2020-12

Development of an early detection tool for banana diseases: A case of Mbeya and Arusha region

Sanga, Sophia

NM-AIST

http://doi.org/10.58694/20.500.12479/1346

DEVELOPMENT OF AN EARLY DETECTION TOOL FOR BANANA DISEASES: A CASE OF MBEYA AND ARUSHA REGION

Sophia Leonard Sanga

A Dissertation Submitted in Partial Fulfillment of the Requirements for the Degree of Master's in Information and Communication Science and Engineering of the Nelson Mandela African Institution of Science and Technology

Arusha, Tanzania

December, 2020

ABSTRACT

In Tanzania, smallholder farmers are mainly involved in farming activities which contribute significantly to food security and nutrition. Kagera, Mbeya and Arusha regions lead in high banana production, however, diseases and pests are threats to the yields. Early detection and identification of banana diseases is still a challenge for smallholder farmers and extension officers due to lack of the necessary tools such as sensors and mobile applications. In this research, an early detection tool for banana fungal diseases was developed. This research presents deep learning models trained for the detection of banana diseases of Fusarium wilt and Black sigatoka and deployed on a smartphone for the early detection of the diseases in real time. The five models selected and trained include VGG16, Resnet18, Resnet50, Resnet152 and InceptionV3. The VGG16 model achieved an accuracy of 97.26%, Resnet50 achieved an accuracy of 98.8%, Resnet18 achieved an accuracy of 98.4%, InceptionV3 achieved an accuracy of 95.41% and Resnet152 achieved an accuracy of 99.2%. We therefore, used InceptionV3 model for deployment in mobile phones because it has low computation cost and low memory requirements of the all models. The developed tool was capable of detecting diseases with 99% of confidence of the captured leaves from the real world environment. The system was developed on Android based application in English language. The developed tool has the potential to support smallholder farmers and extension officers to detect banana fungal disease at early stages. We conclude that early detection of the diseases is important. Hence control and management of banana fungal diseases will be done early for the improvement of banana yield.

DECLARATION

I, Sophia Leonard Sanga do hereby declare to the Senate of the Nelson Mandela African Institution of Science and Technology that this dissertation is my own original work and that it has neither been submitted nor being concurrently for degree award in any other institution.

Sophia Leonard Sanga ..... 17/12/2020 .....

Name and Signature of Candidate **Date**

The above declaration is confirmed

Dr. Dina Machuve ..... 17/12/2020 .....

Name and Signature of Supervisor **Date**

COPYRIGHT

CERTIFICATION

The undersigned certify that has read the dissertation titled "Development of an Early Detection Tool for Banana Diseases" and found the dissertation acceptable by the Nelson Mandela African Institution of Science and Technology (NM-AIST).

Dr. Dina Machuve ..... 17/12/2020 .....

Name and Signature of the Supervisor **Date**

ACKNOWLEDGMENTS

First and foremost, I am very grateful to my Lord for His grace, love, mercy, blessing all the time during my studies.

Secondly, I am grateful for the financial support from the DAAD German that covered my scholarship during the duration of my master's degree studies.

My supervisor Dr. Dina Machuve provided me great support and mentorship throughout my studies and I am thankful. I also extend my appreciation to Mr. Kennedy Jomanga for his expert advice, time, constructive comments and encouragement, during data collection and preparation of manuscript as an agricultural expert from Agriculture IITA.

I especially thank my colleagues Victor Mero, Mpawe Mleke, and Angelica Kayanda for their support and cooperation.

I am also indebted to recognize the role played by staff and Management from NM-AIST for their support and assistance which helps us to achieve our goals.

DEDICATION

I dedicate my dissertation to my husband Joachim Nchimbi Kimisha, my daughters Jocelyne and Jordine and my son Joshua for their support, motivation and love.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF APPENDICES

LIST OF ABBREVIATIONS

App	Application
CNN	Convolutional Neural Network
CPUs	Central Processing Units
DCNN	Deep Convolutional Neural Network
ECA	East and Central Africa
ERD	Entity Relationship Diagram
FP	False Positive
FN	False Negative
GPUs	Graphical Processing units
ICT	Information and Communication Technology
IITA	International Institute of Tropical Agriculture
SGD	Stochastic Gradient Decent
TP	True Positive
TN	True Negative
UI	User Interface
XML	Extensible Markup Language
DL	Deep Learning
DTL	Deep Transfer Learning
ANN	Artificial Neural Network

CHAPTER ONE

INTRODUCTION

1.1 Background of the Study

Bananas are one of the staple foods and cash crops for about 70 million people especially in East Africa Counties (Etebu & Young-harry, 2014; Ordonez, Seidl, Waalwijk, Drenth & Kilian, 2015). Banana is largely produced by small-holder farmers in East and Central Africa (ECA) including Uganda, Rwanda, Burundi, Western Kenya, Tanzania and the Democratic Republic of Congo (Nkuba *et al.,* 2015). Regardless of its impact, the yields are largely affected by diseases (Swennen, 2004) commonly are Fusarium wilt race 1 and Black Sigatoka (Ramadhani, Machuve & Jomanga, 2017; Shimwela *et al.,* 2016).

Black Sigatoka is a leaf spot disease of banana caused by an airborne fungal pathogen called *Cercospora fijiensis* (Deltour *et al.,* 2017; Etebu & Young-harry, 2014). In black Sigatoka disease, the first symptoms appear as dark brown specks on the lower surface of the leaf (Gutierrez-Monsalve *et al.,* 2015). Fusarium wilt race 1 is a destructive soil-borne disease caused by *Fusarium oxysporum f*.sp. Cubense (Foc), can be identified by the symptoms such as wilting of older leaves and yellowing color which progresses to the young leaves until the whole plant is affected and eventually die. Another external symptom often linked to banana Fusarium wilt is the splitting of the pseudostem and an internal symptom in the rhizome and pseudostem, irrespective of the cultivar affected (Ssali, Potato & Agriculture, 2017; Thangavelu & Mustaffa, 2014). However, the process of identifying and detecting these diseases using human eyes at an early stage is very difficult.

The production of bananas in East and Central Africa (ECA) has declined since the 1970s and now yields are a fraction of its potential (Ssali *et al.,* 2017). The low production of bananas has resulted in significant household food insecurity and loss of income (Nkuba *et al.,* 2015). Studies in Tanzania indicate that the infestation of banana fungal diseases is still high (Ramadhani *et al.,* 2017; Shimwela *et al.,* 2016). In Tanzania production of banana is about 2.5 million tons of fruit for each year and ~290000 ha (Mgonja *et al.,* 2020) is the total area under banana production. Also in Bukoba district, about 11 876 farms were affected by these diseases (Nkuba *et al.,* 2015).

Early detection of diseases allows for the control and management of diseases properly with the potential to save crops from damage (Mishra, Mishra & Santra, 2016; Ramchearan *et al.,*

2019). Smallholder farmers and extension officers rely mainly on traditional knowledge to detect and identify the diseases (Ramadhani *et al.,* 2017). Diseases can be effectively treated by the time of diagnosis but the availability of tools and methods for early detection, control and management of the Diseases is a challenge (Dyrmann, Karstoft & Midtiby, 2016; Patil & Pawar, 2017; Ramcharan *et al.,* 2017).

Machine learning techniques and computer vision have been used to detect diseases from different crops such as tomatoes, cotton, oranges, grapes and potatoes (Amara, Bouaziz & Algergawy, 2017). Computer vision systems in agriculture provide useful information in real-time about diseases and reduce costs and attributes of the product (Amara *et al.,* 2017). Moreover, image processing using computer vision methods can decrease the computational costs hence, leaf disease detection can be made easy and faster (Lokesh, Naveenkumar, Rajesh, Kamath & Rathnam, 2017). In machine learning, Deep Convolutional Neural Network (DCNN) is used in the detection, identification, and prediction of pests and diseases from crops (Mg, Hanson, Joy & Francis, 2017). It is also used to provide knowledge and understanding of different plant pests and diseases response to pathogen effectors (Yang & Guo, 2017). The recent advancements of a smartphone, advances in deep transfer learning techniques, and computer vision have indicated deployment of the models on smartphones for disease detection and identification (Amara *et al.,* 2017; Ramcharan *et al.,* 2017). The uses of smartphone devices provide many advantages of quick response time and low communication bandwidths.

Deep Convolutional Neural Network has been used in many fields of computer vision such as natural language processing, speech recognition and face recognition (Voulodimos, Doulamis, Doulamis & Protopapadakis, 2018). Nevertheless, in computer vision, DL was found more effective for object detection, image recognition, image segmentation tasks, self-driving cars and object recognition (Liu *et al.,* 2017).

In agriculture, deep learning has successfully been deployed on smartphone assisted disease detection based on leaf images (Eli-chukwu, 2019; Ramcharan *et al.,* 2019). There are efforts made using a mobile application to provide information to smallholder farmers regarding banana disease management to prevent crop loss due to diseases (Ramadhani *et al.,* 2017). The study further indicated information on disease identification is disseminated by local plant clinics and agricultural extension officers.

Mobile penetration in Tanzania had reached 95% by September 2019, with more than 43.67 million people using mobile phones (Ng'wanakilala, 2019). This fact promises the smartphone

infrastructure can be used for the deployment of a deep transfer learning model for banana disease detection.

The developed tool which is a smartphone-based application will help smallholder farmers and extension officers detect banana diseases in actual time. Therefore, this research study focused on the development of an early detection tool (smartphone-based app) for detecting diseases from banana using leaf images, using DCNN, and transfer learning. Hence, the developed tool can support smallholder farmers and extension officers in the early detection of banana diseases to improve banana yield.

The purpose of this research is to develop an early detection tool to support smallholder farmers in banana diseases detection. The proposed smartphone-based application is expected to also reduce the workload on the extension officers in disseminating information to farmers regarding the banana diseases occurrence. The application of deep learning-based techniques and transfer learning techniques have shown promising results in the detection of crop diseases, among many types of machine learning techniques using image datasets (Hwan *et al.,* 2014; Mishra *et al.,* 2016; Ramcharan *et al.,* 2017).

Hence this research is motivated by a challenge on early detection of the most commonly banana diseases namely, Fusarium wilt race 1 and Black Sigatoka grown by smallholder farmers in Arusha and Mbeya regions in Tanzania (Ramadhani *et al.,* 2017; Shimwela *et al.,* 2016). Early detection, control, and management of plant diseases are potentially more important than the classification of diseases, due to their implications in the agriculture sector (Rumpf *et al.,* 2010; Yang & Guo, 2017).

1.2 Statement of the Problem

There have been efforts by researchers in providing banana disease outbreak information to the smallholder farmers and extension officers (Nkuba *et al.,* 2015). Banana is a cash crop and staple food cultivated in Tukuyu district in Mbeya Region and Arumeru District in Arusha region in Tanzania. Banana production in these regions is affected by the most two common diseases of Fusarium wilt race 1 and Black Sigatoka (Gallez *et al.,* 2004; Ganry *et al.,* 2012; Ramadhani *et al.,* 2017; Shimwela *et al.,* 2016).

Different studies reveal that early detection of plant diseases in the agriculture sector is a big challenge that needs to be treated with special attention (Fuentes *et al.,* 2017; Mg *et al.,* 2017; Singh *et al.,* 2016).

A few mobile phone tools were developed to detect crop diseases for tomato, Brinjal crop, cassava and plant diseases identification (Gajanan, Shankar & Keshav, 2018; Gorad & Kotrappa, 2019; Ramcharan *et al.,* 2019; Ramcharan *et al.,* 2019; Ramcharan *et al.,* 2019; Ramcharan *et al.,* 2019; Ramcharan *et al.,* 2019; Verma, 2019). Model development work on banana diseases of Sigatoka and Fusarium wilt by (Owomugisha, Quinn & Mwebaze, 2019) suggested that the work can be extended to work on mobile phones.

There is a big challenge on early detection of banana fungal diseases for smallholder banana farmers who live in Arusha and Mbeya regions, due to the lack of necessary tools. Smallholder farmers in these regions mainly use traditional methods to identify disease occurrence (Ramadhani *et al.,* 2017). Smartphone-based applications are low cost, user friendly, and computationally efficient for the deployment of deep learning models such as detection of banana diseases using image datasets (Deng, 2019; Sharada *et al.,* 2016).

Therefore, this study proposes a smartphone-based application for the deployment of deep learning models to detect banana diseases of Sigatoka and Fusarium wilt. Machine learning techniques and computer vision methods have proved to be effective in addressing diagnostics problems in agriculture (Amara *et al.,* 2017; Jagan *et al.,* 2016; Prabha *et al.,* 2014). Though in the next chapter, we were able to discuss the disease overview, deep CNN for image classification, deep learning architectures, and mobile application used to detect diseases for crops.

1.3 Rationale of the Study

Different studies reveal that early detection of plant diseases in the agriculture sector is a big challenge that needs to be treated with special attention (Fuentes *et al.,* 2017; Mg *et al.,* 2017; Singh *et al.,* 2016). There is a big challenge on early detection of banana fungal diseases for smallholder banana farmers who live in Arusha and Mbeya regions, due to the lack of necessary tools. Smallholder farmers in these regions mainly use traditional methods to identify disease occurrence (Ramadhani *et al.,* 2017). Smartphone-based applications are low cost, user friendly, and computationally efficient for the deployment of deep learning models such as detection of banana diseases using image datasets (Deng, 2019; Sharada *et al.,* 2016).

1.4 Objectives

1.4.1 Main Objective

The main objective of this research is to develop an early detection tool for banana diseases in Mbeya and Arusha regions.

1.4.2 Specific Objective

(i) To determine the features of banana fungal diseases using the leaves image dataset.

(ii) To develop an image classification model for early detection of banana diseases by using deep learning.

(iii) To develop a smartphone-based application for the deployment of the proposed model.

(iv) To assess user acceptance of the developed model when deployed on a mobile application

1.5 Research Questions

(v) What features are required for the early detection of banana fungal diseases?

(vi) How model selection will be conducted?

(vii) How can banana fungal disease detection tool be developed to support farmers in early detection of banana disease from the proposed model?

(viii) How will the developed tool be tested?

1.6 Significance of the Study

The proposed application has the potential to support extension officers and smallholder banana farmers in the early detection of banana fungal diseases in real time. The detection, control, and management of banana diseases are potentially more important than the identification and classification of diseases, in the future due to the implications in the agriculture sector (Yang & Guo, 2017). The expected outcome of this research work is supporting smallholder banana farmers and extension officers in Arusha and Mbeya regions in Tanzania with the provision of a smartphone-based application for early detection of banana diseases. The tool will enable them to:

(i) Take pictures of an affected leaf using a mobile phone and upload it to the app for detection and receive feedback.

(ii) Detect and recognize the type of disease the banana plant is affected by using an early detection tool deployed on a mobile phone.

(iii) Reduce the work of the extension officers in disseminating information to farmers regarding the banana fungi disease occurrence.

1.7 Delineation of the Study

The study is focused on the development of an early detection tool for two banana diseases, namely, Fusarium wilt race 1 and Black Sigatoka. The dataset for the development of the tool was banana leaves images collected at the farm level in Tanzania. The machine learning models developed were deployed as a mobile application to support smallholder farmers in detecting the occurrence of banana fungal diseases.

CHAPTER TWO

LITERATURE REVIEW

2.1 Overview of Banana Fungal Diseases

Black Sigatoka is a leaf spot disease of banana caused by an airborne fungal pathogen called *Cercospora fijiensis* (Deltour *et al.,* 2017; Etebu & Young-harry, 2014). The initial symptoms look like dark brown specks on the lower surface of the leaf (Gutierrez-Monsalve *et al.,* 2015).

Banana plants with Fusarium wilt race 1 is caused by the soil-borne pathogen, called *Fusarium oxysporum f.* sp. Cubense is recognized by the symptoms such as wilting and yellowing of the older leaves (Deltour *et al.,* 2017; Thangavelu & Mustaffa, 2014). This effect progresses to the youngest leaves until the whole banana plant dies, also the splitting of the pseudostem irrespective of the cultivar affected (G.Region, 2017; Gutierrez-Monsalve *et al.,* 2015).

However, the majority of smallholder banana farmers rely on traditional methods for disease identification or detection from banana plants (Ramadhani *et al.,* 2017). Historically, disease detection or identification has been distributed by agricultural officers and other sectors eg. local plant clinics (Mohanty *et al.,* 2016). Furthermore, these agricultural experts or officers must visit plantations regularly to avoid disease spreading. Usually, the method of monitoring or detecting these diseases by using an agricultural expert or local plant clinics is time consuming and difficult task for experts (Brahimi, Arsenovic, Laraba & Sladojevic, 2018).

Banana fungal diseases are still a big challenge for smallholder farmers causing high crop losses of up to 100% (Nkuba *et al.,* 2015). Similar studies have reported different cases of crop loss as a result of the failure of early disease detection (Jagan *et al.,* 2016; Lokesh *et al.,* 2017; Martinelli *et al.,* 2015). These challenges makes the need of developing an early detection tool which is a smartphone-based app to detect and identify diseases, hence control and management can be done properly to protect banana plants.

2.2 Image-Based Detection of Plant disease Using Deep convolution Neural Network

Patil and Pawar (2017) proposed a deep learning method for detecting diseases from different plants by using leaves images. They used different detectors for disease detection and classification using architecture called Region-based Fully Convolutional Network. The architecture was tested with datasets from images that some were downloaded from the internet and captured by using camera devices from different places.

Sladojevic *et al.* (2016) proposed a deep learning method that was able to classify 13 different types of plant diseases using diseased and healthy leaves images, with the capability to distinguish plant leaves from their environments. CNN approach was used, with leaf images diseases from the plants were classified and recognized. The model had a performance of 91% and the precision value was 98% with an average of 96.3%.

A study conducted by Selvaraj *et al.* (2019) addressing banana pest diseases (based on leaf and the banana fruits). He proposed an Artificial Intelligence used to detect pests and diseases from banana plants by using deep machine learning techniques that in the future can be integrated on a mobile phone. They retrained several convolution network model architectures and finally, chose ResNet 50 and InceptionV3 for model development. The study finally was capable to train DTL to create a system that can make correct predictions.

Deep learning and the advances in computer vision allow increasing plant protection from diseases, in the agriculture sector (Mg *et al.,* 2017). Deep CNN was trained and fine-tuned and achieved an accuracy of 96.3% while after 100 epochs the training accuracy was 95.8% but this was done without fine-tuning the parameters. The DCNN method was used to enable an easy and quick system implementation during training. The datasets were trained by DCNN and the model was able to differentiate between healthy leaves and unhealthy leaves and the final results were 95% accurate.

Deep Convolutional Neural Network (DCNN) architectures on a public dataset were applied for plant disease classification (Brahimi *et al.,* (2018). Three architectures from DCNN were selected and trained to detect diseases from plants at an accuracy of 99.7%. The saliency maps visualization technique was used to understand and interpret the result of the DCNN. The visualization technique increases the transparency of the DL modes and it gives additional details of the disease detected from the plant.

The study was done by Sharada *et al.* (2016) used GoogLeNet and AlexNet architectures from deep learning. The aim was to classify both species and identify diseases on images from plat leaves that the model has never seen before. Plant Village data set of about 54306 images contains 26 diseases and 38 classes of 14 crops were used. The accuracy of the trained models was 99.35%, however, without feature engineering, the trained model succeeded in classifying disease and crop from 38 classes in 993 out of 1000 images.

Moreover study conducted by Fuentes *et al.* (2017) was able to detect pests and diseases on tomato leaves. With a deep learning technique, tomato leaves images were captured by the

camera at different resolutions were analysed, processed, and tested for pests and disease presence and the model was able to give the results (positive or negative pests and or diseases presence).

2.3 Mobile Application for Supporting Farmers in Disease Detection

There are Some tools have been developed by researchers on the problem of detecting, and identifying banana leaves diseases.

Ramcharan *et al.* (2017, 2019) used deep learning techniques and mobile phones for the detection of cassava disease using images taken from the real environment in Tanzania. The training for the deep CNN model was used to classify 2 pest occurrence and 3 damage diseases. However, the best-trained models achieved an accuracy of 98% for brown leaf spot, 96% for cassava mosaic disease and 98% for cassava brown streak disease damage, during testing the model accuracy was 93% for the dataset that were not used during the training process. The models were validated in the real field with a mobile phone application embedded with TensorFlow android inference. However, the developed mobile application was limited to detect diseases for cassava only.

Another study conducted by Gorad and Kotrappa (2019) discusses a machine learning model that was integrated with mobile phones for crop disease prediction in India. By using a smartphone, camera plant leaves images were captured then transmitted to the back end model for processing. New images in clusters of different disease categories with historical data were presented by using K-means algorithm. Through this application, farmers were able to request information using a desktop GUI form or web browser and through messaging or app service and access the results.

Selvaraj *et al.* (2019), addressed a banana pest diseases detection challenge (based on leaf and the banana fruits) by proposing an AI-banana disease and pest detection using deep machine learning techniques that in the future was integrated on a mobile phone. Convolution network model architectures were retrained and finally, ResNet 50 and InceptionV3 were selected for the model development. Hence authors were able to train DTL to create a model that can make correct predictions.

A deep learning mobile based on plant disease diagnostic was proposed using CNN after training five CNN models on tomato leaf images. The study ended up with ResNet 50 as the

accurate prediction model and it was employed in the mobile application for tomato diseases classification and identification (Verma *et al.,* 2019).

According to Shriram *et al*. (2019), a deep learning mechanism integrated with a mobile phone as a means of early diagnosis of leaf diseases was developed. A convolutional neural network that uses leaf images as input, analyses, and categories was employed. The sole purpose was to detect plant diseases in early stages through leaf images that are captured by a farmer's mobile phone camera.

Another study conducted by Gajanan *et al.* (2018) developed an intelligent system that can be used for the diagnosis of disease from the plant at an early stage. The system used images of the plant captured with visible infrared light. The system was able to recognize the diseases from plants using symptoms such as lesion or spots in a different part of a plant.

Moreover, Owomugisha and Mwebaze (2016) and Owomugisha *et al.* (2019) proposed an automated platform to detect the occurrence of black sigatoka and Fusarium wilt. Various Computer vision techniques were used to diagnose these diseases. The randomized trees method achieved an accuracy of 0.96 for BBS and 0.91 for Fusarium wilt. The performance of the selected classifier was assessed from the covered area under the curve. They conclude mobile deployment process remains as future work.

2.4 Deep Learning

In machine learning deep learning is a method used to train a single ANN and it performs a task by frequently testing its performance of the task and altering internal parameters within the network between each test (Voulodimos *et al.,* 2018). The DCNN has been used in several areas example in computer vision, faces recognition, natural language processing and speech recognition. Though in computer vision, DL was found more effective for self-driving cars, image recognition, robotic, object recognition and image segmentation tasks (Liu *et al.,* 2017). Its architecture is inspired by how the neurons in the human brain work. Each neuron in the Convolutional Neural Networks will either activate or deactivate when observing a certain object. Each layer will focus on different features. In a computer vision task for example, the 1st layers may focus on lines and edges, and while we progress towards the last layers the focus will change to e.g. colours. Putting all the neurons together and the network is able to recognize even small details in an image shown on Fig. 1.

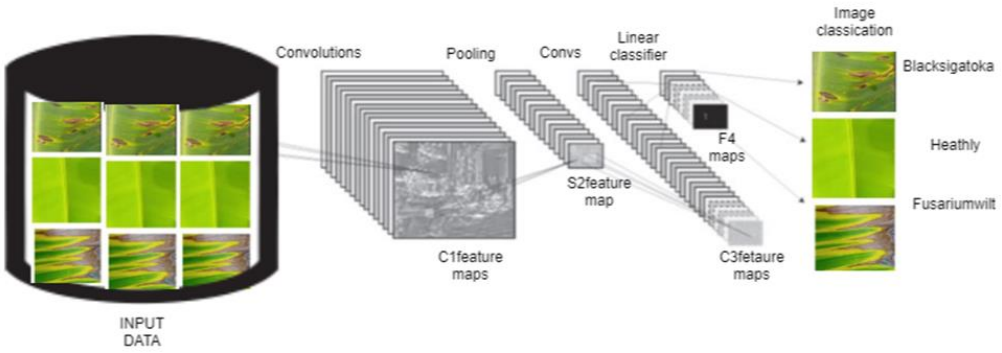


Figure 1: CNN architecture for a computer vision task (Voulodimos *et al.*, 2018)

The architecture of CNN consists of different layers and each layer will focus on different work the first layer is called Convolutional Layers. This layer uses various kernels to train the image and its internal feature maps and creates various feature maps. Because of its advantages, the convolution method is used for many works (Voulodimos *et al.*, 2018). The second layer is called Pooling layer, the aim of using this layer is to reduce the spatial dimensions (width × height) of the input size for the next convolutional and make it strong to variations for previously learned features. However, this technique works out in computing the maximum value in each area at different positions (Bayar & Stamm, 2016).

The next layer after the pooling layer is called a fully connected layer. This layer is involved in making a high-level of reasoning in the neural. All Neurons in a fully connected layer have a complete connection to all activation in the previous layer, as their name implies (Krizhevsky, Sutskever & Hinton, 2012). Their method is calculated with a matrix calculation followed by a bias offset. The layer can transform the 2D feature maps into a 1D feature vector. The created vector either could be fed onward into a certain number of classes for classification or could be used as a feature vector for extra processing (Girshick, Donahue, Darrell, Berkeley & Malik, 2012).

2.5 Deep Learning Network Architectures

This section will explain the deep learning network architectures used in training our datasets

2.5.1 VGG16

The first architectures trained are VGG16 is a convolutional neural network. VGG16 model has the capability of training about 140 million images from the Image Net database. The VGG16 consists of 16 layers deep shown in Fig. 2. The network is capable of classifying images into 1000 object classes, such as house, car, mouse, keyboard, bottle of water and animals

(Simonyan & Zisserman, 2014). VGG16 has many features used for representing a large group of images. This network takes an image input size of 224 x 224 and it is considered state-of-the-art. The implementation of VGG16 architectures in JAVA contains 13 layers, 5 Max pooling layers and 3 fully-connected layers, to make a total of 21 layers. To implement in python 3 x 3 filter sizes are used with a stride of 1 pixel for all convolutional and Max-pooling layers are executed with a 2 x 2 filter with a stride of 2 pixels. The ReLU activation is used in every convolutional, the AGD optimization algorithm is used for the implementation process and fully-connected layer in the network. The AGD uses a probability of 0.5 along with the dropout regularization method to reduce the overfitting occurrence in two of the three fully-connected layers. The Softmax layer is used as a final layer and a negative log-likelihood loss function.

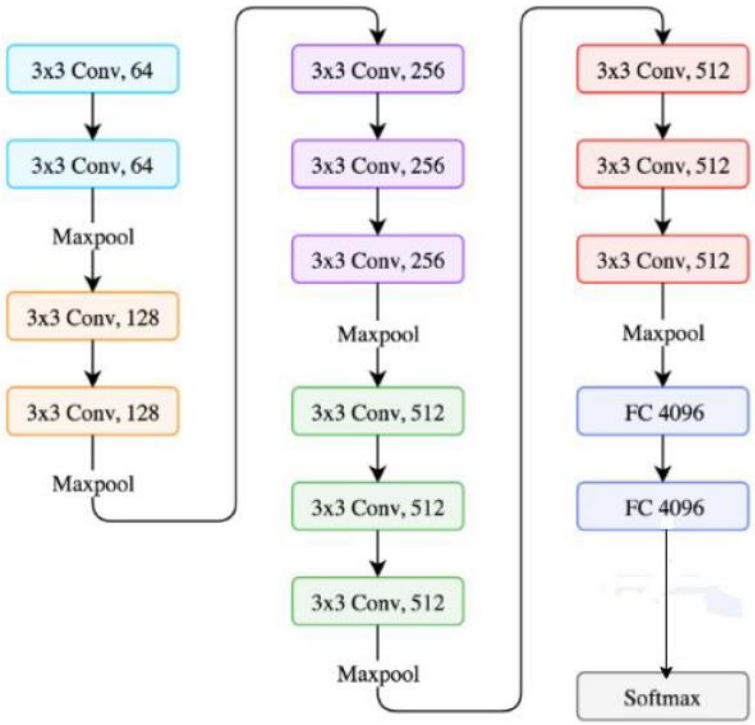


Figure 2: VGG16 (Andersen, 2019)

2.5.2 Residual Network (ResNet)

It is a convolutional neural network work by shortcutting with identity functions (He, Zhang, Ren & Sun, 2015). The network allows more features to be reused directly also it improves the training efficiency of the model. This network is motivated by checking that its neural network works to acquire higher training error when the depth increases to big values. However, residual network shows to improve the training speed very well (Szegedy, Ioffe, Vanhoucke & Alemi, 2016) the network has different versions such as ResNet18, ResNet50, ResNet152.

2.5.3 InceptionV3

It is a deep convolutional network, in recent years the network show higher performance in image recognition. Also, the Inception architecture has achieved very good performance with a low computational cost (Szegedy, Ioffe, Vanhoucke & Alemi, 2017) when compared to other deep convolutional networks. However, the introduction of residual connection together with traditional architecture has shown the best performance and become the winner of the ILSVRC 2015 challenge. To reduce the computation cost from residual learning, InceptionV3 uses a 1 × 1 convolution method illustrated in Fig. 3.

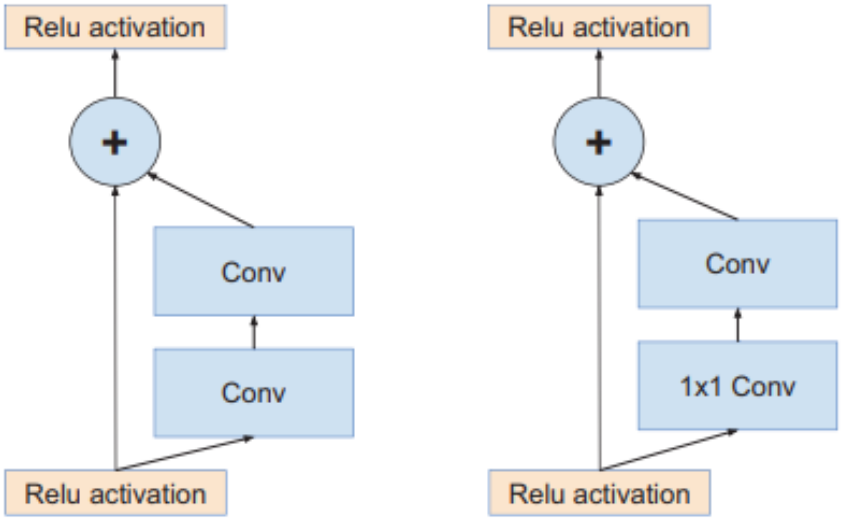


Figure 3: Residual (inception) connections (He *et al.*, 2015)

A residual (inception) connection from Fig. 3 left sides is the first residual connection. While from the right side is an improved one that decreases the computational cost with the use of a 1 × 1 convolution called InceptionV3 (Szegedy *et al.*, 2017) and InceptionV3 is among the transfer Learning examples shown in Fig. 4.

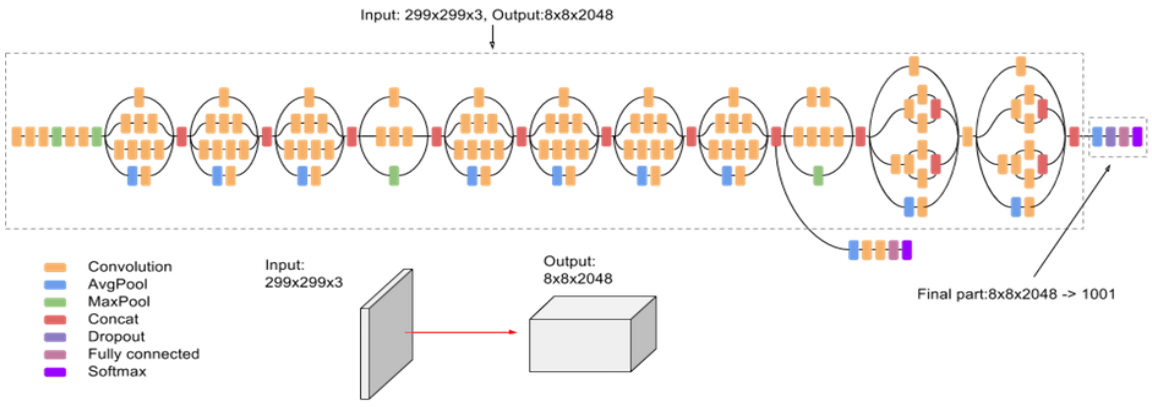


Figure 4: InceptionV3 architecture

The transfer learning technique is used to speed up the process of classifying Image by using InceptionV3 as a starting point for transferring modules (Medium.com, 2019). Also, deep convolution neural network can be implemented by using python as a programming language; also TensorFlow platform is used for handling machine learning algorithms.

2.6 The Research Gap

The literature indicates that a lot of deep learning models used to detect and identify diseases from the plant were not deployed to reach stakeholders mainly farmers and extension officers. Model deployment is mainly recommended as an area of future work.

A Mobile-Based Deep Learning Model for Cassava Disease Diagnosis that exits in Tanzania was used to detect a disease from cassava. The app uses the cassava leaf dataset collected from IITA in Bagamoyo District, Tanzania. Authors suggest that the developed mobile app needs to be evaluated in real environment conditions for plant disease diagnostic. Based on the result they recommend that mobile Convolutional Neural Network models be used on the collected cassava data set.

Hence still smallholder banana farmers are not able to use the developed app to detect banana disease occurrence. Smallholder farmers in the selected regions still use historical methods to identify disease occurrence.

The few mobile apps that were developed to detect diseases on crops use a single crop (Ramcharan *et al.,* 2019) in Tanzania. The use of a dataset downloaded from the internet makes the models fail to detect the disease in a real-world environment. Also, the use of segmentation techniques reduces the performance of the image captured since the background was removed during images pre-processing. However, different studies have reported different cases of crop loss as a result of the failure of early disease detection (Jagan *et al.,* 2016; Lokesh *et al.,* 2017; Martinelli *et al.,* 2015). Hence, early detection and classification of diseases from crops using Leaves image is still a big challenge to many smallholder farmers as well as extension officers (Dyrmann *et al.,* 2016; Patil & Pawar, 2017; Ramcharan *et al.,* 2017), due to lack of necessary tools that will help to prevent the possible outbreak of pests and diseases on real-time (Jagan *et al.,* 2016; Rumpf *et al.,* 2010).

There are many benefits of detecting diseases from the crop by using automatic methods at its initial stage to facilitate the control and management of the diseases at early stages (Mishra *et al.,* 2016; Ramcharan *et al.,* 2019). Due to the challenges mentioned in research gaps, there is a

need for developing an early detection tool which is a smartphone-based app by using Machine learning techniques and Computer vision to support farmers in disease detection (Jagan *et al.*, 2016; Ramcharan *et al.*, 2017). In order to achieve the detection of banana disease at an early stage, we need to have a supportive tool which is a smartphone-based app which is less expensive, real-time, automatic, user friendly and fast for the detection of banana diseases by using image datasets (Al Hiary *et al.*, 2011).

The literature review revealed that the availability of tools for early detection of banana diseases is a challenge for smallholder farmers and extension officers and as a result, its impact is low productivity and low yields of banana.

Therefore, this research will take the advantage of DCNN and transfer learning technique from machine learning techniques and computer vision to develop a tool which is a smartphone-based app in addressing this problem. The developed tool aims to support farmers to detect and recognize the disease occurrence from the banana plants at an early stage. Also, the tool will reduce the work of the extension officers to disseminate information to farmers regarding the fungi disease occurrence.

CHAPTER THREE

MATERIALS AND METHODS

3.1 Study Area

This study was conducted in Rungwe District in Mbeya and Arumeru District in Arusha region, Tanzania from November 2018 to February 2019. Rungwe District which involves two wards Tukuyu (that include Ikama, Katumba, Itagata and Ilinga villages) and Itete wards located at Latitude in 9° 15' 00" S, Longitude in 33° 40' 00" E, and altitude of 2 981m. In Arumeru District, two wards with six villages were visited; Akheri (that includes Patandi, Akheri, Nguruma and Ngyani villages) and Nkoaranga (which has Nshupu and Nkoaranga villages) located at Latitude: -3° 07' 60.00" S, Longitude: 36° 51' 59.99" E and altitude of 1724 m. These regions were selected due to the high cultivation of banana crops. During data collection, we used the Open Data Kit (ODK) tool installed on Tecno wx3 mobile phone camera to capture banana leaves images shown in Fig. 5 under different conditions depending on the seasons and time (i.e. temperature and humidity) Table 1 with the percentage of images in each category.

(a)Black Sigatoka at late stage (b) Fusarium wilt race 1 at late stage (C) Healthy leaf

(d) Black Sigatoka at early stage

(e) Fusarium wilt race 1 at earlystage

Figure 5: Banana leaves images

The possible spreading of banana Fusarium wilt 1 and black Sigatoka disease in Tanzania are shown in Appendix 4. Circles with bolded dashes are major regions producing banana crops (Kilimanjaro and Mbeya in Tanzania); dotted circles without bold show the areas with average banana production (Shimwela *et al.*, 2016).

Table 1: Banana leaves images captured under different condition

Banana Classes	Early stage images/ %	Mid stage images/ %	Late stage images/%	Lighting stage images /%
Black Sigatoka	343/15.37	793/34.99	980/43.24	150/6.619
Fusarium wilt race1	259/15.35	530/31.41	758/48.93	140/8.299
Healthy	-	-	-	2400/100

The dataset was collected with the help of two banana specialists from Arusha region and one expert from Mbeya region from IITA, Tanzania.

3.2 Research Framework

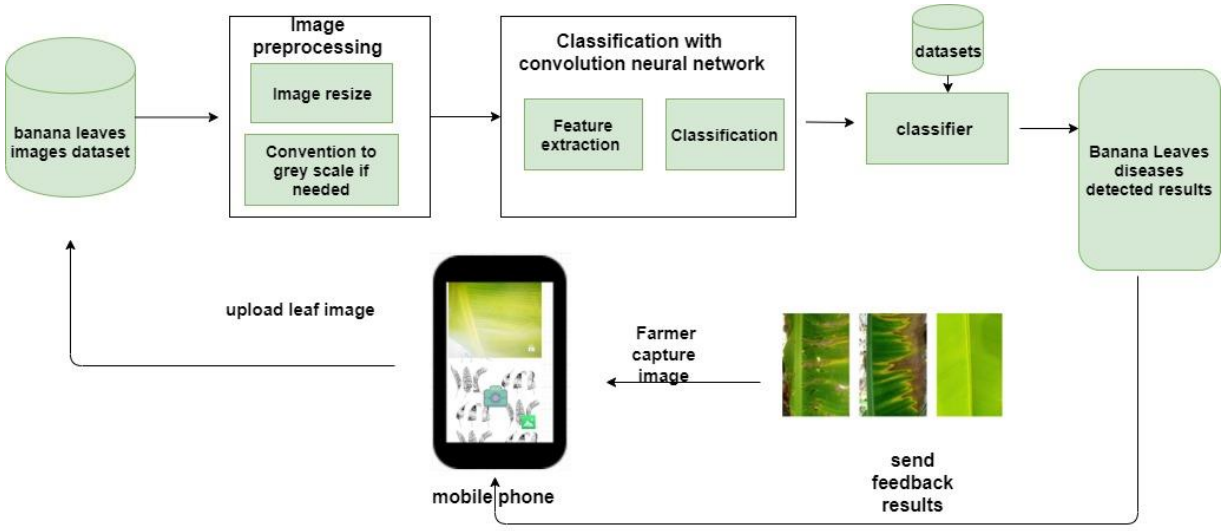


Figure 6: Conceptual Framework

The research framework above indicates the different activities on this research were conducted from data collection (section 3.1), data analysis and pre-processing, model selection, and finally how the developed system was validated to detect banana diseases illustrated on Fig. 6. All activities done from this research framework was indicated in Section 3.3 to Section 3.10 and Fig. 7.

3.3 Data Analysis and Preprocessing

3.3.1 Labeling

The collected dataset was labeled manually with the help of three agricultural banana experts IITA by naming the folder containing images with their respective class. Then the labeled images were processed by using transfer learning techniques. The process of labeling the dataset helped the model to classify well each image during training.

3.3.2 Resizing Images

Data analysis was performed to understand the statistical and general properties of the collected images. The collected images were RGB in jpeg format with a height of 1920 pixels and a width of 2560 pixels.VGG16 were used to fix image size 224 x 224 pixel RGB to the cov1 layer during training. ResNet18, ResNet50, ResNet152 were used for an image size of 224 x 224

pixel RGB on training while InceptionV3 was used on an image of size 299 X 299 pixel as an input.

3.3.3 Reducing Overfitting

To increase the size of our dataset for training the selected models and reducing the overfitting problem data augmentation technique was used by adding slight variations of changes to the existing data (image) to generate more image datasets for the model well for generalization. The augmentation technique for image data generator was l rotating-left-bottom, horizontal flipping, rotating –right-bottom, shear of 0.1 ranges, zoom of 0.2 ranges, cropping, and rescaling ration of 1/255 factor for each image was done. Augmentation process each class contain 3000 leaves images was conducted to make a total of 9000 datasets.

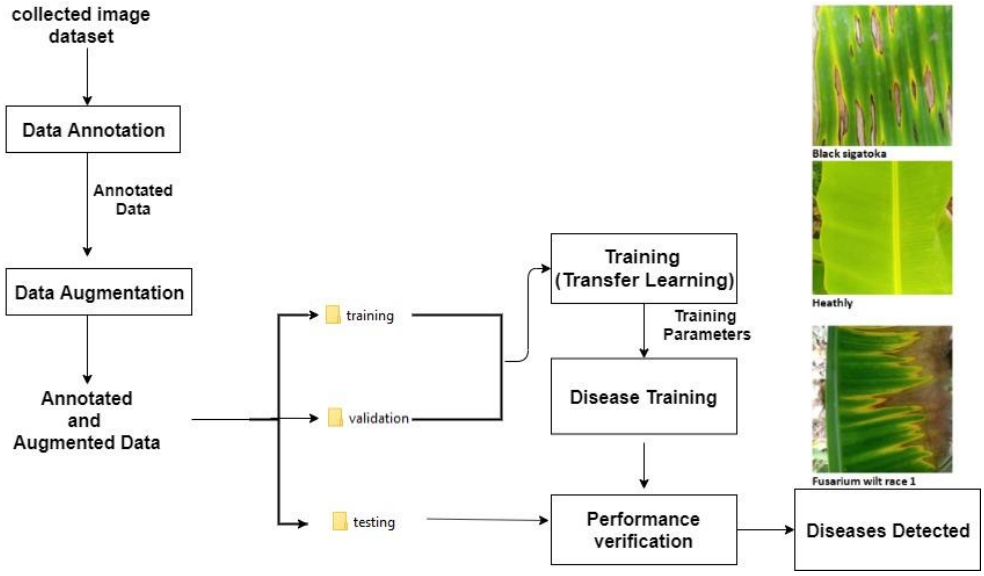


Figure 7: Data analysis and preprocessing steps

3.4 Experimental Setup

The experiments for this research were carried out using a desktop computer installed with Ubuntu 18.04 with the specifications summarized in Table 2. The programming language used during training was Python. Keras library was used as a baseline during implementation with Tensor Flow on the backend to enable models to have high performance for numerical computation during training. The software library used to train the selected models was the Google collab machine with runtime type as Python3, Notebook size of 20Mb and hardware accelerator as GPU.

Table 2: Experimental setup specifications

Device name	Description
Memory	251.8GiB
Processor	Intel xeon(R)CPU E5-26200@2.00GHz*12
Graphics	NVD9
GNOME	3.32.1
OS type	64-bit
Disk	1.0TB

3.5 Retraining Selected Models

In this research, five pre-trained (re-used) models were trained using the transfer learning technique. These models were ResNet18, ResNet50, ResNet152, VGG16 and InceptionV3. From deep convolutional neural networks, we assess the applicability of these techniques for the classification problem described in this research. The selected five models are among the DCNN and they are foremost state-of-art in computer vision tasks. Instead of using traditional ways of training a classifier to train feature extraction by using the hand-designed method, the Convolutional neural network (CNN) "learns feature hierarchy from pixels to classier and train layers jointly up to the final output" (Szegedy *et al.,* 2016). However, as a result of its density, CNN takes even more time for complete model training with millions of image samples. To overcome this challenge, The selected five models was used for model development (Tan *et al.,* 2018). Studies revealed that the " transfer learning technique is effective for many applications and reduces training time than training or learning from scratch" (Karpathy *et al.,* 2014; Shijie, Peiyi, Siping & Haibo, 2017).

In comparison to mobile-friendly architectures such as MobileNet, the InceptionV3 is optimized for correctness when making predictions (Woff, 2017). MobileNet architecture is optimized for speed (Gavai, Jakhade, Tribhuvan & Bhattad, 2017). The MobileNet model utilizes depth-wise separable convolutions, which is similar to InceptionV3 architecture (Bankar & Gavai, 2018). Since the study focused on the detection of banana diseases, the accuracy (correctness) on prediction is more important than speed. Hence the deployment of the model was on InceptionV3.

3.5.1 Training Procedures

To get the best hyper-parameter that would give better performance these parameters were selected with consideration of batch size, optimizer, momentum, learning rate and weight decay.

To train the models, the dataset was split into two ratios. To avoid bias in the re-used models, the datasets was randomly arranged before splitting. Banana leaves image dataset was divided in the following ratio 70:20:10 and 85:10:5 for training, validation, and testing respectively. The 70% and 85% contained a training dataset, and our model was validated on the 20% and 10% validation dataset to ensure that the training process was the correct way and made it easier to detect over fitting problems. While the test set with 10% and 5% test dataset was used to assess the performance (performance verification stage) of the selected model. All hyper-parameter searches were done on the validation set to avoid overfitting and degradation problem. The selected model was trained with a new dataset by taking only the last layer of the re-used (transfer learning) model and exchanged it with the new untrained model as illustrated in Fig 8. The last two layers called Softmax was trained by using the early weight of the model, for recognizing and classifying the new images. The retrained final layer was used because the old information was needed to differentiate between the earlier 1000 classes and the new object trained.

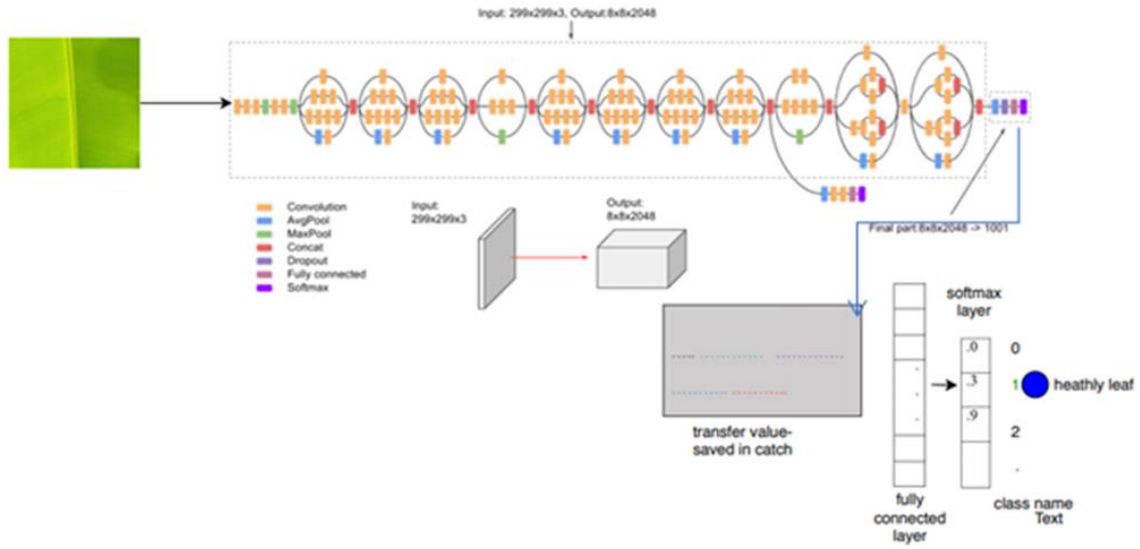


Figure 8: Transfer learning techniques (Jonsson & Jonsson, 2018)

3.6 Evaluation of the Classifier

Each model was evaluated after every 5 iterations by checking the classification accuracy and loss during the training process. To evaluate the model's performance, confusion-matrix metrics and accuracy were used. During training, the model parameters were updated, and at the same time tuning the hyper parameters for the models to generalize well.

3.6.1 Confusion Matrix

The confusion matrix is the table that shows how the model confuses during training and it explaining the performance of a model from its classification. It gives a better idea of what your model is getting right and what types of errors the models make during the classification process. Hence to visualize our pre-trained models' confusion matrix was used shown in Fig. 25 to Fig. 29. During training, the validation set contained 309 images for Black Sigatoka, 309 images for Fusarium Wilt race 1, and 309 images for Healthy, and the test set contains 150 images for Black Sigatoka, 150 images for Fusarium Wilt race 1, and 150 images for Healthy. Both validation set and test set were considered to evaluate model performance, predicting unseen data, and visualize the predicted result by using the confusion matrix.

3.6.2 Accuracy

Accuracy is the rate at which the model is capable to predict the correct value for a given data idea or observation defined in equation 1.

Equation 1: Accuracy equation

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FN}$$

Where TP =True positive showed the number of leaf images with Black Sigatoka and it classifies as Black Sigatoka as well as for Fusarium Wilt race 1 and Healthy class. FN = False Negative showed the number of leaf images with Fusarium Wilt race 1 but classified as Healthy class, TN=True Negative showed the number of leaf images with Black Sigatoka and classified as Fusarium wilt while FP = False Positive showed the number of leaf images with Black Sigatoka and classified as Healthy.

3.7 Development of a Mobile Application

After testing the model's performance we found that Resnet152 was able to meet our objective since it was able to detect a disease from the trained dataset and archived an accuracy of 99.2%. Mobile deployment on Resnet152 is complex and it has a high computational cost and requires large memory space. Hence, we selected inceptionV3 model for mobile application deployment due to its low computational cost and it has low memory requirements.

An early detection tool which is a smartphone-based application to support smallholder farmers and extension officers to detect diseases occurrence on real-time was developed. Tensor flow is

an open-source deep learning framework that offers Applications Program Interfaces for mobile deployment in IOS-based smartphones and Android. This framework was used to train the InceptionV3 model for banana disease detection. Also, Extensible Markup Language and JAVA was used as programming language in the Android application. The tool makes use of the leaf image capture using a mobile phone. It processes the captured image and gives feedback according to the leaf captured if it has diseases of it is a healthy one.

During software development, a user requirement of the entire system was considered and the software engineering approach was followed during the development of the tool. The Software development life cycle (SDLC) was selected as our way to the development of the tool which is a smartphone-based app. The flow of activities and progress of each task is presented using SDLC.

3.7.1 System Analysis

This is the first layer of SDLC. User needs are considered in this phase for developing new software also problems in the current system are identified. In this research the system study phase was done through these steps:

Existing system: explained in the research gap section 2.6.

Proposed system: FUSI SCANNER is an android based mobile app. It is a smartphone-based app used to support smallholder farmers and extension officers in the early detection of disease from the banana plant. The app was developed to work offline so that farmers can reduce the cost of buying internet bandwidth when visiting their farms, also the developed app is user-friendly and it can detect the disease on real time.

The major activities performed by FUSI SCANNER app in banana diseases detection are

- The app provides basic information about the disease detected with percent confidence.
- The app uses a camera as a user interface for disease detection.

3.7.2 Feasibility Analysis

Feasibility is defined as the level to which a project can be done successfully. The strengths and weaknesses of the proposed research were analyzed in this phase. To estimate feasibility, a feasibility study is implemented, to decide whether the solution considered is practical and

feasible in the software. The feasibility study is aimed to establish the motives for developing the software that will satisfy users, conformable and flexible to change to established standards. Technical feasibility, economic feasibility and operational feasibility was considered in the development of FUSI SCANNER.

3.7.3 Technology and System Feasibility

In this phase, we check if our research is within the boundaries of current technology and does the technology exist at all, or if it is obtainable within given resource constraints (i.e., budget, schedule). The developed tool on a smartphone is using Android which is stable and recognized technology.

3.7.4 Operational Feasibility

This phase was used to measure if the proposed tool will solve the problem, and takes advantage of the opportunities recognized during scope definition and if it uses the requirements recognized in the requirements analysis phase of system development. The user interface of FUSI SCANNER is in English language.

3.7.5 Economic Feasibility

The developed app will improve the economic conditions of smallholder farmers and extension officers by detecting the disease occurrence from banana plants on real time also the app is developed to operate offline.

3.7.6 Requirement Analysis

This phase is used to define the user needs and the whole planned framework for client use recognized system characteristics and environments to determine requirements for system functions. Requirements were analysed, restructured, refining the mission statement and environment to uphold the system definition explained in section 4.1 chapter 4.

Fix System Boundaries

The scope of the proposed tool was decided. As the developed tool was able to detect disease from banana plants and its scope is to support farmers in Tanzania.

To Identify the Customers

From requirements analysis, another step is to identify users of the developed software. This tool is designed for smallholder farmers and extension officers so it can support them in early detection disease from banana plants hence control and management can be done properly.

Requirement Gathering

At this stage, the requirement for the tool to be developed was banana leaves images. They have been collected from Arusha and Mbeya region in Tanzania by using a smartphone camera installed with an ODK kit.

Requirement Analysis Process

Structure analysis and modelling of the requirement gathered was done at this stage.

Requirement Specification

This is an initial point for software design. It includes the function of the developed system, performance, user interface, and operational constraints that will assist in system development. Hence the design phase was done and the tool was developed.

3.7.7 Functional Requirements

A functional requirement is defined as the core functions of the system it includes data manipulation, calculations or components of the system. It defines what a system can accomplish. The following are the functional requirements of FUSI SCANNER: the sample size of the dataset used was 9000 images.

- Capturing images.
- Uploading images.
- Performing inference of captured or uploaded images to a deep transfer learning model embedded within the application for classification of the disease.
- Displaying results (Healthy leaves, Fusarium wilt race 1 and Black Sigatoka disease) with the percentage of confidence as shown in Fig. 31 and appendix 7 and appendix 8 respectively.

3.7.8 Non-Functional Requirements of the System

- Reliability: The system intended to function under a certain state without failure.
- Flexibility: Possibility of the system to adapt future changes in its requirements.
- Performance: Response time, throughput and memory usage.
- Availability: In software engineering, availability is the ratio of time a system is required or expected to function.

3.7.9 Use Case Diagrams

Use case methodology is used in system analysis to clarify, identify, as well as to organizing the system requirements. Figure 9 shows the use-case for the FUSI SCANNER application.

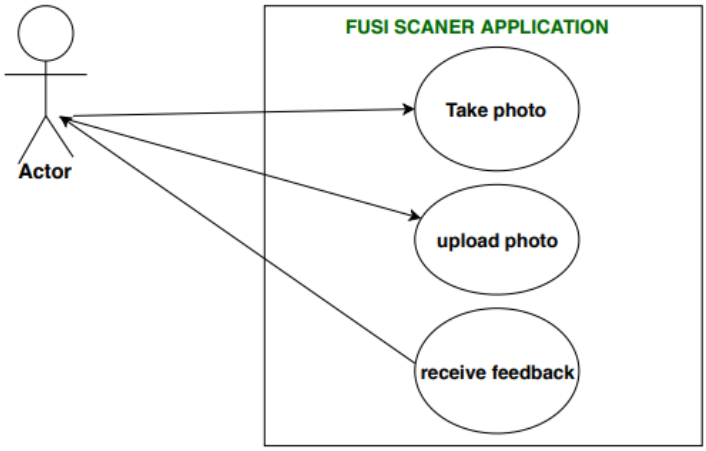


Figure 9: Use case

Above is the use case of the FUSI SCANNER which is a mobile application it shows how the actors interact with the application. This mobile application has two actors which are smallholder farmers and Extension officers.

From the above use case diagram Actor performs the following task in disease detection; FUSI SCANNER application through a developed model and analyzed algorithm will give a detection of banana disease using leaf images depending on the types of leaf image were captured /uploaded by the user.

Take Photo

- Primary actors

Farmers/Extension officer: take photo/capture banana leaf and upload it to the application.

Upload Photo

- Farmers/Extension officer selected banana leaf images stored in the galley and uploading to the application.

Receive Feedback

- Farmers: receive feedback from a mobile application.
- The extension officer receives feedback from a mobile application and provides suggestions to farmers.

3.8 System Design

System design is a process of defining the elements of a system like architecture, modules, components and different interfaces for a system to satisfy the defined requirement. The system design on this research is based on two types from the design phase, the first design is logical design with entity relationship diagrams and the second is a data flow diagram and activity diagram.

3.8.1 Data Flow Diagrams (DFDs)

The data flow diagram in Fig. 10 and Fig. 11 respectively shows the data stored, external entities and processes in the system. It explains where information comes from and where it ends also it shows how part of the system transforms inputs into output. The Fig.10 demonstrate the entire idea of how smallholder farmers/extension officers captured image and undergo the process as per system model to results.

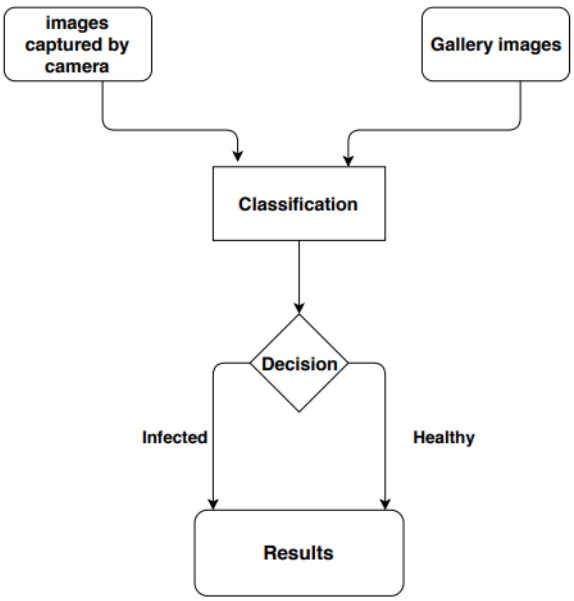


Figure 10: Data flow diagram of the FUSI Scanner mobile application

The data flow diagram in Fig. 11 presents the overall idea of how Extension officers and smallholder farmers they will interact with the application. The function of both users is to open the app then from the home page, the user selected the option to use e.g. camera option/uploading option then capture /upload the image to the application the app processing the input and return the result finally the user view the feedback.

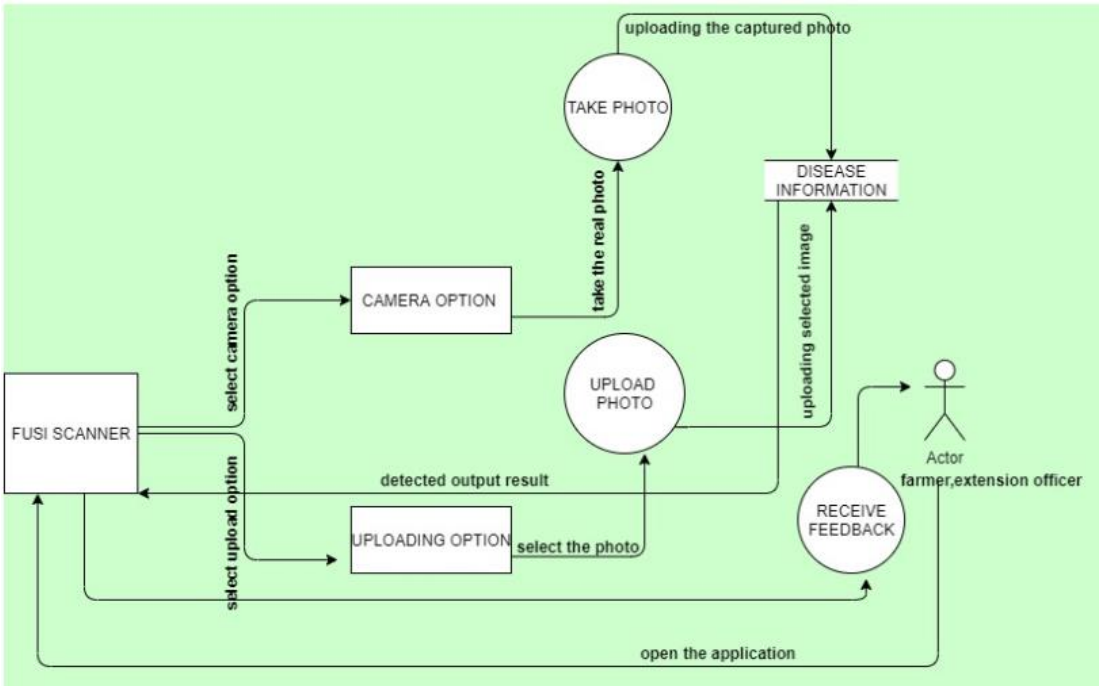


Figure 11: Data flow diagram 2 level of the FUSI Scanner mobile application

3.8.2 Activity Diagram

Figure 12 shows the activity diagram of the users (farmer/extension officers) in disease detection.

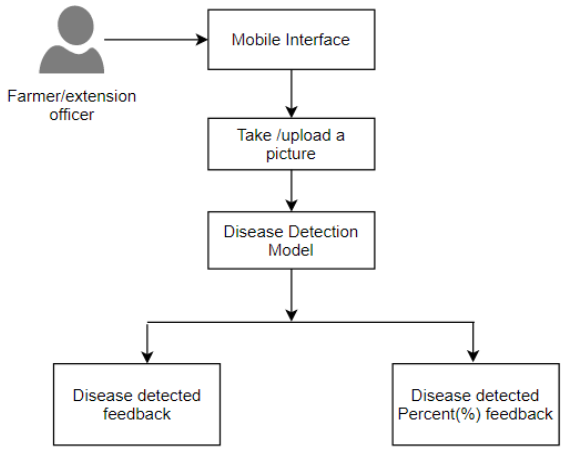


Figure 12: Activity diagram of users in disease detection

3.9 System Implementation

The developed tool smartphone-based application will enable access to information and support farmers for the detection of banana fungal diseases on real time. Figure 13 shows the conceptual design of the mobile application and illustrates how the data moves within the mobile application to the users (smallholder farmers and extension officers). InceptionV3 model was used during the deployment of this system due to its low computation cost and less memory requirement when deployed in mobile devices as compared to four remained trained models. InceptionV3 model was integrated with a mobile phone to facilitate access and detect diseases in real-time for easy control and management of the disease to improve the banana yield.

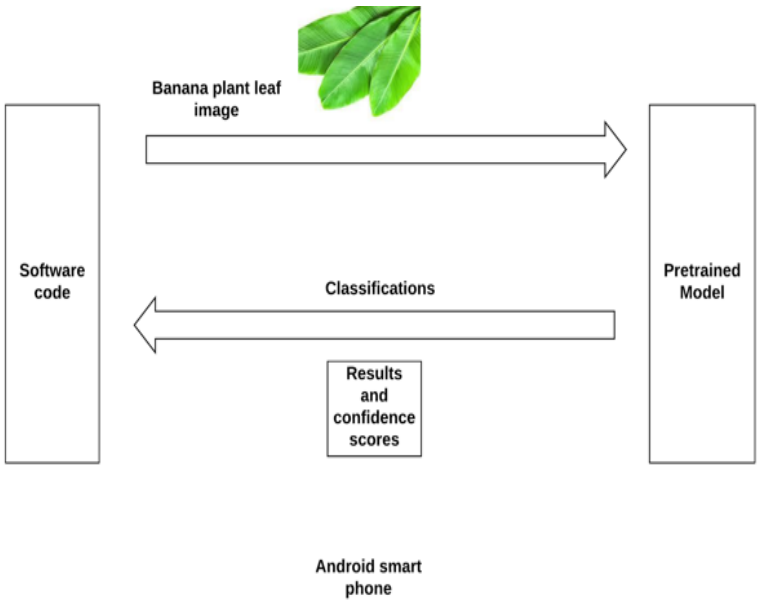


Figure 13: Conceptual design of FUSI SCANNER application

In the development of this smartphone-based application, Python programming language has been used to create the deep learning model. Also, Extensible Markup Language (XML) and Java in the android studio (explained in Chapter 2) was used to create an application interface that gives an easy interactive observation. The application was developed to work in the Android operating system, the globally most popular and open-source operating system developed and owned by Google Company.

3.10 System Interface

The system interface is defined as the whole presentation of the developed system or a means of interaction among the application and user. Figure 29 in section 4.5.1 demonstrates the default interfaces of a smartphone-based application called FUSI SCANNER. Chapter 4 presents the result obtained when testing the developed tool.

CHAPTER FOUR

RESULTS AND DISCUSSION

4.1 Data Collection Results

In this research, the collected dataset contains images with two categories of banana disease namely Fusarium wilt race 1 and Black Sigatoka diseases and one category of healthy banana leaves collected by an expects as shown in Fig. 5 and explained in section 3.1. Table 3 below indicates the total number of images collected from Arusha and Mbeya regions in Tanzania.

The requirement of this research was images of banana leaves, which were collected using mobile phone camera Tecno WX3 installed with ODK kit with the help of three experts from the international institute of tropical agriculture (IITA). All datasets were collected from Arusha and Mbeya regions in Tanzania and stored in a Goggle drive. The sample size of the collected data set was 6353.

Table 3: Images collected from Arusha and Mbeya regions in Tanzania

Images collected	Mbeya region	Arusha region
Black sigatoka	1048	1218
Healthy leaves	1300	1100
Fusarium wilt race 1	1050	637
Total images	3398	2955

According to the nature of the study, the dataset was collected from the banana leaf plants and not the soil. Currently, smallholder farmers still use the traditional method in detecting disease occurrence in their fields. Smallholder farmers, have to make timely decisions for the control and management of their crops. Monitor the field situation is done at least once in a week by checking (plants, weather factors, pests, natural enemies, soil, water etc.). Once the farmers observe the occurrence of the disease in their field farmers they take direct action that is needed such as, remove infested plants, and remove infected banana leaves and collecting egg masses.

4.2 Augmented Dataset Results

From section 3.3.3 our dataset collected from both regions was augmented and we get a total of 9000 datasets. Figure 14 shows the distribution image from each class to different set of training, validation and testing.

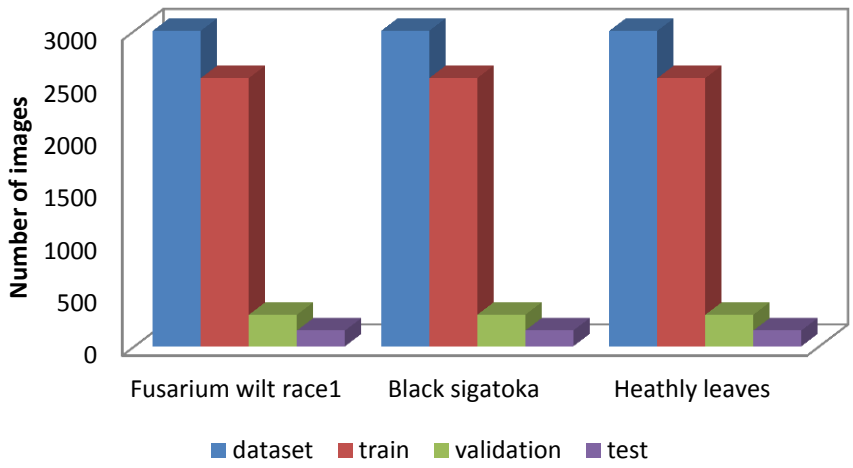


Figure 14: Dataset division before and after splitting

4.3 Models Results

4.3.1 Training VGG16 Model on 85:10:5 Dataset Results

Hyper-parameters summarized in Table 4 were selected and used to train VGG16 model. The first parameters set were 20 epochs and a batch size of 16. Dataset was divided into this ratio for training 85%, for validation10% and test set 5%. During training, we found that 8121 images belong to 3 classes and 927 images belonging to 3 classes.

Table 4: Hyper-parameter used in training VGG16 model

Parameter	Value
Optimizer	Adam
Batch-size	16
Epoch	20
Metrics	Accuracy
Model Architecture	VGG16
Images size	244*244 pixel

4.3.2 Accuracy Graph Results 1 from VGG16 Model

VGG16 model was trained with these hype-parameters 20 epochs and a batch size of 16. However, during the training process at Epoch 15 the validation accuracy was 94.40% and train accuracy was 90.72% and loss values are shown in Fig. 15 and Fig. 16 respectively. However, at 20 epochs we have found that the validation accuracy decreased to 93.85% and train accuracy increased to 92.15%. Due to these results, our model was not able to generalize well during training due to a large gap between train and validation dataset.

Figure 15: Model accuracy results for VGG16

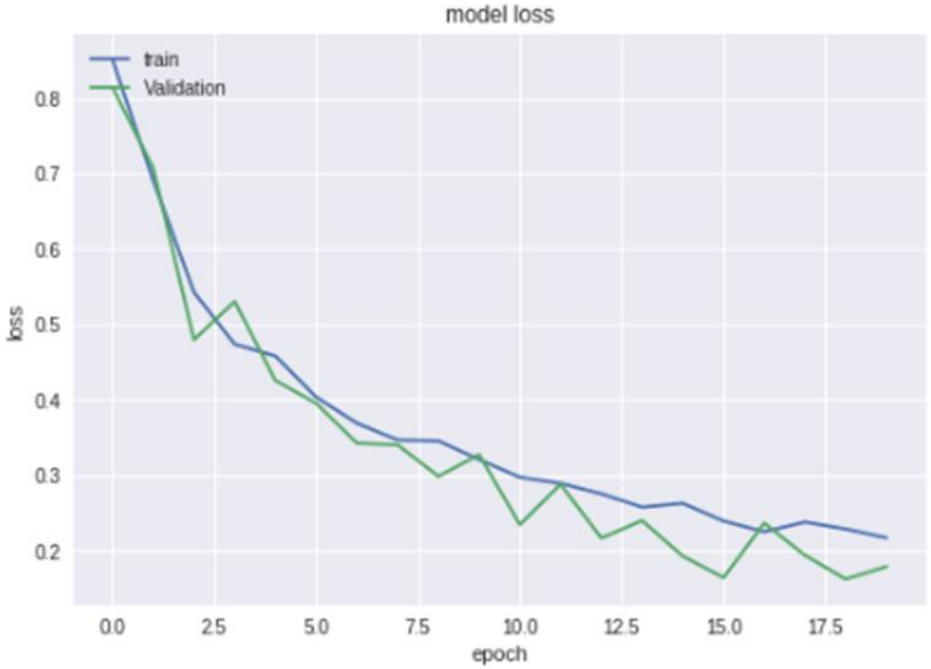


Figure 16: Model loss results for VGG16

4.3.3 Accuracy Graph Results 2 for VGG16 Model

Then we adjust our hyper-parameter to 35 epochs and a batch size of 16. However during the training process at Epoch 25 the validation accuracy was 97.37% and train accuracy was 93.95%

and loss values are shown in Fig. 17 and Fig. 18 respectively. However, at 35 epochs we have found that the validation accuracy decreased to 96.05% and train accuracy increased to 95.56%. Due to these results, our model was not able to generalize well during training due to a large gap between train and validation dataset.

Figure 17: Model accuracy results for VGG16

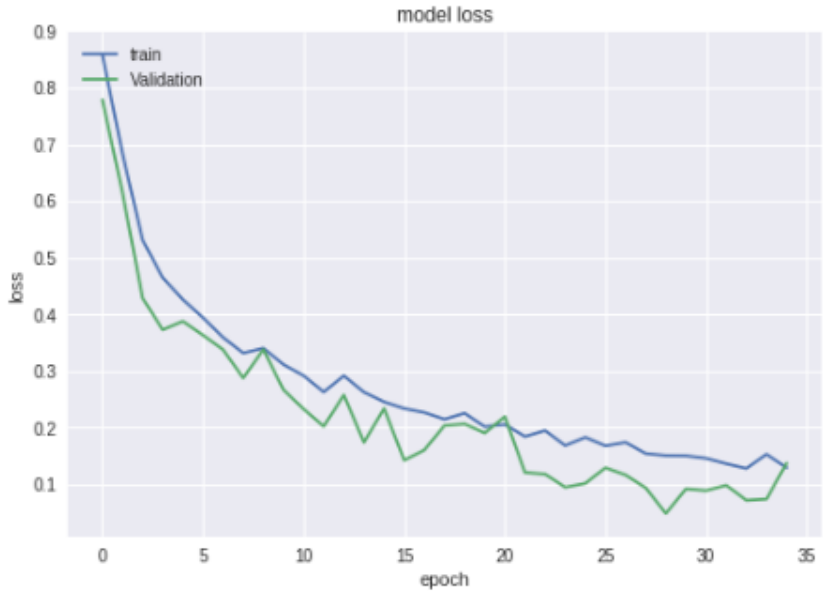


Figure 18: Model loss results for VGG16

4.3.4 Accuracy Graph Results 3 for VGG16 Model

Finally, we adjust our hyper-parameters with 30 epochs and a batch size of 15. However, during the training process at Epoch 23 the validation accuracy was 95.39% and train accuracy was

92.63% and loss values are shown in Fig. 19 and Fig. 20 respectively. However, at 30 epochs we have found that the validation accuracy continued to increase to 97.26% and train accuracy increased to 94.82%. Due to these results, our model was able to generalize well during training hence there is no gap between training and validation datasets.

Figure 19: Model accuracy results for VGG16

Figure 20: Model loss results for VGG16

4.3.5 Training ResNet Models on 85:10:5 Dataset Results

The hyper-parameters used to train Resnet models are summarized in Table 5. Since the problem at hand is multi-class classification, the first parameter to set was 50 epochs and batch size of 4, However, Resnet models succeeded to maintain the accuracy performance and loss value at epoch 50 was 0.0842 and accuracy was 98.4 for Resnet18, Resnet50 achieved an

accuracy of 98.8%, Resnet152 achieved an accuracy of 99.2%. The objective of the Resnet models was to increase true positives and decrease false negatives. The objective function used was Cross-Entropy Loss as defined in equation 2 whose value rises as the predicted probability deviates from the real label.

Equation 2: Cross-Entropy Loss

$$L_O(y,p) = \sum_{i=1}^{M}(y_i.logp_i)(1)$$

Where: M represents the number of classes (Black Sigatoka, Fusarium wilt, healthy leaves), y represents the true label and p represents the predicted probability for the observed image used in confusion matrix results.

Table 5: Hyper-parameter used in training Resnet models

Parameter	Value
No-epoch	50, 100, 1500
Optimizer	SGD
Batch-size	4
Learning rate	1e-3
Weight Decay	1e-9
Model Architecture	Resnet 18,50,152
Momentum	0.9
Images size	244*244 pixel

To optimize the above objective function, stochastic gradient descent (SGD) optimizer was used after searching for the best optimizer from amongst many optimization techniques. In each experiment for Resnet models, training was done for changing the number of epochs and saving the best model only. The input image was normalized using z-score normalization with standard deviation = [0:229; 0:224; 0:225] and mean = [0:485; 0:456; 0:406].

4.3.6 Training InceptionV3 Model on 85:10:5 Dataset Results

Hyper-parameters used to train InceptionV3 model are summarized on Table 6. The training phase for TensorFlow with InceptionV3 was the shortest method as illustrated in Fig. 8. The best accuracy for InceptionV3 model was 96.61%. Tensor Board was used as another way of checking the training history, during and after training. Tensor Board made it possible to see visually with graphs how the training history was gone as shown in Fig. 21 to Fig. 23.

Table 6: Hyper -parameter used in training inception V3 model

Parameter	Value
No-epoch	500,1000,1500
Smoothing	0.6 and 0.75
Model Architecture	InceptionV3
Images size	299*299 pixel

4.3.7 InceptionV3 model Results 1

InceptionV3 model was trained with epoch 500; however, during the training process at Epoch 390 the validation accuracy was 96.61% and train accuracy was 92.72% as shown in Fig. 21.

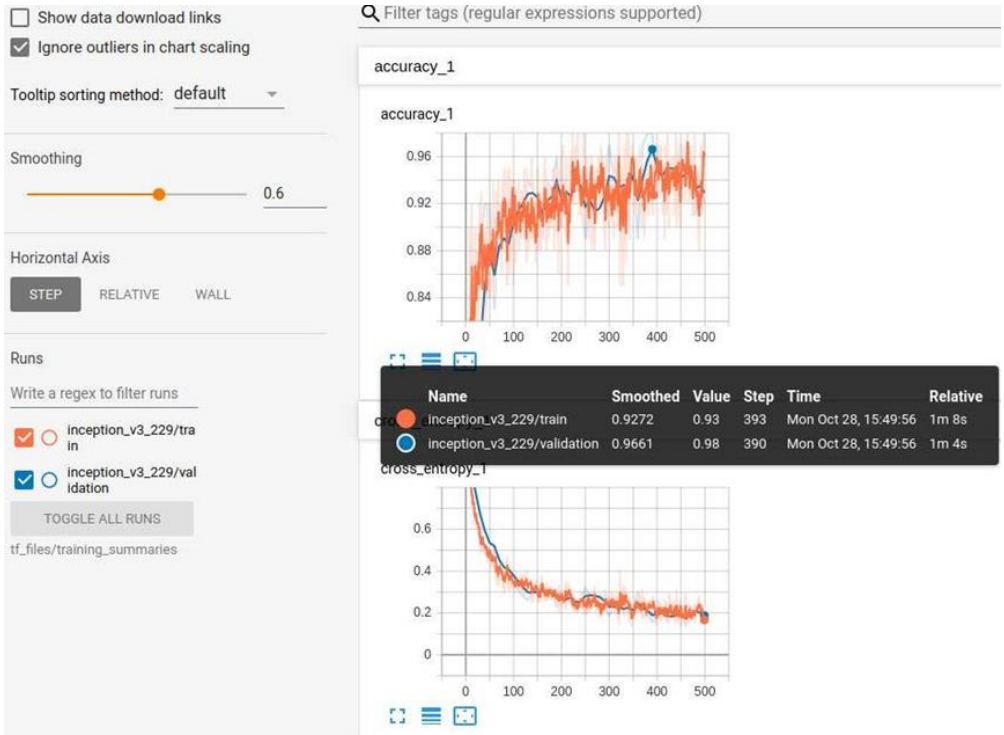


Figure 21: InceptionV3 training and validation accuracy results 1

At epoch 500 the validation accuracy was 92.92% and train accuracy was 96.32%. This means that the model will start to degrade by having higher training error as well as testing error. While loss value obtained during training was 0.1957 for validation results.

4.3.8 InceptionV3 Model Results 2

Parameters were adjusted to epoch 1500 from Fig. 22 InceptionV3 achieve an accuracy of 95.41% for the validation set and accuracy of 95.541% for the training set. This indicates that

InceptionV3 model generalized well during training and even loss value has decreased as shown from Fig. 23.

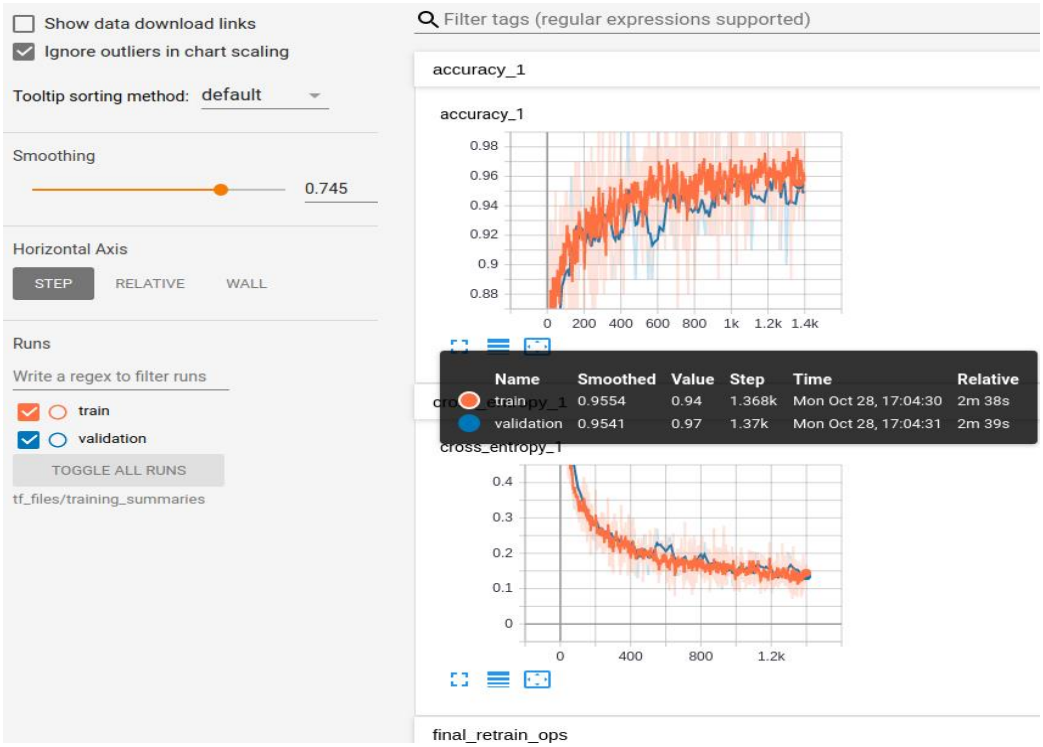


Figure 22: InceptionV3 training and validation accuracy results 2

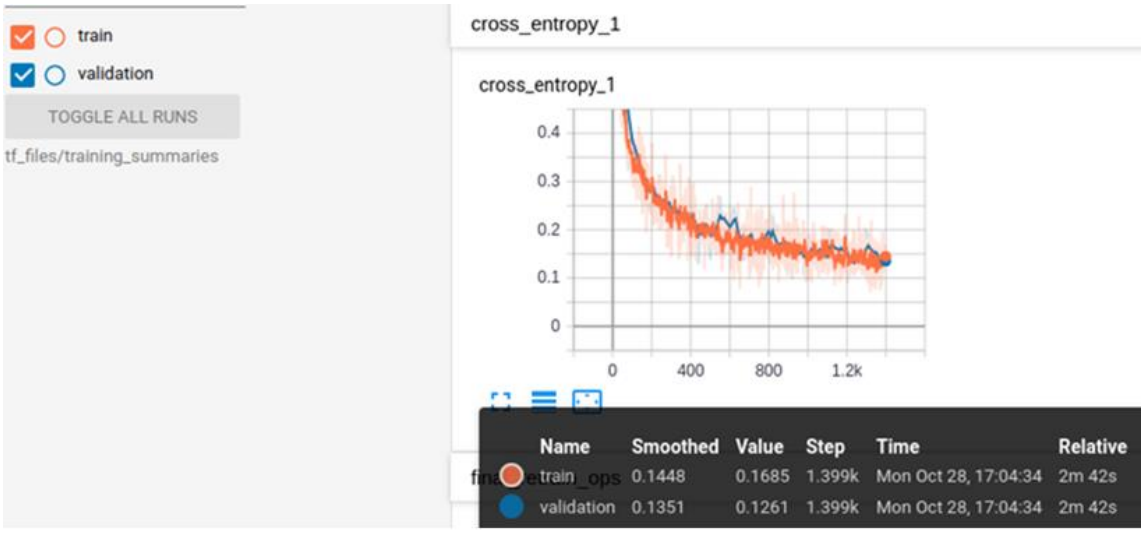


Figure 23: InceptionV3 training and validation loss results 3

4.4 Models Performance Results

The objective/ the evaluation of the models were accomplished by conducting performance analysis and the results are presented in Table 7. The five CNN architecture presented in section

2.5 were trained and models results are presented in section 4.3. Datasets collected were trained with the selected convolutional neural network models and allow them to extract and learn features from the input data. With the validation set, we were able to evaluate the training process and determine how well it will fit the new data by using tensor board. Validation set and Testing set results was used to assess the models for disease detection and results were plotted by using confusion matrix presented in section 4.4.2. Also, classification accuracy was defined in Equation 1 and confusion matrix explained in section 3.6.1 was used.

4.4.1 Classification Accuracy Results

Five models were selected and trained are VGG16, Resnet18, Resnet50, Resnet152 and InceptionV3 deep learning architectures. After the training process was done the model was tested with on test set images. Results for the first set of 70:20:10 was Resnet152 achieved an accuracy of 92.6%, Resnet50 achieved an accuracy of 89.7%, VGG16 achieved an accuracy of 86.7%, InceptionV3 achieved an accuracy of 89.9% and Resnet18 achieved an accuracy of 87.9%.

Moreover, after observing the result from the first set we decided to add more training samples to the training dataset and reduce data from the validation and test set to check if our models can give the best results (generalize well) compared to the first results. After training the second set of 85:10:5 the results was Resnet152 attained an accuracy of 99.2%, Resnet50 attained an accuracy of 98.8%, VGG16 attained an accuracy of 97.26%, Resnet18 attained an accuracy of 98.4%, and InceptionV3 attained an accuracy of 95.41%. Finally, we saved only models that succeeded to generalize. To achieve greater classification accuracy, from the five models trained the complexity of the models was considered as a significant factor in selecting the best model architecture from the training datasets. Since to bring out more features from the trained leaf images, it is associated with the complexity of the model, Resnet152 meets this objective as it achieved an accuracy of 99.2% as presented in Table 7.

Table 7: Performance of the model's architectures

Models	Validation accuracy%	Test accuracy%	Epoch	Time (s/epoch)	Loss
VGG16	98.4	98.7	35	30minsec13	0.197
Resnet18	98.8	98.8	50	08min sec 35	0.256
Resnet50	98.8	98.9	50	08min sec 35	O.180
Resnet152	99.2	99.8	50	08min sec 35	0.0539
Inception v3	95.41	95.5	1500	02min sec 39	0.1351

4.4.2 Confusion Matrix Results

In the field of Deep learning, another method used to check the statistical classification of the trained modes is called confusion matrix (CM). The method was used to control the problem of statistical classification also it is called error matrix. CM is in form of a table that shows the visualization of the performance of the model. Metric used was: False Negative=FN, False

Positive=FP, True Positive=TP, True Negative=TN. These values were defined in Eqn 2 depend on the results from validation and test set datasets, confusion matric was generated for all models. A confusion matrix was created for the validation set with 309 images for Black Sigatoka, 309 images for Fusarium Wilt race 1 and 309 images for Healthy and test set the confusion matrix contain 150 images for Black Sigatoka, 150 images for Fusarium Wilt race 1 and 150 images for Healthy. Both validation and test set results were considered while evaluating model's performance, predicting unseen data and visualize the predicted results by using confusion matrix. Each confusion matrix shows the accuracy per disease by using true label and the predicted label representation of the class where the model is confused or misclassified. Confusion matrix plot result for VGG16 indicated that the model was capable to detect diseases of the class from test data with great accuracy while on validation the model confuses more between black sigatoka and healthy leaves show on Fig. 24.

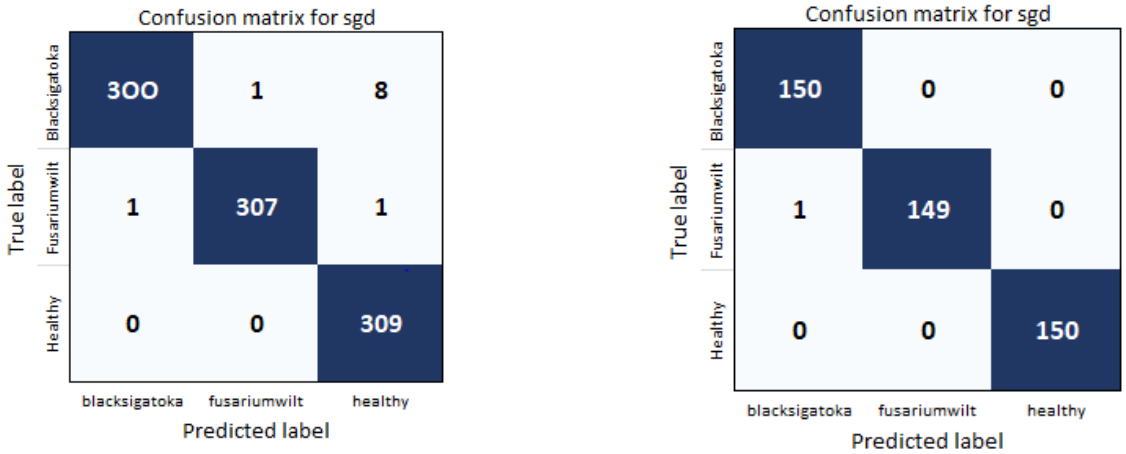


Figure 24: (a) Validation confusion matrix for VGG16 (b) Test confusion matrix for VGG16

Confusion matrix plot result for Resnet18 show that the model succeeded to detect diseases of the class from test data with great accuracy while on the validation set the model confuses more with 11 images between black sigatoka and healthy leaves show on Fig. 25.

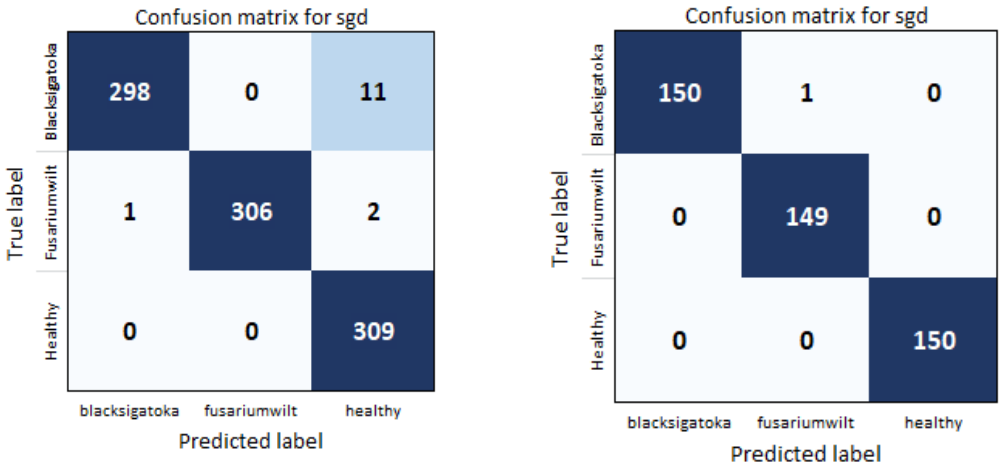


Figure 25: (a) Validation confusion matrix for Resnet18 (b) Test confusion matrix for Resnet18

Confusion matrix plot result for Resnet50 show that the model succeeded to detect diseases of the class from the test set with great accuracy and validation set. The model confuses only with few images between the classes shown on Fig. 26.

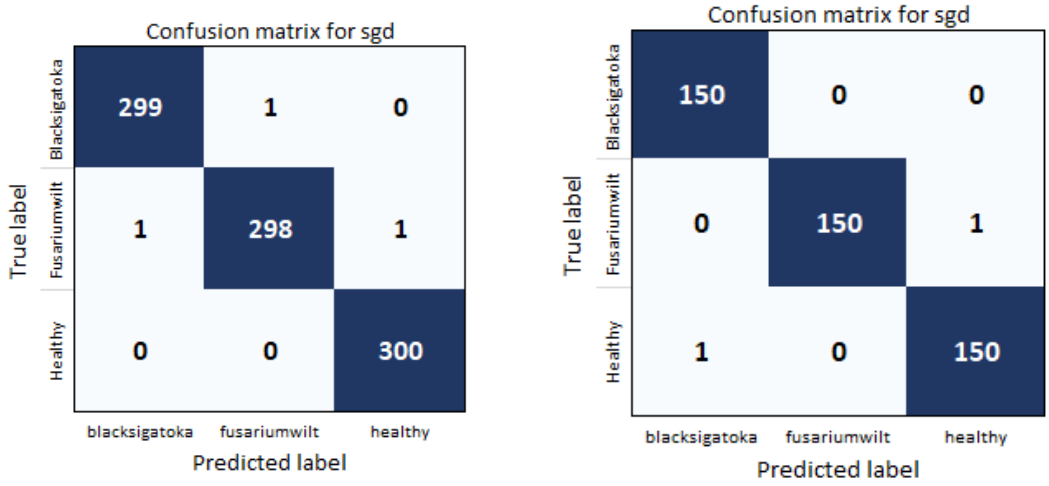


Figure 26: (a) Validation confusion matrix for Resnet50 (b) Test confusion matrix Resnet50

Confusion matrix plot result for Resnet152 show that the model succeeded to detect diseases of the class from the test set with great accuracy. While on validation set the model confuses only with few images between the classes shown on Fig. 27.

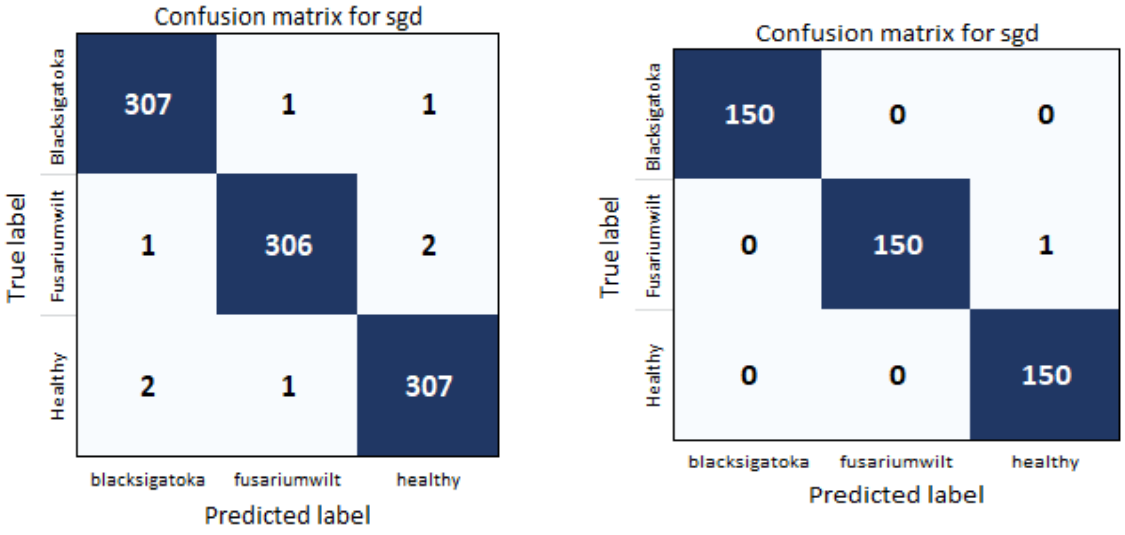


Figure 27: (a) Validation confusion matrix for Resnet152 (b) Test confusion matrix Resnet152

Confusion matrix plot result for InceptionV3 show that the model succeeded to detect diseases of the class from test data with great accuracy. While on validation set the model confuses only with few images between the classes shown on Fig. 28.

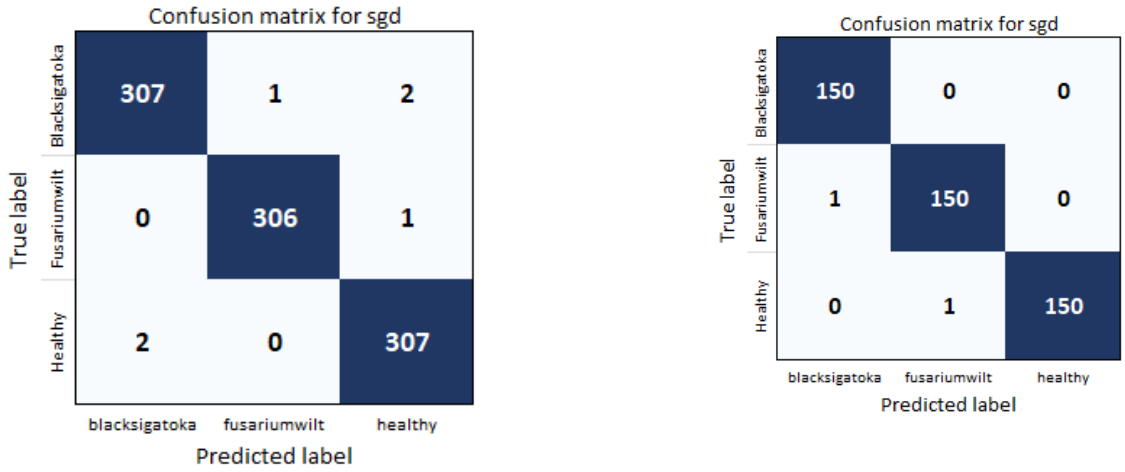


Figure 28: (a) Validation confusion matrix for InceptionV3 (b) Test confusion matrix for InceptionV3

4.5 Mobile Deployment Results

Objective 3 was accomplished by developing a mobile system and the results are presented in the section below.

4.5.1 Activities of the user Mobile Application Results

This part explains how the developed smartphone-based application was working in disease detection. The interface of the application is user-friendly to allow interaction between smallholder farmers/extension officers. Figure 29 shows home page interfaces of the FUSI SCANNER app for Farmer and extension to interact with the application. The interface consists of two buttons the first button is the camera option, this camera will allow the farmer to take picture of the banana leaf and upload it to the application, the application will process the uploaded image and return feedback to the farmer. The second button is the uploading option this option is used to upload images to the application stored in the gallery, then the app will process them and return feedback to the farmer as well.

The first step is for the farmer to open the app from his/her smartphone then clicks the application and the application is opened, the farmer takes a photo of the leaf upload it to the application, then the App will process the uploaded picture and give feedback if the captured leaf has a disease or it is healthy. This process takes at least three minutes to give results shown on Fig. 30. The developed application was designed to operate offline. This method will reduce the cost of buying the internet because some of the smallholder farmers/extension officers may not afford to buy the internet whenever they visit their farmer to check for banana diseases occurrence.

Figure 29: FUSI SCANNER APP disease detection interface

The tool was verified to check if it is working as expected by smallholder farmers and extension officers who live in Arusha and Mbeya region, Tanzania.

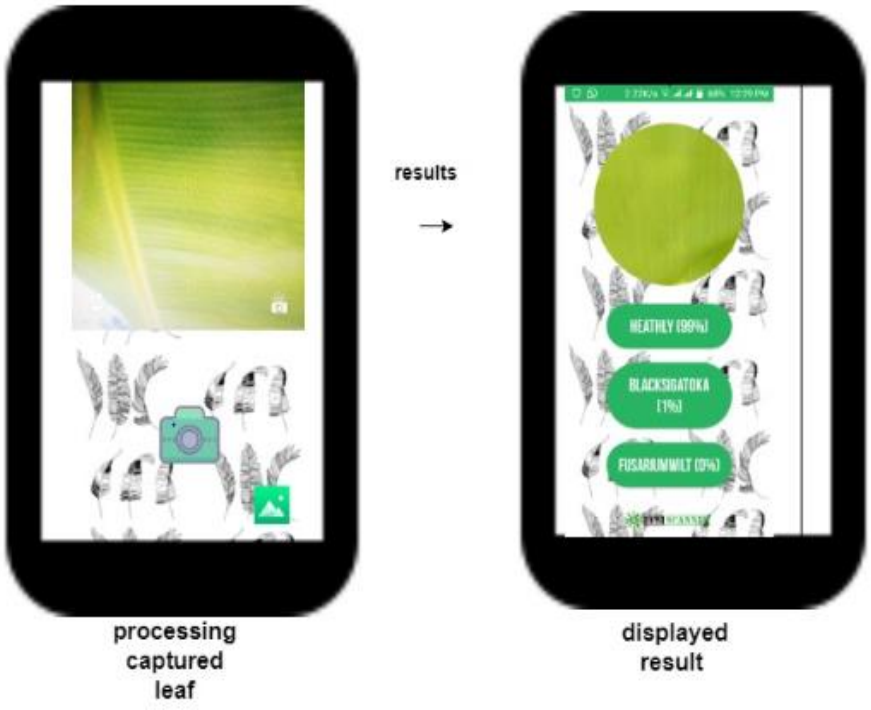


Figure 30: Implementation of FUSI SCANNER app and its results

4.5.2 Disease Detection and Disease Details Interface Results

Figure 31 shows the disease detection and details interface results. Disease detection and information interface provide disease details (the type of disease detected and it's percentage). In this mobile app, the detected disease is predicted with a percentage confidence of more than 70%. The confidence measure was included to provide the best results during the detection process, if the result is below the target e.g. 50% the voice message will be prompt to alert for a clear picture.

Figure 31: Screenshots result of the FUSI SCANNER app for healthy banana leaves captured from the real environment

4.5.3 Source Code Screenshot Taken in an Android Studio for Mobile Deployment

Figure 32 shows the output results of InceptionV3 model that was deployed into Android smartphone. The private static final output is the final result from the InceptionV3 model with

the graph and label called graph.pb and labes.txt respectively. The graph file contained a trained dataset while the labels file contains labels of the dataset trained.

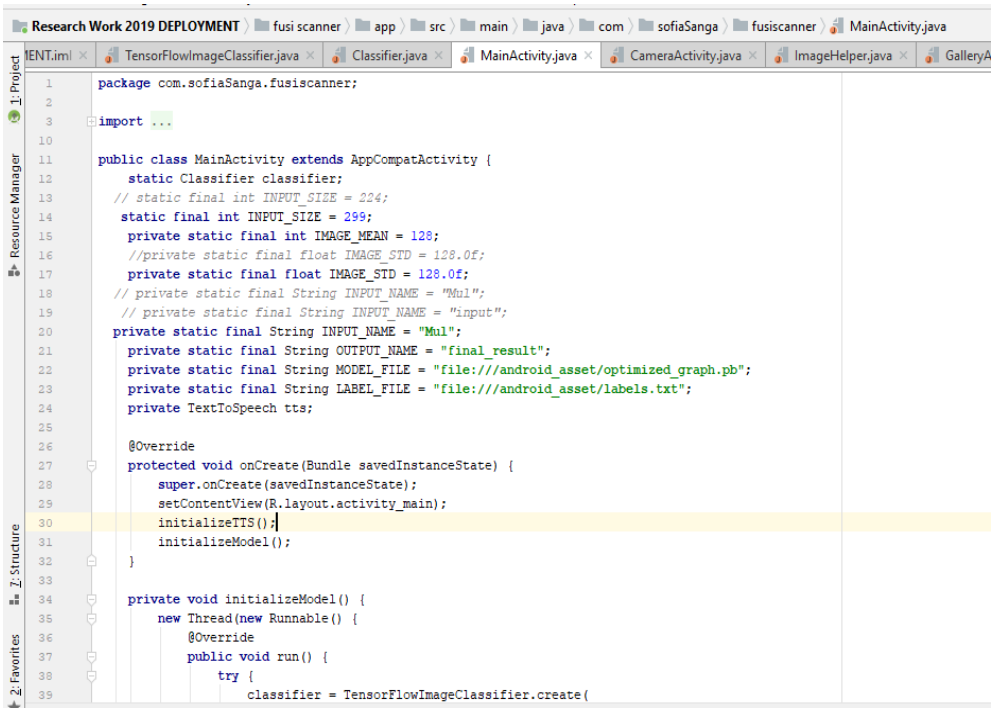


Figure 32: Screenshots result source code for the main activity in Android studio

Figure 33 illustrates the Android source code for the output results that were displayed by the FUSI SCANNER for detecting banana disease. The code in green color indicates the text and voice displayed in the app. However, if the input image is unrecognizable then the application responds with the voice, "That image seems unrecognizable maybe; I have to update my database". If the application understands the input image the result indicates the level of confidence of the disease detected.

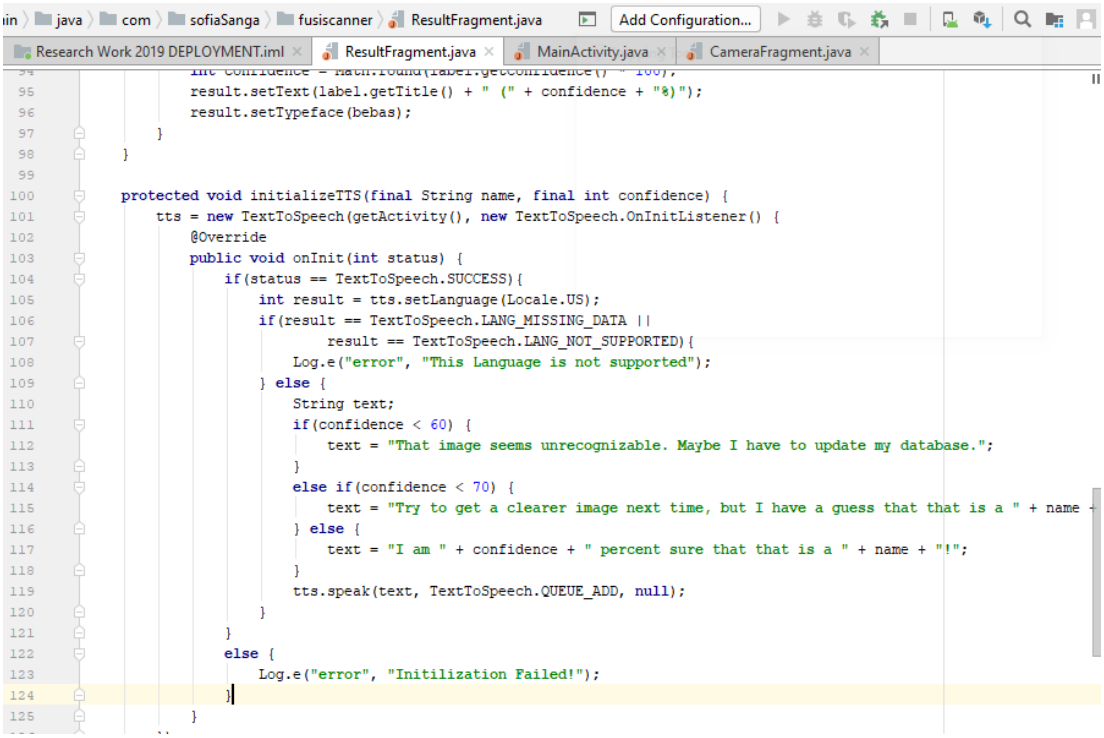


Figure 33: Screenshots result source code for Result Activity in Android studio

4.6 User Acceptance Testing Results

A survey was conducted with extension officers and smallholder farmers in Mbeya and Arusha region, Tanzania to test the tool. Participants completed the survey by giving comments from the provided questionnaire. Through the questionnaire, evaluation data were collected and the developed tool were given to Extension officers and Smallholder banana farmers they were able to open the app and test to detect the banana disease and were able to get the result of the captured image and understand the results returned by the tool. Hence the mentioned user strongly agrees that the application is interactive and user-friendly with an overall 76.47% and 63.89% respectively they like the application. Table 8 and Table 10 respectively represent usability testing results for extension officers and Smallholder banana farmers after testing the app.

As well as for acceptance testing Extension officers and Smallholder banana farmers accepted that the tool is useful and it will help in detecting black sigatoka and Fusarium wilt race1 diseases with 88.24% and91.67%. Table 9 and Table 11 respectively represent acceptance testing results for extension officers and Smallholder banana farmers after accepting the app. Smallholder banana farmers and extension officers strongly agree with the usefulness of the developed smartphone-based app called FUSI SCANNER.

4.6.1 Agricultural Extension Officers Results

From the study conducted on the testing of this tool smartphone-based app, we found that the majority of the extension officers agreed that the system is useful and of great help towards the detection of the two commonly banana fungal diseases namely black Sigatoka and Fusarium wilt race 1. Seventeen (17) extension officers (8 in Arusha and 7 in Mbeya) responded to the questionnaire, their responses are dissipated in Table 8 and Table 9.

4.6.2 Farmers Results

Thirty-six (36) farmers (from both regions) responded to the questionnaire for usability testing and acceptance testing. Their results are summarized in Table 10 and Table 11 in the appendices.

4.7 Discussion

In this research, we have discovered that the majority of smallholder farmers and extension officers rely on traditional methods to identify and detect disease from banana plants. The production of bananas crop has declined since the 1970s, and now yields a fraction of its potential due to diseases (Ssali *et al.,* 2017). Banana is a cash crop and staple food for people who live in Arusha and Mbeya region. However, disease occurrence in these plants results in banana production losses which significantly cause a reduction in household food security and incomes (Nkuba *et al.,* 2015). Early detection, identification, and control of diseases from plant leaf is a big challenge to many smallholder farmers as well as extension officers (Dyrmann *et al.,* 2016; Patil & Pawar, 2017; Ramcharan *et al.,* 2017), due to lack of necessary tools that will help to prevent the possible outbreak of pests and diseases on real-time (Jagan *et al.,* 2016; Rumpf *et al.,* 2010). However, recent growth of deep learning and transfer learning techniques has shown a possible way to detect diseases on plants. Computer vision systems in agriculture provide useful information in real-time and it offers substantial information about nature, reduces costs and attributes of the product (Amara *et al.,* 2017). Moreover, deep learning technology has shown a successful way for smartphones to assist in disease detection based on leaf image(Eli-chukwu, 2019; Ramcharan *et al.,* 2019).

The literature review indicated that research on crop disease detection mainly ends up on model development and not end-user tools for farmers. Another shortcoming we have found that a lot of researchers used dataset collected from Plant village while others used the dataset

downloaded from the internet. This research work focused on the real-life dataset and developed a tool for end-users i.e. smallholder banana farmers and extension officers. The developed system works in real-world conditions.

The performance of the five models tested in this research was compared using the accuracy and confusion matrix. From the two data set that we have first set of 70:20:10 and the second set of 85: 10: 5, the data set with 85: 10: 5 shows that our models succeeded to generalize well in disease detection. However, Resnet152 was able to meet our objective as it was able to detect disease from the trained dataset and archived an accuracy of 99.2% but for mobile deployment, Resnet152 is a complex model and it has a higher computational cost and requires large memory space. Hence due to this reason, we decide to use the InceptionV3 model (Szegedy *et al.*, 2016, 2017) because of its low computation cost and it takes less memory (Meng, Sun, Yang, Qiu & Gu, 2017) when deployed to a mobile application. However, it is always a competition between simple architecture and complex especially when thinking about mobile app.

CHAPTER FIVE

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

A study conducted by Ramadhani (2017), reveals that almost 94% of banana farmers in Tanzanian specifically at the Arumeru district in the Arusha region responded on the presence of banana fungal diseases in their plantations. Also in Mbeya, it was observed to 80% of respondents had a similar response. Diseases detection has been a challenge to both farmers as most of them still rely on traditional knowledge to detect those diseases.

Studies reveal several efforts that have been made (Ramadhani *et al.,* 2017; Shimwela *et al.,* 2016). With their contributions, still, the impact is not that much in terms of increasing yields and economy of farms and country in general. In our study area, a study by Ramadhani (2017) proposed a tool that is much based on weather to predict results. This had some challenges, especially when considering the internet and weather data that sometimes are difficult to be access in rural areas. Thus, to address this challenge and others, deep learning and transfer learning from machine learning technique was proposed and applied on mobile phones to develop a smartphone-based application called FUSI SCANNER. The developed tool depends on the datasets (leaf image that is captured by the farmer/extension officer). The tool smartphone-based app uses a camera to capture leaf image that once captured it processes with the algorithm trained and return the result as per image that was captured with percentage confidence. So, this application has provided an easy way to support smallholder farmers and extension officers in the detection of banana fungal diseases.

Chapter two explains different literature reviews that describe several issues concerning banana fungal disease management and machine learning techniques, their implementations, and how they provide a reasonable solution. In chapter three the formulation and training of the model were done. With several formulas and several data sets, stable values were obtained that were used in mobile development. Chapter four presents results, for the dataset collected, models selected to train the dataset, models results, a full design, and implementation of the tool (FUSI). Finally, in chapter five conclusions, limitations, and future work are presented, these conclude according to this study and the current situation that banana farmers are facing in the early detection of the two banana fungal diseases namely black Sigatoka and Fusarium wilt race 1. Hence, findings explained in this dissertation might be limited to the environment and participants. Further studies are needed to address other issues that came short in this study.

Computer vision approaches have reported being used for automatic crop disease detection and classification, but still, the exploration of real-time detection and classification is lagging behind. In this research deep convolution neural network and transfer learning technology was used to automatically detect diseases from banana plants by using leaf images captured from the real field. The developed tool provides practical and applicable ways for detection of the diseases from the captured leaf with the class of the diseases and percentage confidence, which represents a big difference from other tools developed for banana diseases detection. The developed smartphone-based application was able to detect diseases from banana leaf and give feedback on real-time according to the disease detected. The developed smartphone-based application archived an accuracy of 99% confidence of the captures leaf from the real environment.

In this research we conclude that early detection and recognition of the diseases is very important, hence the control and management of banana fungal diseases to be done early for the improvement of banana yields. FUSI SCANNER was being tested in Arusha and Mbeya region in Tanzania and smallholder farmers and extensions officers agreed that the tool is very useful in the detection of black Sigatoka and Fusarium wilt race 1 diseases. With a critical investigation of the requirements survey, datasets collected, literature review, direct observation, and agricultural expert opinions from the international institute of Tropical Agriculture (IITA) a mobile application (FUSI SCANNER) with android technology to detect the presence of two mainly banana fungal diseases was developed. Now, with reference to previously banana management tools, the functionality of the proposed detection smartphone-based application (FUSI SCANNER), the tool is thought to be able to support the banana grower to better control and manage banana fungi diseases incidents easily. The developed tool smartphone-based application from this research can be easily transferred to other mandatory crops.

5.2 Limitation

This research develops an early detection tool which is a smartphone-based application for banana disease and the tool is limited to detect disease for banana leaf only.

5.3 Future Work

The addition of content on the mobile application to educate farmers on the control and management of banana diseases is recommended. This could be used to provide awareness to farmers and extension officers and people who are involved in banana cultivation. Furthermore,

dissemination on the usage of the tool by training farmers and extension officers in the detection of diseases from banana crops.

This research study can be extended to other crops such as potato, bean, tomato cassava, sweet potato and maize, for which diseases can be detected using leaf images. The work will have a valuable impact on food security in the developing country.

Moreover, suggestions that different initiatives targeting farmers and extension officers on how to increase yield and control several diseases have to be put in place. Also, awareness of the impact of ICT in agriculture and the need to adapt it for both farmers and extension officers for more production and disease management should be infer sized. Awareness information among banana smallholder farmer's communities with respect to different banana diseases (not only Fusarium wilt race 1 and black Sigatoka) should be given the highest priority. Last but not least we encourage farmers and extension officers to be active players in matters regarding those projects/initiatives as it is there to benefit them. The study recommended encouraging farmers in stabilizing and expanding their banana crop production.

REFERENCES

Al Hiary, H., Bani Ahmad, S., Reyalat, M., Braik, M., & AlRahamneh, Z. (2011). Fast and Accurate Detection and Classification of Plant Diseases. *International Journal of Computer Applications*, *17*(1), 31–38. https://doi.org/10.5120/2183-2754

Amara, J., Bouaziz, B., & Algergawy, A. (2017). A Deep Learning-based Approach for Banana Leaf Diseases Classification. *In BTW (Workshops)*, 79–88.

Andersen, C. (2019). *Dermoscopy Images Using Deep Learning*. (June).

Bankar, J., & Gavai, N. R. (2018). *Convolutional Neural Network based Inception v3 Model for Animal Classification*. 142–146. https://doi.org/10.17148/IJARCCE.2018.7529

Bayar, B., & Stamm, M. C. (2016). *A Deep Learning Approach To Universal Image Manipulation Detection Using A New Convolutional Layer*. 5–10.

Brahimi, M., Arsenovic, M., Laraba, S., & Sladojevic, S. (2018). *Deep Learning for Plant Diseases : Detection and Saliency Map Visualisation Deep Learning For Plant Diseases : Detection and Saliency map Visualization*. (June).

Deltour, P., França, S. C., Liparini Pereira, O., Cardoso, I., De Neve, S., Debode, J., & Höfte, M. (2017). Disease suppressiveness to Fusarium wilt of banana in an agroforestry system: Influence of soil characteristics and plant community. *Agriculture, Ecosystems and Environment*, *239*, 173–181. https://doi.org/10.1016/j.agee.2017.01.018

Deng, Y. (2019). *Deep learning on mobile devices: a review*. (April), 11. https://doi.org/10.1117/12.2518469

Dyrmann, M., Karstoft, H., & Midtiby, H. S. (2016). Plant species classification using deep convolutional neural network. *Biosystems Engineering*, *151*. https://doi.org/10.1016/j.biosystemseng.2016.08.024

Eli-chukwu, N. C. (2019). *Applications of Artificial Intelligence in Agriculture : A Review*. *9*(4), 4377–4383.

Etebu, E., & Young-harry, W. (2014). *Control of black Sigatoka disease : Challenges and prospects Control of black Sigatoka disease : Challenges and prospects*. (June).

Fuentes, A., Yoon, S., Kim, S. C., & Park, D. S. (2017). A robust deep-learning-based detector

for real-time tomato plant diseases and pests recognition. *Sensors (Switzerland)*, *17*(9). https://doi.org/10.3390/s17092022

Region. (2017). *Banana Pests and Diseases Field Guide for Disease Diagnostics and Data Collection Great Lakes Region of Africa*.

Gajanan, D. E., Shankar, G. G., & Keshav, G. V. (2018). *Android Based Plant Disease Identification System Using Feature Extraction Technique*. 861–864.

Gallez, A., Runyoro, G., Mbehoma, C. B., Van den Houwe, I., & Swennen, R. (2004). Rapid mass propagation and diffusion of new banana varieties among small-scale farmers in north Western Tanzania. *African Crop Science Journal*, 12, 7–17.

Ganry, J., Fouré, E., De Lapeyre de Bellaire, L., & Lescot, T. (2012). An integrated approach to control the Black leaf streak disease (BLSD) of bananas, while reducing fungicide use and environmental impact. *Dhanasekaran, D. Thajuddin, N. Panneerselvam, A. Fungicides for Plant and Animal Diseases*, 193–226.

Gavai, N. R., Jakhade, Y. A., Tribhuvan, S. A., & Bhattad, R. (2017). *MobileNets for Flower Classification using TensorFlow*. 154–158.

Girshick, R., Donahue, J., Darrell, T., Berkeley, U. C., & Malik, J. (2012). *Rich feature hierarchies for accurate object detection and semantic segmentation*. 2–9.

Gutierrez-Monsalve, J. A., Mosquera, S., González-Jaramillo, L. M., Mira, J. J., & Villegas-Escobar, V. (2015). Effective control of black Sigatoka disease using a microbial fungicide based on Bacillus subtilis EA-CB0015 culture. *Biological Control*, *87*, 39–46. https://doi.org/10.1016/j.biocontrol.2015.04.012

He, K., Zhang, X., Ren, S., & Sun, J. (2015). *Deep Residual Learning for Image Recognition*. Retrieved from http://arxiv.org/abs/1512.03385

Hwan, Y., Joon, S., Hyeon, Y., Hee, J., & Han, D. (2014). Crop Pests Prediction Method Using Regression and Machine Learning Technology : Survey Crop Pests Prediction Method using Regression and Machine Learning Technology : Survey. *IERI Procedia*, *6*(December), 52–56. https://doi.org/10.1016/j.ieri.2014.03.009

Jagan, K., Balasubramanian, M., & Palanivel, S. (2016). Detection and Recognition of Diseases

from Paddy Plant Leaf Images. *International Journal of Computer Applications*, *144*(12), 34–41. https://doi.org/10.5120/ijca2016910505

Jonsson, N., & Jonsson, N. (2018). *Ways to use Machine Learning approaches for software development*.

Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Li, F. F. (2014). Large-scale video classification with convolutional neural networks. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1725–1732. https://doi.org/10.1109/CVPR.2014.223

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). AlexNet. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Kumar, V., Dave, V., Nagrani, R., Chaudhary, S., & Bhise, M. (n.d.). *Crop Cultivation Information System on Mobile Devices*.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., & Alsaadi, F. E. (2017). *NNs Archtectures review*. 1–31.

Lokesh, S., Naveenkumar, D., Rajesh, K., Kamath, G. A. R., & Rathnam, M. J. (2017). *Leaf Disease Detection and Grading using*. *6*(3), 279–287.

Martinelli, F., Scalenghe, R., Davino, S., Panno, S., Scuderi, G., Ruisi, P., … Dandekar, A. M. (2015). Advanced methods of plant disease detection. A review. *Agronomy for Sustainable Development*, Vol. 35. https://doi.org/10.1007/s13593-014-0246-1

Medium.com. (2019). An Introduction to Transfer Learning in Machine Learning. Retrieved from https:// medium. com/ kansas-city-machine-learning-artificial-intelligen/an-introduction-to-transfer-learning-in-machine-learning-7efd104b6026

Meng, C., Sun, M., Yang, J., Qiu, M., & Gu, Y. (2017). *Training Deeper Models by GPU Memory Optimization on TensorFlow*. (Nips), 1–8.

Mg, A., Hanson, J., Joy, A., & Francis, J. (2017). Plant Leaf Disease Detection using Deep Learning and Convolutional Neural Network. *International Journal of Engineering Science and Computing*, *7*(3). Retrieved from http://ijesc.org/

Mgonja, D. M., Temu, G. E., Lyantagaye, S. L., Makaranga, A., Joseph, C., & Luambano, N. D. (2020). *Plant parasitic nematodes occurrence and genetic diversity of banana cultivars grown in Tanzania*. *13*(01), 21–29. https://doi.org/10.21475/POJ.13.01.20.p2085

Mishra, S., Mishra, D., & Santra, G. H. (2016). *Applications of Machine Learning Techniques in Agricultural Crop Production : A Review Paper*. *9*(October). https://doi.org/10.17485/ijst/2016/v9i38/95032

Mohanty, S P, Hughes, D. P., & Salathe, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, *7*, 10. https://doi.org/10.3389/fpls.2016.01419

Mohanty, Sharada P., Hughes, D. P., & Salathé, M. (2016). Using Deep Learning for Image-Based Plant Disease Detection. *Frontiers in Plant Science*, *7*(September), 1–10. https://doi.org/10.3389/fpls.2016.01419

Ng'wanakilala, F. (2019). Tanzania's mobile phone subscriptions rise to nearly 44 million. Retrieved from https://www.reuters.com/article/tanzania-telecoms/tanzanias-mobile-phone-subscriptions-rise-to-nearly-44-million-idUSL3N26G26H

Nkuba, J., Tinzaara, W., Night, G., Niko, N., Jogo, W., Ndyetabula, I., … Box, P. O. (2015). *Adverse impact of Banana Xanthomonas Wilt on farmers ' livelihoods in Eastern and Central Africa*. *9*(July), 279–286. https://doi.org/10.5897/AJPS2015.1292

Ordonez, N., Seidl, M. F., Waalwijk, C., Drenth, A., & Kilian, A. (2015). *Worse Comes to Worst : Bananas and Panama Disease — When Plant and Pathogen Clones Meet*. 1–7. https://doi.org/10.1371/journal.ppat.1005197

Owomugisha, G., & Mwebaze, E. (2016). *Machine Learning for Plant Disease Incidence and Severity Measurements from Leaf Images*. https://doi.org/10.1109/ICMLA.2016.126

Owomugisha, G., Quinn, J. A., & Mwebaze, E. (2019). *Automated Vision-Based Diagnosis of Banana Bacterial Wilt Disease and Black Sigatoka Disease*. (June).

Patil, M. A. N., & Pawar, M. V. (2017). Detection and Classification of Plant Leaf Disease. *Iarjset*, *4*(4), 72–75. https://doi.org/10.17148/iarjset/nciarcse.2017.20

Prabha, D. S., & Kumar, J. S. (2014). Study on Banana Leaf Disease Identification Using

Image Processing Methods. *Ijrcsit*, *2*(2(A)). https://doi.org/ISSN 2319-5010

Ramadhani, K., Machuve, D., & Jomanga, K. (2017). *Identification and analysis of factors in management of banana fungal diseases : Case of Sigatoka (Mycosphaerella fijiensis . Mulder) and Fusarium (Fusarium oxysporum f . sp . cubense (Foc) diseases in Arumeru District*. *11*(1), 69–75.

Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., & Hughes, D. (2017). *Using Transfer Learning for Image-Based Cassava Disease Detection*. *8*(October), 1–7. https://doi.org/10.3389/fpls.2017.01852

Ramcharan, A., McCloskey, P., Baranowski, K., Mbilinyi, N., Mrisho, L., Ndalahwa, M., … Hughes, D. P. (2019). A Mobile-Based Deep Learning Model for Cassava Disease Diagnosis. *Frontiers in Plant Science*, *10*(March), 1–8. https://doi.org/10.3389/fpls.2019.00272

Rumpf, T., Mahlein, A. K., Steiner, U., Oerke, E. C., Dehne, H. W., & Plümer, L. (2010). Early detection and classification of plant diseases with Support Vector Machines based on hyperspectral reflectance. *Computers and Electronics in Agriculture*, *74*(1), 91–99. https://doi.org/10.1016/j.compag.2010.06.009

Shijie, J., Peiyi, J., Siping, H., & Haibo, Sl. (2017). Automatic detection of tomato diseases and pests based on leaf images. *Proceedings - 2017 Chinese Automation Congress, CAC 2017, 2017-Janua*, 3507–3510. https://doi.org/10.1109/CAC.2017.8243388

Shimwela, M. M., Blackburn, J. K., Jones, J. B., Nkuba, J., Narouei-khandan, H. A., Ploetz, R. C., … Bruggen, A. H. C. Van. (2016). *Local and regional spread of banana xanthomonas wilt (BXW) in space and time in Kagera, Tanzania*. 1–12. https://doi.org/10.1111/ppa.12637

Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 1–14. Retrieved from http://arxiv.org/abs/1409.1556

Singh, V., & Misra, A. K. (2016). Detection of Plant Leaf Diseases Using Image Segmentation and Soft Computing Techniques. *Information Processing in Agriculture*. https://doi.org/10.1016/j.inpa.2016.10.005

Sladojevic, S., Arsenovic, M., Anderla, A., Culibrk, D., & Stefanovic, D. (2016). Deep Neural

Networks Based Recognition of Plant Diseases by Leaf Image Classification. *Computational Intelligence and Neuroscience*, *2016*. https://doi.org/10.1155/2016/3289801

Ssali, R., Potato, I., & Agriculture, T. (2017). *Field Guide for Diagnostics and Data Collection*.

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). *Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning*. Retrieved from http://arxiv.org/abs/1602.07261

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. A. (2017). *the Impact of Residual Connections on Learning*. 4278–4284.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., & Liu, C. (2018). A survey on deep transfer learning. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *11141 LNCS*, 270–279. https://doi.org/10.1007/978-3-030-01424-7_27

Thangavelu, R., & Mustaffa, M. M. (2014). *Current Advances in the Fusarium Wilt Disease Management in Banana with Emphasis on Biological Control*. (April 2012). https://doi.org/10.13140/2.1.1941.0723

Verma, S. (2019). *Deep Learning-Based Mobile Application for Plant Disease Diagnosis : A Proof of Concept With a Case Study on Tomato Plant*. 2–5. https://doi.org/10.4018/978-1-5225-8027-0.ch010

Voulodimos, A., Doulamis, N., Doulamis, A., & Protopapadakis, E. (2018). *Deep Learning for Computer Vision : A Brief Review*. *2018*.

Woff, C. (n.d.). Comparing Image-Classification Systems: Custom Vision Service vs. Inception. Retrieved from https://devblogs.microsoft.com/cse/2017/12/05/comparing-transfer-learning-systems-custom-vision-service-vs-inception-vs-mobilenet/

Yang, X., & Guo, T. (2017). *Machine learning in plant disease research*. (August), 6–9. https://doi.org/10.18088/ejbmr.3.1.2017.pp6-9

APPENDICES

Appendix 1: Smalholder farmer's questions during data collection

1. Sex
   - ☐ Female
   - ☐ Male
2. How older are you?
   - ☐ 18-30
   - ☐ 31-56
   - ☐ >56
3. What is your education level?
   - ☐ Primary school
   - ☐ Secondary education
   - ☐ University
   - ☐ Never been to school
4. Is agriculture your main activity?
   - ☐ Yes
   - ☐ No
5. Which crop did you cultivate?
   - ☐ Maize
   - ☐ Rice
   - ☐ Banana
   - ☐ Tomatoes
   - ☐ Vegetables
   - ☐ None
6. If you cultivate banana what varieties did you plant?
   - ☐ Kisukari
   - ☐ Uganda
   - ☐ Mshale
   - ☐ Ngómbe
   - ☐ Other

   If other please specify……
7. How long have you been staying here?
   - ☐ 5years
   - ☐ 10 years
   - ☐ 20 years
   - ☐ More than 40 years

8. Apart from cultivating banana do you have another work?

- ☐ Yes
- ☐ No

9. If yes, then what do you do apart from cultivating banana

……

10. Do you have any information of banana fungal diseases?
    - ☐ Yes
    - ☐ No
11. What reasons do you think are affecting these diseases?
    - ☐ Low rainfall
    - ☐ High humidity
    - ☐ High Temperature
      Other please specify……
12. Have you heard about any tool used to detect banana diseases?
    - ☐ Yes
    - ☐ No

      If yes please mention it……
13. Do you have a mobile phone?
    - ☐ Yes
    - ☐ No
14. If yes from question 14 above, which kind of mobile do you own?
    - ☐ feature phone
    - ☐ Smartphone
15. Which network operator do you use?
    - ☐ Halotel
    - ☐ Vodacom
    - ☐ Tigo
    - ☐ Airtel
    - ☐ TTCL
16. Which method did you use to detect the banana diseases?
    - ☐ Using experience
    - ☐ Using a traditional method
    - ☐ None

17. if using a traditional method from question 16 please specify

……..

18. Is there any traditional approach that exists to notify smallholder farmers about the occurrence of banana diseases? Put Tick for your answer

- ☐ Extension officer
- ☐ Radio
- ☐ Farmers groups
- ☐ Forums
- ☐ Mobile phone

19. From question 18 which approach do you think the best and easy for you to get information about disease detection and management?
    - ☐ Farmers groups
    - ☐ Social media
    - ☐ Extension officer
    - ☐ Radio
    - ☐ Forums
    - ☐ Mobile phone
20. If the new system has been developed for banana disease detection will you be able to use it?
    - ☐ Yes
    - ☐ No
21. If No from question 20 explain why?

    ……..
22. If yes do you think the tool will help you to detect banana diseases at an early stage so that you can take action for management procedure?

    ……

Appendix 2: Extension officer's questions during data collection

1. Sex
   - ☐ Female
   - ☐ Male
2. How older are you?
   - ☐ 18-30
   - ☐ 31-56
   - ☐ >56
3. What is your education level?
   - ☐ Primary school
   - ☐ Secondary education
   - ☐ University
   - ☐ Never been to school
4. What is your occupation?
   - ☐ Agricultural district officer
   - ☐ Extension officer
   - ☐ Researcher
5. How many smallholder farmers do you save?
   - ☐ 5-20 farmers
   - ☐ 20-40 farmers
   - ☐ 40 farmers
   - ☐ >40 farmers
6. How many numbers of villages did you visit this year?
   - ☐ 4 village
   - ☐ 10 village
   - ☐ >10
   - ☐ none
7. How long have you been staying here?
   - ☐ 5years
   - ☐ 10 years
   - ☐ 20 years
   - ☐ More than 40 years
8. Do you have any knowledge of banana diseases?
   - ☐ Yes
   - ☐ No

9. Have you heard about any tool used to detect banana diseases?
   - ☐ Yes

- ☐ No

10. If yes please mention it……
11. Do you have a mobile phone?
    - ☐ Yes
    - ☐ No
12. If yes from question 14 above, which kind of mobile do you own?
    - ☐ feature phone
    - ☐ Smartphone
13. Which network operator do you use?
    - ☐ Halotel
    - ☐ Vodacom
    - ☐ Tigo
    - ☐ Airtel
    - ☐ TTCL
14. Which method did you use to inform smallholder farmers about banana disease occurrence?
    - ☐ Mobile phone
    - ☐ Visiting on their farm
    - ☐ Radio
    - ☐ Farmers groups
    - ☐ Exhibitions
    - ☐ none
15. What help do you deliver to farmers in your area?
    - ☐ Training
    - ☐ Advice
    - ☐ Consultation

    Other, response …….

Appendix 3: Questionnaire for the evaluation of FUSI application

I am Sophia Leonard Sanga a student pursuing a Master's Degree in Information and Communication Science and Engineering at The Nelson Mandela African Institute of Science and Technology. My research based on the development of an early detection tool for banana diseases which will help smallholder farmers /extension officers in the detection of banana fungal diseases at an early stage by using mobile application .our study has coved two areas for this research these areas were Rungwe district located in Mbeya region and Arusha region, Tanzania.

The purpose of these questions is for validation and testing of the developed tool called "FUSI (FUSARIUM SIGATOKA) SCANNER" to receive feedback from users by filling these questions.

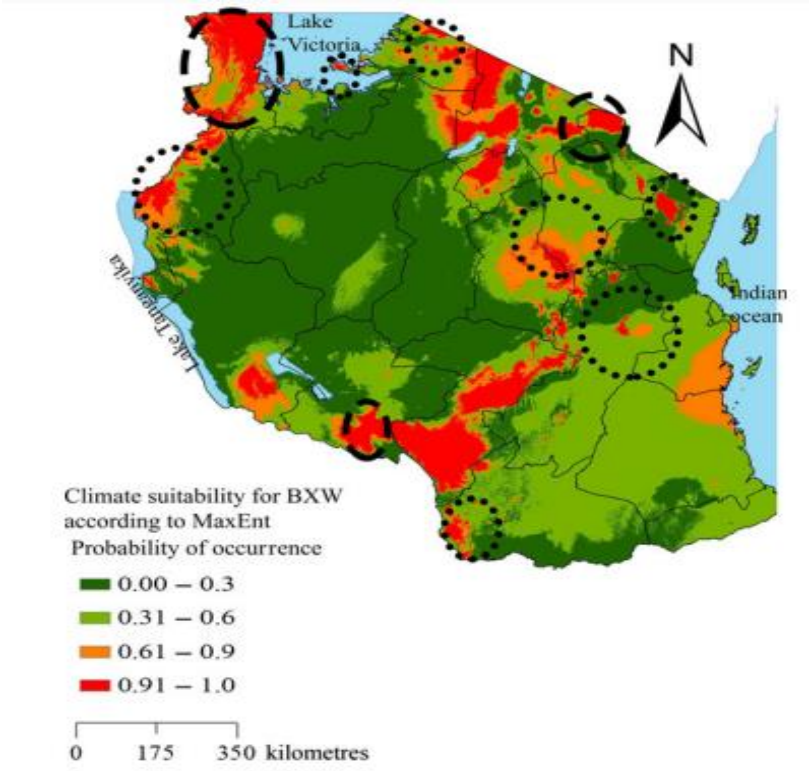
Your name ……..

Title/Occupation ……..

Note**:** Tick ✓ to the box relevant to your views

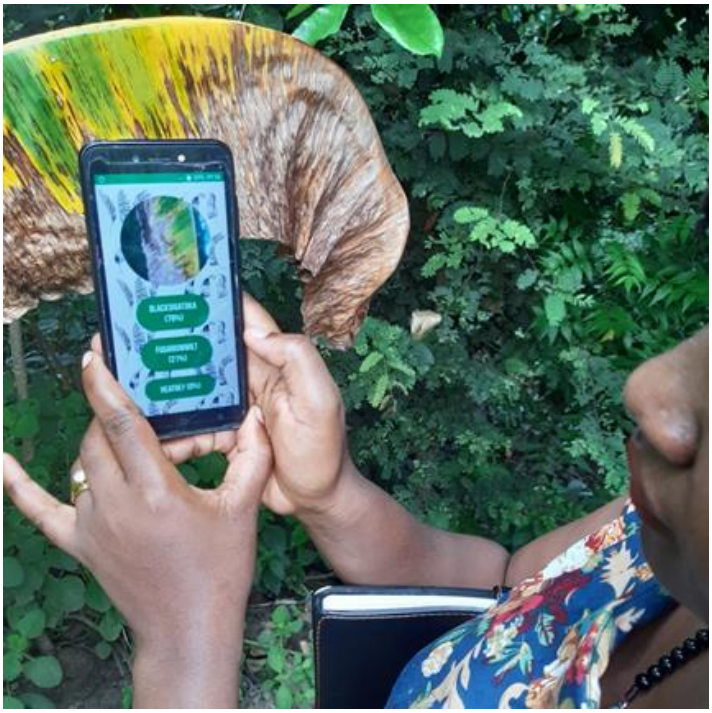
Please choose the best answer from the table below for the testing of the FUSI SCANNER

Questions	Strongly agree	Agree	Disagree	Neutral
Application is easy to use?				
Application interactive and user-friendly				
Functionalities of the application (Contents and features) are useful				
Did you get any difficulty while using a system?				
The application will help in the early detection of banana diseases				
Overall, did you like the application?				

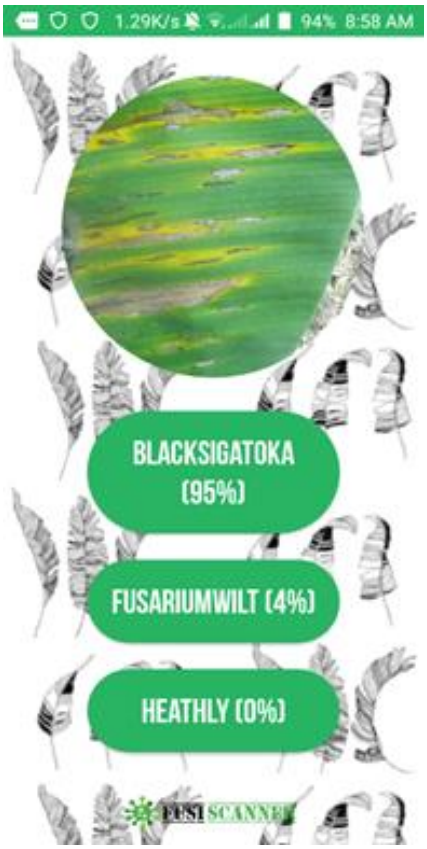

Lake
Victoria
N
Indian
ocean
Lake Tanganyika
Climate suitability for BXW
according to MaxEnt
Probability of occurrence
0.00 – 0.3
0.31 – 0.6
0.61 – 0.9
0.91 – 1.0
0
175
350 kilometres

Appendix 4: The geographical distribution of banana Fusarium wilt disease in Tanzania (Shimwela *et al.,* 2016)

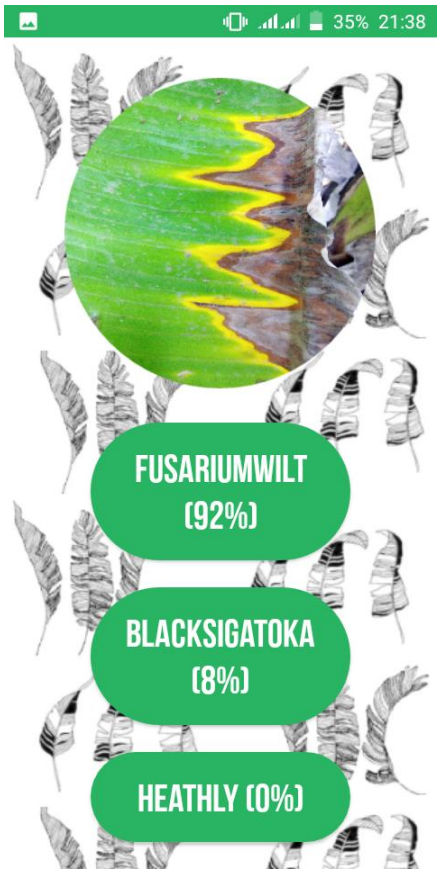
Appendix 5: An expert collecting data from the field in Arusha region



Appendix 6: Verification of FUSI SCANNER App in the real word environment



Appendix 7: Screenshots result of FUSI SCANNER app for uploaded image and captured from real environment respectively



Appendix 8: Screen shoots result of FUSI SCANNER app for Fusarium wilt race 1 captured from the real environment

Table 8: Usability testing results for extension officers

Questions	Strongly agree	Agree	Disagree	Neutral
	%	%	%	%
The application is interactive and user-friendly	76.47	23.53	0	0
Functionalities of the application (Contents and features) are useful	41.18	52.94	0	5.88
Did you get any difficulty while using your phone with the application?	0	5.88	94.12	0
Before using this application, first I need to learn about it.	0	5.88	94.12	0
Did you like the application?	58.82	41.18	0	0

Table 9: Acceptance testing results for the evaluation of tool by extension officers

Questions	Strongly agree	Agree	Disagree	Neutral
	%	%	%	%
I found the system useful	70.59	23.53	5.88	0
The system will help in detecting Sigatoka and Fusarium diseases	76.47	23.53	0	0
The application address the key problem that we are facing	88.24	11.76	0	0
I found too much inconsistency in this system	5.88	5.88	76.47	11.76
Overall, I like this application	88.24	11.76	0	0

Table 10: Farmers usability testing results

Questions	Strongly agree	Agree	Disagree	Neutral
	%	**%**	**%**	**%**
The application interactive and user-friendly	63.89	27.78	0	8.33
Functionalities of the application (Contents and features) are useful	72.22	22.22	0	5.56
Did you get any difficulty while using your phone with the application?	5.56	11.11	83.33	-
Before using this application, first I need to learn about it.	11.11	5.56	80.56	2.78
Did you like the application?	83.33	2.78	2.78	11.11

Table 11: Farmers acceptance testing results

Questions	Strongly agree	Agree	Disagree	Neutral
	%	%	%	%
I found the system useful	58.33	13	0	5.56
The system will help in detecting Sigatoka and Fusarium diseases	83.33	16.67	0	0
The application address the key problem that we are facing	86.11	11.11	0	5.56
I found too much inconsistency in this system	0	0	91.67	8.33
Overall, I like this application	91.67	2.78	5.56	0

Appendix 9: Source code for mobile deployment in Adroid

```
<?xml version="1.0" encoding="UTF-8"?>
<module type="JAVA_MODULE" version="4">
  <component name="NewModuleRootManager" inherit-compiler-output="true">
    <exclude-output />
    <content url="file://$MODULE_DIR$" />
    <orderEntry type="inheritedJdk" />
    <orderEntry type="sourceFolder" forTests="false" />
  </component>
</module>
```

Main Activity source code

```
package com.sofiaSanga.fusiscanner;

import android.content.Intent;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

import java.util.Locale;

public class MainActivity extends AppCompatActivity {
    static Classifier classifier;
  // static final int INPUT_SIZE = 224;
   static final int INPUT_SIZE = 299;
    private static final int IMAGE_MEAN = 128;
    //private static final float IMAGE_STD = 128.0f;
    private static final float IMAGE_STD = 128.0f;
  // private static final String INPUT_NAME = "Mul";
   // private static final String INPUT_NAME = "input";
  private static final String INPUT_NAME = "Mul";
    private static final String OUTPUT_NAME = "final_result";
    private static final String MODEL_FILE =
"file:///android_asset/optimized_graph.pb";
    private static final String LABEL_FILE = "file:///android_asset/labels.txt";
    private TextToSpeech tts;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        initializeTTS();
        initializeModel();
    }

    private void initializeModel() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                try {
                    classifier = TensorFlowImageClassifier.create(
                            getAssets(),
                            MODEL_FILE,
                            LABEL_FILE,
                            INPUT_SIZE,
                            IMAGE_MEAN,
                            IMAGE_STD,
                            INPUT_NAME,
                            OUTPUT_NAME);
                    startCameraActivity();
                } catch (final Exception e) {
                    throw new RuntimeException("Error initializing TensorFlow
Model!", e);
                }
            }
        }).start();
```

```
    }

    // Start the camera

    private void startCameraActivity() {
        new Thread(new Runnable() {
            @Override
            public void run() {
                Intent intent = new Intent(getApplicationContext(),
CameraActivity.class);
                startActivity(intent);
            }
        }).start();
    }

//Make the app speak
    protected void initializeTTS() {
        new Thread(new Runnable(){
            @Override
            public void run() {
                tts = new TextToSpeech(getApplicationContext(), new
TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status == TextToSpeech.SUCCESS) {
                            int result = tts.setLanguage(Locale.US);
                            if (result == TextToSpeech.LANG_MISSING_DATA || result
== TextToSpeech.LANG_NOT_SUPPORTED) {
                                Log.e("error", "Language not supported.");
                            } else {
                                String text = "This is Fusi scanner, the banana
disease identification app.";
                                tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
                            }
                        } else {
                            Log.e("error", "Initilization failed.");
                        }
                    }
                });
            }
        }).start();
    }
// close all
    @Override
    protected void onDestroy() {
        if(tts != null) {
            tts.stop();
            tts.shutdown();
        }
        finish();
        super.onDestroy();
    }

    @Override
    public void onBackPressed() {
        finish();
    }
}
```

camera activity source code

```
package com.sofiaSanga.fusiscanner;

import android.app.Activity;
import android.app.Fragment;
import android.app.FragmentManager;
import android.app.FragmentTransaction;
import android.graphics.Bitmap;
import android.os.Bundle;
```

```
import android.support.v7.app.AppCompatActivity;

import java.util.ArrayList;
import java.util.List;

public class CameraActivity extends AppCompatActivity {
    static Bitmap bitmap;
    static boolean resultsShown;
    static List<Classifier.Recognition> results;

    private static Fragment resultFragment;
    private static Fragment cameraFragment;
    private FragmentManager fragmentManager;

    int IMAGE_PICKER_SELECT = 1;


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_camera);
// classes initialization to perform camera activities.
        results = new ArrayList<>();
        cameraFragment = new CameraFragment();
        resultFragment = new ResultFragment();
        fragmentManager = getFragmentManager();

        FragmentTransaction fragmentTransaction =
fragmentManager.beginTransaction();
        fragmentTransaction.add(R.id.fragment_wrapper, cameraFragment);
        fragmentTransaction.commit();
    }
// return the results from the camera
    public static void showResultFragment(Activity activity) {
        FragmentTransaction fragmentTransaction =
activity.getFragmentManager().beginTransaction();
        fragmentTransaction.setCustomAnimations(android.R.animator.fade_in,
android.R.animator.fade_out);
        fragmentTransaction.replace(R.id.fragment_wrapper, resultFragment);
        fragmentTransaction.addToBackStack(null);
        fragmentTransaction.commit();
        resultsShown = true;
    }

    @Override
    public void onResume() {
        super.onResume();
    }
// go back to previous activities on back press
    @Override
    public void onBackPressed() {
        if(fragmentManager.getBackStackEntryCount() != 0) {
            fragmentManager.popBackStack();
            if(resultsShown) {
                resultsShown = false;
            }
            else {
                finish();
            }
        }
        else {
            finish();
        }
    }
}
```

Camera fagment source code

```
package com.sofiaSanga.fusiscanner;

import android.app.Fragment;
import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.ImageButton;
import android.widget.RelativeLayout;

import com.flurgle.camerakit.CameraKit;
import com.flurgle.camerakit.CameraListener;
import com.flurgle.camerakit.CameraView;

import java.util.List;
import java.util.Locale;

public class CameraFragment extends Fragment {
    private ImageButton openGallery;
    private CameraView cameraView;
    private ImageButton btnToggleCamera;
    private ImageButton btnToggleFlash;
    private ImageButton btnDetectObject;
    private TextToSpeech tts;
    private boolean flashStatus = false;
    public static boolean FROM_GALLERY = false;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        return inflater.inflate(R.layout.fragment_camera, container, false);
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceType) {
        super.onActivityCreated(savedInstanceType);

        cameraView = getFragmentManager()
            .findFragmentById(R.id.fragment_wrapper)
            .getView()
            .findViewById(R.id.live_camera);

        cameraView.setFocus(CameraKit.Constants.FOCUS_TAP);

        cameraView.setCameraListener(new CameraListener() {
            @Override
            public void onPictureTaken(byte[] picture) {
                super.onPictureTaken(picture);
                processImage(picture);
            }
        });

        btnToggleCamera = getFragmentManager()
            .findFragmentById(R.id.fragment_wrapper)
            .getView()
            .findViewById(R.id.btnToggleCamera);

        btnToggleFlash = getFragmentManager()
```

```
                .findFragmentById(R.id.fragment_wrapper)
                .getView()
                .findViewById(R.id.btnToggleFlash);
// camera settings
        btnToggleFlash.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(flashStatus) {
                    btnToggleFlash.setImageResource(R.drawable.flashoff);
                    cameraView.setFlash(CameraKit.Constants.FLASH_OFF);
                    flashStatus = false;
                }
                else {
                    btnToggleFlash.setImageResource(R.drawable.flashon);
                    cameraView.setFlash(CameraKit.Constants.FLASH_ON);
                    flashStatus = true;
                }
            }
        });

        btnToggleCamera.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                cameraView.toggleFacing();
            }
        });

        btnDetectObject = getFragmentManager()
            .findFragmentById(R.id.fragment_wrapper)
            .getView()
            .findViewById(R.id.btnDetectObject);

        btnDetectObject.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                initializeTTS();
                cameraView.captureImage();
            }
        });

        openGallery =
getFragmentManager().findFragmentById(R.id.fragment_wrapper).getView().findViewById(
R.id.open_gallery);
        openGallery.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                Intent intent = new Intent(getActivity(), GalleryActivity.class);
                startActivity(intent);
            }
        });

        fixLayoutSize();
        cameraView.start();
    }
// Image processing
    public void processImage(byte[] picture) {
        Log.i("activity", "PROCESS");
        CameraActivity.bitmap = BitmapFactory.decodeByteArray(
                picture,
                0,
                picture.length);
        CameraActivity.bitmap = Bitmap.createScaledBitmap(
                CameraActivity.bitmap,
                MainActivity.INPUT_SIZE,
                MainActivity.INPUT_SIZE,
                false);
        CameraActivity.bitmap = ImageHelper.getRotatedImage(CameraActivity.bitmap,
90);
        CameraActivity.results =
```

```
MainActivity.classifier.recognizeImage(CameraActivity.bitmap);
        CameraActivity.showResultFragment(getActivity());
    }

    @Override
    public void onPause() {
        super.onPause();
        cameraView.stop();
    }
// recognize an image captured
    @Override
    public void onResume() {
        super.onResume();
        if(FROM_GALLERY) {
            List<Classifier.Recognition> results =
MainActivity.classifier.recognizeImage(CameraActivity.bitmap);
            if(results.isEmpty()) return;
            CameraActivity.showResultFragment(getActivity());
            FROM_GALLERY = false;
        }
        else {
            if(flashStatus) {
                btnToggleFlash.setImageResource(R.drawable.flashon);
                cameraView.setFlash(CameraKit.Constants.FLASH_ON);
            }
            else {
                btnToggleFlash.setImageResource(R.drawable.flashoff);
                cameraView.setFlash(CameraKit.Constants.FLASH_OFF);
            }
            cameraView.start();
        }
    }

    public void fixLayoutSize() {
        DisplayMetrics displayMetrics = new DisplayMetrics();

getActivity().getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);

        int screenHeight = displayMetrics.heightPixels;
        int screenWidth = displayMetrics.widthPixels;

        RelativeLayout captureButtonWrapper = getFragmentManager()
            .findFragmentById(R.id.fragment_wrapper)
            .getView()
            .findViewById(R.id.capture_button_wrapper);

        ViewGroup.LayoutParams cameraViewParams = cameraView.getLayoutParams();
        cameraViewParams.height = screenWidth;
        cameraViewParams.width = screenWidth;
        cameraView.setLayoutParams(cameraViewParams);

        int captureWrapperHeight =  screenHeight - (screenWidth + 50 + 30);
        ViewGroup.LayoutParams captureButtonWrapperParams  =
captureButtonWrapper.getLayoutParams();
        captureButtonWrapperParams.height = captureWrapperHeight;
        captureButtonWrapper.setLayoutParams(captureButtonWrapperParams);
    }

    protected void initializeTTS() {
        new Thread(new Runnable(){
            @Override
            public void run() {
                tts = new TextToSpeech(getActivity(), new
TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status == TextToSpeech.SUCCESS) {
                            int result = tts.setLanguage(Locale.US);
                            if (result == TextToSpeech.LANG_MISSING_DATA || result
```

```
== TextToSpeech.LANG_NOT_SUPPORTED) {
                                Log.e("error", "Language not supported.");
                            } else {
                                String text = "Hold on, This might take a while.";
                                tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
                            }
                        } else {
                            Log.e("error", "Initilization failed.");
                        }
                    }
                });
            }
        }).start();
    }

    @Override
    public void onDestroy() {
        if(tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }

}
```

Gallery activity source code

```
package com.sofiaSanga.fusiscanner;

import android.content.Intent;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.speech.tts.TextToSpeech;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;

import java.io.ByteArrayOutputStream;
import java.util.Locale;

public class GalleryActivity extends AppCompatActivity {
    private static int PICK_IMAGE_REQUEST = 1;
    private TextToSpeech tts;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_gallery);
        Intent intent = new Intent();
        intent.setType("image/*");
        intent.setAction(Intent.ACTION_GET_CONTENT);
        startActivityForResult(Intent.createChooser(intent, "Select Picture"),
PICK_IMAGE_REQUEST);
    }

    @Override
    public void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if (requestCode == PICK_IMAGE_REQUEST && resultCode == RESULT_OK && data !=
null && data.getData() != null) {
            initializeTTS();
            Uri returnUri = data.getData();
            try {
                Bitmap image =
MediaStore.Images.Media.getBitmap(getContentResolver(), returnUri);
                ByteArrayOutputStream stream = new ByteArrayOutputStream();
```

```
                image.compress(Bitmap.CompressFormat.PNG, 100, stream);
                byte[] picture = stream.toByteArray();
                CameraFragment.FROM_GALLERY = true;
                CameraActivity.bitmap = BitmapFactory.decodeByteArray(
                        picture,
                        0,
                        picture.length);
                CameraActivity.bitmap = Bitmap.createScaledBitmap(
                        CameraActivity.bitmap,
                        MainActivity.INPUT_SIZE,
                        MainActivity.INPUT_SIZE,
                        false);
                CameraActivity.results =
MainActivity.classifier.recognizeImage(CameraActivity.bitmap);
                finish();
            } catch(Exception e) {
                Log.e("error", "Error loading image from gallery.");
            }
        }
    }

    @Override
    public void onResume() {
        super.onResume();
    }

    protected void initializeTTS() {
        new Thread(new Runnable(){
            @Override
            public void run() {
                tts = new TextToSpeech(getApplicationContext(), new
TextToSpeech.OnInitListener() {
                    @Override
                    public void onInit(int status) {
                        if (status == TextToSpeech.SUCCESS) {
                            int result = tts.setLanguage(Locale.US);
                            if (result == TextToSpeech.LANG_MISSING_DATA || result
== TextToSpeech.LANG_NOT_SUPPORTED) {
                                Log.e("error", "Language not supported.");
                            } else {
                                String text = "Hold on, This might take a while.";
                                tts.speak(text, TextToSpeech.QUEUE_FLUSH, null);
                            }
                        } else {
                            Log.e("error", "Initilization failed.");
                        }
                    }
                });
            }
        }).start();
    }

    @Override
    public void onDestroy() {
        if(tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }

}
```

Result activity source code

```
package com.sofiaSanga.fusiscanner;
```

```
import android.app.Fragment;
import android.graphics.Typeface;
import android.os.Bundle;
import android.speech.tts.TextToSpeech;
import android.util.DisplayMetrics;
import android.util.Log;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.LinearLayout;

import java.util.List;
import java.util.Locale;

public class ResultFragment extends Fragment {
    private TextToSpeech tts;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle
savedInstanceState) {
        return inflater.inflate(R.layout.fragment_result, container, false);
    }

    @Override
    public void onActivityCreated(Bundle savedInstanceType) {
        super.onActivityCreated(savedInstanceType);

        fixLayoutSize();
        renderResult();
    }

    public void fixLayoutSize() {
        DisplayMetrics displayMetrics = new DisplayMetrics();

getActivity().getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);

        int screenHeight = displayMetrics.heightPixels;
        int screenWidth = displayMetrics.widthPixels;

        ImageView preview = getFragmentManager().
                findFragmentById(R.id.fragment_wrapper).
                getView().
                findViewById(R.id.image_captured_preview);

        preview.setImageBitmap(ImageHelper.getRoundedImage(CameraActivity.bitmap,
1000));
    }

    public void renderResult() {
        final List<Classifier.Recognition> results = CameraActivity.results;
        initializeTTS(results.get(0).getTitle(),
Math.round(results.get(0).getConfidence() * 100));

        DisplayMetrics displayMetrics = new DisplayMetrics();

getActivity().getWindowManager().getDefaultDisplay().getMetrics(displayMetrics);

        int screenHeight = displayMetrics.heightPixels;
        int screenWidth = displayMetrics.widthPixels;

        LinearLayout resultWrapper = getFragmentManager().
                findFragmentById(R.id.fragment_wrapper).
```

```
                getView().
                findViewById(R.id.results_wrapper);

        int[] id = new int[3];
        id[0] = R.id.guess_1;
        id[1] = R.id.guess_2;
        id[2] = R.id.guess_3;

        Typeface bebas = Typeface.createFromAsset(getActivity().getAssets(),
"font/bebas.otf");

        int resultsCount = CameraActivity.results.size();
        for(int i=0; i<resultsCount; i++) {
            final Classifier.Recognition label = results.get(i);
            Button result = getFragmentManager().
                    findFragmentById(R.id.fragment_wrapper).
                    getView().
                    findViewById(id[i]);

           /* result.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    openInsectWiki(label.getTitle());
                }
            }); */

            result.getLayoutParams().width = (int) (screenWidth * 0.6);

            int confidence = Math.round(label.getConfidence() * 100);
            result.setText(label.getTitle() + " (" + confidence + "%)");
            result.setTypeface(bebas);
        }
    }

    protected void initializeTTS(final String name, final int confidence) {
        tts = new TextToSpeech(getActivity(), new TextToSpeech.OnInitListener() {
            @Override
            public void onInit(int status) {
                if(status == TextToSpeech.SUCCESS){
                    int result = tts.setLanguage(Locale.US);
                    if(result == TextToSpeech.LANG_MISSING_DATA ||
                            result == TextToSpeech.LANG_NOT_SUPPORTED){
                        Log.e("error", "This Language is not supported");
                    } else {
                        String text;
                        if(confidence < 60) {
                            text = "That image seems unrecognizable. Maybe I have to
update my database.";
                        }
                        else if(confidence < 70) {
                            text = "Try to get a clearer image next time, but I have
a guess that that is a " + name + "!";
                        } else {
                            text = "I am " + confidence + " percent sure that that
is a " + name + "!";
                        }
                        tts.speak(text, TextToSpeech.QUEUE_ADD, null);
                    }
                }
                else {
                    Log.e("error", "Initilization Failed!");
                }
            }
        });
    }


    @Override
    public void onDestroy() {
```

```
        if(tts != null) {
            tts.stop();
            tts.shutdown();
        }
        super.onDestroy();
    }

}
```

Image help source code

```
package com.sofiaSanga.fusiscanner;

import android.graphics.Bitmap;
import android.graphics.Canvas;
import android.graphics.Matrix;
import android.graphics.Paint;
import android.graphics.PorterDuff;
import android.graphics.PorterDuffXfermode;
import android.graphics.Rect;
import android.graphics.RectF;

public class ImageHelper {

    public static Bitmap getRoundedImage(Bitmap bitmap, int pixels) {
        Bitmap output = Bitmap.createBitmap(bitmap.getWidth(), bitmap
                .getHeight(), Bitmap.Config.ARGB_8888);
        Canvas canvas = new Canvas(output);

        final int color = 0xff424242;
        final Paint paint = new Paint();
        final Rect rect = new Rect(0, 0, bitmap.getWidth(), bitmap.getHeight());
        final RectF rectF = new RectF(rect);
        final float roundPx = pixels;

        paint.setAntiAlias(true);
        canvas.drawARGB(0, 0, 0, 0);
        paint.setColor(color);
        canvas.drawRoundRect(rectF, roundPx, roundPx, paint);

        paint.setXfermode(new PorterDuffXfermode(PorterDuff.Mode.SRC_IN));
        canvas.drawBitmap(bitmap, rect, rect, paint);

        return output;
    }

    public static Bitmap getRotatedImage(Bitmap source, float angle) {
        Matrix matrix = new Matrix();
        matrix.postRotate(angle);
        return Bitmap.createBitmap(source, 0, 0, source.getWidth(),
source.getHeight(),
                matrix, true);
    }

}
```

Classifier source code

```
package com.sofiaSanga.fusiscanner;

import android.graphics.Bitmap;
import android.graphics.RectF;

import java.util.List;

public interface Classifier {
    /**
     * An immutable result returned by a Classifier describing what was recognized.
```

```
     */
    class Recognition {
        /**
         * A unique identifier for what has been recognized. Specific to the class,
not the instance of
         * the object.
         */
        private final String id;

        /**
         * Display name for the recognition.
         */
        private final String title;

        /**
         * A sortable score for how good the recognition is relative to others.
Higher should be better.
         */
        private final Float confidence;

        /**
         * Optional location within the source image for the location of the
recognized object.
         */
        private RectF location;

        public Recognition(
                final String id, final String title, final Float confidence, final
RectF location) {
            this.id = id;
            this.title = title;
            this.confidence = confidence;
            this.location = location;
        }

        public String getId() {
            return id;
        }

        public String getTitle() {
            return title;
        }

        public Float getConfidence() {
            return confidence;
        }

        public RectF getLocation() {
            return new RectF(location);
        }

        public void setLocation(RectF location) {
            this.location = location;
        }

        @Override
        public String toString() {
            String resultString = "";
            if (id != null) {
                resultString += "[" + id + "] ";
            }

            if (title != null) {
                resultString += title + " ";
            }

            if (confidence != null) {
                resultString += String.format("(%.1f%%) ", confidence * 100.0f);
            }
```

```
            if (location != null) {
                resultString += location + " ";
            }

            return resultString.trim();
        }
    }

    List<Recognition> recognizeImage(Bitmap bitmap);

    void enableStatLogging(final boolean debug);

    String getStatString();

    void close();
}
```

TensorFlow image classifier source code

```
package com.sofiaSanga.fusiscanner;

import android.annotation.SuppressLint;
import android.content.res.AssetManager;
import android.graphics.Bitmap;
import android.os.Build;
import android.os.Trace;
import android.support.annotation.RequiresApi;
import android.util.Log;

import org.tensorflow.contrib.android.TensorFlowInferenceInterface;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import java.util.PriorityQueue;
import java.util.Vector;


public class TensorFlowImageClassifier implements Classifier {

    private static final String TAG = "TensorFlowImageClassifier";

    // Only return this many results with at least this confidence.
    private static final int MAX_RESULTS = 3;
    private static final float THRESHOLD = 0.0f;

    // Config values.
    private String inputName;
    private String outputName;
    private int inputSize;
    private int imageMean;
    private float imageStd;

    // Pre-allocated buffers.
    private Vector<String> labels = new Vector<String>();
    private int[] intValues;
    private float[] floatValues;
    private float[] outputs;
    private String[] outputNames;

    private TensorFlowInferenceInterface inferenceInterface;

    private TensorFlowImageClassifier() {
    }
```

```
    /**
     * Initializes a native TensorFlow session for classifying images.
     *
     * @param assetManager  The asset manager to be used to load assets.
     * @param modelFilename The filepath of the model GraphDef protocol buffer.
     * @param labelFilename The filepath of label file for classes.
     * @param inputSize     The input size. A square image of inputSize x inputSize
is assumed.
     * @param imageMean     The assumed mean of the image values.
     * @param imageStd      The assumed std of the image values.
     * @param inputName     The label of the image input node.
     * @param outputName    The label of the output node.
     * @throws IOException
     */
    @SuppressLint("LongLogTag")
    static Classifier create(
            AssetManager assetManager,
            String modelFilename,
            String labelFilename,
            int inputSize,
            int imageMean,
            float imageStd,
            String inputName,
            String outputName)
            throws IOException {
        TensorFlowImageClassifier c = new TensorFlowImageClassifier();
        c.inputName = inputName;
        c.outputName = outputName;

        // Read the label names into memory.
        // TODO(andrewharp): make this handle non-assets.
        String actualFilename = labelFilename.split("file:///android_asset/")[1];
        Log.i(TAG, "Reading labels from: " + actualFilename);
        BufferedReader br = null;
        br = new BufferedReader(new
InputStreamReader(assetManager.open(actualFilename)));
        String line;
        while ((line = br.readLine()) != null) {
            c.labels.add(line);
        }
        br.close();

        c.inferenceInterface = new TensorFlowInferenceInterface( );
        if (c.inferenceInterface.initializeTensorFlow(assetManager, modelFilename)
!= 0) {
            throw new RuntimeException("TF initialization failed");
        }
        // The shape of the output is [N, NUM_CLASSES], where N is the batch size.
        int numClasses =
                (int)
c.inferenceInterface.graph().operation(outputName).output(0).shape().size(1);
        Log.i(TAG, "Read " + c.labels.size() + " labels, output layer size is " +
numClasses);

        // Ideally, inputSize could have been retrieved from the shape of the input
operation.  Alas,
        // the placeholder node for input in the graphdef typically used does not
specify a shape, so it
        // must be passed in as a parameter.
        c.inputSize = inputSize;
        c.imageMean = imageMean;
        c.imageStd = imageStd;

        // Pre-allocate buffers.
        c.outputNames = new String[]{outputName};
        c.intValues = new int[inputSize * inputSize];
        c.floatValues = new float[inputSize * inputSize * 3];
        c.outputs = new float[numClasses];
```

```
        return c;
    }

    @RequiresApi(api = Build.VERSION_CODES.JELLY_BEAN_MR2)
    @Override
    public List<Recognition> recognizeImage(final Bitmap bitmap) {
        // Log this method so that it can be analyzed with systrace.
        Trace.beginSection("recognizeImage");

        Trace.beginSection("preprocessBitmap");
        // Preprocess the image data from 0-255 int to normalized float based
        // on the provided parameters.
        bitmap.getPixels(intValues, 0, bitmap.getWidth(), 0, 0, bitmap.getWidth(),
bitmap.getHeight());
        for (int i = 0; i < intValues.length; ++i) {
            final int val = intValues[i];
            floatValues[i * 3 + 0] = (((val >> 16) & 0xFF) - imageMean) / imageStd;
            floatValues[i * 3 + 1] = (((val >> 8) & 0xFF) - imageMean) / imageStd;
            floatValues[i * 3 + 2] = ((val & 0xFF) - imageMean) / imageStd;
        }
        Trace.endSection();

        // Copy the input data into TensorFlow.
        Trace.beginSection("fillNodeFloat");
        inferenceInterface.fillNodeFloat(
                inputName, new int[]{1, inputSize, inputSize, 3}, floatValues);
        Trace.endSection();

        // Run the inference call.
        Trace.beginSection("runInference");
        inferenceInterface.runInference(outputNames);
        Trace.endSection();

        // Copy the output Tensor back into the output array.
        Trace.beginSection("readNodeFloat");
        inferenceInterface.readNodeFloat(outputName, outputs);
        Trace.endSection();

        // Find the best classifications.
        PriorityQueue<Recognition> pq =
                new PriorityQueue<Recognition>(
                        3,
                        new Comparator<Recognition>() {
                            @Override
                            public int compare(Recognition lhs, Recognition rhs) {
                                // Intentionally reversed to put high confidence at
the head of the queue.
                                return Float.compare(rhs.getConfidence(),
lhs.getConfidence());
                            }
                        });
        for (int i = 0; i < outputs.length; ++i) {
            if (outputs[i] > THRESHOLD) {
                pq.add(
                        new Recognition(
                                "" + i, labels.size() > i ? labels.get(i) :
"unknown", outputs[i], null));
            }
        }
        final ArrayList<Recognition> recognitions = new ArrayList<Recognition>();
        int recognitionsSize = Math.min(pq.size(), MAX_RESULTS);
        for (int i = 0; i < recognitionsSize; ++i) {
            recognitions.add(pq.poll());
        }
        Trace.endSection(); // "recognizeImage"
        return recognitions;
    }

    @Override
```

```
    public void enableStatLogging(boolean debug) {
        inferenceInterface.enableStatLogging(debug);
    }

    @Override
    public String getStatString() {
        return inferenceInterface.getStatString();
    }

    @Override
    public void close() {
        inferenceInterface.close();
    }
}
```

RESEARCH OUTPUT

1. Research article “Mobile-based Deep Learning Modes for Banana Disease Detection”, published in the Journal of Engineering, Technology & Applied Science Research.

2. Poster Presentation “Banana Disease Detection using Deep Learning”.