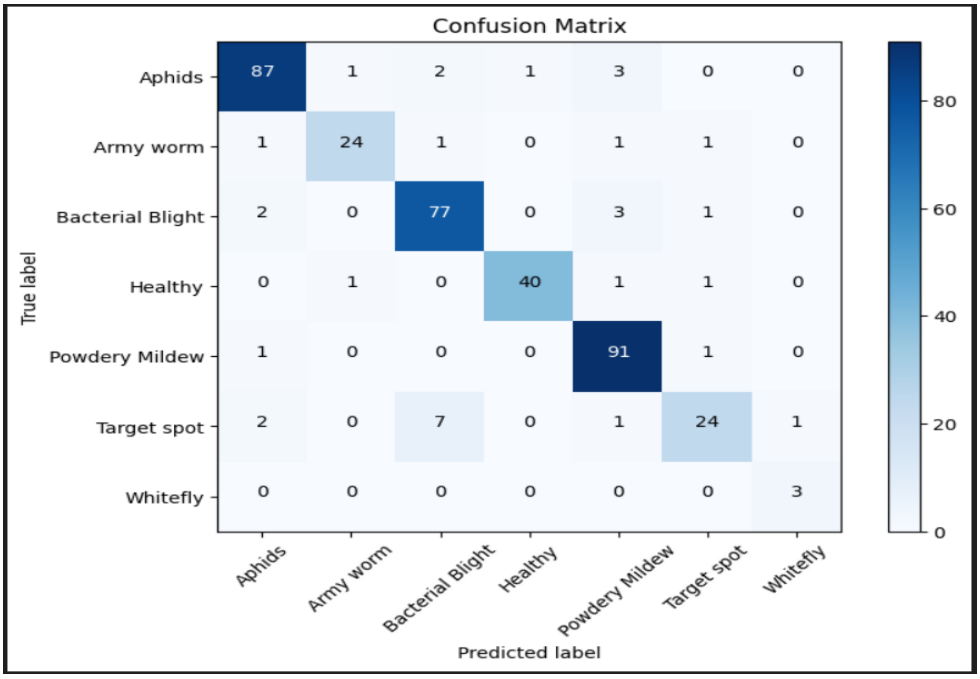


Fig 14: Confusion Matrix

3.1.5 Saving the Model

- The Model after training is saved as trained_model.h5 file.

3.1.6 Error Analysis:

In the framework of deep learning, error analysis is the process of looking at mistakes that your model has made. It entails delving deeply into the instances in which the model made erroneous predictions, comprehending the potential causes of these errors, and formulating correction plans.

3.1.6.1 Importance of Error Analysis

- **Recognizing the Limitations of the Model:** No model is flawless. Error analysis can help you identify areas in which your model is lacking. Guiding Future Work: You can decide which parts of your model or data to improve next by knowing where errors are originating from.

- **Preventing Overfitting:** It can assist in determining whether the model is overfitting to specific types of mistakes or data.
- **Improving Generalization:** Recognizing error trends can reveal model biases, which when fixed, can enhance generalization.
- **Developing Trust**: Providing a thorough error analysis to users of your model will help them feel more confident.

3.1.6.2 Error Analysis of the Model

- **Target Spot:** Misclassified 7 times as bacterial blight and bacterial blight misclassified 1 time as target spot.
- **Aphids and Bacterial Bight Confusion:** Aphids are 3 times misclassified as powdery mildew. Similarly bacterial blight is misclassified 3 times as powdery mildew.
- **High precision:** Whitefly was classified correctly on all 3 occasions whereas powdery mildew is misclassified across only 2 other classes
- **Misclassification Pattern:** Aphids, Army Worm and Target Spot are misclassified across 4 other classes.
- **Potential Improvements:** The misclassified classes indicate that the model needs further investigation to improve its performance.
- **Overall Performance:** The majority predictions on the diagonal indicates that the model has performed well.

3.1.7 Using Django, HTML & CSS for webpage

- Using Django as backend development in order to connect with CNN model that generates the result and using HTML and CSS for Front end development.

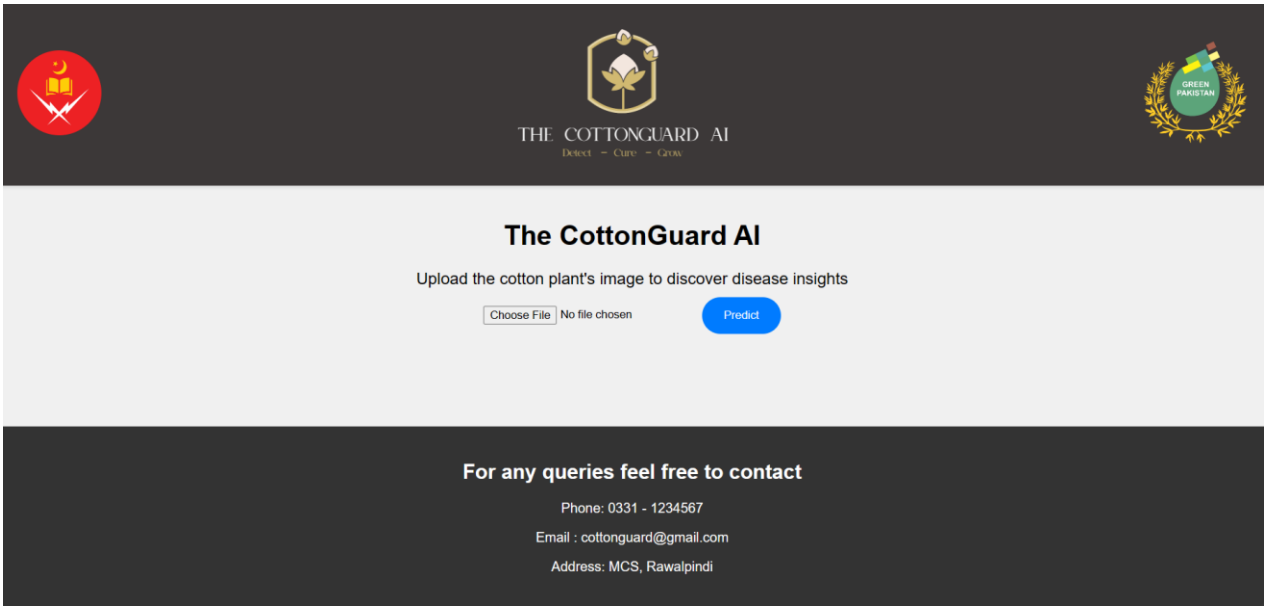


Fig 15: User Interface

Chapter 4: Functional and Non-Functional Requirements

4.1 Functional Requirements

4.1.1 Operating Environment

The system will function within a Google Co lab setting.

4.1.1.1 Hardware Requirements

The system is made to function on typical hardware setups that are frequently used in offices and healthcare facilities.

- The hardware specifications include:

  Processor: a CPU with dual cores or more for effective processing.

  Memory (RAM): 8 GB of RAM minimum required for efficient operation.

  Storage: Sufficient storage space to accommodate data and application files.

- Making use of the scalable GPU resources at Google Co lab. The system can run on desktop and laptop computers with different hardware configurations and is platform-agnostic, giving end users freedom.

4.1.1.2 Software Requirements

- Google Co lab
- Deep learning libraries (TensorFlow, PyTorch)
- Data Set
- Cloud Based Code
- Good Back Up

4.1.1.3 Operating System

- Interoperability with Linux and Windows environments.

4.1.1.4 Integration Corporate or Regulatory Policies

All relevant business and regulatory rules, particularly those pertaining to data security and privacy, must be complied with by the system.

4.1.1.5 Interface Requirements

To satisfy the needs of the user, the interface will comply with the following requirements.

- Easy to use and user friendly interface.
- The web interface enables users to communicate with the system and comprehend its features.
- The interface will incorporate both graphical outputs and user inputs.
- **GPU Interface**: Google Colab's GPU resources are integrated for effective model inference and training.

4.1.1.6 TensorFlow and PyTorch Interfaces: compatibility for training and developing models with these deep learning frameworks.

4.1.1.7 Communications Interfaces

Web-Based API: Makes it easier for the user interface and backend to communicate, allowing for smooth data transfer and result display.

4.2 Nonfunctional Requirements

4.2.1 Performance Requirements

4.2.1.1 Inference Speed

It is imperative that results be furnished in a fair amount of time to guarantee timely diagnosis and intervention.

4.2.1.2 Scalability

More and more requests should be handled by the system, especially when large-scale image recognition activities are involved.

4.2.1.3 Reliability

The system ought to provide dependable, exact, and accurate outcomes.

4.2.1.4 Performance

The system ought to function as planned and anticipated.

4.2.1.5 Safety Requirements

- Misclassification Prevention
- Data Loss Avoidance
- Safety of Data and Privacy
- Model Accuracy and Reliability
- Error Handling and Reporting

Chapter 5: Code Analysis and Evaluation

```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import os
import random
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import confusion_matrix
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.optimizers import Adam

# Define your dataset directory
path = '../MLexp/cotton_dataset/'

# Get the list of classes (subdirectories) in the dataset directory
classes = os.listdir(path)
print('Classes:', classes)

# Load images and their corresponding labels
images = []
labels = []
for label in classes:
    label_path = os.path.join(path, label)
    image_files = os.listdir(label_path)
    for image_file in image_files:
        image_path = os.path.join(label_path, image_file)
        image = plt.imread(image_path)
        # Resize the image to 64x64x3
        image = tf.image.resize(image, [64, 64])
        images.append(image)
        labels.append(label)

# Convert lists to numpy arrays
images = np.array(images)
labels = np.array(labels)

# Split the data into training, validation, and testing sets (70%, 15%, 15%)
X_train, X_temp, y_train, y_temp = train_test_split(images, labels,
test_size=0.3, random_state=42)
```

```
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.5,
random_state=42)

# Convert class labels to one-hot encoded vectors
label_encoder = LabelEncoder()
y_train_encoded =
tf.keras.utils.to_categorical(label_encoder.fit_transform(y_train))
y_val_encoded = tf.keras.utils.to_categorical(label_encoder.transform(y_val))
y_test_encoded = tf.keras.utils.to_categorical(label_encoder.transform(y_test))

# Define model
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', padding='same', input_shape=(64, 64,
3)),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Conv2D(64, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Conv2D(128, (3, 3), activation='relu', padding='same'),
    BatchNormalization(),
    MaxPooling2D((2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dense(len(classes), activation='softmax')
])

# Compile model
model.compile(optimizer=Adam(lr=0.001), loss='categorical_crossentropy',
metrics=['accuracy'])

# Train model
history = model.fit(X_train, y_train_encoded, batch_size=64, epochs=100,
validation_data=(X_val, y_val_encoded))

# Evaluate model on test data
test_loss, test_acc = model.evaluate(X_test, y_test_encoded)
print('Test accuracy:', test_acc)

# Plot training and validation accuracy
plt.plot(history.history['accuracy'], label='Training accuracy')
plt.plot(history.history['val_accuracy'], label='Validation accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
```

```
plt.show()

# Plot training and validation loss
plt.plot(history.history['loss'], label='Training loss')
plt.plot(history.history['val_loss'], label='Validation loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

# Generate and save confusion matrix without seaborn
y_pred = np.argmax(model.predict(X_test), axis=1)
conf_matrix = confusion_matrix(np.argmax(y_test_encoded, axis=1), y_pred)

plt.figure(figsize=(8, 6))
plt.imshow(conf_matrix, interpolation='nearest', cmap='Blues')
plt.title('Confusion Matrix')
plt.colorbar()
tick_marks = np.arange(len(classes))
plt.xticks(tick_marks, classes, rotation=45)
plt.yticks(tick_marks, classes)

for i in range(len(classes)):
    for j in range(len(classes)):
        plt.text(j, i, str(conf_matrix[i][j]), horizontalalignment="center",
color="white" if conf_matrix[i][j] > conf_matrix.max() / 2 else "black")

plt.tight_layout()
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.savefig('confusion_matrix.png')

# Save the trained model in .h5 format
model.save('trained_model.h5')
```

5.1 Code Analysis

These lines import necessary libraries for various functionalities:

- **numpy (np):** offers manipulating arrays and performing numerical operations.
- **matplotlib.pyplot (plt):** utilized to produce plot-like displays.
- **tensorflow (tf):** The main library used in deep learning model construction and training.

- **ImageDataGenerator (from tensorflow.keras.preprocessing.image):** Used not in this code, but for data augmentation (making different versions of photos).
- **os:** utilized to communicate with the operating system (directories and files are accessed).
- **random:** employed for operating system communication (directories and files are accessed).
- **train_test_split (from sklearn.model_selection):** divides data into sets for testing, validation, and training.
- **LabelEncoder (from sklearn.preprocessing):** transforms labels from strings to numbers.
- **confusion_matrix (from sklearn.metrics):** evaluates the performance of the model by computing and plotting the confusion matrix.
- **BatchNormalization (from tensorflow.keras.layers):** data inside a mini-batch is normalized.
- **Sequential (from tensorflow.keras.models):** used to stack layers to create a sequential model.
- **Conv2D, MaxPooling2D, Flatten, Dense (from tensorflow.keras.layers):** Dense layers, convolutional, pooling, and flattening are employed in the CNN architecture.
- **Adam (from tensorflow.keras.optimizers):** It is the optimizer that modifies the model's weights and biases of training.

Data Loading and Preprocessing:

- The code begins by loading the labels and images from the directory (../MLexp/cotton_dataset/).
- Every image is scaled to 64 by 64 pixels, which is a set size.
- There are three sets of the dataset: training, validation, and testing.

Model Architecture:

- Keras is used to describe CNN model.
- Three convolutional layers with progressively larger filters (32, 64, and 128) make up the model.
- Batch normalization and max-pooling come after each convolutional layer.
- Two dense layers that are fully linked and have ReLU activated receive the flattened output.
- The softmax activation is used by last layer for classification of multi-class and contains as many neurons as there are classes.

Compilation and Training of Model:

- To compile the model The Adam optimizer and categorical cross-entropy loss have been used.
- For 100 epochs, it is trained using the training set (X_train and y_train_encoded).
- X_val and y_val_encoded, the validation data, are utilized to track training performance.

Evaluation:

- X_test and y_test_encoded, the testing data, are used to calculate the test accuracy.
- Plotting training and validation accuracy curves allows one to check-out that how this model performs during training.
- Plotting of the training and validation loss curves is also done.

Confusion Matrix:

- Using the prediction of models on test data, a confusion matrix is constructed.
- How effectively the model predicts each class is displayed in the matrix.
- It aids in evaluating the performance of models in perspective of true positives, true negatives, false positives, and false negatives,

Model Saving:

- The trained model gets stored in a file called trained_model.h5.
- Random Predictions: Using the test data, the code creates five random indices.
- Random samples from the test set are correlated with these indices.
- Additional analysis can be done on the model's predictions for these samples.

In summary, this algorithm uses photos to train a CNN model for classifying cotton crop diseases and assesses the model's performance. The model's efficacy for various classes is revealed via the confusion matrix. Future forecasts can be made using the preserved model.

Chapter 6: Conclusion

In this thesis, we discussed the importance of timely and accurate detection of disease cotton crop using the deep learning methods specifically CNNs. It explored the efficacy of Convolutional Neural Networks (CNNs). CNN models were honed using transfer learning and tuning techniques.

Cotton exports significantly contribute to the economies of countries like Pakistan. Traditional methods of disease detection often rely on visual inspection, which can be challenging due to the wide range of illnesses affecting cotton leaves. However, implementing automated disease detection can help sustain cotton production and improve overall agricultural outcomes.

6.1 Effectiveness of the Deep Learning Model Used

The thorough assessment and optimization of several deep learning model has produced very convincing outcomes, confirming the dependability and effectiveness of the model in identifying cotton illnesses. This study's careful choice and application of deep learning algorithms have often shown outstanding accuracy, up to 91%.

Chapter 7: Future Work

7.1 Improvement in Current Model

This work brings up many interesting possibilities. Further study is required to optimize these models specifically for the purpose of cotton disease detection, given the observed improved performance of CNNs. For example, we can investigate how different architectural changes or training approaches would be able to improve their efficiency. The integration of self-supervised learning methods with Vision Transformers would also be interesting to investigate, as this could enhance the model's capacity to derive meaningful representations from the input.

Furthermore, it would be beneficial to test and validate these models on **bigger** and more **diverse datasets**, since the current dataset was created from **public** sources and might not accurately reflect the range of cotton illnesses. This could include various cotton crop growing environments, various stages of disease progression, and various cotton disease kinds. Integrating additional data modalities, such **thermal** or **hyperspectral imaging**, would also be instructive to see whether they can enhance performance.

7.2 Mobile Application

In future mobile application can also be build do that users can view the results on their mobile phones.

7.3 IoT Integration

Implementing IoT enabled devices for real-time tracking.

Work Cited

[1] www.kaggle.com
[2] www.hindawi.com
[3] Submitted to Uttar Pradesh Technical University
[4] G. Dhingra, V. Kumar, and H. D. Joshi, "Study of digital image processing techniques for leaf disease detection and classification," Multimedia Tools and Applications, vol. 77, pp. 19951-20000, 2018.
[5] S. Tripathy, "Detection of cotton leaf disease using image processing techniques," in Journal of Physics: Conference Series, vol. 2062, no. 1. IOP Publishing, 2021, p. 012009.
[6] M. Zekiwos and A. Bruck, "Deep learning-based image processing for cotton leaf disease and pest diagnosis," Journal of Electrical and Computer Engineering, vol. 2021, pp. 1-10, 2021.