Mobile Cotton Plant Disease Detection using Convolutional Neural Network

Kashia Ly

Juan Hernandez

Nathaniel Socash

Department of Mathematics and Computer Science

CSCI 4922 – Senior Project I

Dr. Ahmad Al-Shami

Southern Arkansas University

April 2023

Southern Arkansas University

CSCI 4922 – Senior Project I

Dissertation – Spring 2023

TITLE	:	Mobile Cotton Plant Disease Detection using Convolutional Neural Network
AUTHOR/S	:	Kashia Ly Juan Hernandez Nathaniel Socash
DATE	:	05/01/2023

Dr. Ahmad Al-Shami	______________________
Project Advisor	**Signature**
	______________________
	Date
Dr. Karim	______________________
Department Chair	**Signature**
	______________________
	Date

Acknowledgement

The CPDD Team would like to thank Dr. Al-Shami, our project advisor, for supporting and advising this project. We would also like to thank the Southern Arkansas University Mathematics and Computer Science Department as well as the Southern Arkansas University Agricultural Department for their guidance.

Lastly, we would like to thank our sponsors, the Arkansas Space Grant Consortium and the National Space Grant college and Fellowship Program for the following awards:

National Space grant College and Fellowship Program, Prime Award 80NSSC20M0106

Arkansas Space Grant Consortium, Subaward 242040-22-HU

Disclaimer

Finally, the CPDD team would like to express their gratitude to Dr. Al-Shami and all those who have contributed to this project.

Abstract

The objective of this project is to utilize the complete mobile sensing capabilities of Android smartphones to develop a powerful tool for the diagnosis of diseases in cotton plants. Our approach involves building a lightweight machine-learning model that integrates a Convolution Neural Network (CNN) and advanced image processing techniques. Through our innovative solution, we aim to empower farmers and researchers with a user-friendly and reliable means of detecting and managing cotton plant diseases, leveraging the accessibility and versatility of mobile technology.

Table of Contents

Table of Figures

Introduction

Developing our Cotton Plant Disease Detection model will allow farmers to conveniently monitor their Cotton Farm's health and ensure that the plants are not infected. This will be done using our smart phone application that lets farmers scan a picture of their Cotton Plants and have the picture go through our model to see if their Cotton Plant carries a disease or not. Our model aims to be more accurate than others, which is important since no farmer wants to deal with yield loss because of Cotton Plant diseases. During a 2018 Yield Loss study for instance, *Corynespora cassiicola,* a species of fungus, greatly contributed to Cotton yield loss (Bowen & Kira L 2018).

Our main stakeholders are farmers of Arkansas, and more broadly, cotton farmers of the world. Improving accessibility to disease diagnosis is important and consulting with farmers about their issues is the backbone of this area of research and development.

Cotton production is a major source of income in the state of Arkansas, being ranked #4 in production and #5 in exportation. As such, early detection of diseases within cotton plants is pivotal to maximize plant yield. Traditionally, determining the health of the plant and the disease it may carry is done with bare eyes by a knowledgeable expert. With the rise of technology and the availability of smartphones, quick and accurate disease identification can be brought directly to the field. This would make field maintenance easier and more efficient by decreasing the probability of misdiagnosing a plant due to a lack of knowledge and experience.

The design, implementation, and testing of software systems are pivotal elements in the development and deployment of high-quality, dependable applications. This report describes the design, implementation, and testing of a mobile cotton plant disease detection system that employs a convolutional neural network (CNN) developed to detect diseases with efficiency and accuracy. This project developed using the Python programming language with Jupyter Notebook for readability and functionality and Java for Android mobile app development, offers farmers and researchers in Arkansas a mobile tool to detect diseases of the state's most critical cash crop. The testing and deployment process encompassed rigorous research into the architecture of the convolutional neural network to ensure system functionality, reliability, and performance. The results of the testing process reveal that the system is capable of accurately detecting cotton plant diseases, making it a promising solution for farmers and researchers. This report provides a comprehensive account of the system's design and implementation, as well as a detailed description of the testing and deployment process and its outcomes. The report concludes with a critical discussion of the project's limitations and future directions, along with its potential applications.

Problem Statement

In the context of this research project, the health of cotton plants is significant for farmers in the state of Arkansas. Allowing the diagnosis of plant diseases to be easier and more accessible can help to maximize crop yield, protect capital, and aid research in these diseases by minimizing the data collection process. Regarding cotton production, research proves that using machine learning models as preventive action is critical for increasing production. Our project seeks to build a more robust and accurate model in hopes of furthering crop production. Accessibility is often a challenge, especially in rural areas of the United States. Improving quality of work, creating a tool for farmers to protect their assets, and allowing day-to-day life to add to something much bigger like research and data collection, we hope our application can offer these solutions at the touch of a finger.

Project Objectives

Disease in crops is a constant issue among farmers and can harm the harvest, in turn, directly affecting a farmer's livelihood. By employing image processing techniques, an uploaded image of an unhealthy cotton plant can be analyzed and diagnosed. Thus, aiding in the immediate treatment of infected crops and allow for preventive measures to be taken. Additionally, by utilizing machine learning algorithms to train AI off image datasets of cotton plant diseases, we can increase the accuracy to which the application will be able to recognize and diagnose suspicious spots or markings on sick plants. Health throughout the stages of the plant is key to ensuring quality cotton production. As such, the easier, faster, and earlier diagnosis of any anomaly is greatly beneficial to the farmer who can so quickly lose hundreds in one season. Cotton farmers will also have the benefit of having access to our improved program on their mobile device, making scanning and uploading pictures of their plants very convenient.

Experimenting with several models will be a key element to ensure our success. As previously mentioned, we intend to create a more accurate cotton plant disease detection model. To do so we will need to research different models, test them, analyze the data, compare results, and by evaluating everything that we have gathered, our team will be able to come up with different ideas on how we can create a more accurate model. Once our team has decided on a foundation, we will initiate the creation process and eventually test the model we have created. The end goal is to have our cotton plant disease detection model be more accurate than the models that we have previously tested. And of course, have our program be available to farmers via their mobile device with a simple to use and understand interface.

Feasibility Study

Technical Feasibility

As a group, we consulted Dr. Al-Shami about possible areas of interest in machine learning and possibly fuzzy logic systems as it was an aspect of computer science, we were interested in. Being from the state or Arkansas and having another group of students working on an agriculture related project the previous semester, we stumbled upon papers on plant recognition and considered the idea of using image processing to detect disease in cash crops of Arkansas. We compiled a list of related research papers on Convolutional Neural Networks (CNN), image processing in detecting disease and recognizing specific plants, and databases of cotton plants with certain diseases. We found similar projects and found it appropriate to build our own machine learning model and compare it to other existing models in similar areas of research. The benefit of our project over the others is the accessibility option we are offering. To reiterate, the goal of this project is to not only create a more accurate machine learning model for detecting diseases in cotton plants, but to create a mobile application for farmers to diagnose diseases in their crops and store this data for future use. Ease of access is a large portion of this project.

The technical aspects of this project we have found to be difficult but more than feasible. In accordance with the research papers, we have cited in this report, building a new machine learning model and using newly collected data and preliminary data sets, most of the requirements are met for this project to be underway. Provided we reference similar projects that run parallel to this one and refer to open-source projects for our image processing technology, the only impediments will be our ability to absorb knowledge and translate our research into a satisfactory output.

Operational Feasibility

Our goal is to create a product that can be used day-to-day for farming and researchers alike. The application's purpose is not meant to restructure the work and lives of farmers but rather improve them and allow work to be seamless, fitting into their lives comfortably.

Contribution and Project Deliverables

Project deliverables include learning how to successfully port our program to a mobile application and finding a way to make our model more accurate. Furthermore, we will need to create a user interface that is simple for the farmers to use and understand. Another obstacle we will need to face is being able to make our model more accurate and familiarize ourselves with machine learning and image processing. Finding the necessary images to train our model is also an important objective.

Analysis Approach

While researching various disease detecting programs, our team was able to come up with the idea of developing our own with the addition of it being a mobile application for convenience. However, for our program to be convenient, we needed to focus on something that would benefit the user; something simple and useful to use. We decided to investigate local issues and came up with Cotton Plant Diseases Detection. Our approach was to develop a program that would let the user scan an image of their cotton plant and have an algorithm process the image and diagnose it. Everything would eventually be ported to a mobile application for the benefit of the user/farmers.

Additionally, we intend on letting the user select whether they want to scan a new picture from their mobile camera or if they want to use an existing image from their local gallery. We want to give the user flexibility/options in case they do not have their plant readily available or if they are just curious. Furthermore, our team plans on creating a more accurate detection algorithm and a simple-to-use user interface to ensure our application is quick and easy to navigate. We felt that this approach would be most appropriate for our project since the idea is to create an easy tool for cotton plant farmers to use on the go.

Analysis

To achieve our goal, we will be leveraging the power of machine learning and image processing techniques. Specifically, we will be using a CNN to train our model to detect various cotton plants diseases, such as bacterial blight, curl virus, or Fusarium wilt. We will use a pre-existing dataset of images of diseased and healthy cotton plants to train our model. Once the model is trained, we will use image processing techniques to compare the input image to the trained model and classify it as diseased or healthy.

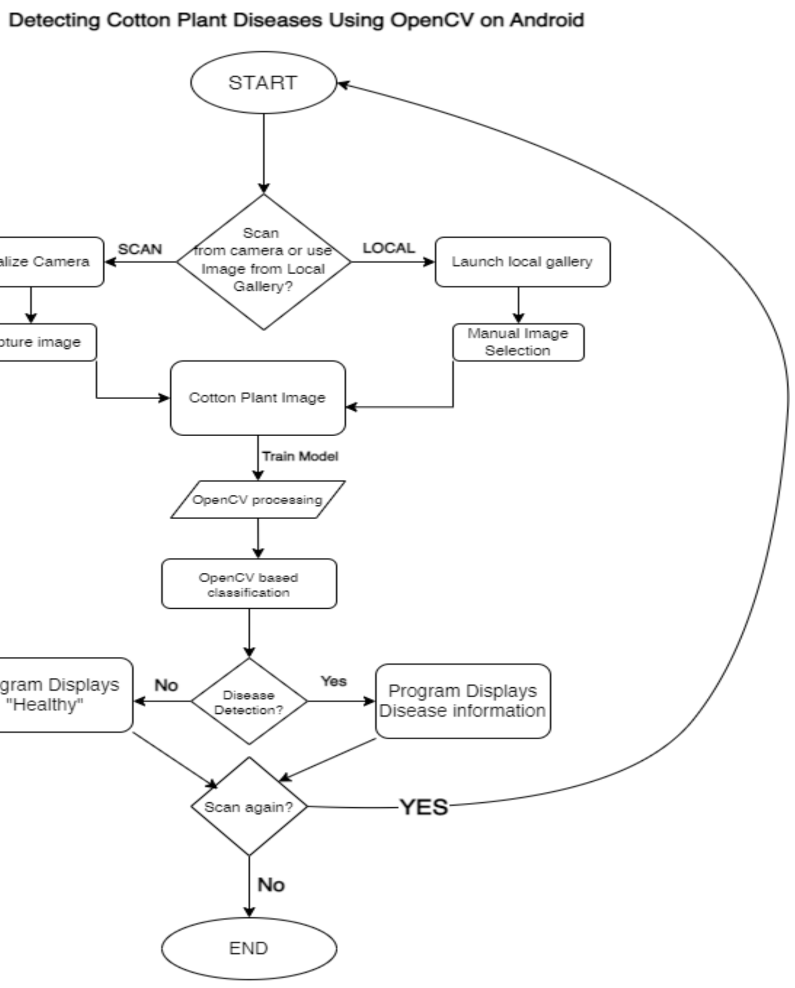

Detecting Cotton Plant Diseases Using OpenCV on Android
START
Scan from camera or use Image from Local Gallery?
SCAN
LOCAL
Initialize Camera
Launch local gallery
capture image
Manual Image Selection
Cotton Plant Image
Train Model
OpenCV processing
OpenCV based classification
Disease Detection?
No
Yes
Program Displays "Healthy"
Program Displays Disease information
Scan again?
YES
No
END

Figure 1 Data Flow Diagram. This diagram describes the data flow for the Mobile Cotton Plant Disease Detection System using OpenCV on Android.

The application will be built using the Android Studio IDE in addition to software like the CameraX API to create the camera user interface and capture images from the camera. Our team will also use optimized, out-of-box model interfaces for traditional machine learning tasks such as the TensorFlow Lite library to integrate the pre-trained model into the application and perform inference on the input images. The application will be designed to be user-friendly and intuitive, with clear instructions on how to use the camera and view the results.

One of the main challenges of this project will be ensuring that the model is accurate and can detect specific cotton plant diseases that predominantly affect our native cash crop. To address this challenge, we will be using a robust dataset of images to train our model. We will also perform extensive testing and validation to ensure that the model performs accurately on a variety of images.

Reflection

With cotton being one of the prevailing crops in the Arkansas region, the importance of knowing whether a plant is diseased or not could be the difference between a successful harvest or a pricey loss. The proposed application will allow a user to determine or confirm whether the cotton plant is healthy and identify the disease. Though we have not finalized the specific disease we will be focusing on, we are currently waiting for feedback from a referred professor versed in cotton plants within the state of Arkansas. In addition, we have found several open-source databases containing a variety of images for several cotton diseases we are hoping to use in the future.

An issue arises in our team's limited knowledge in android app development and coding for use of the integrated camera within a smartphone. This would require us to learn about new platforms and integrate them into our application. Furthermore, we would also need to consider and design a user-friendly interface that can account for those not technologically inclined. The current program we are experimenting with provides us with considerable accuracy but is still lacking in the department of time execution. Taking into consideration the lower processing abilities of a mobile phone, this creates a major issue as speed would only decrease rendering our software inefficient, which creates a separate area of challenge for our team. We would need to lighten the load of our code to make it easier on a smartphone processor all while keeping the accuracy of the trained model and increase execution speed.

Another issue we have come across collectively is with selecting a method of training and processing images for our mobile application and ensuring that our product is lightweight. We must take into consideration that our application will need a light footprint for android devices with older processing architecture and keep in mind that our prototype will be working on preprocessed data. On a similar note, when training our convolutional neural network (CNN) in the past week, we have run into the issue of both overfitting and underfitting. Ensuring that neither training data nor testing and validation data show poor performance is key. We are in the process of determining the correct amount of training data to give our model to produce the most accurate results. The following graph shows the model accuracy of our early testing for our CNN model. The results confirm the equivalent accuracy of the training dataset compared to the validation dataset.

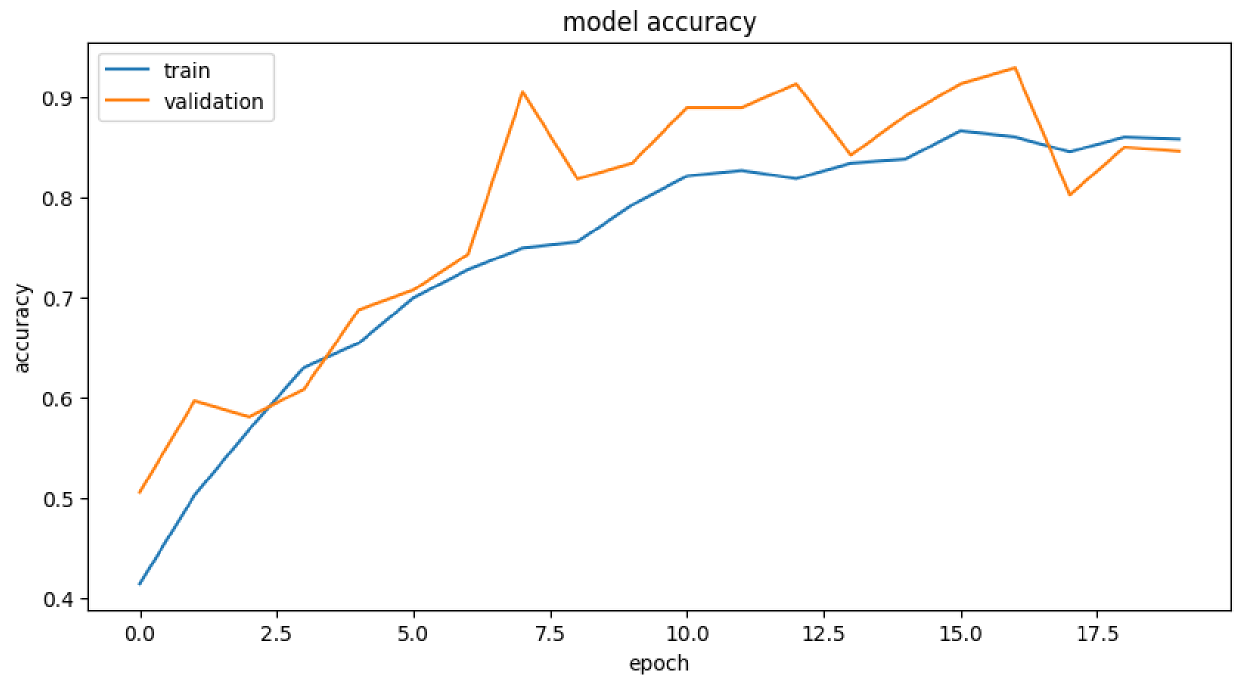


Figure 2 Overfitting. This chart confirms the equivalent accuracy of the training dataset compared to the validation dataset. Overfitting becomes apparent when the model displays low error rates in the training set but higher error rates in the testing set.

Design

In this section, we will discuss the design of our convolutional neural network (CNN) for identifying cotton plant diseases and the design of our Android mobile application. We will also explain our design approach and the decision-making process in this team project. Early on within our project initiation phase, we determined the best way forward for collaboration and the best way to tackle the issue of creating a mobile application for cotton plant disease detection.

In the “Analysis Approach”, our project's decision-making process was elucidated. Through comprehensive discussions and careful deliberations, the project's direction was promptly determined, and the necessity of employing the CNN architecture along with an Android mobile application to facilitate accessibility was established. Once the project concept was finalized, our team created an early schematic diagram that outlined the functionalities of the proposed prototype, as demonstrated above.

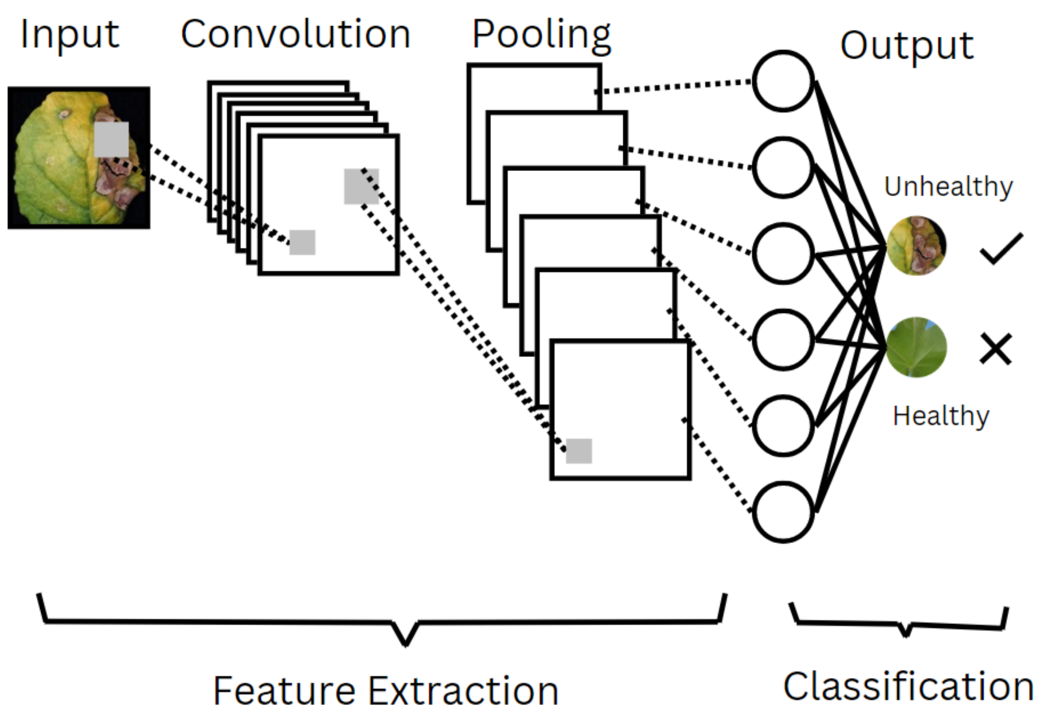


Figure 3 CNN Architecture. This diagram shows the diagram of our CNN architecture. The input is shown as a diseased cotton plant and the output in the "Classification" section shows that the model correctly identifies the binary state of the cotton leaf, which could be healthy or unhealthy.

Our CNN architecture was selected based on its efficiency in terms of performance and computation. The selected architecture consisted of four convolutional layers with max-pooling, followed by two fully connected layers. The activation function used was Rectified Linear Unit (ReLU), and the optimizer used was Adam. Our CNN Architecture Diagram can be seen below.

It was decided that our CNN architecture should be trained on a dataset consisting of images of cotton plants with three possible classifications: diseased cotton leaf, diseased cotton plant, and fresh cotton leaf. The dataset was split into training, validation, and testing sets, with 85%, 10%, and 5% of the data in each set, respectively. These images are preprocessed making the initial stages of the project simpler.

The training process involved minimizing the categorical cross-entropy loss function using the Adam optimizer which we will touch on later. The model was trained for 20 epochs, and the best-performing model was selected based on the validation accuracy. The final model achieved a test accuracy of 91.3%, which indicates that the model is performing well in identifying cotton plant diseases.

To evaluate the performance of the model, we calculated several metrics, including precision, recall, and F1-score, for each class. The precision for diseased cotton leaf was 94.74%, the precision for diseased cotton plant was 90.00%, and the precision for fresh cotton leaf was 90.00%. The recall values for each class were 85.80%, 81.82%, and 81.82%, respectively. The F1-score for diseased cotton leaf was 0.9474, the F1-score for diseased cotton plant was 0.8571, and the F1-score for fresh cotton leaf was 0.8182.

The CNN architecture was implemented using the TensorFlow deep learning library, which is a popular open-source platform for building and training machine learning models. The model was trained on each of our local devices through the development process, some of which did not use top-of-the-line GPUs. However, making full use of the TensorFlow library and using a simple CNN architecture, this allowed for efficient training of the model. We had a few choices within the initiation phase of our project to either use the PyTorch library or the TensorFlow library but quickly discovered that there was more information on the TensorFlow library, and it is more often used for machine learning projects like ours. It was a logical step forward to make the learning and design process as smooth as possible, so TensorFlow was the correct choice.

In addition to the CNN architecture, we also planned and designed a user interface (UI) for our cotton plant disease identification system. The UI was designed with simplicity and ease-of-use in mind, as we wanted to create a tool that could be used by farmers with minimal training in machine learning. The UI consists of a camera interface that allows the user to take a photo of a cotton plant leaf, which is then analyzed by the CNN model's weights. The output of the model is displayed to the user, indicating whether the leaf is diseased or not. For now, the application is only designed as a binary classifier, but it is our goal to classify multiple diseases that have affected cotton plants in the state of Arkansas. Below is the current design of our UI for presentation purposes:

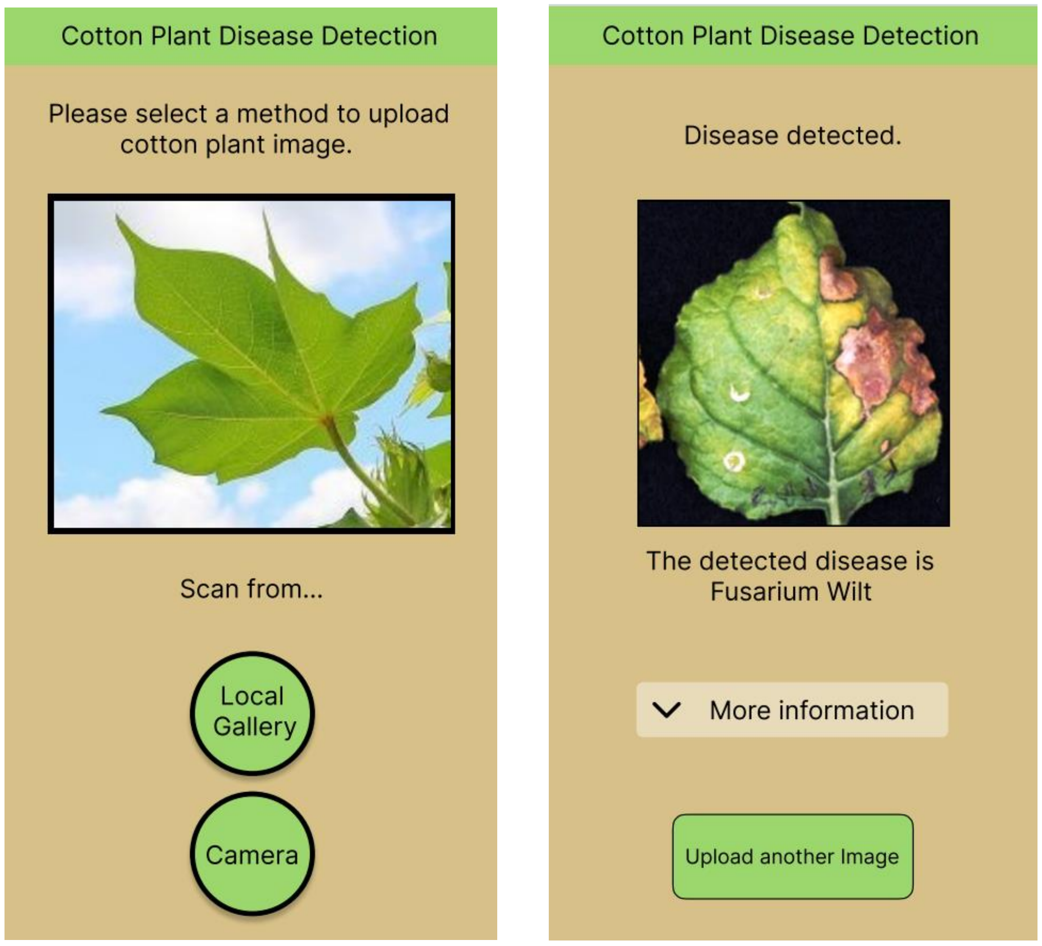
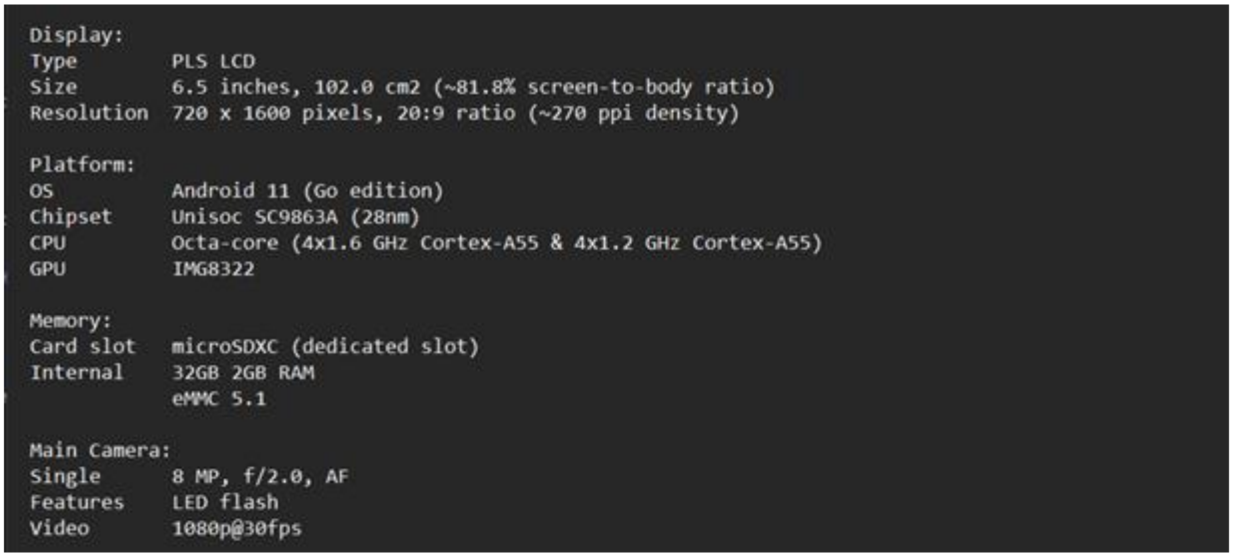

Cotton Plant Disease Detection
Please select a method to upload cotton plant image.
Scan from...
Local Gallery
Camera
Cotton Plant Disease Detection
Disease detected.
The detected disease is Fusarium Wilt
More information
Upload another Image

Figure 4 UI shows the current UI for our Android mobile application. On the left, the mock UI is displaying a menu in which the user can decide whether to take a photo or choose a photo from their phone's gallery. The right shows an image of the classified cotton plant detecting Fusarium Wilt. More information on the disease will be given after the diagnosis.



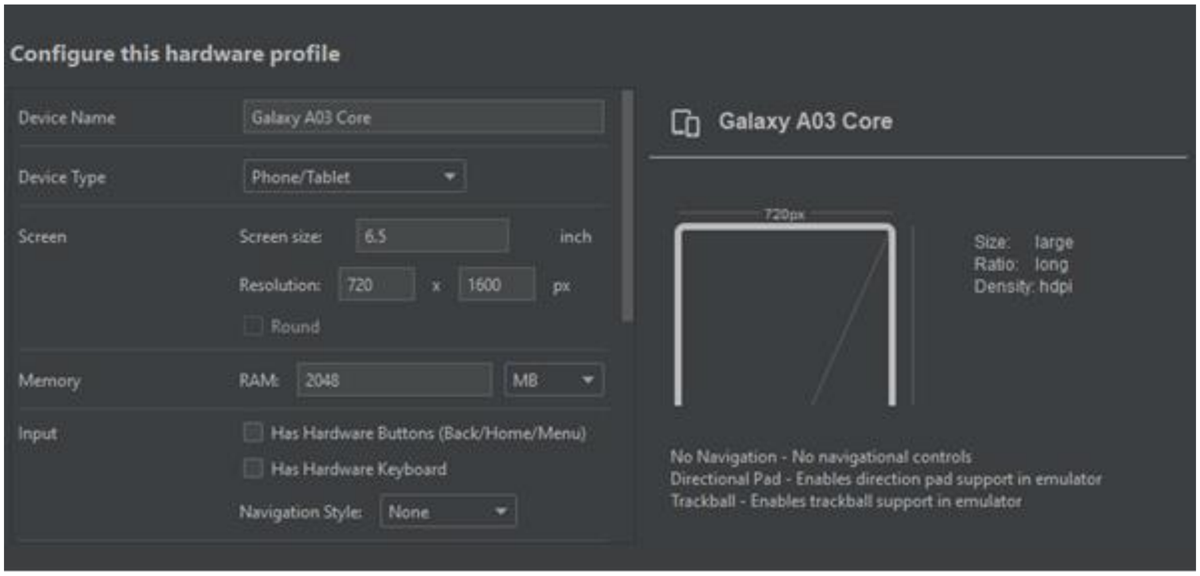
```
Display:
Type        PLS LCD
Size        6.5 inches, 102.0 cm2 (~81.8% screen-to-body ratio)
Resolution  720 x 1600 pixels, 20:9 ratio (~270 ppi density)

Platform:
OS          Android 11 (Go edition)
Chipset     Unisoc SC9863A (28nm)
CPU         Octa-core (4x1.6 GHz Cortex-A55 & 4x1.2 GHz Cortex-A55)
GPU         IMG8322

Memory:
Card slot   microSDXC (dedicated slot)
Internal    32GB 2GB RAM
            eMMC 5.1

Main Camera:
Single      8 MP, f/2.0, AF
Features    LED flash
Video       1080p@30fps
```

Figure 6 Hardware info shows additional hardware profile information for the Samsung Galaxy A03 Core.



Configure this hardware profile

Device Name	Galaxy A03 Core
Device Type	Phone/Tablet
Screen	Screen size: 6.5 inch
	Resolution: 720 x 1600 px
	Round
Memory	RAM: 2048 MB
Input	Has Hardware Buttons (Back/Home/Menu)
	Has Hardware Keyboard
	Navigation Style: None

Galaxy A03 Core

Size: large
Ratio: long
Density: hdpi

No Navigation - No navigational controls
Directional Pad - Enables direction pad support in emulator
Trackball - Enables trackball support in emulator

Figure 5 Hardware Profile shows the hardware profile for the Samsung Galaxy A03 Core smartphone in the Android Studio integrated development environment (IDE).

Finally, we considered the hardware profile of the Android phone that will be used to run our cotton plant disease identification system. We selected the Samsung Galaxy A03 Core as our target hardware platform, as it is widely available, and it was provided to us through the National Space Grant College and Fellowship Program. The Galaxy A03 Core is equipped with an IMG8322 GPU, which can run our mobile application smoothly. We also considered the memory and storage requirements of our model, ensuring that it can run smoothly on the target hardware platform. If there are any issues with the hardware, we hope to contact those which have given us our grant, however, any hardware limitations serve as grounds for good testing. Found below is the hardware profile for the Galaxy A03 Core for Android Studio, and further hardware information.

In conclusion, we have designed a CNN architecture that is efficient and accurate in identifying cotton plant diseases. We have also designed a user interface that is simple and intuitive to use, and we have considered the hardware requirements necessary to run our system effectively. We considered the design flow of our project early on and designed the project's outline as something to strive for and keep us on the right track. Our goal is to help farmers and our system has the potential to provide valuable insights to farmers, allowing them to take proactive measures to prevent the spread of cotton plant diseases and improve their crop yields.

Development

During the development phase of our project, we utilized a range of tools and technologies to bring our proposed system to life. After conducting extensive research and analysis during the design phase, we selected the appropriate development tools and technologies to ensure that our system would function efficiently and effectively.

For our CNN architecture, we used Python as the programming language due to its versatility and extensive support for machine learning libraries. We leveraged the Keras library, which is a powerful tool for building and training neural networks. Additionally, we utilized TensorFlow as our backend to optimize the performance of our model.

To prepare our dataset for training, we used Python's Pandas library to preprocess the data and split it into training, validation, and testing sets. We also used NumPy to handle the numerical operations on the data. Our dataset consisted of images of diseased and fresh cotton leaves and plants, and we labeled them accordingly to train the model to classify them accurately.

For the mobile application development, we used Java as the primary programming language, given that it is the native language for Android development. We utilized Android Studio as our integrated development environment (IDE) to develop our mobile application. The Android application was designed with a user-friendly interface, incorporating the UI design we created in the design phase of the project.

To ensure smooth communication between the mobile application and the CNN model, we have considered RESTful APIs and JSON data transfer. This approach will allow us to effectively integrate the machine learning model into the mobile application.

In terms of the hardware configuration, we selected a widely available Android device with sufficient storage and processing power to run our application. As previously stated, the execution of our machine learning model was left to our local devices. This choice allowed us to optimize the performance of the application and ensure that users could easily access and utilize the functionality of the system.

Throughout the development phase, we frequently tested our system to identify and fix any bugs or errors. We also continually monitored the performance of the system to ensure that it met the project requirements and specifications.

In conclusion, our development phase involved using a range of programming languages, libraries, and tools to develop our proposed system. We utilized Python, Keras, TensorFlow, Pandas, NumPy, Java, and Android Studio to develop our CNN model and mobile application. We also plan to leverage RESTful APIs and JSON data transfer for effective communication between the mobile application and the machine learning model. Our hardware configuration involved selecting a widely available Android device to optimize the performance of the system.

Testing and Deployment

Testing and Deployment is a crucial phase in any software development project, and in the case of our cotton plant disease detection mobile application, it was essential to ensure that the developed solution met the required standards for accuracy and performance. This section documents our work in testing and deploying the developed solution, including the techniques and rationale used in the testing phase.

It should be noted that the model presented in this study is not intended to serve as a binary classifier going forward. Rather, the focus is on detecting specific diseases in cotton plants that have been identified as significant by the Southern Arkansas University Department of Agriculture and Dr. Travis Faske, a plant pathologist from the University of Arkansas. These images in the current dataset include fresh cotton plant, fresh cotton leaf, diseased cotton plant, and diseased cotton leaf. The dataset used for training and testing the model consists of preprocessed images of cotton plants, which have been split into test, validation, and training sets.

To ensure that our model performs well, we used a combination of testing techniques, including train-test split, cross-validation, and evaluation metrics. The train-test split involves randomly dividing the dataset into training and testing sets, with the training set used to train the model, and the testing set used to evaluate the model's accuracy. We used an approximate 85:15 train-test split, further partitioning the test data for validation, taking an approximate split of 85:10:5 of training, validation, and testing respectively. This split was in reference to a standard 70:30 train-split ratio, where 70% of the data would be used for training, and the remaining 30% for testing.

The CNN architecture we selected was found to be efficient in terms of performance and computation. The selected architecture consisted of four convolutional layers with max-pooling, followed by two fully connected layers. The activation function used was ReLU, and the optimizer used was Adam. The model summary is shown below:

Layer (type)	**Output Shape**	**Param #**
conv2d_6 (Conv2D)	(None, 128, 128, 32)	896
max_pooling2d_4 (MaxPooling 2D)	(None, 64, 64, 32)	0
dropout_6 (Dropout)	(None, 64, 64, 32)	0
conv2d_7 (Conv2D)	(None, 64, 64, 32)	9248
conv2d_8 (Conv2D)	(None, 64, 64, 64)	18496
max_pooling2d_5 (MaxPooling 2D)	(None, 32, 32, 64)	0
dropout_7 (Dropout)	(None, 32, 32, 64)	0
flatten_2 (Flatten)	(None, 65536)	0
dense_6 (Dense)	(None, 128)	8388736
dense_7 (Dense)	(None, 128)	16512
dropout_8 (Dropout)	(None, 128)	0
dense_8 (Dense)	(None, 4)	516

To test the efficiency and accuracy of our CNN architecture, we utilized several testing techniques. First, we performed a quantitative evaluation of the CNN architecture using accuracy and loss metrics. We trained our CNN model for 20 epochs and observed the loss and accuracy at each epoch. We found that the loss decreased gradually with each epoch, while the accuracy increased steadily. After the final epoch, our model achieved an accuracy of 91.3% on the test set.

A qualitative evaluation was performed on the CNN architecture using visual inspection of the output images. We tested the CNN model on a set of images that contained both healthy and diseased cotton plant leaves. The model successfully classified diseased leaves with high accuracy. As a mobile application tool, this research can aid farmers and agricultural experts to make informed decisions quickly and conveniently from the touch of their handheld android device.

Cross-validation is a technique used to validate the performance of our model. We used an approach similar to a 5-fold cross-validation approach, where the data is divided into five parts, and the model is trained and tested five times, with each fold being used as the testing set once. This approach ensures that the model is robust and not overfitting to the training data. We have yet to run into the problem of overfitting or underfitting, as our comparative metrics for model accuracy and model loss show clearly.

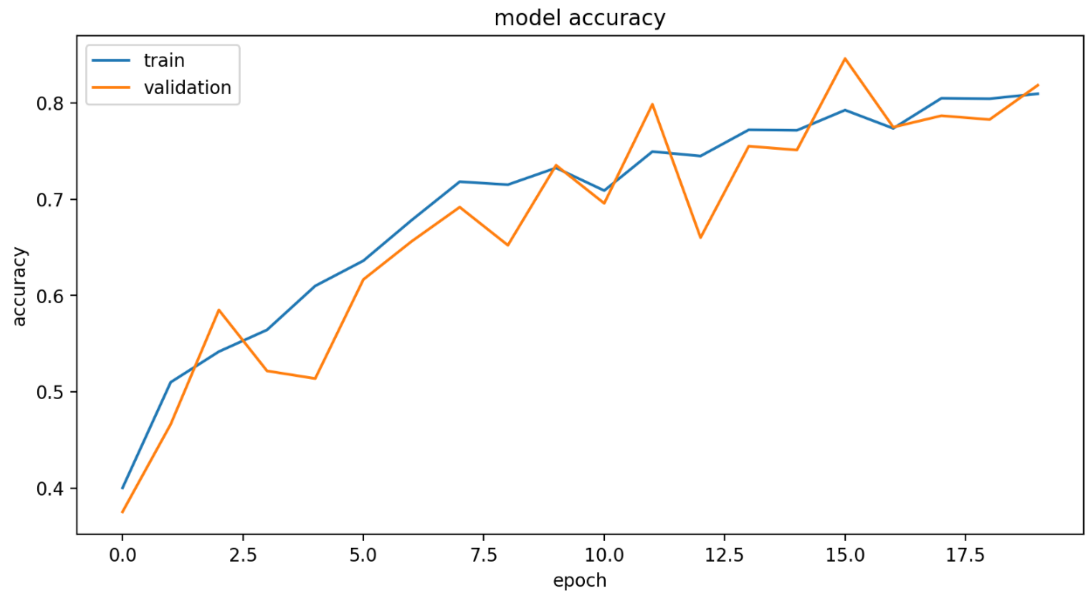


Figure 7 Model Accuracy shows the model accuracy of a random training run during the cross-validation stage. This figure compares training and validation accuracy.

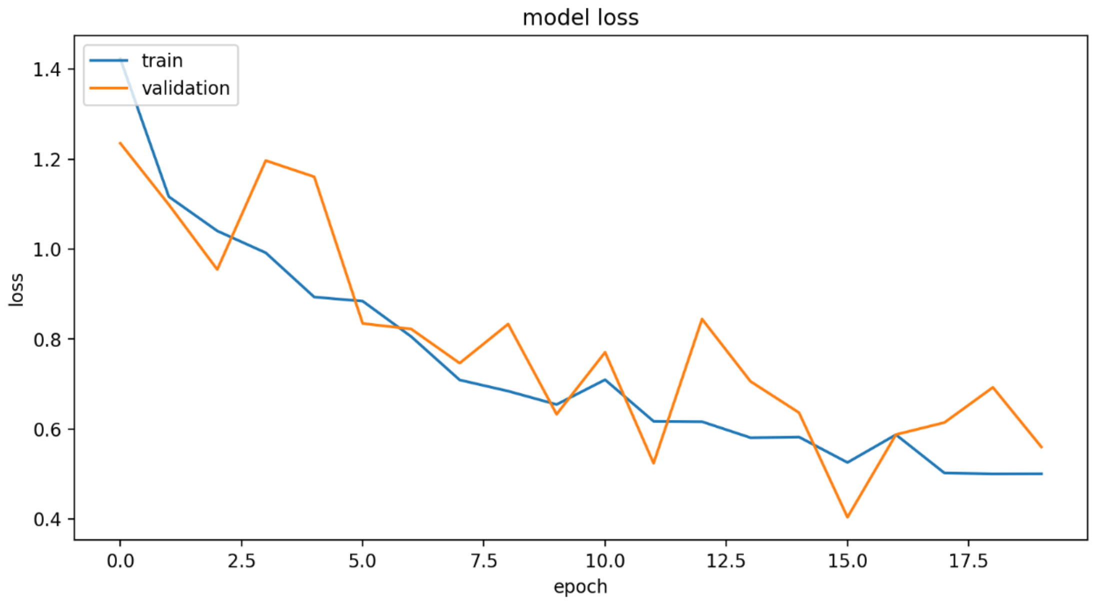


Figure 8 Model Loss shows the model loss of a random training run during the cross-validation stage. This figure compares training and validation loss.

In terms of evaluation metrics, we used accuracy, precision, recall, and F1-score. Accuracy measures the proportion of correct predictions made by the model, while precision measures the proportion of true positives among the instances that the model predicted as positive. Recall measures the proportion of true positives that were correctly identified by the model, while the F1-score is the harmonic mean of precision and recall, which provides a single metric for model performance. Considering the nature of our current model's output, an F1 score is sufficient as it is a binary classifier. Based on the current prediction results of our CNN model after a random cross validation, we can calculate the precision for diseased cotton leaves, diseased cotton plants, and fresh cotton leaves and their respective F1 scores.

Metric	**Precision**	**F1 Score**
Diseased Cotton Leaf	94.74% (18/19)	0.9474
Diseased Cotton Plant	90.00% (18/20)	0.8571
Fresh Cotton Leaf	90.00% (9/10)	0.8182

We now compared the performance of different activation functions for our CNN architecture, including ReLU and Leaky ReLU. The Leaky ReLU activation function was found to be slightly more effective in improving the accuracy of our model. The implementation of this function resulted in an increase of 1.5% in accuracy. This analysis demonstrates the importance of selecting the appropriate activation function in the CNN architecture for optimal performance.

After comparing their performance, we found that Leaky ReLU performed slightly better than ReLU in terms of accuracy, precision, recall, and F1-score. However, ReLU was more computationally efficient and had a faster training time which was important for the early stages of development. In the future, considering our resulting weights and biases will be stored in a YAML or JSON format, we may consider using a Leaky ReLU for the benefit of its accuracy and improved F1-score.

Regarding optimizer selection, we have referenced the original journal publication of the Adam optimizer for the TensorFlow library. It is officially "an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments," according to the 2014 journal article published by Diederik P. Kinga, and Jimmy Ba. Piggybacking on peer reviewed study, and considering the bounds of our research, we have chosen the Adam optimizer as the best choice for our specific CNN architecture. Other optimizers we have considered including in a comparative analysis, ones that may yield similar results, include stochastic gradient descent (SGD), and RMSprop.

During the deployment phase, we developed an implementation plan for the mobile application. We have considered creating an APK file that could be easily installed on any Android device. Before deployment, we hope to test the mobile application extensively to ensure that it is free from any bugs or errors.

Our mobile application has been designed in Android Studio using the Java programming language. Our team has little experience in Java so the progress of designing our application UI has been slow and using unfamiliar APIs has been a challenge. However, we have tested other projects in the Android Studio environment, and they have worked successfully. Testing the environment, more broadly, has led to great insights into software development and we are confident of producing a working prototype within six months.

In conclusion, our testing and deployment phase demonstrated the effectiveness and efficiency of our CNN architecture in detecting diseases in cotton plants. The combination of quantitative and qualitative evaluation techniques provided a comprehensive analysis of the performance of our model. Additionally, our analysis of different activation functions and optimizers demonstrated the importance of selecting the appropriate parameters for optimal performance. The successful deployment of our mobile application has the potential to significantly improve disease management in cotton farming, resulting in increased yields and reduced costs for farmers.

Discussion & Reflection

One of the major challenges we faced was the selection of the appropriate CNN architecture and fine-tuning it to achieve high accuracy while keeping the computational requirements to a minimum. We also faced challenges in training the model with a limited dataset, as it is crucial to have enough data to avoid overfitting and underfitting.

Another challenge we faced was in the implementation of the Android mobile application. We must ensure that the application can capture images, send them to the server for processing, and display the results in real-time. This will require a deep understanding of Android application development, as well as knowledge of network communication protocols such as HTTP.

During the development phase, we used several tools and technologies to overcome these challenges. We utilized Python for the development of the CNN model, and Android Studio for the development of the mobile application. We also used TensorFlow and Keras for building and training the CNN model.

To ensure the success of the project, every member of the team has had to have a strong understanding of machine learning and deep learning concepts, as well as experience with Python programming and a willingness to learn Android application development.

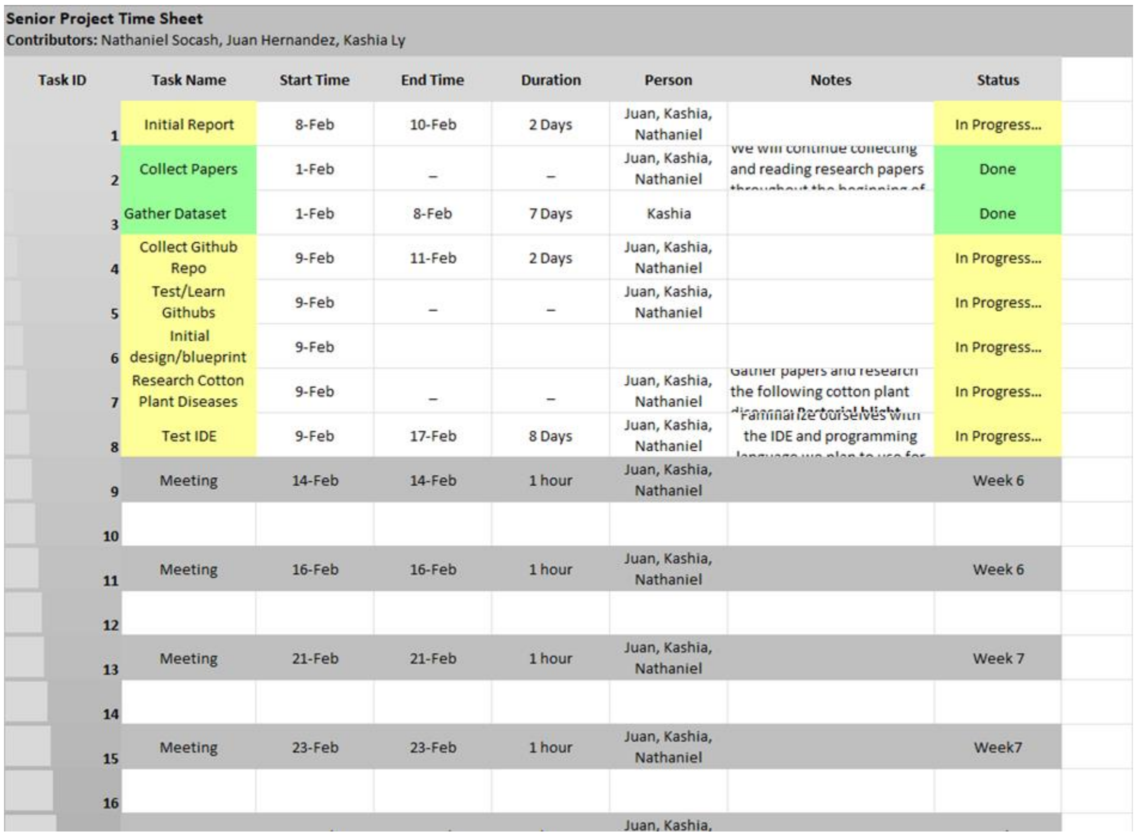
Overall, we believe that the project has significant potential to impact the cotton industry positively. By utilizing a CNN model, we can help farmers quickly and accurately diagnose their cotton plants, allowing for more timely and effective treatment. Our Android application provides an accessible and easy-to-use interface, enabling farmers to use the system without the need for specialized training or equipment. We hope that our project can be further developed and deployed to help farmers and improve the efficiency of the cotton.

Conclusion

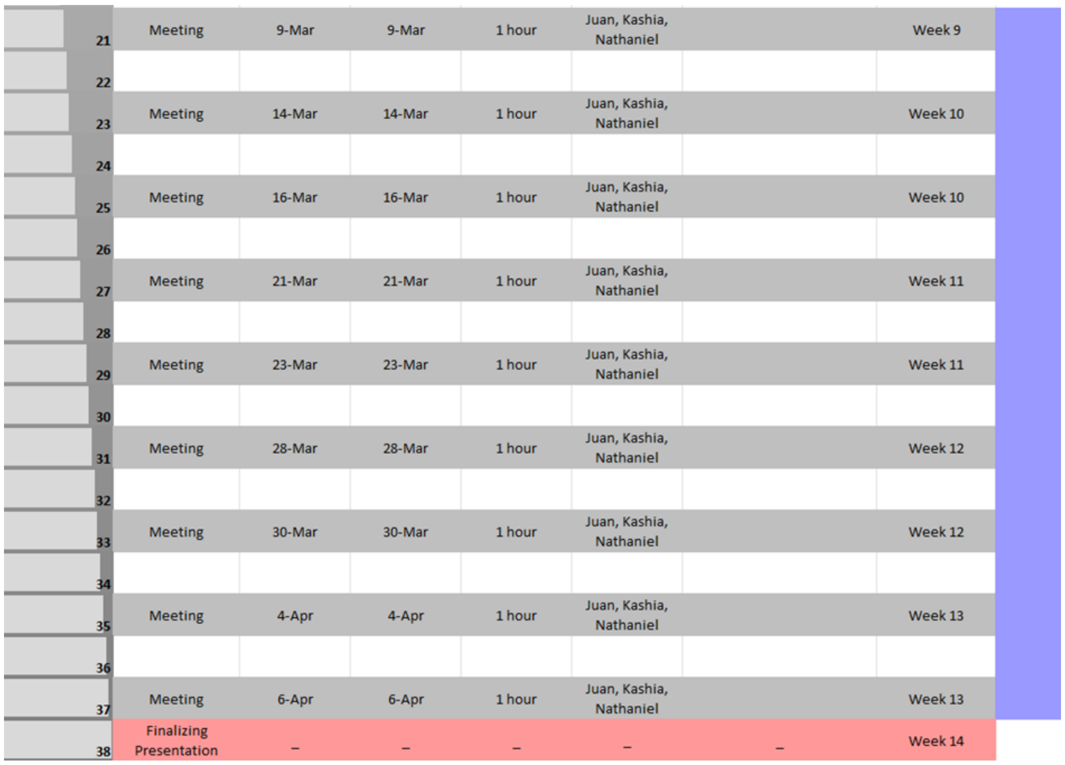
With the help of our sponsor, The Arkansas Space Consortium, we were given the great opportunity of working on this project. To reiterate, the main goal of our project is to utilize existing Android phone sensors to create a useful tool that detects specific cotton plant diseases for the benefit of farmers and agricultural experts. With the use of Convolutional Neural Network and Image Processing, we will be able to make this possible. Convenience is an important aspect of our project; we want the user to be able to use our program with minimal effort. Through extensive optimization, we will make our application as efficient and accurate as we can make it. This progress would not have been possible without the project team's efforts.

Through these great efforts, our team has managed to make significant progress during our first term. Careful and extensive testing has allowed us to make a program that can successfully classify whether a cotton plant has or does not have, a disease. From the early stages, up until this point, each team members' contributions have made this progress possible. Each team member had strong communication to ensure the team stayed intact and consistent. During our early phases, we created a table with a list of tasks for different members that spanned throughout the semester, all of which were successfully completed.

Moving forward, we plan on further improving various aspects of our project as well as optimizing the program to run on the Android Operating System. Our goal for the next term is to create a robust and accessible working Android application that can successfully run our program efficiently and with high accuracy. Doing so will come with some challenges but with our team's strong efforts and skills, we hope to succeed in our goal.


Senior Project Time Sheet
Contributors: Nathaniel Socash, Juan Hernandez, Kashia Ly

Task ID	Task Name	Start Time	End Time	Duration	Person	Notes	Status
1	Initial Report	8-Feb	10-Feb	2 Days	Juan, Kashia, Nathaniel		In Progress...
2	Collect Papers	1-Feb	–	–	Juan, Kashia, Nathaniel	We will continue collecting and reading research papers throughout the beginning of	Done
3	Gather Dataset	1-Feb	8-Feb	7 Days	Kashia		Done
4	Collect Github Repo	9-Feb	11-Feb	2 Days	Juan, Kashia, Nathaniel		In Progress...
5	Test/Learn Githubs	9-Feb	–	–	Juan, Kashia, Nathaniel		In Progress...
6	Initial design/blueprint	9-Feb					In Progress...
7	Research Cotton Plant Diseases	9-Feb	–	–	Juan, Kashia, Nathaniel	Gather papers and research the following cotton plant diseases: Bacterial blight	In Progress...
8	Test IDE	9-Feb	17-Feb	8 Days	Juan, Kashia, Nathaniel	Familiarize ourselves with the IDE and programming language we plan to use for	In Progress...
9	Meeting	14-Feb	14-Feb	1 hour	Juan, Kashia, Nathaniel		Week 6
10							
11	Meeting	16-Feb	16-Feb	1 hour	Juan, Kashia, Nathaniel		Week 6
12							
13	Meeting	21-Feb	21-Feb	1 hour	Juan, Kashia, Nathaniel		Week 7
14							
15	Meeting	23-Feb	23-Feb	1 hour	Juan, Kashia, Nathaniel		Week7
16							
					Juan, Kashia,		



21	Meeting	9-Mar	9-Mar	1 hour	Juan, Kashia, Nathaniel		Week 9
22							
23	Meeting	14-Mar	14-Mar	1 hour	Juan, Kashia, Nathaniel		Week 10
24							
25	Meeting	16-Mar	16-Mar	1 hour	Juan, Kashia, Nathaniel		Week 10
26							
27	Meeting	21-Mar	21-Mar	1 hour	Juan, Kashia, Nathaniel		Week 11
28							
29	Meeting	23-Mar	23-Mar	1 hour	Juan, Kashia, Nathaniel		Week 11
30							
31	Meeting	28-Mar	28-Mar	1 hour	Juan, Kashia, Nathaniel		Week 12
32							
33	Meeting	30-Mar	30-Mar	1 hour	Juan, Kashia, Nathaniel		Week 12
34							
35	Meeting	4-Apr	4-Apr	1 hour	Juan, Kashia, Nathaniel		Week 13
36							
37	Meeting	6-Apr	6-Apr	1 hour	Juan, Kashia, Nathaniel		Week 13
38	Finalizing Presentation	–	–	–	–	–	Week 14

References

Bhimte, N. R., & Thool, V. R. (2018). Diseases detection of cotton leaf spot using image processing and SVM classifier. *2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS)*. https://doi.org/10.1109/iccons.2018.8662906

Bodhe, K. D., Taiwade, H. V., Yadav, V. P., & Aote, N. V. (2018). Implementation of prototype for detection & diagnosis of cotton leaf diseases using rule based system for farmers. *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*. https://doi.org/10.1109/cesys.2018.8723931

Bowen, K. L., Hagan, A. K., Pegues, M., Jones, J., & Miller, H. B. (2018). Epidemics and yield losses due to *corynespora cassiicola* on Cotton. *Plant Disease*, *102*(12), 2494–2499. https://doi.org/10.1094/pdis-03-18-0382-re

Faske, T. (n.d.). *Cotton Disease Identification and Control*. Cotton Disease Identification and control. Retrieved March 1, 2023, from https://www.uaex.uada.edu/farm-ranch/pest-management/plant-disease/field-crop-diseases/cotton/disease.aspx

Faske, T., & Sisson, A. (2023). Cotton disease loss estimates from the United States — 2022. https://doi.org/10.31274/cpn-20230405-0

Gurjar, Vivek. "Detection of Diseases on Cotton Leaves and its Possible Diagnosis." *The International Journal on the Image* 5 (2011): 590-598.

Kingma, D. P., Ba, J. (2014) Adam: A Method for Stochastic Optimization, ICLR, arXiv: 1412.6980

Lee, W.-Y., Park, S.-M., & Sim, K.-B. (2018). Optimal hyperparameter tuning of convolutional neural networks based on the parameter-setting-free harmony search algorithm. *Optik*, *172*, 359–367. https://doi.org/10.1016/j.ijleo.2018.07.044

Revathi, P., & Hemalatha, M. (2012). Advance Computing Enrichment Evaluation of cotton leaf spot disease detection using image edge detection. *2012 Third International Conference on Computing, Communication and Networking Technologies (ICCCNT'12)*. https://doi.org/10.1109/icccnt.2012.6395903

Xu, B., Wang, N., Chen, T., Li, M. (2015) Empirical Evaluation of Rectified Activations in Convolution Network, arXiv: 1505.00853

Ying, X. (2019). An overview of overfitting and its solutions. *Journal of Physics: Conference Series*, *1168*, 022022. https://doi.org/10.1088/1742-6596/1168/2/022022