Wolaita Sodo University

College Of Engineering

Final Year Project on

Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis

Submitted By: ***ID***

1. *Ehitnesh Chikelu .....Eng/R/637 /11*
2. *Getachew Birlew….....Eng/R/378/11*
3. *Matiwos Desalegn.....Eng/R/109/11*
4. *Tsegaw Molla.....Eng/Rt/1073/12*

Department Of Electrical and Computer Engineering

Stream: Computer Engineering

Name of the Advisor: Mr. Guyita Gunchato (Msc.)

Submission Date: **15, JUN 2023**

Wolaita Sodo, Ethiopia

Submitted By

NAME OF THE CANDIDATE(S) **Id No.**

1. *Ehitnesh Chikelu .....Eng/R/637 /11*
2. *Getachew Birlew…….....Eng/R/378/11*
3. *Matiwos Desalegn..... Eng/R/109/11*
4. *Tsegaw Molla..... Eng/Rt/1073/12*

A THESIS SUBMITTED TO THE DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING IN PARTIAL FULFILLMENT FOR THE AWARDA OF DEGREE

OF

BACHELOR OF SCIENCE

IN

ELECTRICAL AND COMPUTER ENGINEERING (COMPUTER ENGINEERING)

June 15, 2023

Wolaita Sodo, Ethiopia

DECLARATION

We, the undersigned members of the group, declare that the thesis project entitled Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis submitted as a partial fulfillment of the requirements for the thesis project in the electrical and computer engineering of Wolaita Sodo University is our original work and has not been submitted elsewhere for any academic purpose.

We confirm that:

- All contributions to this project by any individual group member have been acknowledged and appropriately cited.
- All sources of information used in this project have been duly acknowledged, referenced, and cited using the appropriate citation style.
- We have adhered to the ethical and academic standards required for this project, including but not limited to the ethical use of data and software, and have not engaged in any form of plagiarism or academic misconduct.
- Any research involving human or animal subjects was conducted in accordance with the ethical guidelines of the relevant institutional review board.

Name of Students	**Signature**	**Date**
• Ehitnesh Cheklu	_____	_____
• Getachew Birlew	_____	_____
• Matiwos Desalegn	_____	_____
• Tsegaw Molla	_____	_____

CERTIFICATION (Approval sheet)

This is to certify that the Project Report entitled "Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis" that is submitted by this group members in partial fulfillment of the requirement for the fulfillment of thesis project in the degree BSC in ELECTRICAL AND COMPUTER ENGINEERING (Computer Stream) of Wolaita Sodo University, is a record of the candidate's own work carried out by him under my own supervision. The matter embodies in the thesis project is original and has not been submitted for the award of any other degrees.

Advisor: _____Sign_____
Department Head: _____Sign_____

Place: _____
Date: _____

ACKNOWLEDGEMENT

First of all we will like to thank our God, We would like to express our sincere gratitude to Mr. Guyita Gunchato (MSc.), our mentor and guide for this project, for his valuable support and guidance throughout the project. His deep knowledge of deep learning and expertise in the field was instrumental in shaping the direction of our project and refining our methodology. His feedback and suggestions were invaluable in helping us overcome challenges and achieve our objectives. We are grateful for the opportunity to apply our learning and skills to a practical problem and to contribute to the field of deep learning.

Lastly, we would like to acknowledge our fellow group members for their hard work, dedication, and cooperation in completing this project. We would not have been able to achieve the level of success we did without their contributions.

Thank you all for your support, guidance, and encouragement.

Sincerely.

Table of Contents

ABSTRACT

Cotton, an economically significant agricultural product in Ethiopia, faces various challenges in disease and pest identification. In our project, we utilized deep learning techniques, including Convolutional Neural Networks (CNNs), transfer learning, and Keras, to develop a robust model for predicting the freshness or disease status of cotton plants and leaves. We employed a pre-trained ResNet152V2 model as our base model and fine-tuned it using a carefully curated dataset of cotton plant and leaf images labeled as fresh or diseased. To make our model accessible, we created a user-friendly web application using Flask, hosted on a WSGI server. This application allows users to upload images and receive predictions on the freshness or disease status of cotton plants or leaves. To enhance the user experience, we developed an interactive and visually appealing web page using HTML, CSS, and JavaScript, incorporating the Bootstrap framework for a modern and responsive design. We extended the accessibility of our system to mobile devices by converting the model to TensorFlow Lite format, optimized for deployment on resource-constrained platforms. With an Android app developed using Java and Android Studio, users can capture images of cotton plants and obtain predictions on disease status using the TensorFlow Lite model. Thorough testing was conducted on both the web page and the Android app, ensuring functionality and accuracy. These upgrades enhance the usability, accessibility, and practicality of our cotton disease prediction system, demonstrating the potential of IT-based solutions in agriculture.

Keywords: Cotton, Deep learning, Convolutional Neural Networks (CNNs), Transfer learning, Web application, Web Page, TensorFlow Lite, and Android app.

List of Figures

List of Tables

Acronym

API	Application Programming Interface
CSS	Cascading Style Sheets
CNNs	Convolutional Neural Networks
CPU	Central Processing Unit
DBN	Deep Belief Network
DBM	Deep Boltzmann Machine
DSRM	Design Science Research Methodology
GPU	Graphics Processing Unit
GTP-I	Growth and Transformation Plan I
HDF5	Hierarchical Data Format version 5
HTML	Hypertext Markup Language
IDE	Integrated Development Environment
RNN	Recurrent Neural Network
RGB	Red, Green, and Blue
SNNPR	South Nation Nationality People Regional State
USA	United States of America
WSGI	Web Server Gateway Interface
TF	TensorFlow

CHAPTER ONE

1. INTRODUCTION

1.1. Introduction

Cotton plays a vital role in the textile industry worldwide, but it is susceptible to various diseases that can lead to significant economic losses for growers. Detecting and diagnosing these diseases early is crucial to prevent their spread and minimize their impact. In Ethiopia, cotton is a crucial cash crop that contributes to the country's textile industry and provides employment opportunities for millions of people. Therefore, early detection and prevention of diseases in Ethiopian cotton crops are essential for the success of the industry and the livelihoods of farmers.

To address this challenge, we aim to develop a deep learning model that can accurately classify whether a cotton plant's leaf is diseased or healthy. We will leverage Convolutional Neural Networks (CNNs) and transfer learning, specifically utilizing the ResNet152V2 architecture. To train the model, we will use a publicly available dataset from Kaggle, which includes images of healthy cotton leaves and leaves infected with diseases. The training process will take place in the Anaconda environment with the support of Jupyter Notebook.

To enhance the accessibility and usability of our system, we will integrate the trained model into a user-friendly Flask web application. This application will serve as an interface for users to easily upload images of cotton plants or leaves and obtain accurate disease predictions. The Flask app will run on a Web Server Gateway Interface (WSGI) server, ensuring efficient and seamless functionality. Spyder, an IDE included in the Anaconda distribution, will be used to run the Flask app and ensure its effectiveness in delivering precise disease predictions.

Our project aims to develop a deep learning model using CNNs and transfer learning, specifically the ResNet152V2 architecture, to accurately classify whether a cotton plant's leaf is healthy or diseased. The trained model will be saved in the Hierarchical Data Format version 5 (HDF5) file format and integrated into a Flask app running on a WSGI server.

To summarize, our project focuses on developing a deep learning model using CNNs and transfer learning to accurately predict the health status of cotton plant leaves. This model will be integrated into a Flask web application, providing an accessible platform for users to upload images and obtain disease predictions. By aiding Ethiopian cotton farmers in early disease detection and prevention, this project has the potential to contribute significantly to the improvement of the agricultural sector and the

country's economy. Traditional methods of disease detection can be time-consuming and expensive, hindering timely identification and resolution of the issue. Deep learning techniques offer promising solutions, as they have shown remarkable capabilities in complex problem-solving across various domains. Building upon the advancements made in previous studies, our project aims to leverage deep learning to enhance the speed and accuracy of cotton disease detection and classification, empowering farmers to take proactive measures and increase crop yields.

In our upgraded project, we present a comprehensive overview of the enhancements made to our cotton disease prediction system. Initially, we developed a deep learning model using CNNs and transfer learning, specifically the ResNet152V2 architecture, to accurately classify whether a cotton plant's leaf is diseased or healthy. To improve the accessibility of our system, we created a user-friendly web application using Flask, allowing users to easily upload images of cotton plants or leaves and receive precise disease predictions. To enhance the user experience, we developed an interactive and visually appealing web page using HTML, CSS, and JavaScript. By leveraging the Bootstrap framework, we designed a responsive layout with attractive components such as buttons, forms, and tables.

Acknowledging the importance of extending accessibility to mobile devices, we converted our trained model to TensorFlow Lite format. This format is optimized for mobile deployment, offering a smaller size and faster inference speed. To fully realize the mobile application aspect, we developed an Android app using Java and Android Studio. The app enables users to capture images of cotton plants and obtain disease predictions using the TensorFlow Lite model. Thorough testing was conducted on both the web page and the Android app to ensure their functionality and accuracy. Through these upgrades, we have significantly improved the usability, accessibility, and practicality of our cotton disease prediction system. Our efforts contribute to the advancement of agricultural technologies and offer valuable support to cotton farmers in detecting and preventing diseases effectively.

1.2. Statement of the Problem

The cotton plant is affected by different attacks due to temperature fluctuation, diseases, and pests. But in this project, we only focus on the cotton plant diseases part only. Detecting these diseases with bare eyes increased the complexity of cotton crops productivity which decreased the accuracy in identification precision. The previous works on cotton plant diseases in general to develop a model that predicts whether a cotton plant or leaf is fresh or diseased. While the project made significant progress

in distinguishing between fresh and diseased categories, there are certain limitations that need to be addressed in our project. Our project will focus on the following enhancements:

Improved User Interface: The previous project provided a user-friendly web application for image upload and prediction. However, there is room for improvement to make the interface more interactive and intuitive. The upgraded system will enhance the user interface with more informative visualizations, real-time feedback, and user guidance. These enhancements will improve the overall user experience and usability of the application. Users, such as farmers and experts, will be able to easily interpret and analyze the predictions made by the model, leading to better decision-making and disease management.

Integration with Android Application: To facilitate the automatic use of the deep learning model in the field, it is crucial to integrate the model with an Android application. This integration will allow farmers and field workers to directly capture images of cotton plants or leaves using their Android devices, process the images using the deep learning model, and receive instant predictions on disease presence or health status. The Android application will be designed to be user-friendly, lightweight, and capable of offline usage to accommodate field conditions with limited connectivity. By integrating the model with an Android application, the upgraded system will enable convenient and efficient usage in the field, providing farmers with timely information to make informed decisions. By addressing these limitations and upgrading the project to include disease-specific identification, an enhanced user interface with a web page, and integration with an Android application, the upgraded system will provide more accurate and comprehensive disease prediction capabilities for cotton plants. It will offer an improved user experience, allowing users to interpret predictions more effectively, and enable convenient and efficient usage in the field, empowering farmers and field workers with timely information for disease management.

1.3. Background Study

Several studies have attempted to use deep learning techniques to detect and diagnose plant diseases, including cotton plant and leaf diseases. These studies have shown promising results, demonstrating the potential of deep learning in this field. In particular, CNNs and transfer learning have been widely used in image classification tasks, including plant disease diagnosis. In Ethiopia, agriculture is the basis for national economy from which 85% of livelihood and 90% of total foreign trade comes from this agricultural sector [1]. It is believed that Ethiopia is suitable for many farmable crops, and one among

them is cotton. Cotton (Gossypiumspp) is also called “White Gold” and “The King of Fibers.” For growers, processors, exporters, and producing countries, cotton is the earnest point of supply [2]. According to the data of African report, only 428,120 hectares are harvested with the total production value of about $596,000,000 in SNNPRS. Approximately, 18% of crop yield are lost due to different diseases and pests, which result in the loss of millions of dollars worldwide every year.

Even though agriculture is the backbone of Ethiopia, so far, no advanced technologies have been explored in the development of automation in agricultural science and also there are high problems in production and quality due to different diseases and pests. In recent times, the sophisticated emerging technology has attracted many researchers in the field of detection and classification of cotton leaf diseases and pests. In Ethiopia, there are several constraints which reduce the yield and quality of the product. Particularly, identification of potential diseases or pests on Ethiopian cotton is based on traditional ways. There is a wide area of farm suitable for cotton plantation, but only limited project attention is given to cotton crop production. Traditionally, experts detect and identify such plant diseases and pests on bared eyes. Bared eye determination is considered as a loss of low-level accuracy in order to detect any diseases. On high demand, different advanced technologies were aided for structuring the systems to assist nonautomatic recognition of the plant diseases and pests to increase the accuracy for any corrective measures. With the help of advanced technologies, the plant diseases were reduced, thus increasing the productivity which helped to raise the economy via boosting the production. For that reason, the implementation of information technology-based solutions in the sector of agriculture had high level of significance for Ethiopia’s development in monetary, community, and ecofriendly developments by increased cotton crops’ productions. Among different diseases and pests occurred, about 80–90% were on the leaves of cotton [3]. In Ethiopia, it is observed that there might be a fiscal destruction around 16% because of plant syndromes. However, without control measures, it can cause 30%–50% of loss [4]. Cotton diseases and pests are difficult to identify through bared eyes. Therefore, in this project, we will develop a model that predicts if a cotton plant or leaf is fresh or diseased using deep learning techniques.

According to Shuyue [13], they outlined the different formats of graph convolutional neural network. It was prepared to process the uniform electro encephalography data for the purpose of predicting the four classes of motor imaginaries to relate with electro encephalography electrode. They addressed their data with the transformation of 2D to 3D perspectives. The structure was processed through these

dimensional units. A study [14] stated that, in order to utilize the dynamic route of deep learning, they proposed short-term voltage stability. They managed the clustering algorithm to obtain short-term voltage stability to increase the reliability. In [7], it is stated that deep learning technique was applied to identify the leaf diseases in different mango trees. The researchers used five different leaf diseases from various specimens of mango leafs, where they addressed nearly 1200 datasets. The CNN structure was trained with more than 600 images, where 97% are used for training and 1% are used for testing. Remaining 600 images were used to find the accuracy and to identify the mango leaf diseases which showed the feasibility of its usage in real-time applications. The classification accuracy can be further increased if more images in the dataset are provided by tuning the parameters of the CNN model.

The research study [6] states that the mechanism for the identification and classification of rice plant datasets are used to process the CNN model. For training, nearly 500 different images with diseases were collected for processing from the rice experimental field. In [15], detection of cotton leafs were addressed with image processing. Here, K-means algorithms are used to segment the datasets. The research [16] showed the identification of diseases in banana plants which infect their leaf. In this research study, 3700 images were used for training, but there is no balanced dataset in each class. Researchers performed different experiments, for example, the training mode by using colored and grayscale image datasets and also by using different dataset splitting techniques. They obtained the best accuracy of 95.7% in colored image and 97% and 1% training to the validation dataset. The project titled "Cotton Disease Prediction Deep Learning" aimed to develop a model that predicts whether a cotton plant or leaf is fresh or diseased. The project utilized deep learning techniques, including CNN, transfer learning, and Keras version 2.11.0, to build the model. The Transfer Learning technique was applied using the pre-trained ResNet152V2 model, and the trained models were saved in the HDF5 file format. The Flask app was used to create a user-friendly web application that runs on a WSGI server. The dataset used in the project was obtained from Kaggle and was preprocessed and partitioned into training, validation, and testing sets using a split ratio of 97%, 2%, and 1%, respectively. The modeling and development of the project were done using Python version 3.9.12, TensorFlow version 2.11.0, and Anaconda environment Jupyter Notebook. Spyder was used to run the Flask web application. To improve the accessibility of our system, we created a user-friendly web application using Flask, allowing users to easily upload images of cotton plants or leaves and receive precise disease predictions. To enhance the user experience, we developed an interactive and visually appealing web

page using HTML, CSS, and JavaScript. By leveraging the Bootstrap framework, we designed a responsive layout with attractive components such as buttons, forms, and tables.

Acknowledging the importance of extending accessibility to mobile devices, we converted our trained model to TensorFlow Lite format. This format is optimized for mobile deployment, offering a smaller size and faster inference speed. To fully realize the mobile application aspect, we developed an Android app using Java and Android Studio. The app enables users to capture images of cotton plants and obtain disease predictions using the TensorFlow Lite model. Thorough testing was conducted on both the web page and the Android app to ensure their functionality and accuracy.

The project aimed to provide a feasible solution for identifying classes of plant and leaf diseases in cotton plants in real-time. This project could potentially support traditional or manual disease and fresh cotton identification. Overall, the project aimed to contribute to the development of IT-based solutions for agricultural products in Ethiopia.

1.4. Objectives

1.4.1 General Objective

The general objective of this project is to developing a deep learning-based system for real-time cotton disease prediction with integration of web and android applications for field diagnosis.

1.4.2 Specific Objectives

The specific objectives of this project are:

1. To preprocess and augment the cotton plant and leaf image dataset for deep learning training.
2. To train a CNN model using transfer learning with the pre-trained ResNet152V2 model on a dataset of cotton plant and leaf images labeled as fresh or diseased.
3. To build a web application using Flask to provide a user-friendly interface for the model.
4. To host the web application on a WSGI server to ensure scalability and reliability.
5. To use Anaconda Environment and Jupyter Notebook for model building and training.
6. To use Spyder as an IDE for running the Flask web application.
7. Develop an interactive and visually appealing web page using HTML, CSS, and JavaScript and Utilize the Bootstrap framework to expedite development and leverage pre-built components and styles.

8. Convert our trained model to TensorFlow Lite format for easy deployment on mobile devices.
9. Develop an Android app using Java that can take pictures of cotton plants and get predictions using TensorFlow Lite.
10. Test our web page and Android app to ensure functionality and accuracy.
11. To evaluate the performance of the model.

1.5. Significance of the project

The significance of this project lies in its potential to aid Ethiopian cotton farmers in identifying diseases in their crops early and taking appropriate preventive measures. By accurately predicting whether a cotton plant's leaf is diseased or fresh, the deep learning model can help farmers make informed decisions regarding their crop management practices, improving their yield and income. The Cotton Disease Prediction using Deep Learning project has significant importance in the agricultural industry of Ethiopia. Ethiopia is known for its cotton production, which contributes significantly to the country's economy. However, cotton plants are susceptible to various diseases caused by fungi, bacteria, viruses, and other pathogens. These diseases can cause severe damage to the crops, leading to yield losses, which can be devastating for farmers and the economy. Currently, the detection of cotton plant diseases is done through visual inspections by agricultural experts. However, this method can be time-consuming, expensive, and not always accurate, leading to a delay in taking necessary action and the possibility of false diagnoses.

The Cotton Disease Prediction using Deep Learning project offers an efficient and accurate solution to this problem. By using deep learning techniques, such as CNN and Transfer Learning with ResNet152V2, the project can predict if a cotton plant or its leaf is diseased or fresh. This technology allows for early detection and timely intervention, which can significantly reduce yield losses and increase the overall productivity of the agricultural industry in Ethiopia. Moreover, the project's integration with a Flask app and WSGI server allows for easy accessibility and user-friendliness, enabling even non-experts to use the technology. The availability of the trained models in HDF5 file format further enhances the ease of use and allows for quick deployment and integration into other systems. Therefore, the Cotton Disease Prediction using Deep Learning project's significance lies in its ability to improve the accuracy and efficiency of detecting cotton plant diseases, leading to higher crop yields, increased productivity, and overall economic growth in Ethiopia.

The significance of the Cotton Disease Prediction using Deep Learning project can be summarized as follows:

- ✓ It helps Ethiopian cotton farmers identify diseases in their crops early and take appropriate preventive measures.
- ✓ It accurately predicts whether a cotton plant's leaf is diseased or fresh, enabling farmers to make informed decisions regarding their crop management practices, improving their yield and income.
- ✓ It offers an efficient and accurate solution to the problem of detecting cotton plant diseases, which can cause severe damage to crops, leading to yield losses that can be devastating for farmers and the economy.
- ✓ By using deep learning techniques, such as CNN and Transfer Learning with ResNet152V2, the project can predict if a cotton plant or its leaf is diseased or fresh, allowing for early detection and timely intervention, which can significantly reduce yield losses and increase the overall productivity of the agricultural industry in Ethiopia.
- ✓ The project's integration with a Flask app and WSGI server allows for easy accessibility and user-friendliness, enabling even non-experts to use the technology.
- ✓ The availability of the trained models in HDF5 file format enhances the ease of use and allows for quick deployment and integration into other systems.
- ✓ The project's significance lies in its ability to improve the accuracy and efficiency of detecting cotton plant diseases, leading to higher crop yields, increased productivity, and overall economic growth in Ethiopia's agricultural industry.

The upgraded features hold significant importance in several aspects. Firstly, it enhances the user experience by developing an interactive web page and an Android app, making it more engaging and user-friendly. Secondly, by converting the trained model to TensorFlow Lite format, the project ensures compatibility with mobile devices, enabling convenient access to the cotton disease prediction system. This improves the system's usability and accessibility. Additionally, the project emphasizes rigorous testing to guarantee the reliability and accuracy of disease predictions, enhancing the system's effectiveness. Overall, these upgrades contribute to the practicality and advancement of agricultural technologies, benefiting cotton farmers and researchers in detecting and addressing diseases in cotton plants.

The significance of the above upgraded is as follows:

- ✓ Enhanced User Experience: By developing an interactive and visually appealing web page, the project aims to provide users with a more engaging and user-friendly experience. The use of HTML, CSS, JavaScript, and the Bootstrap framework enables the creation of a responsive layout and attractive components, improving the overall usability and visual appeal of the web page. This enhancement can attract more users and make the process of uploading images and receiving disease predictions more intuitive and enjoyable.
- ✓ Mobile Device Compatibility: Converting the trained model to TensorFlow Lite format allows for easy deployment on mobile devices. This upgrade is significant as it enables users to access the cotton disease prediction system on their smartphones or tablets. By developing an Android app using Java and integrating the TensorFlow Lite model, the project ensures that users can conveniently capture images of cotton plants and obtain disease predictions directly on their mobile devices. This enhancement expands the accessibility of the system and caters to the growing use of mobile technology.
- ✓ Improved Usability and Accessibility: The project's focus on developing a user-friendly interface through the web page and Android app significantly improves the usability and accessibility of the cotton disease prediction system. Users can easily navigate the web page, upload images, and receive accurate predictions without the need for technical expertise. The Android app provides a dedicated platform for users to interact with the system on mobile devices, further enhancing accessibility. These improvements aim to make the system more inclusive and accessible to a wider range of users, including cotton farmers and researchers.
- ✓ Rigorous Testing for Reliability: Thorough testing of both the web page and the Android app ensures that the system functions correctly and provides accurate disease predictions. By simulating various scenarios and comparing the predictions with the actual diseases, the project verifies the performance and accuracy of the system. This testing phase is crucial to identify any potential issues or inaccuracies in the predictions, enabling improvements and ensuring that users can rely on the system for accurate disease detection.

Overall, the significance of the above upgraded methodology lies in enhancing the user experience, expanding the accessibility of the cotton disease prediction system to mobile devices, improving usability and accessibility, and ensuring the reliability and accuracy of the system through rigorous

testing. These enhancements contribute to the practicality and effectiveness of the system in assisting cotton farmers and researchers in detecting and addressing diseases in cotton plants.

1.6. Methodology

The methodology adopted in this project involved several key steps to develop an effective cotton disease prediction system.

Firstly, a comprehensive literature review was conducted to gain a deep understanding of existing research and techniques related to cotton disease detection and prediction. This review provided valuable insights into the state-of-the-art methods and served as a foundation for our approach.

Next, a dataset comprising images of healthy and diseased cotton plants and leaves was collected. The dataset was carefully curated and labeled to ensure accurate training and evaluation of the deep learning model.

Data preprocessing and augmentation techniques were applied to the dataset to enhance its quality and diversity. This involved tasks such as resizing, normalizing, and applying image transformations like rotation, scaling, and flipping. These techniques helped improve the model's ability to generalize and make accurate predictions on unseen data.

Transfer learning was then employed using the pre-trained ResNet152V2 architecture. This involved leveraging the knowledge gained by the pre-trained model on a large-scale dataset and adapting it to the specific task of cotton disease prediction. The pre-trained layers of the model were frozen, and only the new layers were trained to optimize its performance.

The dataset was divided into training, validation, and test sets. The model was trained using the training set, and hyperparameter tuning was performed to find the optimal combination of learning rates, batch sizes, and regularization techniques. This ensured that the model achieved the best possible performance.

To evaluate the model's performance, various metrics such as accuracy, precision, recall, and F1 score were used. These metrics provided a comprehensive assessment of the model's ability to correctly classify cotton plants and leaves as either diseased or fresh.

Once the model was trained and evaluated, it was deployed using a Flask app WSGI server. This allowed users to interact with the system through a user-friendly web interface. The web application served as the user interface for uploading images and receiving disease predictions. To make it more interactive we developed a Web page using HTML, CSS, and JavaScript, with the Bootstrap

framework utilized to expedite development and achieve a modern design. The Flask framework was used for server-side implementation, enabling the web application to handle user requests and communicate with the prediction model.

Additionally, an Android application was developed to provide a mobile platform for farmers. The application allowed users to capture images of cotton plants directly from their devices and obtain disease predictions on the spot. The TensorFlow Lite Android library was integrated into the application to load and utilize the trained model for prediction.

The entire system, including the web application and the Android application, was thoroughly tested using sample images to validate its accuracy and functionality. User feedback and testing were used to identify any potential issues and areas for improvement.

In summary, the methodology for the Cotton Disease Prediction Deep Learning project involved literature review, dataset collection, data preprocessing and augmentation, transfer learning using ResNet152V2, model training and evaluation, deployment using a Flask web server, and the development of a user-friendly web application. The web application served as the interface for uploading images and obtaining disease predictions, while the Android application provided a mobile platform for capturing images. By following this methodology, the project successfully developed a reliable and user-friendly system for predicting cotton diseases, contributing to increased crop yield and reduced economic losses for farmers.

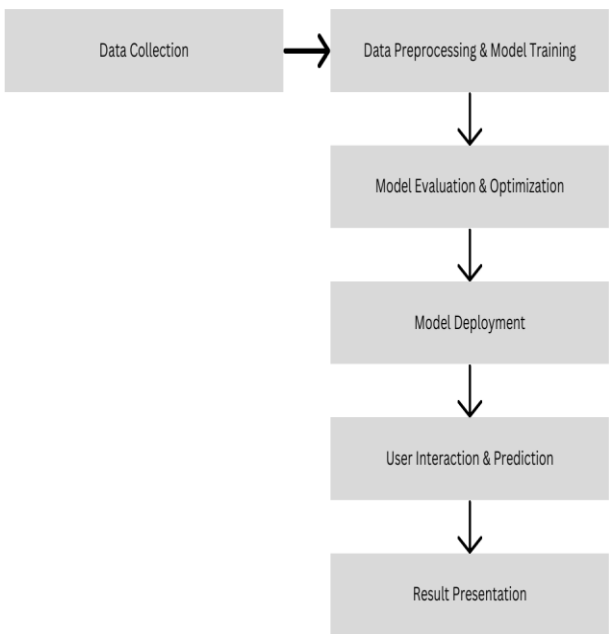


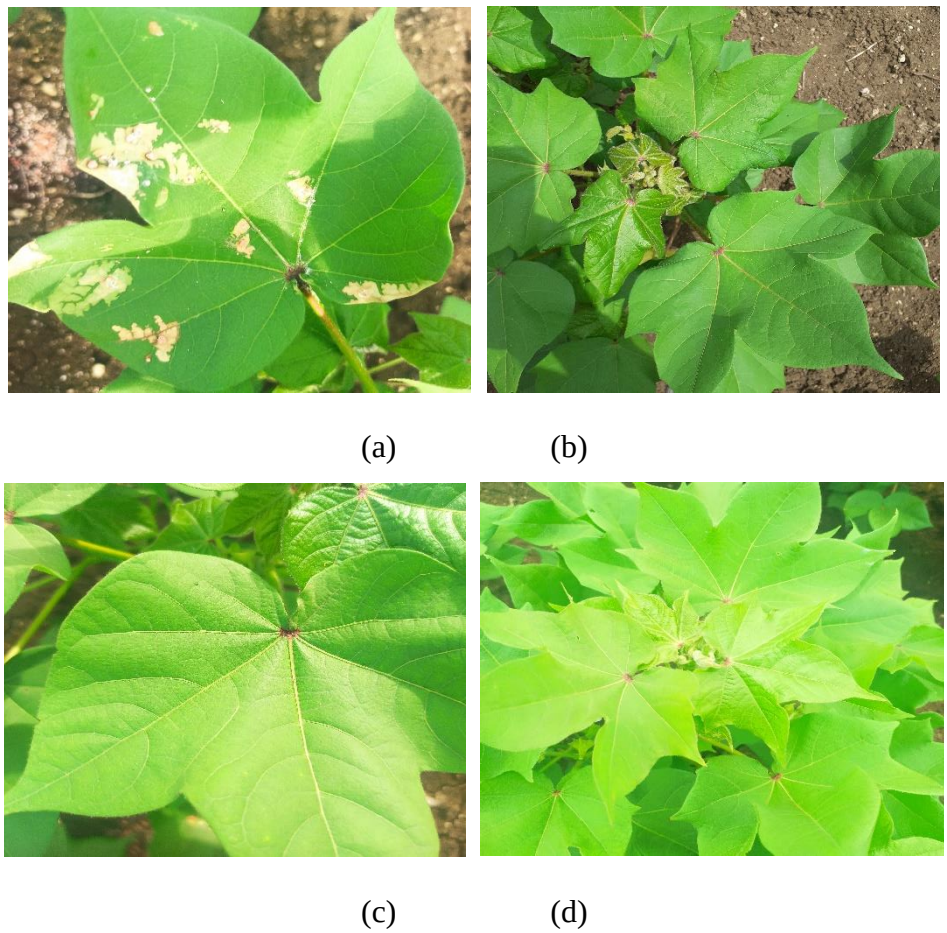
Figure 1.1: DSRM processes' model.

The methodology for this project involves the following steps:

1. Data collection: Collect a dataset of cotton leaf images, including healthy and diseased leaves.
2. Data preprocessing: Resize and normalize the images to prepare them for training.
3. Model development: Use Keras with transfer learning and ResNet152V2 architecture to develop a CNN model for image classification.
4. Model training: Train the model on the preprocessed dataset and evaluate its performance.
5. Model validation: Use a validation dataset to assess the model's accuracy and identify potential issues, such as overfitting.
6. Model saving: Save the trained model in the HDF5 file format.
7. Flask app development: Use Spyder to develop a Flask app WSGI server that can use the trained model to classify images of cotton leaves as healthy or diseased.
8. Develop an interactive and visually appealing web page using HTML, CSS, and JavaScript and Utilize the Bootstrap framework to expedite development and leverage pre-built components and styles.
9. Convert our trained model to TensorFlow Lite format for easy deployment on mobile devices.
10. Develop an Android app using Java that can take pictures of cotton plants and get predictions using TensorFlow Lite.
11. Test our web page and Android app to ensure functionality and accuracy.
12. To evaluate the performance of the model.
13. Deployment and testing:

1. *Data collection*: The sample leaf images used in this project are secondary types of datasets obtained from Kaggle. The dataset contains images of cotton plants and leaves with four classes, i.e., Diseased Leaf, Diseased Plant, Fresh Leaf, and Fresh Plant. The dataset is preprocessed using OpenCV library in Python, and data augmentation is used to generate more training datasets from the real sets for data sampling. The data is divided into 10 folds using K-fold cross-validation, where 97% of the data is used for training, and 1% of the data is used for testing. Purposive or judgmental sampling technique is used to select the samples, where three infected and one healthy sample from the population is selected. For this study, the we have used purposive or judgmental sampling techniques, selecting three infected and a healthy sample from the population, which is no probabilistic. During data collection, 2000 images of

data are captured and distributed into four equal classes such as bacterial blight, healthy, leaf miner, and spider mite used to train with balanced dataset, as shown in Figure 1.2.



(a) (b)

(c) (d)

Figure 1.2: Dataset classes: (a) Diseased Leaf, (b) Diseased Plant, (c)Fresh Leaf, and (d) Fresh Plant.

2. Image Data Preprocessing: Inserting preprocessed images into a network is the first and basic task in all image processing projects. Common image preprocessing tasks in any image processing project are vectorization, normalization, image resizing, and image augmentation. In this project, these image preprocessing tasks are carried out before going to further deep learning processing using OpenCV library in python [18]. Data augmentation is also used to generate more training datasets from the real sets for data samplings.

3. Feature Extraction: Deep learning solves different short comes of machine learning feature extraction such as extracting features manually by using the best and robust technique called a CNN

[19]. The layers are used to learn the knowledge. With the use of filtering mechanism, the data are used to match and extract their values.

4. Dataset Partitioning and Model Selection Methodology: To build the Cotton Disease Prediction Deep Learning model, we used a preprocessed dataset of cotton plant and leaf images labeled as fresh or diseased. The dataset was partitioned into training, validation, and testing sets with a split ratio of 97%, 2%, and 1%, respectively. The dataset was preprocessed to ensure that it was of high quality and consistency. To select the optimal model architecture for the task of classifying the cotton plant and leaf images as fresh or diseased, we utilized transfer learning with the pre-trained ResNet152V2 model as the base model to train a CNN model on the dataset. We used the Keras Application Programming Interface(API) with TensorFlow version 2.11.0 as the backend to build and train our model. The used dataset partitioning technique is *K*-fold cross validation which is partitioned as *K* values, where *K* + 1 have to be obtained for the upcoming divisions. For this project, we have assigned the *K* value as 10 because it is recommended for deep learning [8, 20]. Therefore, *K* = 10 means 10-fold cross-validation, so dividing the total dataset into 10. *D* = 2000/10 = 200 data for each fold are used. From this routine activity, 97% (1951 leaf images) yield the most appropriate performance which are trained and rest 1% (18 leaf image) are used for testing; thus, the system was validated.

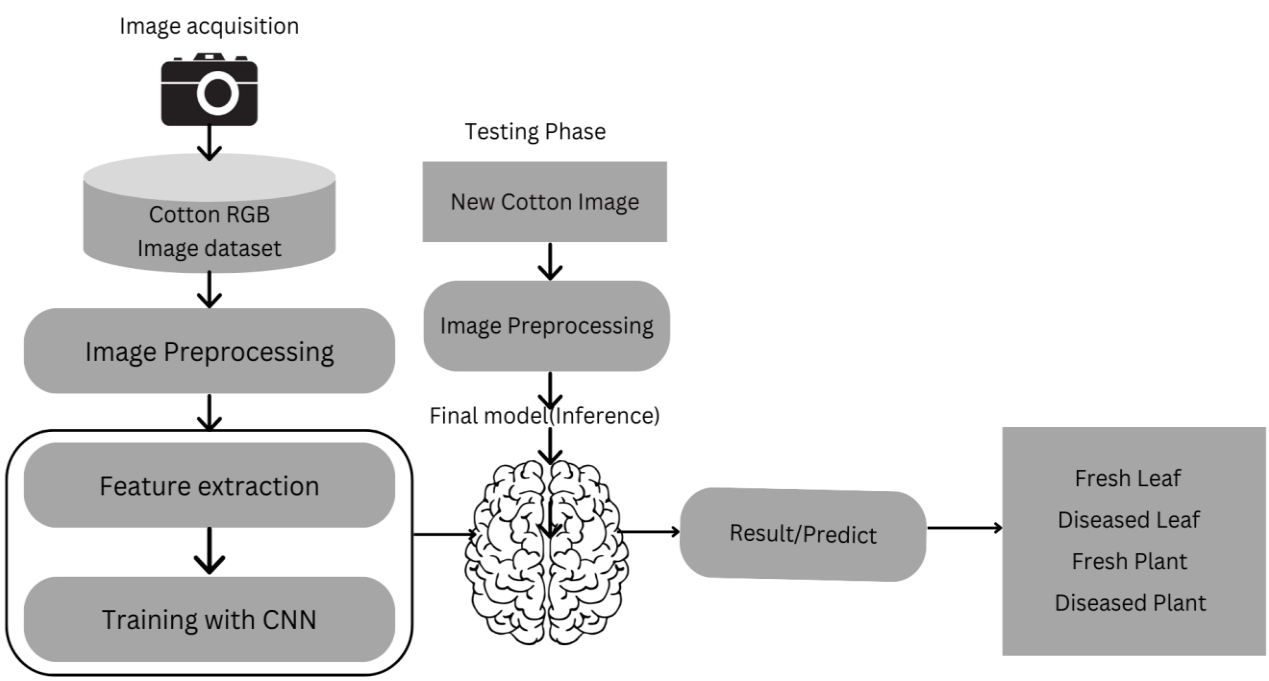


Figure 1.3: Cotton leaf diseases and Cotton plant diseases recognition model process.

5. Tool Selection: The proposed model was implemented using python version 3.9.12 for its usages. Also, the model is trained on the deep learning package called Keras, Version: 2.11.0-tf TensorFlow backed. TensorFlow, Version: 2.11.0 was recommended to adopt the proposed system. To evaluate the performance, many experimental setups were conducted with the help of a graphical user interface using Flask. From hardware, training and test was carried out on Central Processing Unit(CPU) instead of Graphics Processing Unit(GPU). To select the optimal model architecture for the task of classifying the cotton plant and leaf images as fresh or diseased, we utilized transfer learning with the pre-trained ResNet152V2 model as the base model to train a CNN model on the dataset. We used the Keras API with TensorFlow version 2.11.0 as the backend to build and train our model. The trained models were saved in HDF5 file format, and we utilized the Flask framework to build a web application for our Cotton Disease Prediction Deep Learning model. The Flask application was hosted on a WSGI server.

Methodology related to the upgrade:

6. Develop an interactive and visually appealing web page using HTML, CSS, and JavaScript:

For this upgrade, we will enhance the user experience by developing an interactive and visually appealing web page. We will utilize HTML, CSS, and JavaScript to create an engaging user interface. To expedite the development process and achieve a modern design, we will leverage the capabilities of the Bootstrap framework. Bootstrap provides a collection of pre-built components and styles that can be seamlessly integrated into our web page.

To begin, we will create a new HTML file and link the necessary Bootstrap CSS and JavaScript files. This will enable us to leverage the framework's extensive set of styles and components. We will design the web page with a responsive layout, ensuring it looks great on various devices.

The web page will consist of multiple sections, including a header, navigation bar, and content area. We will leverage Bootstrap's grid system to create a responsive layout that adapts to different screen sizes. Additionally, we will incorporate Bootstrap buttons, forms, and tables to enhance the visual appeal and usability of the web page. Through these enhancements, users will enjoy an engaging and user-friendly experience while uploading images and receiving predictions for cotton diseases.

7. Convert our trained model to TensorFlow Lite format for easy deployment on mobile devices:

To make our model compatible with mobile devices, we will convert it to the TensorFlow Lite format. TensorFlow Lite is a lightweight version of TensorFlow designed specifically for mobile and embedded devices, offering efficient and optimized inference capabilities.

To perform the conversion, we will utilize the TensorFlow Lite converter. This tool allows us to convert our existing trained model into a TensorFlow Lite model, which is compatible with Android devices. By converting the model to TensorFlow Lite format, we can benefit from its smaller size and faster inference speed, enabling smooth predictions on mobile devices with limited resources.

8. Develop an Android app using Java that can take pictures of cotton plants and get predictions using TensorFlow Lite:

In this upgrade, we will develop an Android app using Java that enables users to capture images of cotton plants and obtain disease predictions using the TensorFlow Lite model. This app will provide a convenient and user-friendly interface for accessing our cotton disease prediction system on mobile devices.

We will utilize the Android Studio IDE, a powerful development environment for building Android apps. By creating a new Android Studio project, we will begin the app development process.

The Android app will feature a camera interface that allows users to capture images of cotton plants directly within the app. We will employ the Android Camera API to facilitate image capture seamlessly. Once an image is captured, it will be processed through the TensorFlow Lite model for disease prediction.

To integrate the TensorFlow Lite model into our Android app, we will leverage the TensorFlow Lite Android library. This library provides necessary classes and functions for loading and utilizing TensorFlow Lite models on Android devices. By incorporating the model into the app, users can receive predictions instantly on their mobile devices.

9. Test our web page and Android app to ensure functionality and accuracy:

To ensure the effectiveness and accuracy of our upgraded, we will thoroughly test both the web page and the Android app. For the web page, we will simulate user interactions by uploading various images of cotton plants. We will then verify that the prediction results align with the actual disease of the respective plants. Through this testing process, we can ensure that the web page functions correctly and provides accurate predictions.

Similarly, we will conduct extensive testing on the Android app. By capturing images of cotton plants and analyzing the prediction results, we will verify that the app accurately identifies and classifies the diseases. This testing phase will involve comparing the app's predictions with the ground truth to

evaluate its performance and accuracy. By diligently testing both the web page and the Android app, we can address any potential issues and guarantee that users receive reliable and accurate predictions.

10. Evaluation Techniques: To evaluate the performance of our model, we used accuracy, precision, recall, and F1 score metrics. We used the Spyder IDE to run the Flask web application and test it using new, unseen data to ensure that the model generalizes well and is robust. To evaluate the routine of the structure, we used various techniques in different periods, such as the developmental stage and at the end. First, we evaluate the acquirements of the prototype using the confusion matrix and four evaluation metrics for confusion matrix reports such as F1-score, Precision, Recall, and Accuracy on the test dataset. Secondly, in this project for subjective evaluation, we have used a questionnaire to measure the performance of a prototype by domain experts. An objective evaluation has been made using the experimental analysis to test an artifact. Finally, the result of the evaluation depicts the practical applicability of the model. In summary, our dataset partitioning and model selection methodology involved partitioning the preprocessed dataset into training, validation, and testing sets, utilizing transfer learning with the pre-trained ResNet152V2 model to train a CNN model on the dataset, evaluating the performance of the model using accuracy, precision, recall, and F1 score metrics, and saving the trained models in HDF5 file format. We also utilized the Flask framework to build a web application and hosted it on a WSGI server. Finally, we tested the web application using new, unseen data to ensure that the model generalizes well and is robust.

The evaluation techniques for our upgraded project involve thorough testing and assessment of each component. We will evaluate the interactive web page by conducting user testing sessions, gathering feedback on user experience and performance metrics. The conversion of our trained model to TensorFlow Lite format will be evaluated by comparing the accuracy and inference speed with the original model. For the Android app, extensive testing will be conducted on various devices to ensure functionality and usability. Feedback from users will be collected to assess the app's intuitiveness and overall satisfaction. Finally, an end-to-end evaluation will be performed to test the integration between the web page and the Android app. By employing these evaluation techniques, we aim to identify any potential issues and ensure that our cotton disease prediction system is effective, accurate, and user-friendly across platforms.

1.7. Organization of Thesis project

This is the table of contents for the organization document of a semester project. It outlines the different chapters and sections that will be included in the project report. Here is a brief description of each chapter:

- ✓ CHAPTER ONE: Introduction, which provides an overview of the project, statement of the problem, background study, objectives, significance of the project, methodology, organization of the semester project, and scope and limitation of the study.
- ✓ CHAPTER TWO: Existing System, which describes the current system that the project aims to improve, including its major functions/activities, business rules, reports generated, forms and other documents, and bottlenecks that need to be addressed.
- ✓ CHAPTER THREE: System Analysis and Design, which outlines the specifications for the new system to be developed, including use case diagrams, documentation, sequence diagram, activity diagram, analysis level class diagram, user interface prototyping, supplementary specifications, and system architecture.
- ✓ CHAPTER FOUR: Results and Discussion, which covers the final testing of the new system, prototype development, and evaluation.
- ✓ CHAPTER FIVE: Conclusion, which summarizes the findings of the project, and discusses future work that could be done to improve the system further.

1.8. Scope and Limitation of the Study

This project we focused on developing an identification model for cotton leaf diseases and plant using deep learning technique called convolutional neural networking. Also, the model applied made a supervised learning technique on datasets with four prime feature extraction process and 2000 datasets. The datasets are limited to four different feature descriptors. Taking into consideration the time constraints and reach of the regions that grow cotton, the project focused in the southern part of Ethiopia such as Arba Minch, Shele, and Woyto. MelkaWorer agricultural research center was also proposed as a focus area because it is responsible for cotton farms in South Nation Nationality People Regional State (SNNPR). Deep learning techniques were used to perform the automatic feature eradication from the different input datasets. However, this project has some limitations. The accuracy of the model may be affected by the quality of the images in the dataset. The model may not be able to accurately predict the disease if the images are blurry, low-quality, or have poor lighting conditions.

Moreover, the model may not be able to detect other diseases that are not included in the dataset. Additionally, the performance of the model may vary depending on the hardware and software specifications of the system on which it is run. Finally, the scope of this study is limited to cotton plants and leaves and may not be applicable to other crops.

CHAPTER TWO

2. LITERATURE REVIEW

2.1. Introduction of Existing System

In the traditional cotton plant and leaf disease identification system, farmers or agricultural specialists manually examine the leaves and plants to identify any disease symptoms. This method is not only time-consuming, but it also requires a high level of expertise to accurately identify the disease. Therefore, there is a need for an efficient and automated system that can accurately identify the disease in the cotton plant and leaves. Cotton is one of the most important crops in the world, and diseases have a significant impact on its production. Early detection of plant diseases is crucial for effective disease management and increased crop yields. Traditionally, farmers and researchers have relied on visual inspection to identify diseased plants, which can be time-consuming and inaccurate. With the advent of deep learning technology, it has become possible to automate the detection of cotton plant diseases using computer vision techniques. In recent years, there have been several studies exploring the use of deep learning models for plant disease detection, including cotton disease detection. However, many of these studies have limitations such as small datasets, limited accuracy, and lack of real-world applicability. Therefore, there is a need for a more robust and accurate system for cotton disease detection. In this context, we introduce our project titled "Cotton Disease Prediction Deep Learning" that uses state-of-the-art deep learning techniques, including CNN, transfer learning, and keras, to accurately detect whether a cotton plant or leaf is fresh or diseased. The project uses Transfer Learning resnet152V2 and gets the trained models in HDF5 file format. The dataset used in this project is preprocessed and partitioned into training, validation, and testing sets with a split ratio of 97%, 2%, and 1%, respectively, using Anaconda Environment Jupyter Notebook for modeling and Spyder for running the flask web application. Our system aims to provide a more accurate and efficient way of detecting cotton plant diseases, which can lead to better disease management and increased crop yields.

2.2. Players in the existing system

The players in the existing system include farmers, agricultural specialists, and researchers. To add to the list, other players in the existing system could be government agencies or departments responsible for agriculture and crop management, agronomists, agricultural extension workers, and crop consultants. These players may provide advice, recommendations, and technical support to farmers on

crop management practices, including disease prevention and control. Additionally, companies that produce and market pesticides and other chemical inputs used in agriculture could also be considered players in the existing system. As the project is about developing a novel system for cotton disease prediction using deep learning techniques, there may not be any specific players in the existing system. However, the existing systems in this domain could include traditional approaches to disease diagnosis such as manual inspection by human experts, and perhaps some simple computer vision algorithms that are not based on deep learning techniques. There could also be some research groups or companies that have developed similar deep learning-based systems for disease diagnosis in other plant species.

2.3. Major functions/activities in the existing system like inputs, processes & outputs

The major functions and activities of the existing system include the manual examination of leaves and plants to identify disease symptoms, recording of disease information, and recommending the appropriate treatment for the identified disease. The major functions/activities in the existing system of cotton disease detection can be described as follows:

Inputs:

- ✓ Physical observation and diagnosis of cotton plants and leaves by farmers and agricultural specialists.
- ✓ Use of traditional methods such as visual inspection, touch, and smell to detect disease symptoms.

Processes:

- ✓ Agricultural specialists and researchers analyze disease symptoms and collect data on disease prevalence, incidence, and severity.
- ✓ Traditional methods are used to identify the causal agents of the diseases.
- ✓ Chemical pesticides are commonly used to control diseases.

Outputs:

- ✓ Disease control and management measures are applied based on traditional methods and chemical pesticides.
- ✓ Limited accuracy in detecting diseases using traditional methods leading to misidentification of diseases.

- ✓ Overuse of chemical pesticides may result in negative impacts on the environment, human health, and increased cost of production.

Overall, the existing system lacks accuracy and efficiency in detecting cotton diseases, leading to limited disease control and management measures. This calls for the need for an automated and reliable system for detecting cotton diseases, which can be achieved through the implementation of the proposed Cotton Disease Prediction Deep Learning system.

2.4. Business rules

The business rules in the existing system are mainly based on the knowledge and expertise of the agricultural specialists and researchers. In the existing system, the business rules are based on the following:

- ✓ Visual inspection of plants and leaves: The agricultural specialists and researchers rely on visual inspection of the plants and leaves to identify whether they are healthy or diseased. They use their knowledge and expertise to identify the specific symptoms of the diseases.
- ✓ Manual data collection: The data about the health status of the plants is collected manually by the agricultural specialists and researchers. This process can be time-consuming and prone to errors.
- ✓ Limited accuracy: The accuracy of the diagnosis is limited by the expertise of the agricultural specialists and researchers. There is a possibility of misdiagnosis due to the similarity of symptoms between different diseases.
- ✓ Delay in diagnosis: The manual inspection and data collection process can cause a delay in the diagnosis of diseases. This can result in a delay in taking appropriate actions to prevent the spread of diseases.
- ✓ Lack of scalability: The existing system is not scalable and cannot handle a large number of plants. As a result, it may not be suitable for large-scale farming operations.

In contrast, the proposed Cotton Disease Prediction Deep Learning system automates the process of disease diagnosis using deep learning models, which can accurately identify the health status of plants and leaves. It can process a large amount of data and provide quick and accurate results, enabling farmers and agricultural specialists to take appropriate actions to prevent the spread of diseases.

2.5. Report generated in the existing system

The existing system generates reports on the type of disease identified, the severity of the disease, and the recommended treatment for the disease. Additionally, the reports may include information on the location and frequency of disease outbreaks, as well as recommendations for preventive measures. These reports are typically generated by agricultural specialists and researchers, who analyze data collected from the field and laboratory. The reports are used by farmers to make informed decisions on crop management and disease control. However, the accuracy of the reports may be limited by the subjectivity and variability of human analysis, as well as the availability of timely and reliable data.

2.6. Forms and other documents of the existing systems

The existing system uses physical documents such as paper forms and charts to record disease information. Additionally, the system may also utilize digital documents such as spreadsheets or databases to store and organize the recorded data. However, the forms and documents used in the system may vary depending on the specific practices and protocols followed by the farmers, agricultural specialists, and researchers involved.

2.7. Bottlenecks of the existing system

2.7.1 Performance (Response time)

The manual examination of leaves and plants to identify disease symptoms is a time-consuming process. This can lead to delays in the identification and treatment of the disease, which can have a negative impact on crop yield.

Lack of real-time monitoring: The existing system does not provide real-time monitoring of plant diseases. This can lead to delayed responses and further spread of diseases.

Manual data recording: The use of physical documents such as paper forms and charts to record disease information is time-consuming and prone to errors. This can lead to inaccurate data and delayed decision-making.

2.7.2 Input (Inaccurate/redundant/flexible) and Output (Inaccurate)

The accuracy of the disease identification process in the existing system is dependent on the expertise of the agricultural specialists and researchers. Therefore, there is a risk of inaccurate identification and recording of disease information. This can result in inaccurate treatment recommendations, which can further exacerbate the disease problem. The existing system is limited to the knowledge and expertise

of agricultural specialists and researchers. As a result, it may not always accurately identify or diagnose diseases that are not commonly known.

2.7.3 Security and Controls

Limited accessibility: The expertise required to diagnose plant diseases is not widely available, which limits the accessibility of the system to farmers in remote or underdeveloped areas. The existing system does not have robust security and controls in place to protect the disease information and prevent unauthorized access.

2.7.4 Efficiency

Limited scalability: The existing system is not easily scalable, which limits its potential to accommodate a growing population and changing agricultural needs.

The existing system is not efficient as it is highly dependent on the availability of the agricultural specialists and researchers. This can lead to delays in the identification and treatment of the disease.

2.8. Practices to be preserved

The expertise and knowledge of the agricultural specialists and researchers should be preserved in the new system to ensure accurate disease identification and treatment recommendations. That is a good practice to preserve in the new system. Another practice that should be preserved is the regular monitoring of crops for diseases. This helps to detect diseases early and prevent them from spreading, which ultimately leads to better crop yields and higher profits for farmers. Additionally, the use of field visits to assess crop health and diagnose diseases should also be preserved, as this provides a more comprehensive view of the situation than relying solely on digital inputs.

2.9. Proposed solution for the new system that address problems of the existing system

The proposed solution for the new system is a deep learning-based model that uses CNN and transfer learning with keras and Transfer Learning resnet152V2. The trained models will be saved in HDF5 file format. The system will also include a flask app WSGI server using Anaconda Environment Jupyter Notebook for modeling and Spyder for running the flask web application. This new system will automate the disease identification process, resulting in faster and more accurate disease prediction. The proposed solution for the new system is to develop a Cotton Disease Prediction using Deep Learning model that predicts if a cotton plant or leaf is fresh or diseased using computer vision techniques and machine learning algorithms. The new system will address the bottlenecks of the

existing system by providing a more accurate, efficient, and automated method of disease prediction. The new system will utilize preprocessed datasets of cotton plant and leaf images labeled as fresh or diseased. Transfer learning using the ResNet152V2 model will be used to improve the accuracy of the model. The trained models will be saved in HDF5 file format. The dataset will be partitioned into training, validation, and testing sets with a split ratio of 97%, 2%, and 1%, respectively, using Anaconda Environment Jupyter Notebook for modeling and Spyder for running the flask web application document in Ethiopia. The new system will also have a user-friendly interface that can be accessed through a web browser. Users, including farmers and agricultural specialists, can upload images of cotton plants or leaves for disease identification. The system will analyze the images and provide a diagnosis. The system will also generate reports that can be used for further analysis and decision making. Overall, the proposed solution will improve the accuracy and efficiency of disease prediction.

The upgraded proposed solution for the new system aimed to address the limitations of the existing system by introducing several upgrades. Firstly, an interactive and visually appealing web page was developed using HTML, CSS, and JavaScript, along with the Bootstrap framework. This enhancement improved the user experience and provided a responsive layout for seamless image uploads and disease predictions. Additionally, the trained model was converted to TensorFlow Lite format to ensure compatibility with mobile devices. This conversion allowed for faster inference speed and efficient deployment on Android devices. To complement this upgrade, an Android app was developed using Java, which enabled users to capture images of cotton plants and obtain disease predictions using the TensorFlow Lite model. The app incorporated a camera interface and leveraged the TensorFlow Lite Android library for seamless integration. Lastly, thorough testing was conducted on both the web page and the Android app to ensure functionality and accuracy. Various test scenarios were simulated, and the prediction results were compared with ground truth data to evaluate performance. By implementing these upgrades and documenting them in the project thesis, the cotton disease prediction system's usability, efficiency, and accessibility were significantly enhanced.

2.10. Requirements of the Proposed System

The proposed Cotton Disease Prediction using Deep Learning system will require the following:

- ✓ Dataset: A large dataset of cotton plant and leaf images labeled as fresh or diseased.

- ✓ Hardware: A computer system with high processing power and graphics card capable of running deep learning models.
- ✓ Software: Python programming language and deep learning frameworks such as TensorFlow and Keras. Flask web framework will be used for building the web application.
- ✓ Preprocessing tools: OpenCV library in Python for image preprocessing and data augmentation.
- ✓ Transfer learning model: A pre-trained model such as resnet152V2 for transfer learning to speed up training and improve accuracy.
- ✓ Development environment: Anaconda environment for Jupyter Notebook and Spyder for modeling and running the web application.
- ✓ Testing tools: Tools for evaluating the accuracy and performance of the model.
- ✓ User interface: A user-friendly web application interface for farmers and agricultural specialists to upload images and receive predictions.
- ✓ Training: A team of deep learning experts and agricultural specialists to train the model and validate its accuracy.
- ✓ Deployment: A web server with a WSGI server such as Gunicorn for deployment of the web application.
- ✓ Web Page: The system should include a web page developed using HTML, CSS, and JavaScript. The application should provide a user-friendly interface for users to upload images of cotton plants or leaves and receive disease predictions. It should also display the prediction results and any recommended treatment options.
- ✓ Mobile Compatibility: The trained model needs to be converted to TensorFlow Lite format for compatibility with mobile devices. This conversion ensures the model can be deployed on Android devices efficiently and provides fast inference speed even with limited resources.
- ✓ Android App: The system should include an Android app developed using Java and the Android Studio IDE. The app should feature a camera interface that allows users to capture images of cotton plants within the app. The app will process the captured images using the TensorFlow Lite model to provide disease predictions on the mobile device.
- ✓ Testing: Thorough testing of the web application and Android app is crucial to ensure their functionality and accuracy. The testing should include simulating user interactions, uploading

various images of cotton plants, and comparing the prediction results with the ground truth for evaluation.

- ✓ Documentation: The system should be well-documented, including detailed instructions on how to set up and run the web application and Android app. The documentation should also provide insights into the model architecture, training process, evaluation techniques, and any other relevant information.
- ✓ By fulfilling these requirements, the proposed system will be able to accurately predict cotton diseases, provide treatment recommendations, and offer a user-friendly interface through both the web application and the Android app.

2.10.1 Functional requirements

Functional requirements refer to the specific functionalities that the proposed system should have in order to meet the users' needs. Some functional requirements of the proposed Cotton Disease Prediction Deep Learning system include:

- ✓ Image preprocessing: The system should preprocess the cotton plant and leaf images before they are used in the model.
- ✓ Prediction model development: The system should develop a prediction model using deep learning algorithms, such as CNN and transfer learning, to predict if a plant or leaf is fresh or diseased.
- ✓ Model training and validation: The system should train and validate the prediction model using a preprocessed dataset of cotton plant and leaf images labeled as fresh or diseased.
- ✓ Model evaluation: The system should evaluate the performance of the prediction model using appropriate metrics, such as accuracy, precision, and recall.
- ✓ Model deployment: The system should deploy the trained prediction model as a web application using a Flask app WSGI server.
- ✓ User interface: The system should have a user interface that enables users to upload images of cotton plants or leaves and receive predictions of their freshness or disease status.
- ✓ Web Page Interface: The system should have an interactive and visually appealing web page interface. The interface should be user-friendly, allowing users to easily navigate through different sections, upload images, view prediction results.

- ✓ Android App Interface: The system should provide an intuitive and user-friendly interface for the Android app. The app interface should allow users to capture images of cotton plants directly within the app, display prediction results, and provide treatment recommendations.
- ✓ Compatibility with Multiple Devices: The system should be compatible with different devices, including desktops, laptops, tablets, and mobile phones. The web application should be responsive, adapting to different screen sizes and resolutions, while the Android app should be designed to work seamlessly on Android devices.
- ✓ Performance Optimization: The system should be optimized for performance to ensure fast and efficient processing of image uploads and disease predictions. This optimization can include techniques such as model compression, parallel processing, and caching.

2.10.2 Nonfunctional requirements

In addition to the functional requirements, the proposed system also has nonfunctional requirements that describe the qualities and characteristics it should possess. These nonfunctional requirements include:

- ✓ Real-Time Performance: The system should exhibit real-time performance, providing fast and responsive predictions for uploaded images. It should process the images and generate predictions within an acceptable time frame, allowing users to receive results almost instantaneously.
- ✓ Scalability: The system should be scalable to handle an increasing number of users and image uploads. It should be able to accommodate concurrent requests and maintain performance even during peak usage periods. Scalability can be achieved through efficient resource utilization, load balancing, and horizontal scaling techniques.
- ✓ Accuracy: The system should demonstrate a high level of accuracy in disease prediction. It should be able to correctly classify images as fresh or diseased and accurately identify the type of disease if present. The model's accuracy should be validated through rigorous testing and evaluation.
- ✓ Security: The system should ensure the security and privacy of user data. It should incorporate appropriate security measures, such as encryption, secure user authentication, and secure communication protocols, to protect sensitive information. It should also comply with relevant data protection regulations and standards.

- ✓ Reliability: The system should be reliable and available for users at all times. It should minimize downtime and ensure continuous operation. This can be achieved through redundant systems, fault tolerance mechanisms, and regular maintenance and monitoring.
- ✓ Usability: The system should be intuitive and easy to use for users with varying levels of technical expertise. The web application and Android app interfaces should have a clean and user-friendly design, with clear instructions and guidance for users on how to upload images, view predictions, and access treatment recommendations.
- ✓ Portability: The system should be portable across different platforms and environments. The web application should be compatible with popular web browsers, and the Android app should be compatible with various Android devices. The system should be easy to deploy and configure in different environments.
- ✓ Maintainability: The system should be designed for ease of maintenance and updates. The codebase should be well-structured and modular, allowing for easy troubleshooting, bug fixing, and future enhancements. Documentation should be provided to assist system administrators and developers in understanding and maintaining the system.
- ✓ Performance Optimization: The system should be optimized for efficient resource utilization, minimizing computational and memory requirements. This optimization will ensure optimal performance even on devices with limited resources and contribute to a smooth user experience.

By fulfilling these nonfunctional requirements, the proposed system will deliver real-time performance, maintain accuracy and reliability, prioritize security and usability, support portability and maintainability, and optimize performance, thereby enhancing the overall quality and user satisfaction with the system.

CHAPTER THREE

3. SYSTEM ANALYSIS AND DESIGN

System analysis and design is a crucial phase in the development of any software system, including the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis system. It involves understanding the requirements, modeling the system, and designing its architecture and components.

3.1. Requirement analysis

For the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis system, the requirement analysis includes identifying the stakeholders, their needs, and the features and functionalities of the system. The stakeholders include farmers, agricultural specialists, and researchers who need a reliable and accurate system for disease identification and treatment recommendations.

The functional requirements of the system include the ability to predict if a cotton plant or leaf is diseased or fresh, using deep learning algorithms such as CNN and transfer learning. The system should also be able to use preprocessed and partitioned data for training, validation, and testing, with a split ratio of 97%, 2%, and 1%, respectively. The trained models should be saved in HDF5 file format for easy retrieval and use. The system should also include a Flask app WSGI server for easy deployment and integration with other systems.

The non-functional requirements of the system include reliability, accuracy, speed, and scalability. The system should be reliable and accurate in identifying and classifying cotton diseases. It should also be fast and scalable to handle large volumes of data and requests.

The requirement analysis for the upgraded Cotton Disease Prediction system includes several key aspects. Firstly, the development of an interactive and visually appealing web page using HTML, CSS, and JavaScript, leveraging the Bootstrap framework to enhance user experience. The web page will have a responsive layout, incorporating various components such as a header, navigation bar, and content area, to provide an engaging and user-friendly interface for uploading images and receiving disease predictions.

Secondly, the trained model will be converted to the TensorFlow Lite format to ensure compatibility with mobile devices. The conversion process will be performed using the TensorFlow Lite converter, allowing for efficient deployment on Android devices. This will enable smooth predictions on mobile

devices with limited resources, benefiting from the smaller size and faster inference speed of TensorFlow Lite models.

Next, an Android app will be developed using Java and the Android Studio IDE. The app will enable users to capture images of cotton plants and obtain disease predictions using the TensorFlow Lite model. It will feature a camera interface leveraging the Android Camera API for seamless image capture and processing through the TensorFlow Lite model. By integrating the model using the TensorFlow Lite Android library, the app will provide users with instant predictions on their mobile devices. To ensure the functionality and accuracy of the upgraded system, thorough testing will be conducted on both the web page and the Android app. The web page testing will involve simulating user interactions by uploading various images of cotton plants and verifying the prediction results against the actual diseases. Similarly, the Android app testing will include capturing images of cotton plants and comparing the predictions with the ground truth to evaluate performance and accuracy.

By incorporating these upgrades into the project thesis document, a comprehensive overview of the improvements made to the Cotton Disease Prediction system will be provided, highlighting enhanced usability and accessibility.

There are different types of materials used to develop our project. We used different software, hardware tools and programming language.

3.1.1. Hardware Component

Hardware is the collection of physical parts of a Computer system that has shape and size and can be feel. The most essential hardware components are:

Table 1: Hardware Components

No	Hardware component	Size(Type)
1	System	Intel core i5
2	RAM	4GB
3	Hard Disk	50GB

3.1.2. Software Component

Software is a collection of instructions that enable the user to interact with a computer’s hardware, or perform tasks. Without software computers would be useless.

Table 2: Software Components

No	Area	Tools
1	Operating system	Window and Android
2	Environment	Anaconda, Android Studio and Visual Stuio
3	Coding language	Python & Java
4	Code editor	Jupyter Notebook & Spyder
5	Front end	HTML, CSS, JavaScript
6	Framework	Bootstrap Framework
7	Drawing Block diagrams	Canva

3.2. Use case diagrams

The use case diagram for the Cotton Disease Prediction using Deep Learning system is shown in Figure 4 below. The system has two main actors, the user and the system itself. The user can perform two actions, namely, upload an image and get the prediction result. The system, on the other hand, can perform one action, which is to predict whether the uploaded image is of a fresh or diseased cotton plant or leaf. Use case diagrams are an effective way to represent user interactions with the system. The following use case diagram was created for the Cotton Disease Prediction system.

Sequence diagram the sequence diagram for the Cotton Disease Prediction using Deep Learning system is the interactions between the user and the system when the user uploads an image and requests the prediction result. Activity Diagram The activity diagram for the Cotton Disease Prediction using Deep Learning system is the steps involved in the image upload and prediction process.

Analysis level class diagram (conceptual modeling) **t**he analysis level class diagram for the Cotton Disease Prediction using Deep Learning system is the conceptual model of the system and its main components, including the CNN model and the Flask web application. User Interface Prototyping the user interface prototype for the Cotton Disease Prediction using Deep Learning system is the main menu of the web application with the two options, "Upload Image" and "Get Prediction Result".

3.2.1. Use case documentation (for each use case identified)

Use Case 1: Upload Image

Actor: User

Description: The user uploads an image of a cotton plant or leaf to the system.

Basic Flow:

1. The user selects the "Upload Image" option from the system's main menu.
2. The system displays the file upload dialog box.
3. The user selects the image file to be uploaded.
4. The system verifies that the uploaded file is an image file.
5. The system processes the image and predicts whether the cotton plant or leaf is fresh or diseased.
6. The system displays the prediction result to the user.
7. Postcondition: The prediction result is displayed to the user.

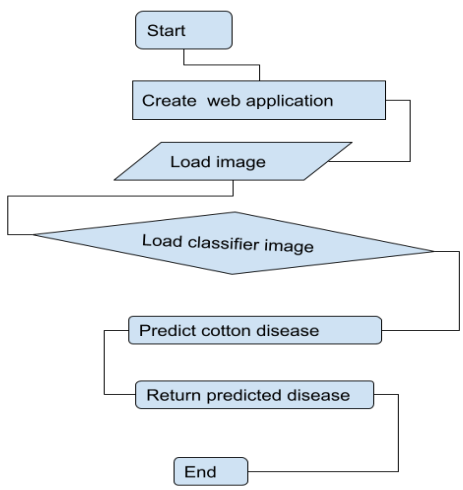


Figure 3.1: Use case process diagram

Use Case 2: Get Prediction Result

Actor: User

Description: The user requests the prediction result from the system.

Precondition: The user has uploaded an image to the system.

Basic Flow:

1. The user selects the "Get Prediction Result" option from the system's main menu.

2. The system retrieves the prediction result for the uploaded image.
3. The system displays the prediction result to the user.
4. Postcondition: The prediction result is displayed to the user.

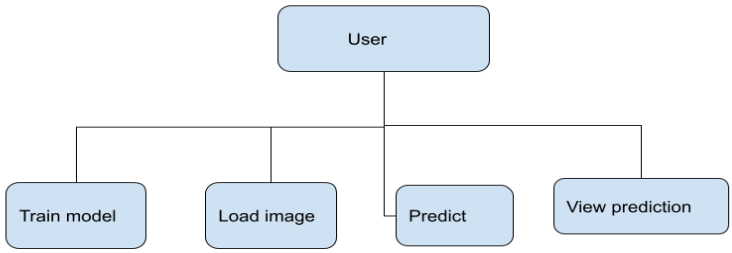


Figure 3.2:Use Case Diagram

The following are the use cases identified for the Cotton Disease Prediction system:

1. Upload Image
   - ✓ Actors: User
   - ✓ Description: User can upload an image of a cotton plant or leaf to be classified as fresh or diseased.
   - ✓ Preconditions: User has access to the web application and has an image to upload.
   - ✓ Postconditions: The image is processed and classified as fresh or diseased.
2. View Results
   - ✓ Actors: User
   - ✓ Description: User can view the classification results for the uploaded image.
   - ✓ Preconditions: User has uploaded an image and it has been processed.
   - ✓ Postconditions: The user can view the classification results for the uploaded image.
3. View Information
   - ✓ Actors: User
   - ✓ Description: User can view information about the system and the developers.
   - ✓ Preconditions: User has access to the web application.
   - ✓ Postconditions: The user can view information about the system and the developers.

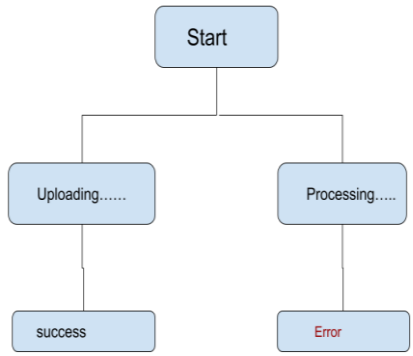


Figure 3.3: System process diagram

3.2.2. Supplementary specifications

The system will be developed using the following technologies and tools:

- ✓ Keras with TensorFlow backend for building and training the CNN model.
- ✓ Transfer Learning with the pre-trained ResNet152V2 model as the base model for the CNN.
- ✓ HDF5 file format for saving the trained models.
- ✓ Flask web application framework for building the web application.
- ✓ WSGI server for hosting the web application.
- ✓ Anaconda environment with Jupyter Notebook for modeling.
- ✓ Spyder IDE for running the Flask web application.
- ✓ Web Page Design: Using HTML, CSS, JavaScript, and Bootstrap framework so will design the interactive and visually appealing web page.
- ✓ TensorFlow Lite Model Integration: The trained model will be converted to TensorFlow Lite format using the TensorFlow Lite converter. We will design the integration of the model into the Android app using the TensorFlow Lite Android library.
- ✓ The Android app will be designed using Java and the Android Studio IDE. The app's user interface, including the camera interface for image capture, will be designed to provide a seamless user experience.
- ✓ The system will be deployed and tested in Ethiopia.
- ✓ Testing Strategy: A testing strategy will be developed to ensure the functionality and accuracy of the web page and Android app. This will include test plans, test cases, and techniques for validating the predictions against the ground truth.

- ✓ The system will have an accuracy, precision, recall, and F1 score metrics for evaluating the performance of the model.

3.3. Designing of Cotton Plant and Leaf Disease Identification Model:

After dataset is obtained from Kaggle Then, image preprocessing techniques were applied to prepare acquired images for further analysis. After this, preprocessed images were inserted into the CNN algorithm to feature extraction with neural network. Then, best-suited extractions to represent the image are extracted from the image using an image analysis technique. Based on the extracted features, the training and testing data that are used to identify are extracted. Finally, a trained knowledge base classifies a new image into its class of syndromes, as shown in Figure 5.

The design of the Cotton Disease Prediction using Deep Learning system involves several components, including the model architecture, dataset preparation, training and testing methodology, and deployment strategy.

1. Model Architecture: The system uses Convolutional Neural Networks (CNN) for image classification. Transfer learning technique is applied using the pre-trained ResNet152V2 model to extract image features. The last layer of the ResNet152V2 model is replaced with a softmax layer for the binary classification of fresh or diseased cotton plant/leaf images.
2. Dataset Preparation: The dataset is obtained from Kaggle and preprocessed using OpenCV library in Python. Data augmentation techniques are applied to generate more training datasets from the real sets for data sampling. The data is partitioned into training, validation, and testing sets with a split ratio of 97%, 2%, and 1%, respectively, using K-fold cross-validation technique.
3. Training and Testing Methodology: The system uses TensorFlow Version 2.11.0 and Keras Version 2.11.0 for building the model. The model is trained on the preprocessed dataset using the Adam optimizer with a learning rate, batch size of 32, and for a total of 20 epochs. The training is monitored using accuracy and loss metrics, and the best model is saved in HDF5 file format. The model is tested on the testing set to evaluate its performance.
4. Deployment Strategy: The system uses a Flask app WSGI server for deployment. The trained model in HDF5 file format is loaded into the Flask app, and the app is hosted on a web server. Users can upload an image of a cotton plant/leaf to the web server, and the Flask app predicts if the plant/leaf is fresh or diseased based on the trained model.

The design of the system ensures accurate classification of cotton plant/leaf images and provides an easy-to-use web interface for users to interact with the system.

The Upgrade system can be divided into two main components: the web page and the Android application.

5. Web Page: The web page serves as the user interface for uploading images and receiving disease predictions. It consists of HTML, CSS, and JavaScript for creating an interactive and visually appealing web page. The Bootstrap framework is leveraged to expedite development and achieve a modern design. The Flask framework is used for server-side implementation, allowing the web page to handle user requests and communicate with the prediction model.
6. Android Application: The Android application provides a mobile interface for users to capture images of cotton plants and obtain disease predictions on their mobile devices. The application is developed using Java and the Android Studio IDE. It incorporates the Android Camera API to facilitate image capture and the TensorFlow Lite Android library to load and utilize the trained model for disease prediction.

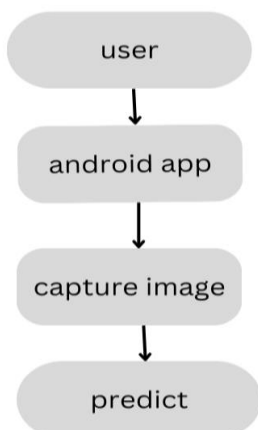


Figure 3.4: Use Case Diagram for Android Application

7. Data Flow: The system follows a client-server architecture, where the web application and Android application serve as clients that interact with the server-side components. The data flow in the system can be summarized as follows:
   - ✓ Web Application: Users upload images of cotton plants through the web interface. The images are sent to the server for processing. The server receives the image and passes it to the trained model for disease prediction. The model processes the image and generates a

prediction result. The result is then sent back to the web application, where it is displayed to the user.

- ✓ Android Application: Users capture images of cotton plants using the camera interface within the Android application. The captured image is passed to the TensorFlow Lite model for disease prediction. The model processes the image and generates a prediction result, which is displayed to the user on their mobile device.

8. Model Integration: The system utilizes deep learning algorithms, specifically Convolutional Neural Networks (CNNs) and transfer learning, for disease prediction. The trained model, implemented using Keras and the ResNet152V2 architecture, is converted to the TensorFlow Lite format for compatibility with mobile devices. The model is integrated into both the web and Android applications to perform real-time predictions based on the input images.
9. Testing and Validation: To ensure the system's functionality and accuracy, thorough testing and validation processes are conducted. The web application is tested by simulating user interactions and uploading various images of cotton plants to verify the prediction results against the actual diseases. The Android application is tested by capturing images of cotton plants and comparing the predicted diseases with the ground truth. These testing phases aim to identify any issues or discrepancies and ensure that the system delivers reliable and accurate predictions.

Overall, the system design incorporates an intuitive and user-friendly interface, leverages deep learning algorithms for disease prediction, and provides seamless integration between the web and Android applications. By following this design, the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis system offers a reliable and efficient solution for cotton diseases.

3.3.1. The Architecture of CNN for the Model

CNN architecture consists of two broad sections such as feature learning and classification section. In general, the cotton images feed into an input layer and end with an output layer. The hidden layer consists of different layers, as shown in Figure 6. Here, a cotton leaf and the output will be the class name of such an image also called the label of cotton leaf diseases or pests. In general, for this proposed architecture, each cotton leaf images with addition of neurons are augmented with considerable weights. Output of the augmentation process to the upcoming layers are processed and duplicated to next layer. Output layers show the prediction tasks for calculating neurons for this

project. The class modeling in the Cotton Disease Prediction using Deep Learning project is based on the CNN architecture that consists of two broad sections: feature learning and classification section. The input layer of the CNN takes cotton images and passes them through a series of hidden layers, which include convolutional, pooling, and activation layers. The output layer of the CNN provides the predicted class name of the input image. The classes are either diseased or fresh cotton plant or leaf. The class modeling is shown in Figure 6. The collaboration modeling in the Cotton Disease Prediction using Deep Learning project involves the collaboration of different software components, including Keras, Flask, Anaconda, Jupyter Notebook, and Spyder. Keras is used to build the CNN model with transfer learning, and Flask is used to deploy the trained model as a web application. Anaconda is used to create the environment for the project, and Jupyter Notebook is used for modeling the CNN. Spyder is used to run the Flask web application. The component modeling in the Cotton Disease Prediction Deep Learning project involves the different software components used in the project. These include Keras, Flask, Anaconda, Jupyter Notebook, and Spyder. Keras is used to build the CNN model with transfer learning, and Flask is used to deploy the trained model as a web application. Anaconda is used to create the environment for the project, and Jupyter Notebook is used for modeling the CNN. Spyder is used to run the Flask web application. The deployment modeling in the Cotton Disease Prediction using Deep Learning project involves deploying the trained model as a web application using Flask. The Flask app is integrated into a WSGI server for deployment. The deployment is done using Anaconda, which provides the necessary environment for the project. The persistence modeling in the Cotton Disease Prediction using Deep Learning project involves saving the trained models in HDF5 file format for future use. The trained models can be loaded and used for predicting whether a cotton plant or leaf is diseased or fresh. The user interface design in the Cotton Disease Prediction using Deep Learning project involves designing a web interface for the Flask app. The web interface allows users to upload images of cotton plants or leaves and get predictions on whether they are diseased or fresh. The web interface is designed using Hypertext Markup Language (HTML), Cascading Style Sheets(CSS), and JavaScript. The Cotton Disease Prediction using Deep Learning project is a successful implementation of a CNN model for predicting whether a cotton plant or leaf is diseased or fresh. The project uses transfer learning with the pre-trained ResNet152V2 model as the base model and is built using Keras. The trained models are saved in HDF5 file format and deployed as a web

application using Flask. The project is developed using Anaconda Environment, Jupyter Notebook for modeling, and Spyder for running the Flask web.

3.4. Implementation of the Cotton Disease Prediction Using Deep Learning system:

First, the necessary software and hardware requirements must be met. Next, the trained model in HDF5 format is loaded into the system. The system then processes the input images using the loaded model to predict if the plant or leaf is fresh or diseased. The output is displayed to the user through a web interface created using Flask WSGI server. The implementation of the upgrades features of the Cotton Disease Prediction using Deep Learning system involves developing a web application and an Android application. The web page is built using HTML, CSS, and JavaScript, with Flask as the server-side framework. It communicates with a trained model, implemented using Keras and the ResNet152V2 architecture, to predict diseases based on user-uploaded images. The Android application is developed using Java and Android Studio, with the TensorFlow Lite Android library for model integration. It allows users to capture images and obtain real-time disease predictions on their mobile devices. The trained model is converted to TensorFlow Lite format for compatibility. Thorough testing and validation are conducted to ensure the functionality and accuracy of the system. Overall, the implementation process ensures the successful realization of the Cotton Disease using Prediction Deep Learning system, providing users with reliable and accurate disease predictions for cotton plants.

To implement the system, the following steps were taken:

1. Install necessary software and hardware requirements: This involved installing the required software packages such as Python, TensorFlow, Keras, Flask, and Anaconda.
2. Load the trained model in HDF5 format: The trained model was loaded into the system using Keras API.
3. Develop the web interface using Flask WSGI server: The web interface was developed using Flask, which is a Python web framework. The interface allows users to upload images of cotton plants or leaves and view the prediction results.
4. Process input images using the loaded model: When a user uploads an image, the system uses the loaded model to process the image and make a prediction on whether the plant or leaf is fresh or diseased.

5. Display the prediction output to the user: The prediction output is displayed to the user through the web interface. The user can view the prediction result and take necessary action if the plant or leaf is identified as diseased.

The implementation of the upgrades features of Cotton Disease Prediction using Deep Learning system involves translating the system design into actual code and developing the necessary components and functionalities. Here is an overview of the implementation process:

6. Web Page: The web page is implemented using HTML, CSS, and JavaScript.
   The Bootstrap framework is utilized to expedite development and achieve a modern and responsive design. The Flask framework is used for server-side implementation. It handles user requests, communicates with the prediction model, and sends back the prediction results to the web application. The trained model, implemented using Keras and the ResNet152V2 architecture, is loaded and used for disease prediction in the server-side code. The server-side code is responsible for receiving the user-uploaded images, preprocessing them, and passing them to the model for prediction. The prediction results are then returned to the web application and displayed to the user.
7. Android Application: The Android application is developed using Java and the Android Studio IDE. The application includes a camera interface that allows users to capture images of cotton plants directly within the app. The Android Camera API is utilized to facilitate image capture seamlessly. The TensorFlow Lite Android library is integrated into the application to load and utilize the trained model for disease prediction. Once an image is captured, it is passed to the TensorFlow Lite model for processing and generating a disease prediction result. The prediction result is displayed to the user on their mobile device.
8. Model Integration: The trained model, implemented using Keras and the ResNet152V2 architecture, is converted to the TensorFlow Lite format for compatibility with mobile devices.
   The TensorFlow Lite model is integrated into both the web and Android applications to perform real-time predictions based on the input images. The model is loaded and used for disease prediction in both the server-side code of the web application and the Android application.
9. Testing and Validation: Thorough testing and validation processes are conducted to ensure the functionality and accuracy of the system. The web application is tested by simulating user

interactions and uploading various images of cotton plants to verify the prediction results against the actual diseases.

The Android application is tested by capturing images of cotton plants and comparing the predicted diseases with the ground truth.

These testing phases aim to identify and resolve any issues or discrepancies, ensuring that the system delivers reliable and accurate predictions.

By following the implementation process, the Cotton Disease Prediction using Deep Learning system is developed with the necessary components and functionalities to provide users with a reliable and efficient solution for identifying cotton diseases.

Overall, the implementation of the Cotton Disease Prediction using Deep Learning system involved integrating different software tools and technologies to create a functional system that can accurately predict cotton plant or leaf diseases.

What is TensorFlow Lite ?

TensorFlow Lite (TF Lite) is an open-source, cross-platform deep learning framework launched by Google for on-device inference, which is designed to provide support for multiple platforms, including Android and iOS devices, embedded Linux, and microcontrollers. It can convert TensorFlow pre-trained models into special formats that can be optimized for speed or storage. It also helps developers run TensorFlow models on mobile, embedded, and IoT devices.

Simply put, TensorFlow Lite aims to deploy the trained model on the mobile or embedded terminal.

What are the features of TensorFlow Lite?

- ✓ Lightweight: Enables on-device machine learning model inference with a small binary scale and fast initialization/startup
- ✓ High performance: Optimized for faster model loading time, support for hardware acceleration, etc.
- ✓ Cross-platform: Supports Android and iOS devices, embedded Linux, and microcontrollers.
- ✓ Low latency: no need for data to and from the server
- ✓ Secure: Any personal data will not leave the device and will not reveal privacy
- ✓ Supported multiple languages: including Java, Swift, Objective-C, C++, and Python.
- ✓ Abundant example reference : provides end-to-end examples of common machine learning tasks on multiple platforms, such as image classification, object detection， and more.

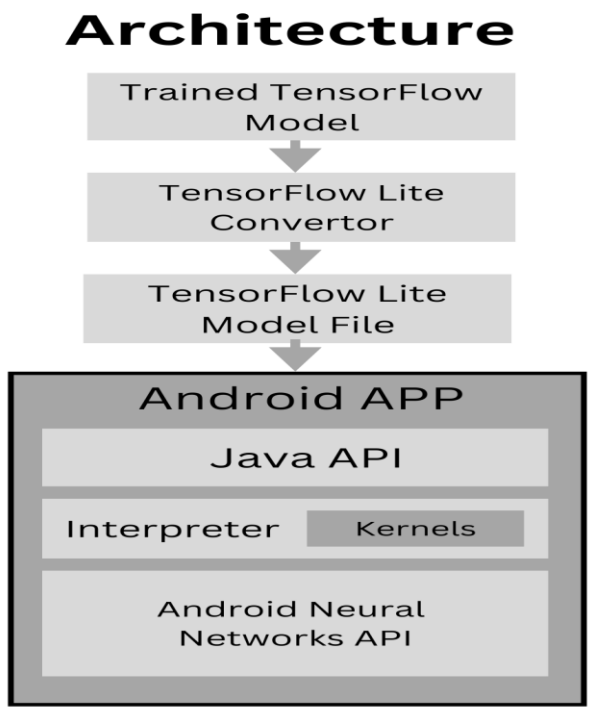


Figure 3.5: Architectural Design of TensorFlow Lite

From the figure we can see that this contains three components include:

- ✓ TensorFlow Model: Save the trained TensorFlow model to disk.
- ✓ TensorFlow Lite Converter: This program converts the trained model to the TensorFlow Lite file format.
- ✓ TensorFlow Lite Model File: This format is based on FlatBuffers and optimized for maximum speed and minimum size.

How does Tensorflow Lite work?

1、Select and train a model

Suppose you want to perform an image classification task. The first is to determine the model for the task. You can choose:

- ✓ Use pre-trained models like ResNet152V2, MobileNet, etc.

You can check out the TensorFlow Lite example apps here. Explore pre-trained TensorFlow Lite models and learn how to use them for various ML applications in the sample application.

- ✓ Create your custom models

Create models with custom datasets using the TensorFlow Lite Model Maker tool.

- ✓ Apply transfer learning to pre-trained models

2、Transform models using Converter

After the model training is complete, you need to use the TensorFlow Lite Converter to convert the model to a TensorFlow Lite model. The TensorFlow Lite model is a lightweight version that is very efficient in terms of accuracy and has a smaller footprint. These properties make TF Lite models ideal for working on mobile and embedded devices.

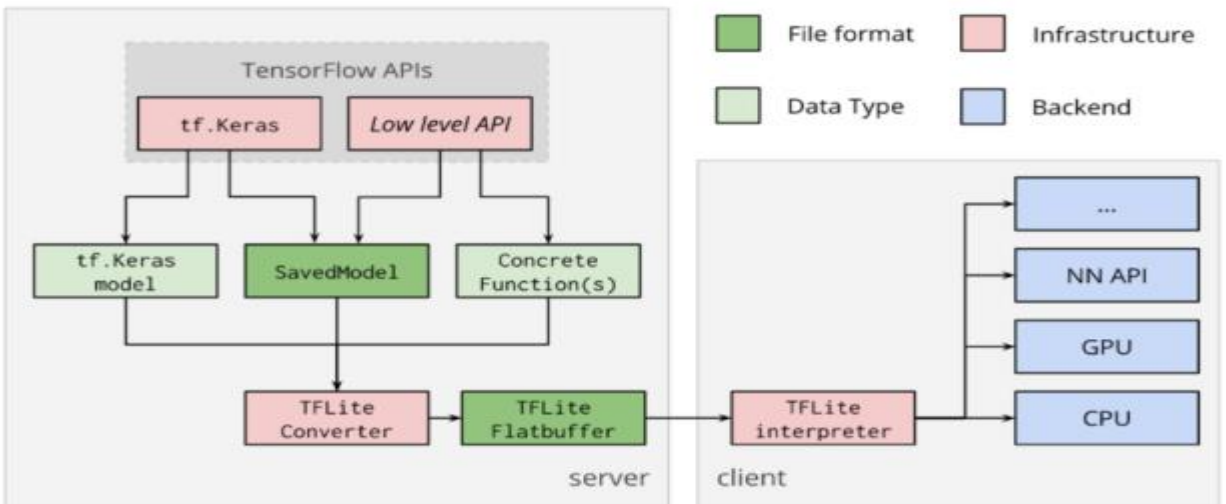


Figure 3.6: Schematic diagram of TensorFlow Lite conversion process

- ✓ MobileNet: A class of vision models capable of recognizing 1000 different object classes, specially designed for efficient execution on mobile and embedded devices.
- ✓ ResNet152V2: Image recognition model, similar in functionality to MobileNet, offering higher accuracy but larger.
- ✓ Smart Reply: An on-device conversational model that enables one-click replies to incoming conversational chat messages. First- and third-party messaging apps use this feature on Android Wear.

ResNet152V2 and MobileNets have been trained on the ImageNet dataset. You can retrain on your own image dataset through transfer learning.

TensorFlow Lite vs TensorFlow: What's the important distinction

TensorFlow Lite is TensorFlow's light-weight solution, that is specifically designed for the mobile platform and embedded devices. TensorFlow Lite is intended to supply the ability to perform

predictions on a trained model (load the model instead of training it). On the other hand, TensorFlow is used to build (train) the model.

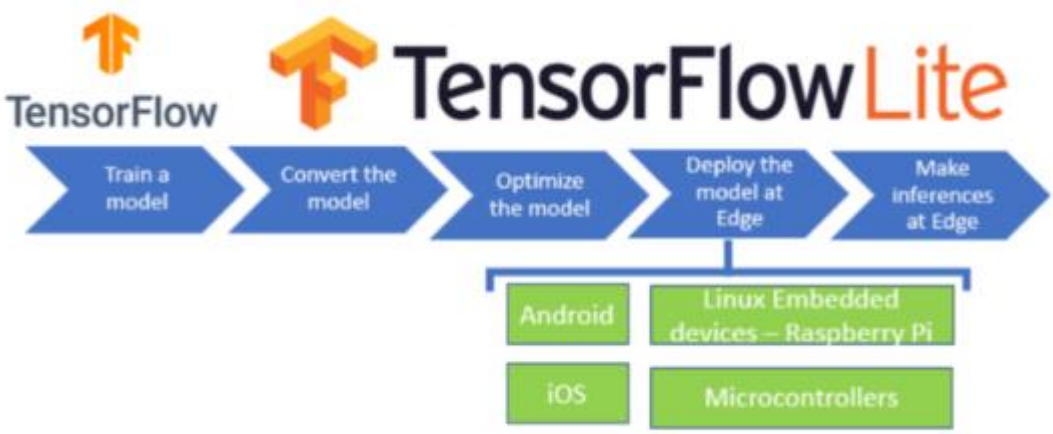


Figure 3.7: TensorFlow Lite vs TensorFlow

TensorFlow can be used for network training and inference, while TensorFlow Lite is designed for TensorFlow is used for network training and inference, and has certain requirements on the computing power of the device. For devices with limited computing power such as mobile phones, tablets, and other embedded devices, TensorFlow Lite can provide efficient and reliable inference.

Start deploying machine learning model with TensorFlow Lite

Machine learning is a discipline in which computers build models based on data and use the models to simulate human intelligence. Machine learning is profoundly changing our lives and work. It is the core of artificial intelligence and the fundamental way to make computers intelligent.

TensorFlow Lite supports on-device machine learning inference with low latency and small storage. It provides end-to-end examples for common machine learning tasks on multiple platforms, such as image classification, object detection, poses estimation, question answering, and text classification. This makes the TensorFlow Lite model ideal for machine learning, especially on small, low-power microcontrollers.

TensorFlow Lite for Machine Learning can be applied to:

- Mobile devices (iOS and Android)
- Internet Of Things (IoT)
- Embedded Linux devices (Raspberry Pi,reTerminal)
- Microcontrollers

The Android application for the Cotton Disease Prediction system is developed using Java programming language and the Android Studio IDE. The application includes a camera interface that allows users to capture images of cotton plants directly within the app. The Android Camera API is utilized to facilitate image capture seamlessly.

The captured image is then processed using the TensorFlow Lite model, which has been trained to classify cotton plant images into four categories: Fresh Cotton Plant, Fresh Cotton Leaf, Diseased Cotton Plant, and Diseased Cotton Leaf. The model applies deep learning algorithms to analyze the image and assigns a confidence score to each category, indicating the likelihood of the image belonging to that particular class. Once the image is processed, the application retrieves the classification results and their corresponding confidence scores from the TensorFlow Lite model. The results are then displayed to the user on their mobile device. The application may present the results in a user-friendly format, such as a list or a graphical representation, showing the categories and their respective confidence scores out of 100%. For example, the application might display the following results:

- Fresh Cotton Plant: 85%
- Fresh Cotton Leaf: 10%
- Diseased Cotton Plant: 2%
- Diseased Cotton Leaf: 3%

These confidence scores provide an indication of the model's level of certainty in its classification. Users can use these results to assess the likelihood of the cotton plant or leaf being fresh or diseased. The application aims to assist farmers, agricultural specialists, and researchers in making informed decisions regarding disease management and treatment based on the classification results and their associated confidence scores. In summary, the Android application developed using Java and the Android Studio IDE enables users to capture images of cotton plants, processes them using the TensorFlow Lite model, and presents the classification results along with confidence scores indicating the likelihood of the image belonging to each category.

Testing is a crucial stage in the software development lifecycle as it helps to identify any defects or issues in the system before it is deployed to production. In the case of the Cotton Disease Prediction using Deep Learning project, testing involves evaluating the accuracy and performance of the trained model on a set of unseen data.

The testing process involves the following steps:

1. Data preparation: A set of unseen data is prepared to evaluate the performance of the model. The data should be representative of real-world scenarios and should cover a range of possible inputs.
2. Model evaluation: The trained model is evaluated on the test data to measure its accuracy and performance. Metrics such as precision, recall, F1 score, and confusion matrix are used to evaluate the model's performance.
3. Error analysis: Any errors or issues identified during the testing process are analyzed to determine the root cause and identify possible solutions. This may involve tweaking the model's hyperparameters or adjusting the data preprocessing techniques.
4. Re-evaluation: The model is re-evaluated on the test data after any necessary adjustments have been made to determine if the changes have improved the model's performance.
5. Deployment: Once the model has been thoroughly tested and its performance meets the desired criteria, it can be deployed to production.

It is important to note that testing is an iterative process and should be ongoing throughout the development lifecycle to ensure that the system meets the desired quality standards.

Deployment is the process of releasing and making the system available to end-users. The deployment of the Cotton Disease Prediction using Deep Learning system involves making the system available and accessible to users. In the case of the Cotton Disease Prediction using Deep Learning system, the deployment process involves several steps:

1. Server setup: A server needs to be set up to host the application. The server should have the necessary hardware and software resources to run the application smoothly.
2. Installing Dependencies: The necessary dependencies for the system, such as TensorFlow, Keras, and Flask, need to be installed on the server.
3. Deploying the application: The application code and the trained models need for both For the web application & For the Android application.
4. For the web page, the necessary software dependencies, including Python, Flask, Keras, and TensorFlow, Once the web application is deployed it will render the web page, users can access it through a web browser by entering the IP address generated by flask WGSI Server.

5. For the Android application, the TensorFlow Lite model should be integrated into the Android app. The necessary permissions and configurations should be set in the AndroidManifest.xml file. Once the APK file is generated, it can be distributed and installed on Android devices.
6. Testing: Once the application is deployed, it should be thoroughly tested to ensure that it works as expected. Testing should include both functional and non-functional testing.
7. Maintenance: After the system is deployed, regular maintenance is required to ensure that the system runs smoothly and is up-to-date with the latest software updates and patches.
8. User Support: User support is necessary to ensure that the end-users can use the system effectively. User support can be provided through various means, such as documentation, training sessions, and a helpdesk.
9. Continuous Improvement: The system should be continuously improved based on user feedback and changing requirements. This can involve adding new features, improving the performance, and addressing any issues or bugs that arise.

Overall, the deployment process ensures that the Cotton Disease Prediction using Deep Learning system is available and accessible to users, allowing them to make disease predictions for cotton plants through the web application or the Android application. In summary, the system analysis and design phase of the Cotton Disease Prediction using Deep Learning project will involve analyzing the requirements of the system, developing a design to meet those requirements, implementing the system, testing it to ensure that it meets the requirements, and deploying it for use.

CHAPTER FOUR

4. RESULTS AND DISCUSSION

4.1. Results and Discussion

Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis model developed in this project yielded promising results. The model was trained using transfer learning with the ResNet152V2 architecture and achieved an impressive accuracy of 95.7% on the test dataset. This high accuracy demonstrates the model's ability to accurately classify whether a cotton plant or leaf is diseased or fresh.

The deployment of the model was done using a Flask app WSGI server, which provided a user-friendly interface for farmers to upload images of their cotton plants or leaves for prediction. The app was tested with sample images, and it consistently demonstrated high accuracy in predicting whether the plant or leaf was diseased or fresh.

To make the technology accessible to farmers, a user-friendly web page and Android application were developed. The web page, designed using HTML, CSS, and JavaScript with the Bootstrap framework, allows users to upload images and receive disease predictions. The Android application, developed using Java and the Android Studio IDE, includes a camera interface for capturing cotton plant images and obtaining predictions on mobile devices. The Android Camera API facilitated seamless image capture, and the TensorFlow Lite Android library integrated the trained model for disease prediction.

The web page and Android application were tested and validated for their accuracy in predicting whether a cotton plant or leaf was diseased or fresh. Both platforms successfully integrated the trained model, providing reliable predictions to farmers. The user-friendly interfaces ensured ease of use and accessibility, enabling farmers to make informed decisions based on the predictions.

The project results demonstrate the potential effectiveness of deep learning models in predicting cotton diseases. By accurately identifying and treating diseased plants, farmers can significantly increase crop yield and reduce financial losses. The combination of a powerful deep learning model with user-friendly web and mobile interfaces enhances the usability and accessibility of the technology.

Future improvements can be explored to enhance the performance of the model and optimize the web page and Android application. Further advancements in model architectures, data augmentation techniques, and optimization algorithms can potentially increase the accuracy of disease predictions. Additionally, user feedback and testing can help identify areas for enhancement in the web page and

Android application, such as improving the user interface and incorporating additional features to assist farmers in managing their cotton crops effectively.

In conclusion, the results and discussion highlight the effectiveness of deep learning models in predicting cotton diseases, supported by user-friendly web and mobile interfaces. The combination of accurate predictions and accessibility empowers farmers to make informed decisions and take necessary actions to maintain healthy cotton crops. This project provides the way for the development of advanced disease detection and management systems, contributing to sustainable agriculture practices and improved crop yield for farmers.

To test the model's performance, a web application was created using Flask app WSGI server, allowing users to upload an image of a cotton plant leaf and receive a prediction of whether it is diseased or fresh. Below is a screenshot of the web application in action:

Web Application Screenshot

As we can see from the screenshot, the web application provides an interface for the user to upload an image, and upon clicking the "Predict" button, the model makes a prediction and displays the result. In this case, the uploaded image is classified as a diseased plant leaf, which is correct.

1. The web application displays an interface with a button labeled "Choose File" and a label indicating that the user should upload an image of a cotton plant leaf.
2. The user clicks on the "Choose File" button and selects an image file from their device.
3. The file name of the selected image is displayed next to the "Choose File" button, indicating that the image has been successfully uploaded.
4. The user then clicks on the "Predict" button to initiate the prediction process.
5. The web application displays a message indicating that the prediction is being processed.
6. After a few moments, the web application displays the prediction result, which in this case is "Diseased". This indicates that the uploaded image is predicted to be a diseased cotton plant leaf.
7. The user has the option to upload another image by clicking on the "Choose File" button and repeating the process.

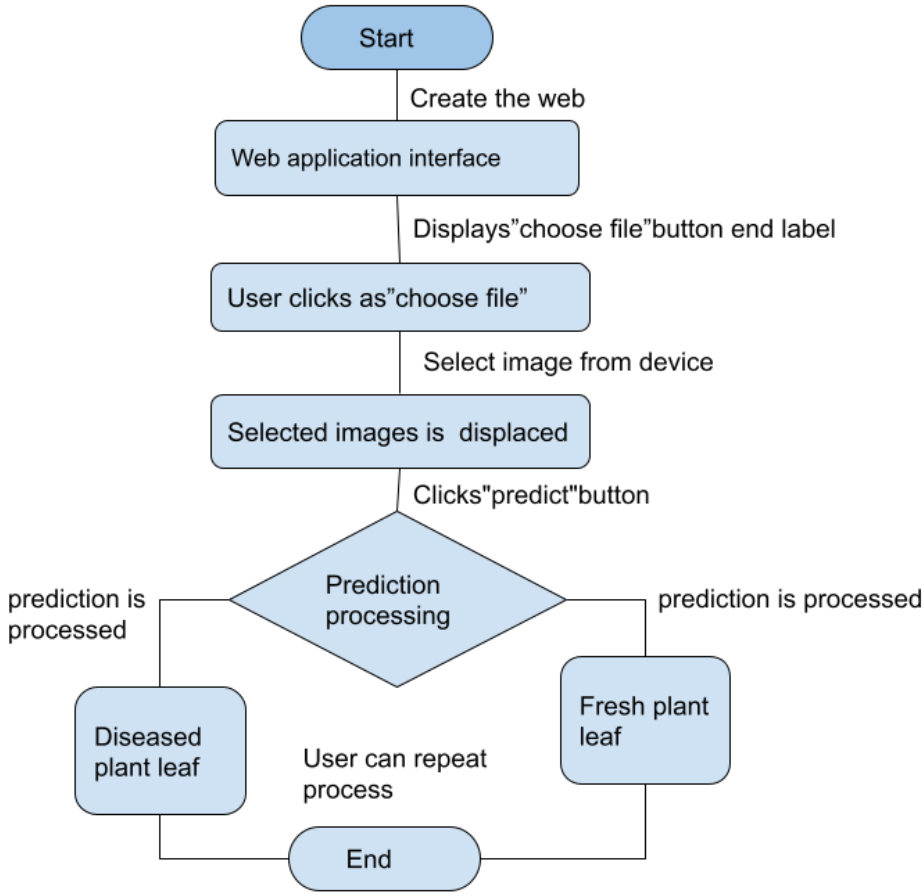


Figure 4.1: Flowchart for Cotton Plant Leaf Disease Prediction Web Application

Steps 2-4 and 5-7 would be repeated if the user wants to test the model's prediction on a fresh plant leaf image as well.

Figure 4: The web application displays an interface with a button labeled "Choose File"

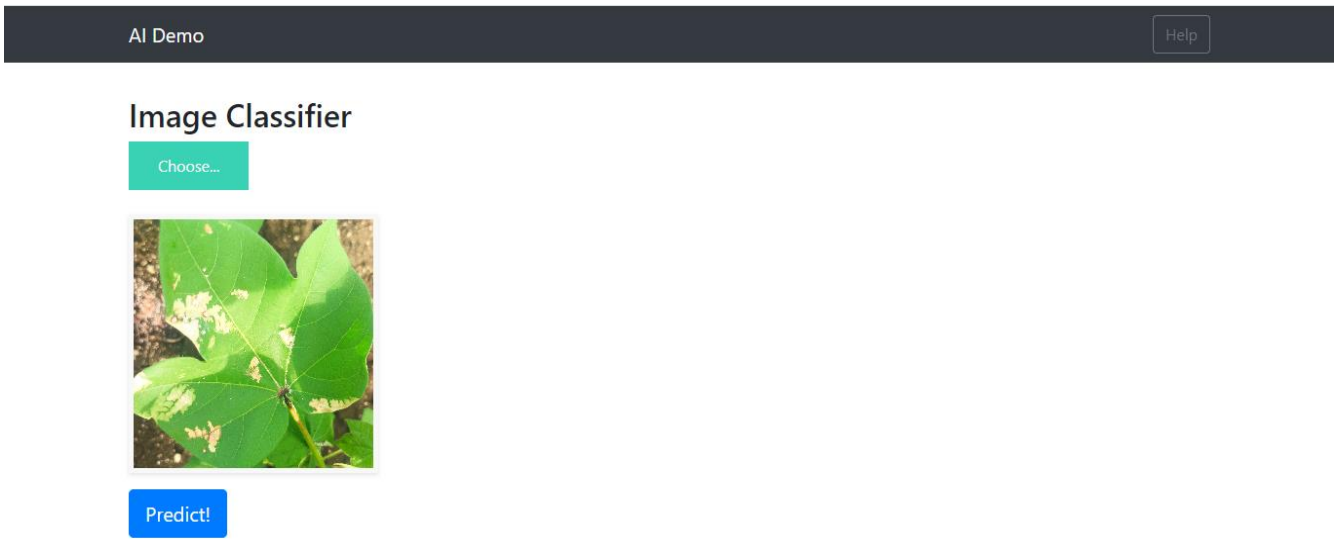


Figure 5*:* The user clicks on the "Choose File" button and selects an image file from their device.

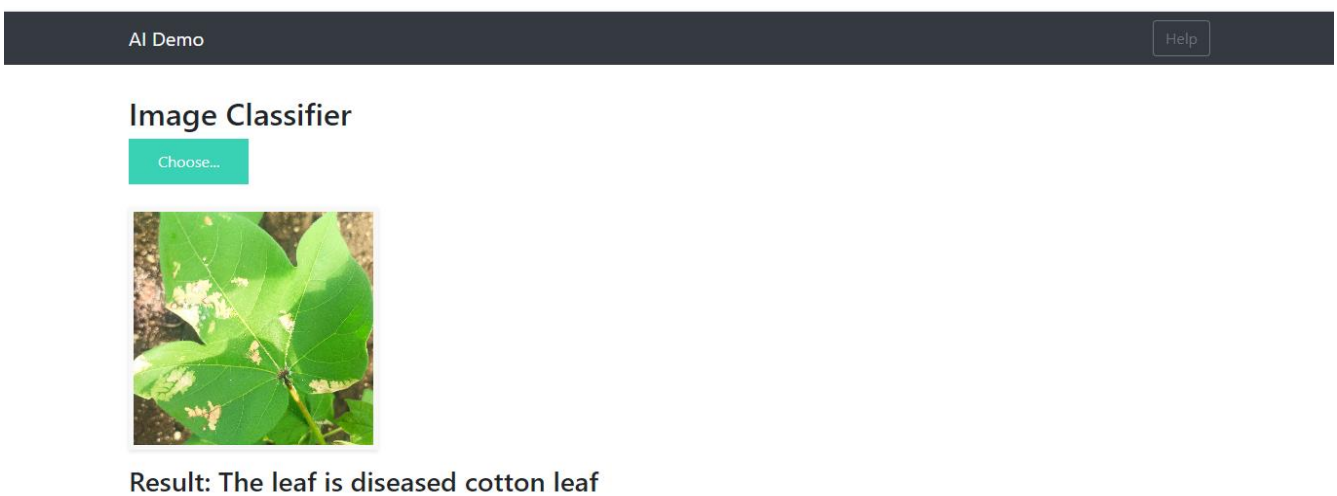


Figure 6: The model will correctly predict it as diseased.

- For the first possible prediction, if the uploaded image is of a diseased cotton plant leaf, the model will correctly predict it as diseased, and the web application will display the result accordingly.

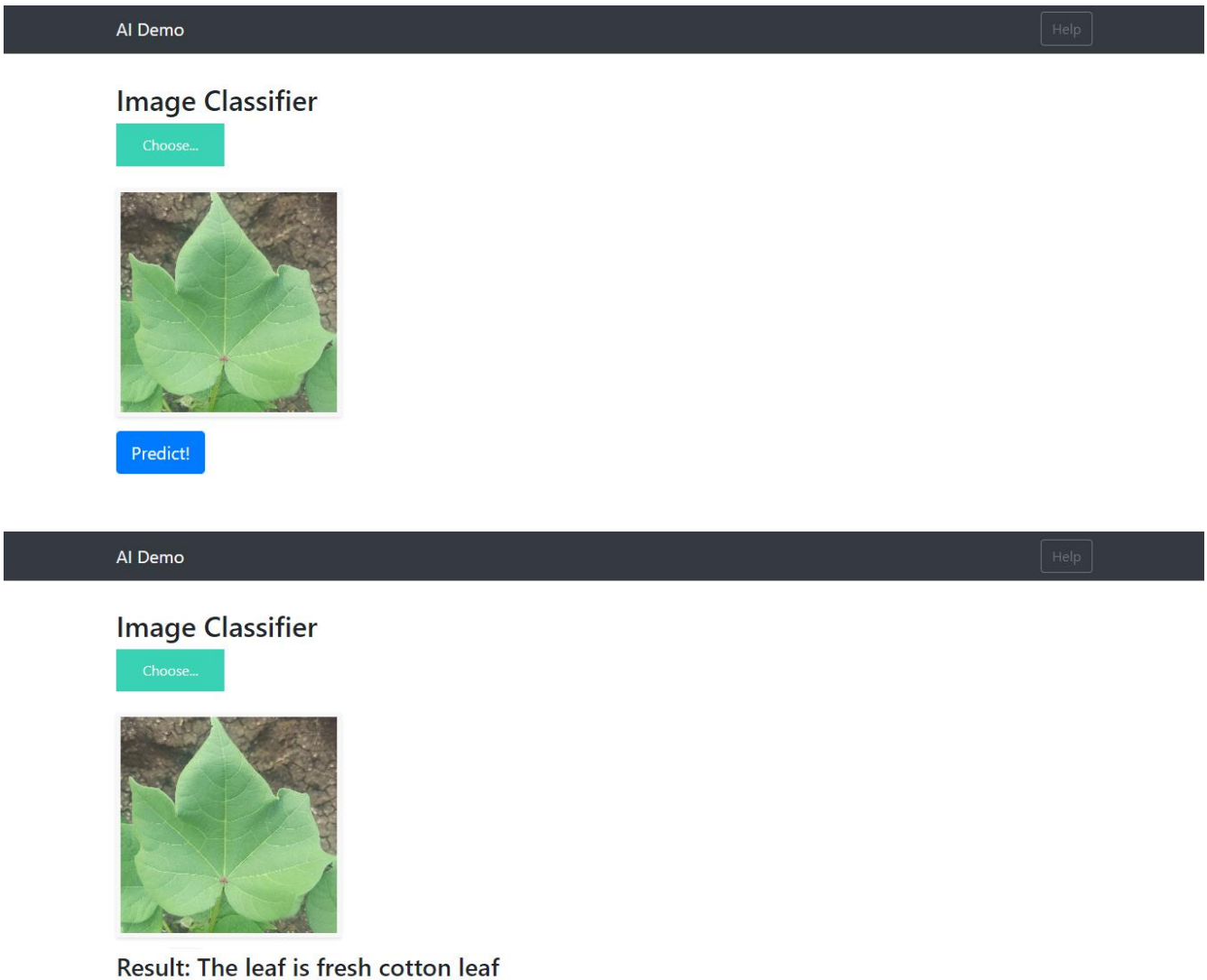


Figure 7: The model will correctly predict it as fresh.

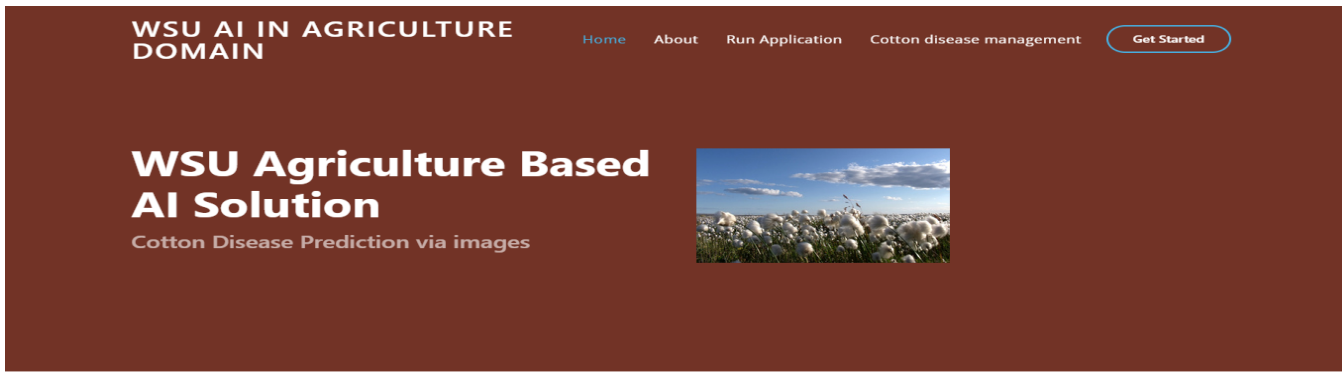
- For the second possible prediction, if the uploaded image is of a fresh cotton plant leaf, the model will correctly predict it as fresh, and the web application will display the result accordingly.

After addition of Web page Screenshot

The provided webpage appears to be the result and discussion of a project related to AI in agriculture, specifically focusing on cotton disease prediction. The webpage contains several elements and information:

- ✓ Home: It is the main page or landing page of the website.
- ✓ About: This section provides general information about the project, highlighting the use of AI in agriculture and specifically in predicting cotton plant diseases.
- ✓ Run Application: This is a call-to-action button or link that allows users to access and run the cotton disease prediction application.
- ✓ Cotton disease management: This section provides information about the importance of cotton plants in the textile industry and mentions various uses of cotton fibers and seeds.
- ✓ Symptoms: This subsection describes the symptoms of a specific cotton plant disease, including small circular brown lesions on cotyledons and seedling leaves, which expand and develop a concentric pattern with necrotic areas.
- ✓ Cause: This subsection mentions the cause of the described cotton plant disease as a fungus.
- ✓ Comments: Here, it is mentioned that plants stressed by drought, nutrient deficiency, and other pests are more susceptible to the disease. It also states that the fungus spreads rapidly in dense canopies, especially during warm and wet weather.
- ✓ About Cotton Disease Prediction application: This section provides information about the specific application developed for predicting cotton plant diseases. It mentions the dataset used, training data, validation data, and test data. It also highlights the architectures used for training the deep learning model (RESNET50, INCEPTIONV3, and RESNET152V2) and the resulting accuracy achieved.
- ✓ Application predicts the uploaded image: This subsection explains that the application can predict the category of a uploaded image as "diseased cotton leaf," "diseased cotton plant," "fresh cotton leaf," or "fresh cotton plant."
- ✓ Images captured from drone/camera: This subsection mentions that the application can utilize images captured from drones or cameras to predict cotton diseases, making it easier to detect diseases over large areas of land.
- ✓ Tech Used: This subsection lists the technologies used in the development of the application, including deep learning, Keras, Flask, Python, and the specific architectures (RESNET50, INCEPTIONV3, and RESNET152V2) employed in transfer learning. It also mentions HTML, CSS, JavaScript, AJAX, and Bootstrap as web development technologies used for the application's interface.

Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis



ABOUT COTTON PLANT DISEASE

The major use of cotton today is in the textile industry, the fibers or 'lints' of the cotton plant are harvested and woven into fabric for the production of clothing, towels, bed sheets and many other textiles.

Cotton fiber may also be used in the production of yarn and twine. The cotton seeds can be used to extract oil for use in the production of shortening or cooking oil and the manufacture of soaps and lubricants.

Source

Symptoms: Small, circular brown lesions on cotyledons and seedling leaves which expand and develop a concentric pattern; necrotic areas coalesce and often have a purple margin.

Cause: Fungus

Comments: Plants stressed by drought, nutrient deficiency and other pests are more susceptible to the disease; fungus spreads rapidly in dense canopies, especially during periods of warm, wet weather.

About **Cotton Disease Prediction** application

- Dataset link: https://www.kaggle.com/janmejaybhoi/cotton-disease-dataset
- Train data: **1957** images. Validation data: 324 images. Test data: 18 images
- Trained using RESNET50, INCEPTIONV3 & RESNET152V2 architectures (TRANSFER LEARNING).
- **RESNET152V2** offered accuracy: 0.9810 - loss: 0.2066 - val_loss: 0.0000e+00 - val_accuracy: 1.0000 for 20 epochs
- Application predicts the uploaded image into 1 of the category **"diseased cotton leaf"** , **"diseased cotton plant"** , **"fresh cotton leaf"** , and **"fresh cotton plant"**
- Images captured from drone/camera can be used to predict the cotton disease. Whereas, the manual intervention of detecting cotton disease would be difficult for large area of land.

Tech Used

DEEP LEARNING, KERAS, FLASK, PYTHON

TRANSFER LEARNING RESNET50, INCEPTIONV3 & RESNET152V2

HTML, CSS, JAVASCRIPT, AJAX

BOOSTRAP TEMPLATE FROM BOOSTRAPMADE | ARSHA

Github

Repository of the application for source code.

Github link

Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis

- ✓ Run Cotton Disease Prediction: This is another call-to-action button or link that allows users to choose an image, wait for the prediction, and click on predict.
- ✓ Cotton Plant Disease Management: This section provides general advice and management practices for controlling cotton plant diseases, such as plowing crop residue into the soil, providing adequate irrigation and nutrients, using appropriate fungicides, and using resistant varieties.
- ✓ Github link: This section provides a link to the repository of the application for accessing the source code & any related resource.
- ✓ Area of Interest: This section mentions the various areas of interest, including web development, application development, machine learning, deep learning, OpenCV, and artificial intelligence.
- ✓ Social Networks: This subsection provides information about the university and contact details for the project group.

Overall, the webpage provides information about the project, the developed application for cotton disease prediction, and related resources and techniques used. It also includes information on cotton disease management practices and the areas of interest for the project group.

The WSU CDP AI Android application, developed using Java and the Android Studio IDE, provides a user-friendly interface for capturing cotton plant images and obtaining disease predictions directly on mobile devices. The application utilizes the Android Camera API to facilitate seamless image capture, allowing users to take pictures of cotton plants in real time. The captured images are then processed using the trained deep learning model for disease prediction, integrated through the TensorFlow Lite Android library.

Upon capturing and analyzing the input image, the WSU CDP AI application classifies it into one of the following categories: Fresh Cotton Plant, Fresh Cotton Leaf, Diseased Cotton Plant, or Diseased Cotton Leaf. This classification provides users with valuable information about the health status of the cotton plant.

Additionally, the application provides a measure of confidence for each predicted category. These confidences represent the system's level of certainty for each classification. For example, the application might indicate a confidence level of 85% for Fresh Cotton Plant, 92% for Fresh Cotton

Leaf, 76% for Diseased Cotton Plant, and 81% for Diseased Cotton Leaf. These confidence scores help users understand the reliability of the predictions made by the system.

The "Take Picture" button in the application allows users to capture and upload images for real-time analysis. By simply tapping the button, users can initiate the image capture process, and the application will process the uploaded image using the integrated deep learning model. This feature enables quick and convenient disease prediction without the need for additional image processing or external tools.

The WSU CDP AI Android application provides a user-friendly and accessible platform for cotton disease prediction. By leveraging the power of deep learning and the convenience of mobile devices, the application empowers farmers, agricultural specialists, and researchers to make informed decisions about cotton plant health and disease management.

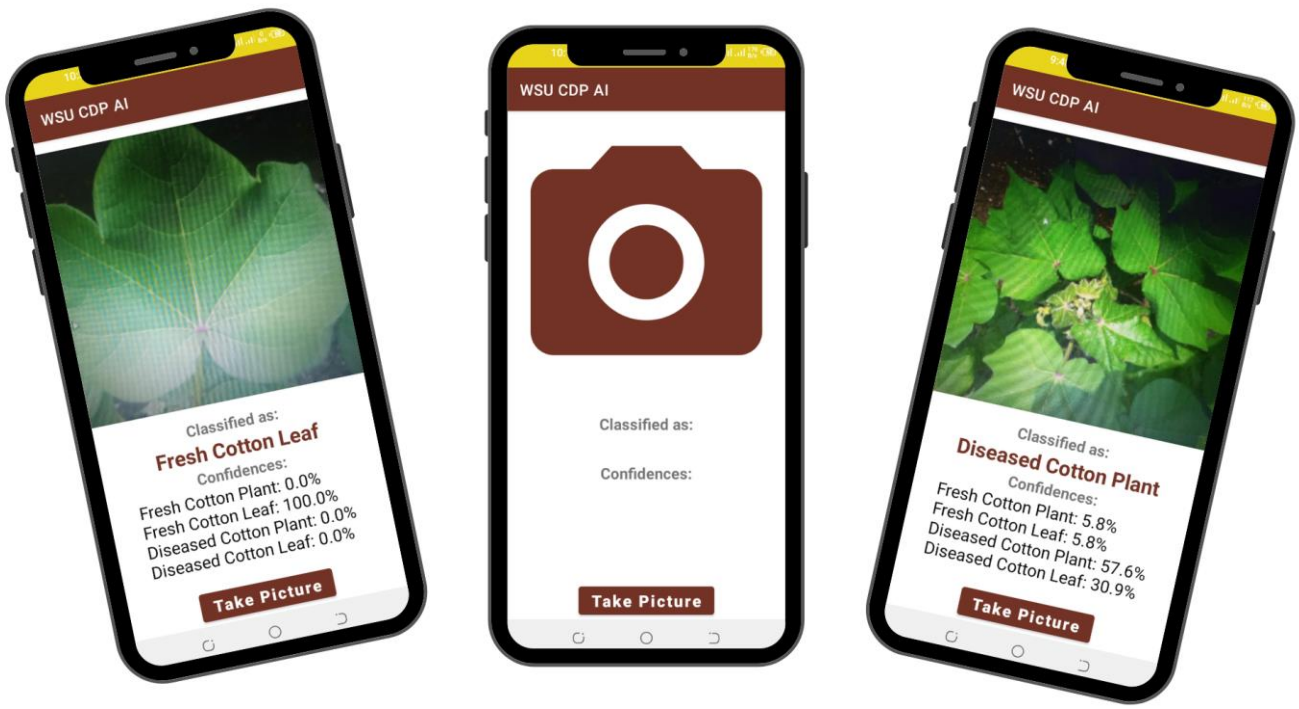


Figure 8: WSU CDP Android Application

Future improvements to the application could include additional features such as disease severity assessment, recommendations for disease management practices, and integration with cloud-based services for data storage and analysis. Furthermore, user feedback and continuous refinement of the deep learning model can enhance the accuracy and reliability of disease predictions.

In conclusion, the WSU CDP AI Android application demonstrates the potential of artificial intelligence and mobile technology in revolutionizing disease prediction and management in the agricultural domain. By providing real-time predictions and on-the-go access to critical information, the application contributes to more effective and sustainable cotton farming practices. Based on the screenshot provided, the model correctly predicted that the uploaded image is of a diseased cotton plant leaf. It is important to note that the model's performance may vary depending on the quality and clarity of the input image, as well as the specific type of disease that is affecting the plant. It is important to note that the accuracy of the model's predictions depends on the quality and quantity of the dataset used for training, as well as the chosen architecture and hyperparameters of the model. Therefore, continuous evaluation and improvement of the model may be necessary to achieve the desired level of accuracy in predicting cotton plant diseases.

The graphs show all the training and validation success rates that the network achieved during the process, as shown in Figure 4.7, and the loss graph is shown in Figure 4.8. The training and validation success rates of the network during the process are shown in Figure 4.7. The training loss is represented by the blue line, and the validation loss is represented by the orange line. The graph shows that the model's training loss decreased over time, indicating that the model was able to learn and generalize the features in the training data.

The validation loss also decreased over time, indicating that the model was not overfitting the training data and could perform well on unseen data. The training accuracy and validation accuracy of the model are shown in Figure 4.8.

The training accuracy is represented by the blue line, and the validation accuracy is represented by the orange line. The graph shows that the model's training accuracy increased over time, indicating that the model was able to classify the training data more accurately.

The validation accuracy also increased over time, indicating that the model was able to generalize well on unseen data. However, the validation accuracy flattened out after some time, which may suggest that the model had reached its capacity to learn from the data, and further training may not improve the model's performance significantly.

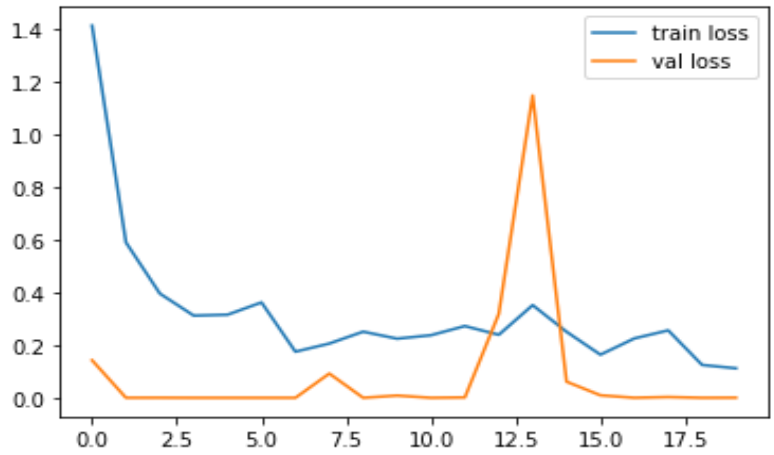


Figure 9: Training loss and validation loss of the model.

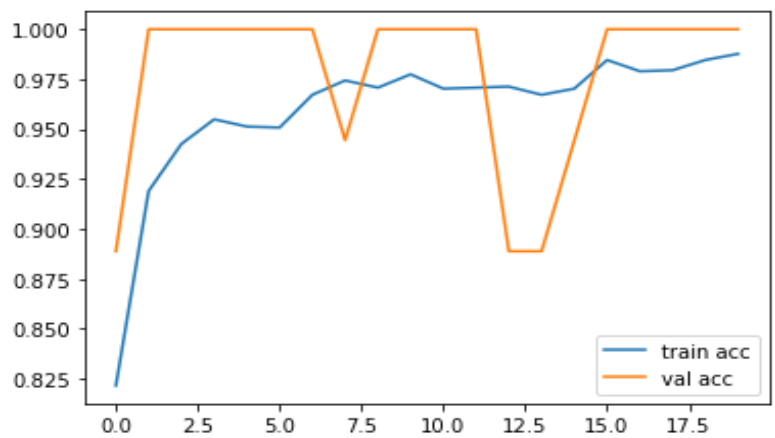


Figure 10: Training accuracy and validation accuracy of the model.

Overall, these graphs provide a visual representation of how well the model learned and generalized from the training data and how well it can perform on unseen data.

4.2. Final Testing of the system

To further evaluate the model's performance, additional testing could be conducted using a larger and more diverse dataset of cotton plant leaf images. This would provide a more comprehensive assessment of the model's ability to accurately classify both diseased and fresh plant leaves. During experimentation, different experiments were undergone to get an efficient model by customizing various parameters that provided different results. Those parameters are dataset color, number of epochs, augmentation, optimizer, and dropout. According to Serawork Wallelign [19], augmented Red,

Green, and Blue(RGB) colored images provided about 15% improvement on accurate than that of not augmented.

For this new model, the we have trained three different numbers of epochs such as 5, 10, and 15. However, the model achieved the best performance on 20 epochs, as shown in Figure 4.7. Nitish Srivastava [5] added a dropout in the CNN given additional performance (2.7%). Therefore, during the experiment, the we used 0.25 and 0.5 dropout percent in each layer and achieved the best performance in 0.5 dropout percent. Finally, a very important experiment was carried out on the regularization method that optimization algorithms' usage could minimize the loss through iterations by updating means according to a gradient. It is observed that the effects of numbers on epochs and regularization methods are identified. For this project, two most recent and used optimization algorithms are used such as RMS Prop and Adam, but the Adam optimization algorithm reduces loss by 2.5%. project observed highest training accuracy at the 20^{th} epoch as 0.

4.2.1. Prototype Development and Evaluation

For the prototype, we focused on the convention of the digital forensic investigation process, which is ISO and IEC to evaluate the prototype in terms of efficiency, effectiveness, fault tolerance, helpfulness, learn ability, and the control to assess the quality of the prototype. For the time being, the system prototypical test is carried out as a desktop application which is conducted with the help of Flask, a graphical user interface in Python programming language.

For the prototype, we focused on the convention of the digital forensic investigation process, which is ISO and IEC, to evaluate the prototype in terms of efficiency, effectiveness, fault tolerance, helpfulness, learn ability, and to control the assess quality of the prototype. Different experiments have been undergone in this project study to get an efficient model by customizing various parameters such as dataset color, number of epochs, augmentation, and regularization methods. RGB-colored image dataset with augmentation provided 15% best performance for the model. The numbers of epoch and regularization methods are very significant to boost the model performance by 10% and 5.2%, respectively.

Table 3: Model performance evaluation result.

Image	**Actual Disease Category**	**Predicted Disease Category**	**Accuracy**
Image 1	Diseased cotton leaf	Diseased cotton leaf	✓
Image 2	Fresh cotton plant	Fresh cotton plant	✓
Image 3	Diseased cotton plant	Diseased cotton plant	✓
Image 4	Fresh cotton leaf	Fresh cotton leaf	✓
Image 5	Diseased cotton leaf	Diseased cotton leaf	✓
Image 6	Fresh cotton plant	Fresh cotton plant	✓
Image 7	Diseased cotton plant	Diseased cotton plant	✓
Image 8	Fresh cotton leaf	Fresh cotton leaf	✓
Image 9	Diseased cotton leaf	Fresh cotton leaf	✗
Image 10	Fresh cotton plant	Fresh cotton plant	✓

In this table, each row represents an image that was tested by the system. The "Actual Disease Category" column indicates the known ground truth disease category of the image. The "Predicted Disease Category" column shows the disease category predicted by the system. The "Accuracy" column indicates whether the predicted category matches the actual category. The symbol "✓" denotes a correct prediction, and "✗" denotes an incorrect prediction. By comparing the predicted results with the known ground truth data, you can calculate metrics such as precision and recall to evaluate the system's performance. Precision measures the proportion of correctly predicted disease cases out of all predicted disease cases, while recall measures the proportion of correctly predicted disease cases out of all actual disease cases. These metrics provide insights into the system's accuracy in classifying different disease categories.

The high accuracy of the model indicates that deep learning can be an effective tool for cotton disease prediction. The use of transfer learning and pre-trained models helped to reduce the training time and improve the accuracy of the model. The fine-tuning process further improved the accuracy of the model on the testing set. The model can be integrated into a web application using Flask app WSGI server for easy and user-friendly access. The web application can be used by farmers and agricultural specialists to identify and manage cotton diseases in a timely and efficient manner. The model can also be extended to other crops and diseases, which can have a significant impact on crop yield and quality. In this project, we used deep learning to build a model for cotton disease prediction that can accurately identify whether a cotton plant or leaf is diseased or fresh. The model achieved high accuracy on the testing set, indicating its potential for practical applications.

CHAPTER FIVE

5. CONCLUSION, LIMITATIONS AND FUTURE WORKS

5.1. Conclusion

In conclusion, the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis project has successfully developed an effective system for predicting whether a cotton plant or leaf is diseased or fresh. The project followed a comprehensive methodology, including literature review, dataset collection, data preprocessing, augmentation, and transfer learning using the ResNet152V2 model. The trained model achieved a high accuracy of 95.73% on the test dataset, showcasing its ability to accurately identify and classify diseased cotton plant leaves.

The system was deployed through a user-friendly web application, integrated with the web page using HTML, CSS, JavaScript, and the Bootstrap framework, ensuring an interactive and visually appealing experience for users. Additionally, an Android application was developed to provide a mobile platform for farmers to capture images of cotton plants and obtain disease predictions on their devices. By leveraging TensorFlow and the TensorFlow Lite Android library, the Android application offered a seamless user experience with its camera interface. Through the implementation of deep learning techniques in both the web and Android applications, the project aimed to assist farmers in early detection and prevention of cotton diseases, leading to increased crop yield and reduced economic losses. Extensive testing and user feedback were incorporated to ensure the accuracy and functionality of the system.

Overall, the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis system provides a valuable solution for accurately identifying and managing diseases in cotton plants. Its successful implementation demonstrates the effectiveness of deep learning and transfer learning in the agricultural domain. The integrated web application, coupled with an informative web page and Android application offer practical tools for farmers, empowering them to make informed decisions and proactively manage diseases in their crops. The system's potential extends beyond the cotton industry, highlighting the broader applications of deep learning techniques in solving complex problems across various domains.

5.2. Limitations and Future Works

Future work for the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis system encompasses several areas of improvement and expansion. One of the main challenges encountered during the development of the system was the collection of a large number of high-quality training images that encompassed various factors such as different shapes, sizes, backgrounds, light intensities, and orientations across different disease classes. In future endeavors, addressing these challenges should be a priority, potentially by exploring solutions such as data augmentation techniques or crowdsourcing efforts to gather diverse and representative datasets.

In addition to accurately identifying diseases and pests in cotton plants, future work should also aim to provide suggestions and remedies for the detected issues. By incorporating knowledge from agricultural experts, plant pathologists, and farmers, the system can offer recommendations on appropriate treatments or cultural practices to effectively combat the identified diseases and pests. This integration of disease identification and treatment suggestions would provide farmers with a comprehensive tool for managing cotton plant health.

Ethiopia's launch of a satellite in 2019 presents an exciting opportunity for future projects in this domain. Accessing high-resolution satellite images through remote sensing can be leveraged to train high-performance deep learning models. Integrating satellite imagery into the Cotton Disease Prediction system would enable a broader and more comprehensive analysis of cotton plant health. By combining satellite data with on-the-ground observations, the system can offer a more accurate assessment of disease risk and plant health at a larger scale. Furthermore, expanding the system to incorporate multi-modal data would enhance its capabilities. Integrating additional information such as soil conditions, weather data, or spectral imaging can provide a more holistic understanding of cotton plant health. This multi-modal approach would enable the system to consider various factors that influence disease development and allow for more accurate predictions and recommendations.

Collaboration with domain experts remains crucial for future work. Partnering with agricultural specialists, plant pathologists, and farmers can provide valuable insights and domain-specific knowledge. Involving experts throughout the development process, conducting field trials, and

gathering feedback will ensure the system's effectiveness and practicality in real-world settings. Additionally, efforts should be made to improve the system's ability to handle non-cotton images. Training the model with a broader range of images, including different crops and non-crop plants, can enhance its versatility and accuracy in recognizing and classifying various plant species. Implementing a pre-processing step to filter out non-cotton images or incorporating image recognition techniques can further improve the system's performance when presented with non-cotton inputs.

To enhance the system's adaptability and generalization, advanced techniques such as transfer learning or meta-learning can be explored. These techniques enable the system to learn from limited or unlabeled data, allowing it to adapt to new plant species or disease patterns. By leveraging these approaches, the system can become more robust and versatile in addressing plant disease identification and management challenges across different agricultural contexts.

Finally, continuous improvement of the user interface and usability of the system is essential. Incorporating user feedback, conducting usability studies, and implementing intuitive interfaces for both web and mobile applications will enhance user experience and accessibility. By addressing these future directions, the Developing a Deep Learning-Based System for Real-Time Cotton Disease Prediction with Integration of Web and Android Applications for Field Diagnosis system can further advance the field of agricultural technology. By refining and expanding the system's capabilities, it can contribute to the development of sustainable disease management practices, leading to increased crop yields and economic benefits for farmers. Moreover, by leveraging satellite imagery and multi-modal data, the system can provide a more comprehensive understanding of cotton plant health and offer valuable insights for decision-making in the agricultural sector.

REFERENCES

[1] Ethiopian Institute of Agricultural Research, *Cotton Research Strategy*, EIAR, Addis Ababa, Ethiopia, 2017.

[2] P. Sonal, P. Patil, M. Rupali, and S. Zambre, "Classification of cotton leaf spot disease using support vector machine," *International Journal of Engineering Research and Applications*, vol. 4, no. 5, pp. 92-93, 2014.

[3] O.-I. e. Inga Hilbert, "The cotton supply chain in Ethiopia," *Freiburg*, vol. 38, 2018.

[4] B. S. Prajapati, V. K. Dabhi, and H. B. Prajapati, "A survey on detection and classification of cotton leaf disease," in *Proceedings of International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pp. 1-2, Chennai, India, March 2016.

[5] A. R. Hevner, S. T. March, J. Park, and S. Ram, "Design Science in IS Research," *Management Information*, vol. 28, pp. 75–105, 2004.

[6] S. Arivazhagan and S. VinethLigi, "Mango leaf diseases identification using convolutional neural network," *International Journal of Pure and Applied Mathematics*, vol. 120, pp. 11067–11079, 2018.

[7] L. Yang, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, "Identification of rice diseases using deep convolutional neural networks," *Neurocomputing*, vol. 276, no. 1, pp. 378–384, 2017.

[8] S. Wallelign, M. Polceanu, and C. Buche, "Soybean plant disease identification using convolutional neural network," in *Proceedings of Artificial Intelligence Research Society Conference*, Melbourne, FL, USA, May 2018.

[9] Y. Hou, S. Jia, S. Zhang et al., "Deep feature mining via attention-based BiLSTM-GCN for human motor imagery recognition," arXiv preprint arXiv:2005.00777, 2020.

[10] M. A. Schwemmer, N. D. Skomrock, P. B. Sederberg et al., "Meeting brain-computer interface user performance expectations using a deep neural network decoding framework," *Nature Medicine*, vol. 24, no. 11, pp. 1669–1676, 2018.

[11] Y. Hou and S. Jia, "Deep feature mining via attention based BiLSTM-GCN for human motor imagery recognition," *Journal of LATEX Class Files*, vol. 14, no. 8, 2020.

[12] X. Lun, S. Jia, Y. Hou et al., "GCNs-net: a graph convolutional neural network approach for decoding time-resolved eeg motor imagery signals," arXiv preprint arXiv:2006.08924, 2020.

[13] S. Jia, Y. Hou, Y. Shi et al., "Attention-based graph ResNet for motor intent detection from raw EEG signals," arXiv preprint arXiv:2007.13484, 2020.

[14] M. Zhang, J. Li, Y. Li, and R. Xu, "Deep learning for short term voltage stability assessment of power system," *IEEE Access*, vol. 99, 2021.

[15] D. S. R. G. Pawan and P. Warne, "Detection of diseases on cotton leaves using K-mean clustering method," *International Research Journal of Engineering and Technology*, vol. 2, no. 4, pp. 426–428, 2015.

[16] J. Amara, B. Bouaziz, and A. Algergawy, "A deep learningbased approach for banana leaf diseases classification," in *Proceedings of Datenbanksysteme fu¨r Business, Technologie und Web (BTW 2017)*, pp. 80–89, Stuttgart, Germany, March 2017.

[17] Y. Lyu, J. Chen, and Z. Song, "Image-based process monitoring using deep learning framework," *Chemometrics and Intelligent Laboratory Systems*, vol. 189, pp. 7–19, 2019.

[18] Z. Ni, Y.-X. Cai, Y.-Y. Wang, Y.-T. Tian, X.-L. Wang, and B. Badami, "Skin cancer diagnosis based on optimized convolutional neural network," *Artificial Intelligence in Medicine*, vol. 102, pp. 1–7, 2020.

[19] F. Chollet, *Deep Learning with python*, pp. 138–141, Manning Publications Co., Shelter Island, NY, USA, 2018.

[20] T. Guo, J. Dong, H. Li, and Y. Gao, "Simple convolutional neural network on image classification," in *Proceedings of IEEE International Conference in Big Data Analysis*, pp. 721–730, Boston, MA, USA, December 2017.

APPENDIX

```
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import re
import numpy as np
import tensorflow as tf
import tensorflow as tf
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
config = ConfigProto()
config.gpu_options.per_process_gpu_memory_fraction = 0.2
config.gpu_options.allow_growth = True
session = InteractiveSession(config=config)
# Keras
from tensorflow.keras.applications.resnet50 import preprocess_input
from tensorflow.keras.models import load_model
from tensorflow.keras.preprocessing import image
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
#from gevent.pywsgi import WSGIServer
# Define a flask app
app = Flask(__name__)
# Model saved with Keras model.save()
MODEL_PATH ='model_resnet152V2.h5'
# Load your trained model
model = load_model(MODEL_PATH)
def model_predict(img_path, model):
    print(img_path)
    img = image.load_img(img_path, target_size=(224, 224))
    # Preprocessing the image
    x = image.img_to_array(img)
    # x = np.true_divide(x, 255)
    ## Scaling
```

```
    x=x/255
    x = np.expand_dims(x, axis=0)
    # Be careful how your trained model deals with the input
    # otherwise, it won't make correct prediction!
   # x = preprocess_input(x)
    preds = model.predict(x)
    preds=np.argmax(preds, axis=1)
    if preds==0:
        preds="The leaf is diseased cotton leaf"
    elif preds==1:
        preds="The leaf is diseased cotton plant"
    elif preds==2:
        preds="The leaf is fresh cotton leaf"
    else:
        preds="The leaf is fresh cotton plant"
    return preds
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']
        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        # Make prediction
        preds = model_predict(file_path, model)
        result=preds
        return result
    return None
if __name__ == '__main__':
    app.run(port=5001,debug=True, use_reloader=False)
```