Plant Leaf Disease Detection

TECHNICAL REPORT

SUBMITTED BY

Horair Ahmad
AG # 2018-ag-8206
Mabtoor ul Shafiq
AG # 2018-ag-8208
Muhammad Waqas
AG # 2018-ag-8224

ADVISED BY

Miss Raheela Nasim

A TECHNICAL REPORT SUBMITTED IN PARTIAL FULFILLMENT OF REQUIREMENT FOR THE DEGREE OF

BACHELOR OF SCIENCE

IN

COMPUTER SCIENCE

DEPARTMENT OF COMPUTER SCIENCE

FACULTY OF SCIENCES

UNIVERSITY OF AGRICULTURE FAISALABAD

DECLARATION

We hereby declare that the contents of the report “Plant Leaf Disease Detection” are project of our own research and no part has been copied from any published source (except the references). We further declare that this work has not been submitted for award of any other diploma/degree. The university may take action if the information provided is found false at any stage. In case of any default the scholar will be proceeded against as per UAF policy.

Horair Ahmad,
Mabtoor ul Shafiq,
Muhammad Waqas

CERTIFICATE

To,
The Controller of Examinations,
University of Agriculture,
Faisalabad.

The supervisory committee certify that Horair Ahmad **AG: 2018-ag-8206,** Muhammad Waqas **AG: 2018-ag-8224,** Mabtoor ul shafiq **AG: 2018-ag-8208** has successfully completed his project in partial fulfillment of requirement for the degree of BSc. Computer Science under our guidance and supervision.

____

Mis Raheela Nasim
Supervisor

____

Horair Ahmad
Mabtoor ul Shafiq
Muhammad Waqas
Member

____

Dr. Muhammad Ahsan Latif
Incharge,
Department of Computer Science

ACKNOWLEDGEMENT

We thank all who in one way or another contributed in the completion of this report. First, we thank to ALLAH ALMIGHTY, most magnificent and most merciful, for all his blessings. Then we are so grateful to the Department of Computer Science for making it possible for us to study here. Our special and heartily thanks to my supervisor, *Miss Raheela Nasim* who encouraged and directed us. Her challenges brought this work towards a completion. It is with her supervision that this work came into existence. For any faults we take full responsibility. We are also deeply thankful to my informants. We want to acknowledge and appreciate their help and transparency during my research. We are also so thankful to my fellow students whose challenges and productive critics have provided new ideas to the work. Furthermore, we also thank our family who encouraged us and prayed for us throughout the time of our research. May the Almighty God richly bless all of you.

ABSTRACT

Plant leaf disease detection is an innovative technology that helps improve the quality and quantity of agricultural production in the country. Plant leaf disease has been one of the major threats to food security since long ago because it reduces the crop yield and compromises its quality. The existing method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms. At the same time, in some countries, farmers do not have proper facilities or even idea that they can contact to experts. Due to which consulting experts even cost high as well as time consuming too. In such conditions, the recent advances in computer vision made possible by deep learning has paved the way for camera assisted disease diagnosis for plant leaf. It described the innovative solution that provides efficient disease detection and deep learning with Transfer learning has achieved great success in the classification of various plant leaf diseases. A variety of neuron-wise and layer-wise visualization methods were applied and trained using a Transfer learning, with a publicly available plant disease given image dataset. So, it observed that deep learning model can predict to classify different plants and diseases of the plants. The proposed system is able to detect 39 different classes of healthy or disease plants with 98% accuracy.

Table of Contents

List of Figures

List of Tables

Chapter 1 - INTRODUCTION

1.1 Background:

Human society needs to increase food production by an estimated 70% by 2050 to feed an expected population size that is predicted to be over 9 billion people. Currently, infectious diseases reduce the potential yield by an average of 40% with many farmers in the developing world experiencing yield losses as high as 100%. The widespread distribution of smartphones among crop growers around the world with an expected 5 billion smartphones by 2020 offers the potential of turning the smartphone into a valuable tool for diverse communities growing food. One potential application is the development of mobile disease diagnostics through machine learning and crowdsourcing.

1.2 Description:

The Goal is to minimize the yield loss caused by different diseases of plants. Objective is to create a mobile application to identify and detect diseases accurately and fastly. The existing method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms. At the same time, in some countries, farmers do not have proper facilities or even idea that they can contact to experts. Due to which consulting experts even cost high as well as time consuming too. In such conditions, the suggested technique proves to be beneficial in monitoring large fields of crops. Automatic detection of the diseases by just seeing the symptoms on the plant leaves makes it easier as well as cheaper. For this purpose, we will use deep learning model to classify different plants and diseases of the plants.

1.3 Problem Statement:

Plant leaf disease has been one of the major threats to food security since long ago because it reduces the crop yield and compromises its quality. The existing method for plant disease detection is simply naked eye observation by experts through which identification and detection of plant diseases is done. For doing so, a large team of experts as well as continuous monitoring of plant is required, which costs very high when we do with large farms. At the same time, in some countries, farmers do not have proper facilities or even idea that they can contact to experts. Due to which consulting experts even cost high as well as time consuming too. For this purpose, we will use deep learning model to classify different plants and diseases of the plants.

1.4 Scope:

First of all, data(images) of healthy and diseased leaves should be collected. Preprocessing of the data(images) which includes annotating and labelling of the data must be done. A model should be trained/developed on the training data and must be evaluated on test data. A mobile application should be developed which must be able to identify and detect diseased leaves of different plants.

1.5 Objectives:

- Goal: To build a mobile application to identify and detect the diseases of Plants.
- Objective 1: Data must be collected first.
- Objective 2: Data should be labelled and annotated before we pass it to model.
- Objective 3: Deep Learning model must be created.
- Objective 4: Model be evaluated using the test set of the data.
- Objective 5: Our trained model must be used in the mobile application.

1.6 Feasibility:

1.6.1 Technical Feasibility

For our project, we have made models using TensorFlow an end to end open source library for deep learning and artificial intelligence on Google colab. Google colab allows anybody to write and execute arbitrary python code through the project and is especially well suited to machine learning and deep learning. It also provides free of cost access to the GPU which makes the training of any deep learning model much faster and efficient.

1.6.2 Schedule Feasibility

The whole application divided into three modules.

1. Create and build a model
2. Create application user interface
3. Integrate our model in application

By using these techniques, we complete our project in required time.

1.6.3 Resource Feasibility

GPU(Graphic processing unit) is required for creating and building deep learning model. For this purpose, google colab was used. For creating application in android studio SSD(secondary storage device) is required.

1.7 Requirements:

1.7.1 Functional Requirements

FR01: Take/select a picture

FR01-01	System should request to allow the camera.
FR01-02	System should check the alignment of the image.
FR01-03	System should request the user to allow the storage media.

FR02: Evaluate the Picture

FR02-01	System should send the captured picture to the model for prediction
FR02-02	System should evaluate the picture either it is a healthy image or diseased.

FR03: Display Output

FR03-01	System should display the category of the Plant & disease.
FR03-02	System should display the causes of the disease.

1.7.2 Non- Functional Requirements

NFR01: System shall remain available 24/7 to its users.
NFR02: System should take different image formats (jpeg, png)
NFR03: System shall process one image at a time.
NFR04: System should display error if the given image does not belong to the specified category of plants diseases.

1.7.3 Hardware Requirements

Smartphone
RAM: 1gb or above

1.7.4 Software Requirements

Android version: 4.0 or above
Internet Connectivity

1.8 Stakeholders:

The stakeholder for our project that will be interested for our end product are:

1. Farmers
2. Students
3. Researcher

Chapter 2 – MATERIALS & METHODS

2.1 Process Model:

The V model (Verification and Validation model) is an extension of the waterfall model. All the requirements are gathered at the start and cannot be changed. You have a corresponding testing activity for each stage. For every phase in the development cycle, there is an **associated testing phase.** The V model is highly disciplined, easy to understand, and makes project management easier. But it isn't good for complex projects or projects that have unclear or changing requirements. This makes the V model a good choice for software where downtimes and failures are unacceptable.
This process model was used because in our project requirements are completely specified in the beginning and there is no need to change them. As this model is sequential so it gives the option to complete one phase before move to the next phase which is an ideal case to our project. As stated above, it gives us the option to test each phase before move to next the phase which gives the flexibility.

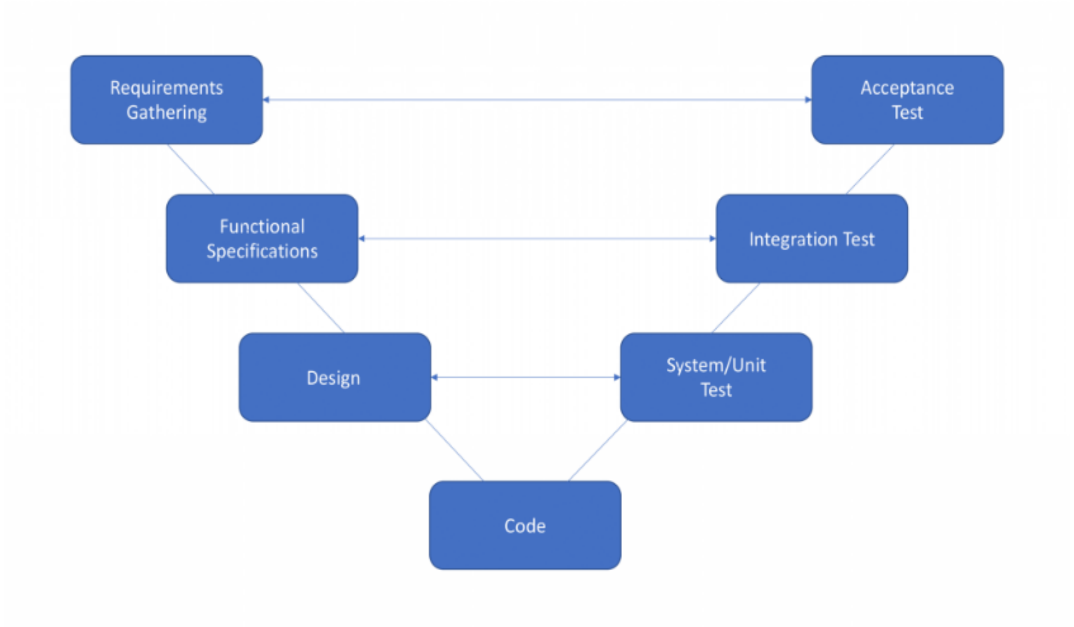


Figure 2.1 V-model Activities

2.2 Tools & Technologies

Google colab: Platform to create deep learning models.
Vs code: IDE to build mobile application.

TensorFlow: TensorFlow is a free and open-source software library for machine learning and artificial intelligence.
NumPy: NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices.
Pandas: Pandas is a software library written for the Python programming language for data manipulation and analysis.
Matplotlib: Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy.
Keras: Keras is an open-source software library that provides a Python interface for artificial neural networks.
Java : Programming language to create an android app.

2.3 Design:

2.3.1 Use Case Diagrams:

Use Case Diagram:

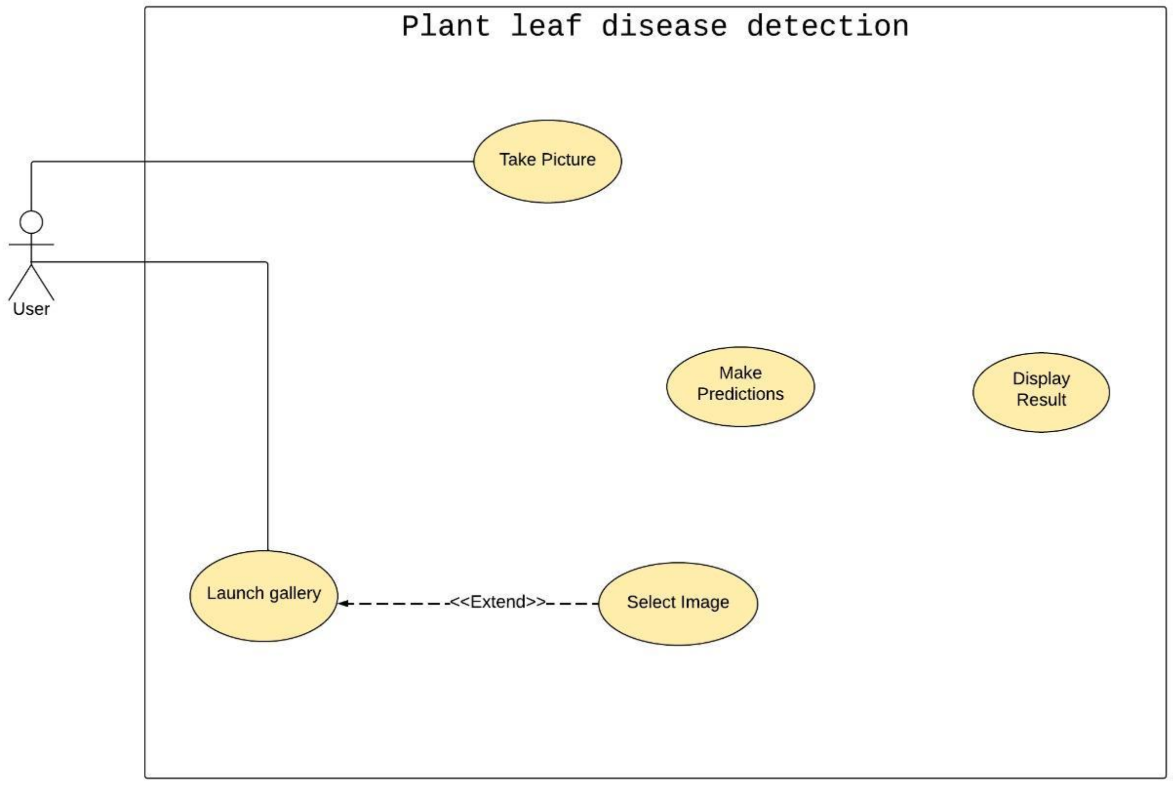


Figure 2.2 Use Case Diagram

2.3.2 Usage Scenario

Use Case Title	Take picture	
Use Case Id	1	
Requirement Id	1	
Description: This use case is about taking image through device camera.		
Pre Conditions: 1. User must request for camera access.		
Task Sequence		**Exceptions**
1. User should click on take picture button.		
2. System should request user to grant permission for accessing camera.		Camera access denied.
3. User clicks on okay button to confirm image.		
Post Conditions: a. System should pass the taken image to model for prediction.		
Unresolved issues:		
Authority: User		
Modification history: 1.0 **Author: <Plant leaf disease detection>** **Description:**		

Table 2. 1: Take picture

Use Case Title	Launch gallery	
Use Case Id	2	
Requirement Id	2	
Description: This use case is about selecting image through device gallery.		
Pre Conditions: 1. User must request for storage media access.		
Task Sequence		**Exceptions**
1. User should click launch gallery button.		
2. System should request user to grant permission for accessing storage media.		Gallery access denied.
3. User tap on desired image.		
Post Conditions: a. System should pass the selected image to model for prediction.		
Unresolved issues:		
Authority: User		
Modification history: 1.0 **Author: <Plant leaf disease detection>** **Description:**		

Table 2. 2: Launch Gallery

Use Case Title	Make Predictions	
Use Case Id	3	
Requirement Id	3	
Description: This use case is about making predictions on selected image.		
Pre Conditions: 1. An image must be taken or selected.		
Task Sequence		**Exceptions**
1. A system should preprocess the selected image.		
2. System should predict the class of selected image.		
Post Conditions: a. System should display the predicted class to the user.		
Unresolved issues:		
Authority: User		
Modification history: 1.0 **Author: <Plant leaf disease detection>** **Description:**		

Table 2. 3: Make Predictions

Use Case Title	Display result	
Use Case Id	4	
Requirement Id	4	
Description: This use case is about displaying result.		
Pre Conditions: 1. A prediction must be made on selected image.		
Task Sequence		**Exceptions**
1. A system should display the predicted class and selected image to the user.		
2. System should display the causes and remedies to the user.		
Post Conditions: -		
Unresolved issues:		
Authority: User		
Modification history: 1.0 **Author: <Plant leaf disease detection>** **Description:**		

Table 2. 4: Display Results

2.3.3 Sequence Diagram:

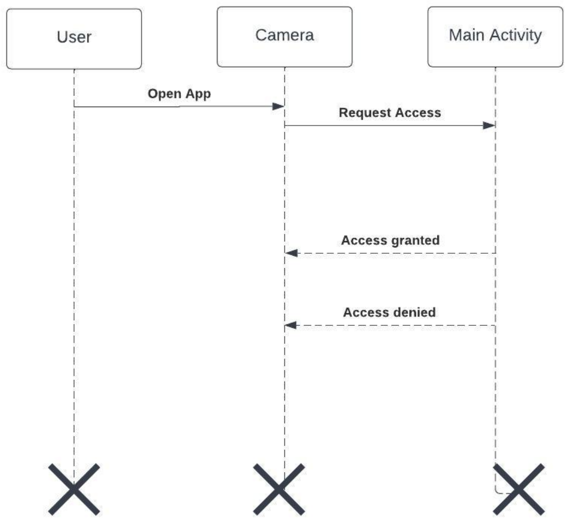


Figure 2.3.1 Sequence Diagram

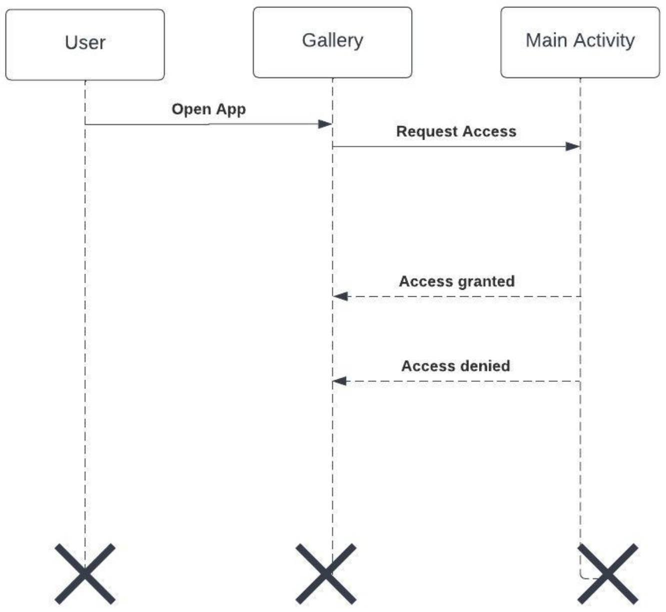


Figure 2.3.2 Sequence Diagram

2.3.4 Class Diagram:

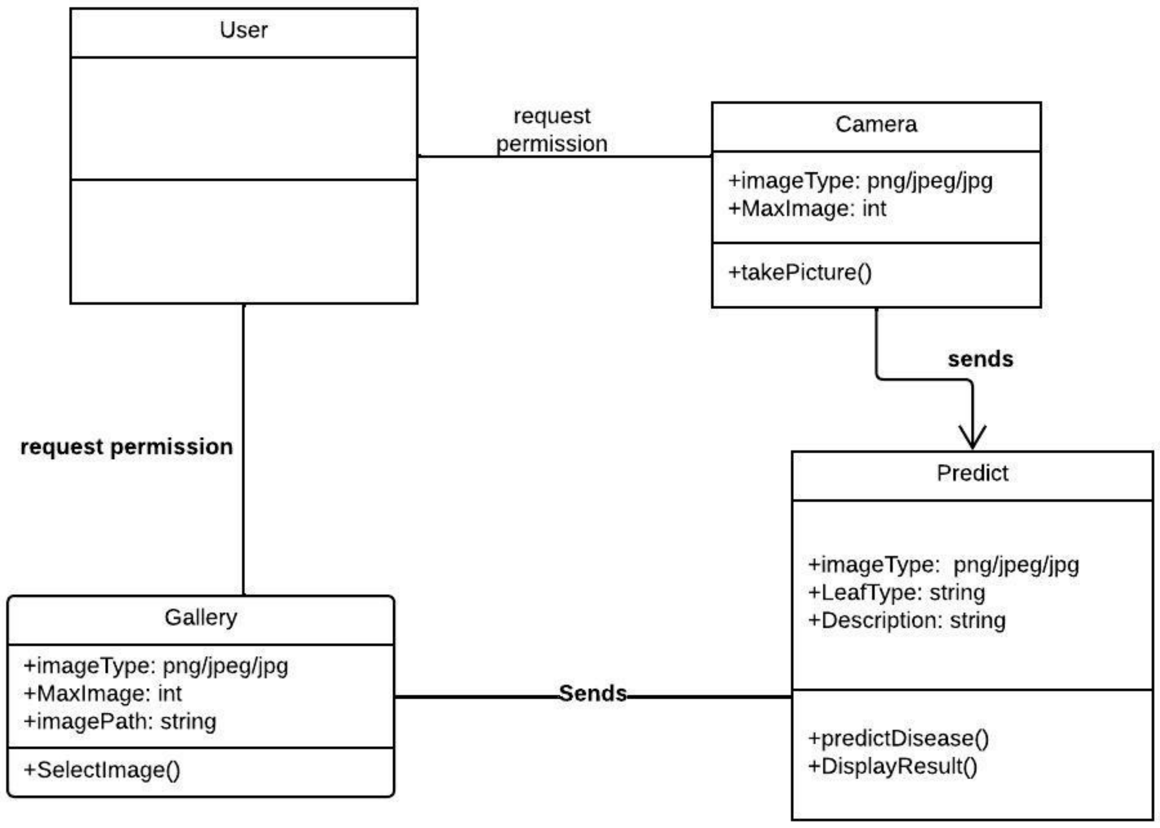


Figure 2.4 Class Diagram

2.3.5 Data Flow Diagram:

Context Diagram:

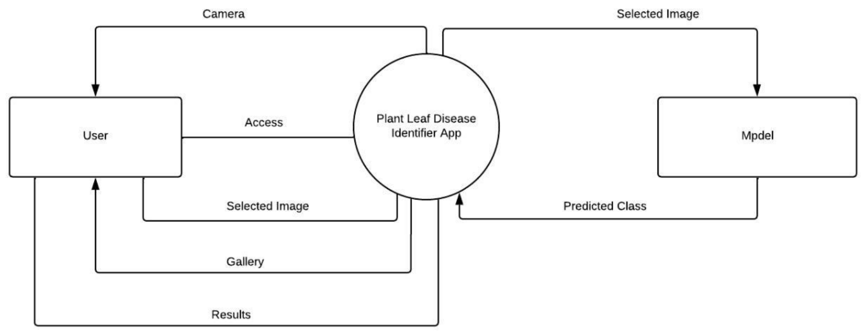


Figure 2.5 Context Diagram

Level 1:

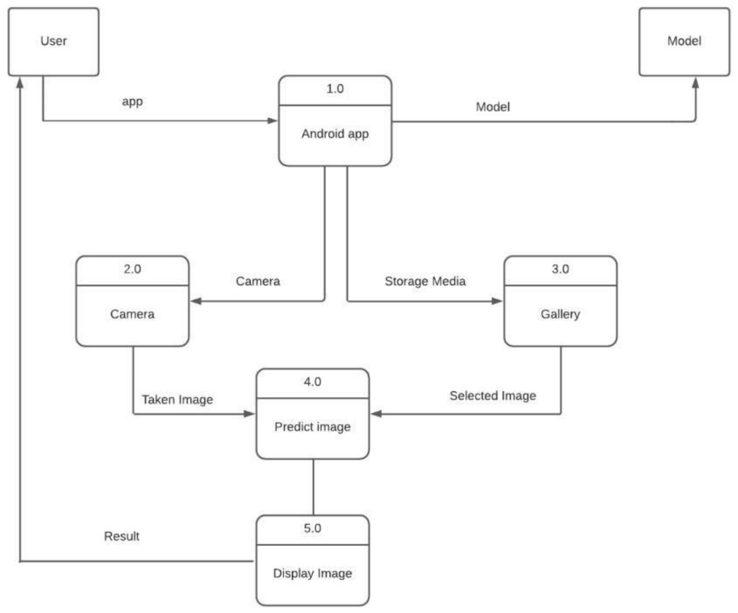


Figure 2.6 Level 1 DFD

2.3.6 Architecture:

3-Tier: (or N- Tier/multitier, chose whatever best suits your project nature)

Presentation Tier- The presentation layer contains the front-end of app which is made in java language. Java language is popular language for making mobile applications.

Application Tier- The application tier contains the deep learning model which we have trained using TensorFlow and keras API.

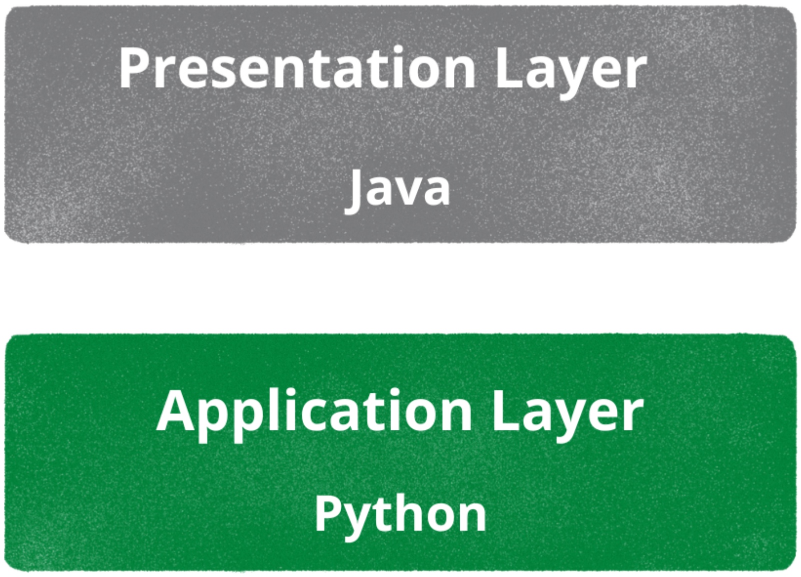


Figure 2.10 Application Architecture

Chapter 3 - RESULTS & DISCUSSION

3. Testing:

3.1 Test Cases:

Test Case: Open camera

Test Case ID:	**TC-1**
Test Case Title:	To verify application take image from camera
Test Case Priority:	High
Requirement:	Camera
Test Description:	This test will verify application take image from camera.
Test Date:	05/20/2022
Pre-Conditions:	1. Open the application. 2. Click Take image button. 3. Request camera access.
Dependencies:	
Test Steps:	1. Open app. 2. Click on take image button. 3. Grant access to camera. 4. Take image.
Test Data	Images
Expected Results:	When user clicked on take image button a pop up appears to ask for camera access. If the camera access is granted user must be able to click an image
Actual Results:	As above
Post Conditions:	The taken image should be pass to model for prediction.
Status: (Pass/Fail)	Pass

Table 3. 1: Open camera Test Case

3.2 Test Case: Launch gallery:

Test Case ID:	**TC-2**
Test Case Title:	To verify application select image from gallery
Test Case Priority:	High
Requirement:	Gallery
Test Description:	This test will verify application select image from gallery.
Test Date:	05/20/2022
Pre-Conditions:	1. Open the application. 2. Click Take image button. 3. Request storage media access.
Dependencies:	
Test Steps:	1. Open app. 2. Click on launch gallery button. 3. Grant access to storage media. 4. Select image.

Test Data	Images
Expected Results:	When user clicked on take launch gallery button a pop up appears to ask for storage media access. If the storage media access is granted user must be able to select an image
Actual Results:	As above
Post Conditions:	The select image should be pass to model for prediction.
Status: (Pass/Fail)	Pass
Other Comments:	None

Table 3. 2: Launch gallery Test Case

3.3 Test Case: Model:

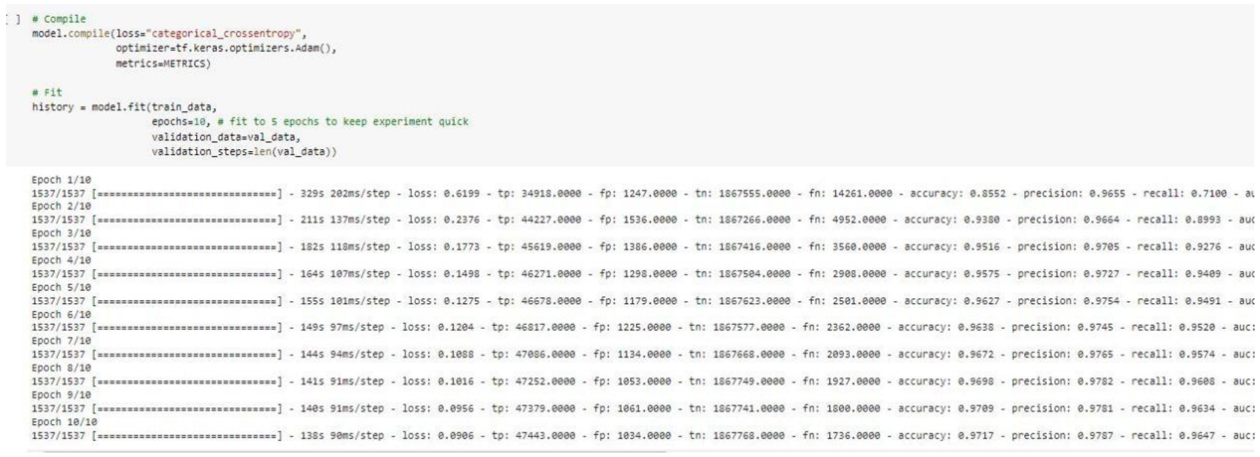
Test Case ID:	**TC-3**
Test Case Title:	To verify model is predicting accurately on selected image.
Test Case Priority:	High
Requirement:	Image
Test Description:	This test will verify model is predicting accurately.
Test Date:	05/20/2022
Pre-Conditions:	1. Image must be taken/selected. 2. Image should be passed to the model.
Dependencies:	Camera or gallery.
Test Steps:	1. Image is preprocessed. 2. Model perform prediction on preprocess image. 3. Model returns predicted class of an image.
Test Data	Images
Expected Results:	The class predicted by the model matches with the actual class.
Actual Results:	As above
Post Conditions:	The System display the result.
Status: (Pass/Fail)	Pass
Other Comments:	None

Table 3. 3: Model Test Case

3.2 Conclusion:

There are many developed methods in detecting and classifying plant diseases using diseased leaves of plants. However, there is still no efficient and effective commercial solution that can be used to identify the diseases. In our work, we used 6 different DL models (InceptionV3, Resnet50, MobileNetV2, EfficientNetB0, DenseNet, EfficientNetV2b3) for the detection of plant diseases using healthy- and diseased-leaf images of plants. To train and test the model, we used the Kaggle platform to get the standard PlantVillage dataset with ~61,000 images. This dataset consists of 39 different classes of different healthy- and diseased-leaf images of 14 different species. After splitting the dataset into 80–20 (80% of whole data for training, 20% whole images for testing),

By applying Feature Extraction, We've got 97% of accuracy and 96% of validation accuracy

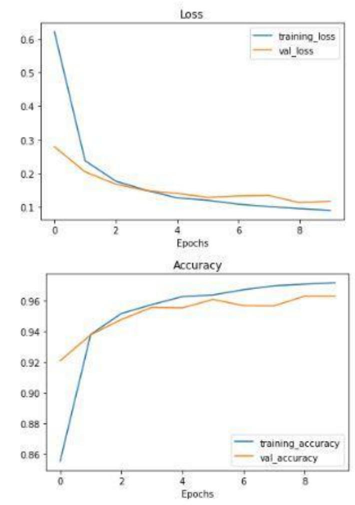


```
[ ] # Compile
    model.compile(loss="categorical_crossentropy",
                  optimizer=tf.keras.optimizers.Adam(),
                  metrics=METRICS)

    # Fit
    history = model.fit(train_data,
                        epochs=10, # fit to 5 epochs to keep experiment quick
                        validation_data=val_data,
                        validation_steps=len(val_data))
```

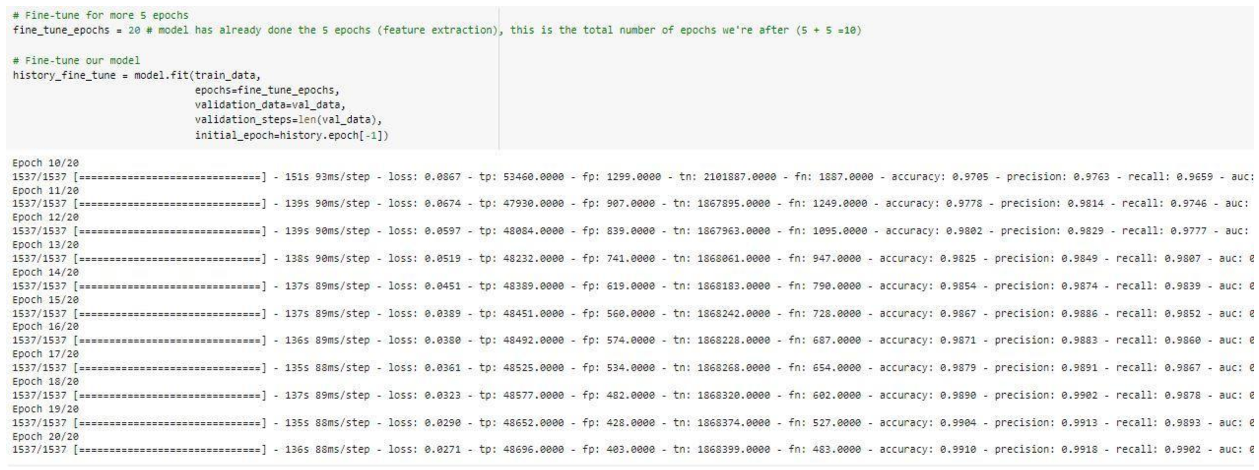
```
Epoch 1/10
1537/1537 [==============================] - 329s 202ms/step - loss: 0.6199 - tp: 34918.0000 - fp: 1247.0000 - tn: 1867555.0000 - fn: 14261.0000 - accuracy: 0.8552 - precision: 0.9655 - recall: 0.7100 - au
Epoch 2/10
1537/1537 [==============================] - 211s 137ms/step - loss: 0.2376 - tp: 44227.0000 - fp: 1536.0000 - tn: 1867266.0000 - fn: 4952.0000 - accuracy: 0.9380 - precision: 0.9664 - recall: 0.8993 - auc
Epoch 3/10
1537/1537 [==============================] - 182s 118ms/step - loss: 0.1773 - tp: 45619.0000 - fp: 1386.0000 - tn: 1867416.0000 - fn: 3560.0000 - accuracy: 0.9516 - precision: 0.9705 - recall: 0.9276 - auc
Epoch 4/10
1537/1537 [==============================] - 164s 107ms/step - loss: 0.1498 - tp: 46271.0000 - fp: 1298.0000 - tn: 1867504.0000 - fn: 2908.0000 - accuracy: 0.9575 - precision: 0.9727 - recall: 0.9409 - auc
Epoch 5/10
1537/1537 [==============================] - 155s 101ms/step - loss: 0.1275 - tp: 46678.0000 - fp: 1179.0000 - tn: 1867623.0000 - fn: 2501.0000 - accuracy: 0.9627 - precision: 0.9754 - recall: 0.9491 - auc
Epoch 6/10
1537/1537 [==============================] - 149s 97ms/step - loss: 0.1204 - tp: 46817.0000 - fp: 1225.0000 - tn: 1867577.0000 - fn: 2362.0000 - accuracy: 0.9638 - precision: 0.9745 - recall: 0.9520 - auc:
Epoch 7/10
1537/1537 [==============================] - 144s 94ms/step - loss: 0.1088 - tp: 47086.0000 - fp: 1134.0000 - tn: 1867668.0000 - fn: 2093.0000 - accuracy: 0.9672 - precision: 0.9765 - recall: 0.9574 - auc:
Epoch 8/10
1537/1537 [==============================] - 141s 91ms/step - loss: 0.1016 - tp: 47252.0000 - fp: 1053.0000 - tn: 1867749.0000 - fn: 1927.0000 - accuracy: 0.9698 - precision: 0.9782 - recall: 0.9608 - auc:
Epoch 9/10
1537/1537 [==============================] - 140s 91ms/step - loss: 0.0956 - tp: 47379.0000 - fp: 1061.0000 - tn: 1867741.0000 - fn: 1800.0000 - accuracy: 0.9709 - precision: 0.9781 - recall: 0.9634 - auc:
Epoch 10/10
1537/1537 [==============================] - 138s 90ms/step - loss: 0.0906 - tp: 47443.0000 - fp: 1034.0000 - tn: 1867768.0000 - fn: 1736.0000 - accuracy: 0.9717 - precision: 0.9787 - recall: 0.9647 - auc:
```

Plot loss curves of feature extraction on plant disease data:

```
[ ]  plot_loss_curves(history)
```



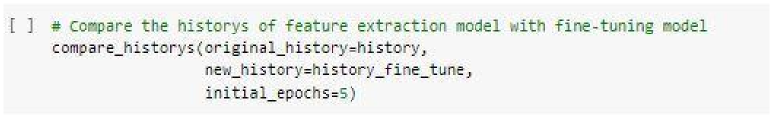
After feature Extraction, We tried to unfreeze last 5 layers and apply fine tuning. After applying fine tuning, We’ve got 99% accuracy and 99% validation accuracy.



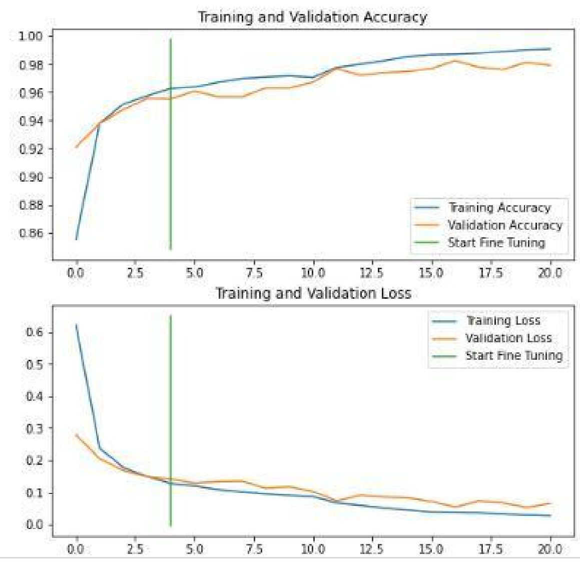
```
# Fine-tune for more 5 epochs
fine_tune_epochs = 20 # model has already done the 5 epochs (feature extraction), this is the total number of epochs we're after (5 + 5 =10)

# Fine-tune our model
history_fine_tune = model.fit(train_data,
                              epochs=fine_tune_epochs,
                              validation_data=val_data,
                              validation_steps=len(val_data),
                              initial_epoch=history.epoch[-1])
```

```
Epoch 10/20
1537/1537 [==============================] - 151s 93ms/step - loss: 0.0867 - tp: 53460.0000 - fp: 1299.0000 - tn: 2101887.0000 - fn: 1887.0000 - accuracy: 0.9705 - precision: 0.9763 - recall: 0.9659 - auc:
Epoch 11/20
1537/1537 [==============================] - 139s 90ms/step - loss: 0.0674 - tp: 47930.0000 - fp: 907.0000 - tn: 1867895.0000 - fn: 1249.0000 - accuracy: 0.9778 - precision: 0.9814 - recall: 0.9746 - auc:
Epoch 12/20
1537/1537 [==============================] - 139s 90ms/step - loss: 0.0597 - tp: 48084.0000 - fp: 839.0000 - tn: 1867963.0000 - fn: 1095.0000 - accuracy: 0.9802 - precision: 0.9829 - recall: 0.9777 - auc:
Epoch 13/20
1537/1537 [==============================] - 138s 90ms/step - loss: 0.0519 - tp: 48232.0000 - fp: 741.0000 - tn: 1868061.0000 - fn: 947.0000 - accuracy: 0.9825 - precision: 0.9849 - recall: 0.9807 - auc:
Epoch 14/20
1537/1537 [==============================] - 137s 89ms/step - loss: 0.0451 - tp: 48389.0000 - fp: 619.0000 - tn: 1868183.0000 - fn: 790.0000 - accuracy: 0.9854 - precision: 0.9874 - recall: 0.9839 - auc:
Epoch 15/20
1537/1537 [==============================] - 137s 89ms/step - loss: 0.0389 - tp: 48451.0000 - fp: 560.0000 - tn: 1868242.0000 - fn: 728.0000 - accuracy: 0.9867 - precision: 0.9886 - recall: 0.9852 - auc:
Epoch 16/20
1537/1537 [==============================] - 136s 89ms/step - loss: 0.0380 - tp: 48492.0000 - fp: 574.0000 - tn: 1868228.0000 - fn: 687.0000 - accuracy: 0.9871 - precision: 0.9883 - recall: 0.9860 - auc:
Epoch 17/20
1537/1537 [==============================] - 135s 88ms/step - loss: 0.0361 - tp: 48525.0000 - fp: 534.0000 - tn: 1868268.0000 - fn: 654.0000 - accuracy: 0.9879 - precision: 0.9891 - recall: 0.9867 - auc:
Epoch 18/20
1537/1537 [==============================] - 137s 89ms/step - loss: 0.0323 - tp: 48577.0000 - fp: 482.0000 - tn: 1868320.0000 - fn: 602.0000 - accuracy: 0.9890 - precision: 0.9902 - recall: 0.9878 - auc:
Epoch 19/20
1537/1537 [==============================] - 135s 88ms/step - loss: 0.0290 - tp: 48652.0000 - fp: 428.0000 - tn: 1868374.0000 - fn: 527.0000 - accuracy: 0.9904 - precision: 0.9913 - recall: 0.9893 - auc:
Epoch 20/20
1537/1537 [==============================] - 136s 88ms/step - loss: 0.0271 - tp: 48696.0000 - fp: 403.0000 - tn: 1868399.0000 - fn: 483.0000 - accuracy: 0.9910 - precision: 0.9918 - recall: 0.9902 - auc:
```

Now we compare both feature extraction and fine tuning results:

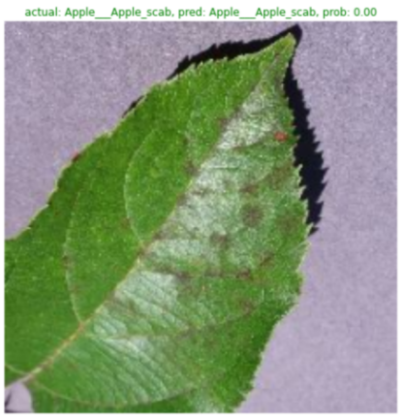


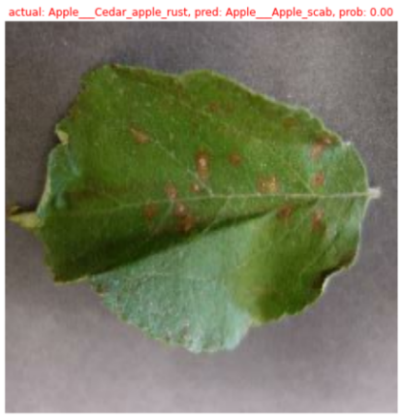
```
[ ]  # Compare the historys of feature extraction model with fine-tuning model
     compare_historys(original_history=history,
                      new_history=history_fine_tune,
                      initial_epochs=5)
```

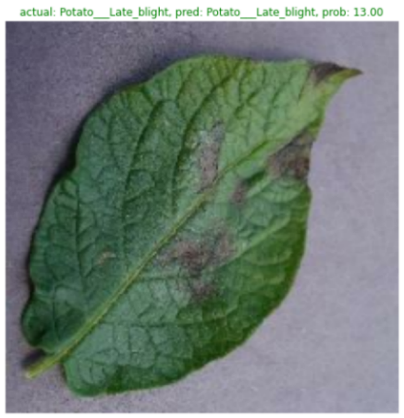


we achieved the best accuracy rate of nearly 99% in the EfficientNetB0 model. On average, less time was required to train the images in the MobileNetV2 and EfficientNetB0 architectures, and it took 5and 564 s/epoch, respectively. In comparison with other deep-learning approaches, the implemented deep-learning model has better predictive ability in terms of both accuracy and loss.

Here, Model makes prediction on series of images

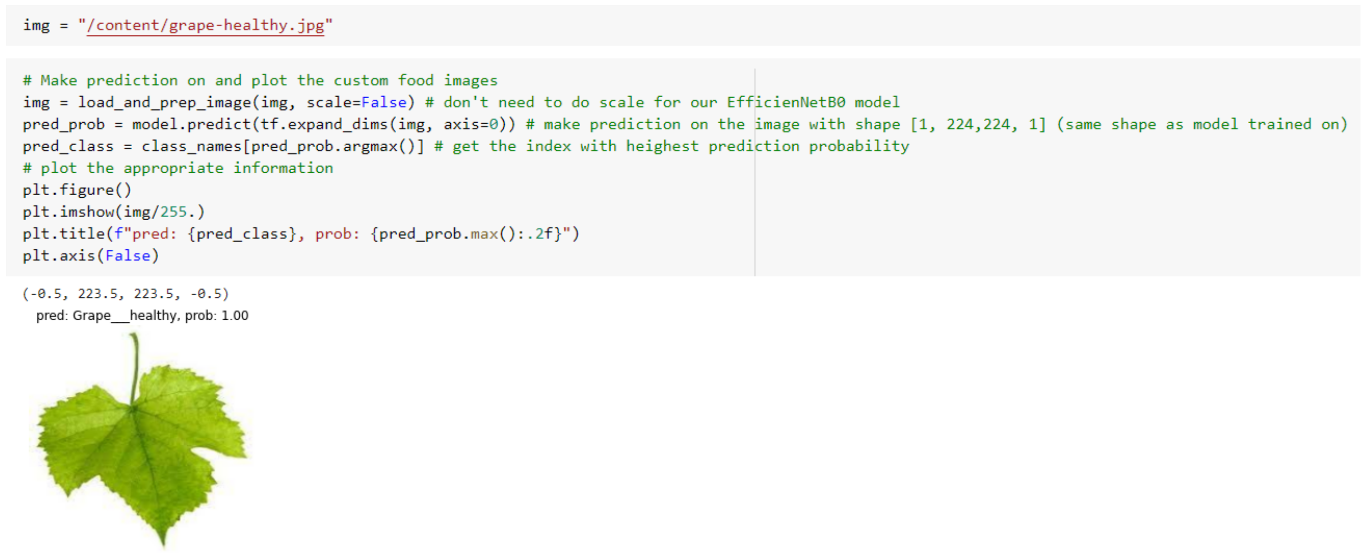

actual: Apple__Apple_scab, pred: Apple__Apple_scab, prob: 0.00


actual: Apple__Cedar_apple_rust, pred: Apple__Apple_scab, prob: 0.00


actual: Potato__Late_blight, pred: Potato__Late_blight, prob: 13.00

The required time to train the model was much less than that of other machine-learning approaches. Moreover, the EfficientNetB0 architecture is an optimized deep convolutional neural network that limits the parameter number and operations as much as possible, and can easily run on mobile devices.

Here, model makes prediction on custom images:

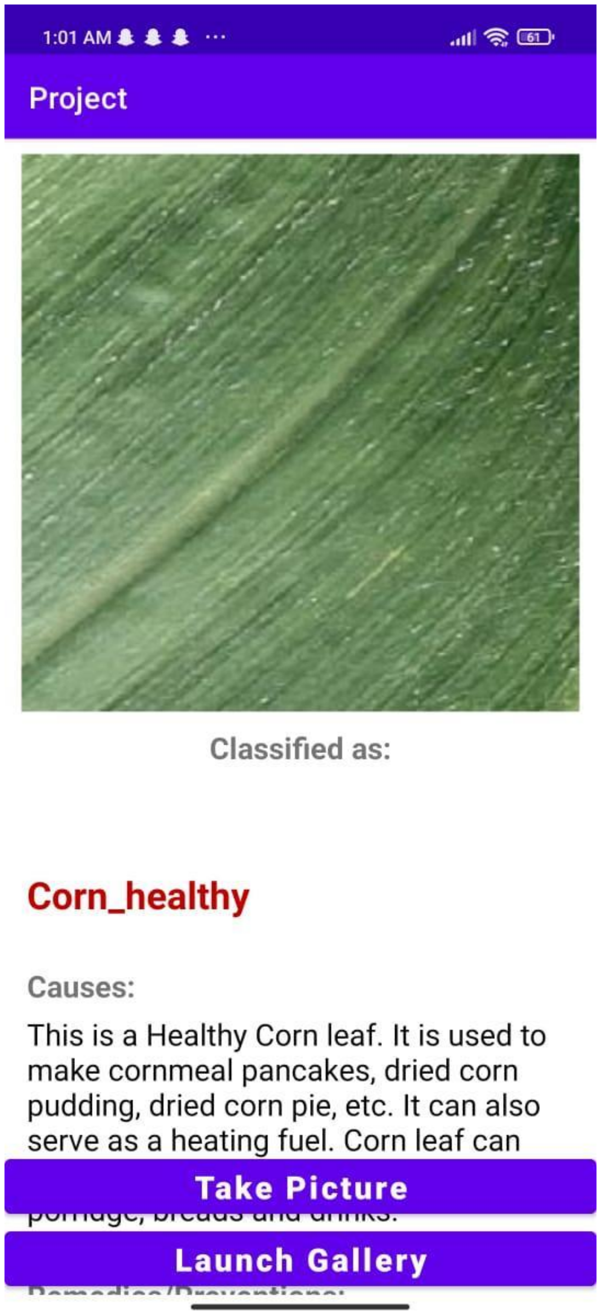


```
img = "/content/grape-healthy.jpg"
```

```
# Make prediction on and plot the custom food images
img = load_and_prep_image(img, scale=False) # don't need to do scale for our EfficienNetB0 model
pred_prob = model.predict(tf.expand_dims(img, axis=0)) # make prediction on the image with shape [1, 224,224, 1] (same shape as model trained on)
pred_class = class_names[pred_prob.argmax()] # get the index with heighest prediction probability
# plot the appropriate information
plt.figure()
plt.imshow(img/255.)
plt.title(f"pred: {pred_class}, prob: {pred_prob.max():.2f}")
plt.axis(False)
```

```
(-0.5, 223.5, 223.5, -0.5)
```

After that we create a android application which is user friendly. Which take image through camera and gallery. The image is predicted and shows the result whether it is healthy leaf or disease leaf.



Classified as:

Corn_healthy

Causes:

This is a Healthy Corn leaf. It is used to make cornmeal pancakes, dried corn pudding, dried corn pie, etc. It can also serve as a heating fuel. Corn leaf can

porridge, breads and drinks.

Chapter 4 - USER MANUAL

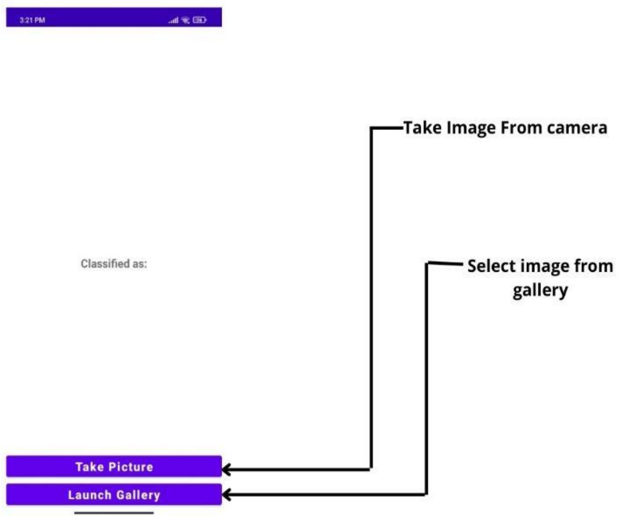


Figure 4.1 Take or select picture

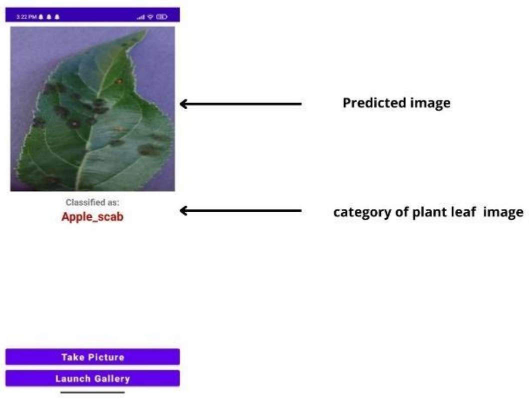
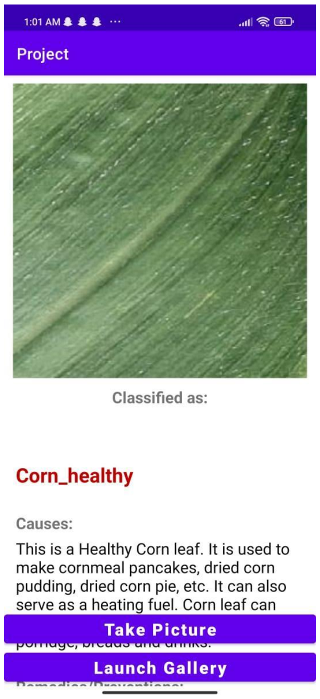


Figure 4.2 Predicted image



Corn_healthy

Causes:

This is a Healthy Corn leaf. It is used to make cornmeal pancakes, dried corn pudding, dried corn pie, etc. It can also serve as a heating fuel. Corn leaf can

porridge, breads and drinks.

Follow UAF reference style to cite book, research paper (journal as well as conference paper) and website references. At least 3 to 4 reference must be present in Chapter 1 and 2.

Formatting instructions:

Following formatting is already applied on the document. However, it is explicitly mentioned below:

- Minimum length of the Report: 30 pages
- Font style : Times New Roman
- Paragraph font size: 12pt
- Main Heading Size: 14pt + Bold (before and after spacing 12pt)
- Sub Heading Size: 13pt + Bold (before and after spacing 8pt)
- Sub sub heading size 12pt + bold (before and after spacing 6pt)
- Paragraph Alignment: Justified
- Picture/Chart Alignment: Center
- Picture/chart/table heading font: Times New Roman
- Picture/chart/table heading font size: 10pt, Italic, center alignment
- Picture caption goes under the picture without any extra line and line space
- Table caption goes above the table without any extra line and line space
- Table heading: Times New Roman, 10pt, Bold
- Table text: Times New Roman, 10pt
- Line Spacing: 1.15
- Left/Right/Top/Bottom Margins: 1 inch
- Table of Contents, List of Tables, List of Figures Heading: Times New Roman. 14pt, Bold
- Table of Contents, List of Tables, List of Figures: Times New Roman. 12pt
- For Table of Contents, List of Tables & List of Figures use Roman number as page number format in footer (center align)
- For Introduction onwards, use integer number as page number format in footer (center align)

For Submission:

Three hard binded copies of technical report are required with memory card (attached with each copy) containing technical report and project (exe file + all source code).