

**BUSITEMA
UNIVERSITY**
Pursuing Excellence

P.O. Box 236, Tororo, Uganda
Gen: +256 - 45 444 8838
Fax: +256 - 45 4436517
Email: info@adm.busitema.ac.ug

www.busitema.ac.ug

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

FINAL YEAR PROJECT REPORT

STUDENT NAME: IKONDE NEKEMIAH ARNOLD

Email: arnoldnek@gmail.com

Tel: 0700388074/0789370238

Project Title:

**MOBILE-BASED MACHINE LEARNING SYSTEM TO RECOGNIZE POULTRY
DISEASES.**

CASE: NEW CASTLE DISEASE, SALMONELLA, AND COCCIDIOSIS

SUPERVISOR: DR. GODLIVER OWOMUGISHA

**PROJECT REPORT SUBMITTED TO THE DEPARTMENT OF COMPUTER
ENGINEERING IN PARTIAL FULFILLMENT FOR THE AWARD OF A
BACHELOR'S DEGREE IN COMPUTER ENGINEERING OF**

BUSITEMA UNIVERSITY

ACKNOWLEDGEMENT

With gratitude, I thank the almighty God for his love towards me and the unending guidance and protection he has provided me throughout the entire duration of my study while at Busitema University. I also continue to thank him for his support which has enabled me to carry out this project successfully.

Appreciation also goes to my Supervisor Dr. Godliver Owomugisha for her help and continued support while carrying out my research.

My appreciation also goes to my fellow students for the help and ideas offered.

DECLARATION

I IKONDE NEKEMIAH ARNOLD Reg No. **BU/UP/2018/2913** hereby declare that this project is my original work except where explicit citation has been made and has never been published or submitted for any other degree award to any other university or institution of higher learning for any academic award.

Sign:

Date:

APPROVAL

This is to certify that the project report entitled “MOBILE-BASED MACHINE LEARNING SYSTEM TO RECOGNIZE POULTRY DISEASES” has been done under the supervision and is submitted to the board of Examiners with my approval.

DR. GODLIVER OWOMUGISHA

DEPARTMENT OF COMPUTER ENGINEERING

Sign:

Date:

TABLE OF CONTENTS

Table of Contents

ACKNOWLEDGEMENT	i
DECLARATION	ii
APPROVAL	iii
TABLE OF CONTENTS	iv
CHAPTER ONE	1
1.1 BACKGROUND	1
1.2 Problem Statement	1
1.2 Objectives	1
1.2.1 Main Objective.....	1
1.2.2 Specific Objectives	2
1.3 Justification	2
1.4 Scope	2
1.4.1 Geographical scope.....	2
1.4.2 Technical Scope	2
1.5.3 Time Scope	2
CHAPTER TWO: LITERATURE REVIEW	3
2.1 MAIN CONCEPTS OF THE PROJECT	3
2.1.1 Poultry Diseases overview	3
2.1.2 Machine learning methods in poultry	3
2.2 Existing System Comparison Table	5
2.3 Proposed System	5
CHAPTER THREE: METHODOLOGY	6
3.1 Requirement Collection (data gathering)	6
3.1.1 Online Resources.....	6
3.2 Requirement Analysis	6
3.2.1 Functional Requirements.	6
3.2.2 Non-functional requirements.	6
3.3 System design	7
3.3.1 Conceptual Design for the System.....	7
3.3.2 System Flow Chat.....	8
CHAPTER FIVE: IMPLEMENTATION AND TESTING	9
5.1 DESIGN AND DEVELOPMENT PLATFORMS	9
5.1.1 Google Colab	9

5.1.2 Kaggle.....	9
5.1.3 Android Studio.....	9
5.2 Testing and Validation	9
5.2.1 Unit testing.....	9
5.2.2 Integration Testing.....	9
5.2.3 System Testing.....	10
5.2.4 Verification	10
5.2.5 Validation.....	10
5.3 System validation.....	10
5.3.1 Results.....	11
CHAPTER SIX: DISCUSSIONS AND RECOMMENDATIONS	12
6.1 Introduction	12
6.2 Summary of the work done	12
6.3 Critical Analysis and Comparison with other Systems	12
6.4 Recommendations	12
6.5 Challenges.....	12
6.6 Conclusions.....	13
7.0. References.....	14
8.0 APPENDICES	15

LIST OF FIGURES

Figure 1. Block Diagram.....	7
Figure 2. Flow Chart.....	8
Figure 3.Salmonera recognized	11
Figure 4.Healthy	11
Figure 5.Training process of the model	24
Figure 6.Graph showing both Training Vs Validation Loss and Training Vs Validation Accuracy	26
Figure 7. Confusion Matrix.....	27
Figure 8. Classification Report	27

CHAPTER ONE

1.1 BACKGROUND

The livestock sector in Uganda contributes 3.2% to the national gross domestic product (GDP) and is projected to be rising [1]. A report in 2009 showed that 4.5 million households (70.8%) owned livestock or poultry

In the recent past, a large number of people in and around Kampala and other major towns have taken up poultry farming. To meet the requirements of the country's fast-growing population, there is a need for improved and increased poultry production. However, several factors put a great constraint on increased production in the country. These factors are mainly disease, lack of appropriate drugs and vaccines, improper management of farms, and wrong motives for poultry farming. Diseases and pathological conditions affecting poultry are many and varied. [3]

Some of the common diseases that bring a setback in poultry farming include;

Newcastle, Coccidiosis [4][5], Salmonella, Aspergillosis, Gumboro disease, and so on.

Respiratory diseases such as Newcastle require early detection so the whole poultry farm is not lost in a short time. Having a system that would detect the disease increases production. [6]

With the current development in information technologies, Artificial intelligence has become a promising tool in the real-time recognition of poultry disease monitoring systems due to its non-intrusive and non-invasive properties and ability to present a wide range of information. Hence, we will use the Tensor flow to deploy the d into the mobile application, we will be able to identify some of the most common poultry diseases.[7]

1.2 Problem Statement

Whether you're raising a large flock of chickens, or only a few, the process can be extremely rewarding. Being able to detect the disease outbreak in the livestock at earlier stages allows the farmer to find the best solution as soon as possible.

There is a need to develop a system that would recognize which disease is responsible for the outbreak. We need this system to be as accessible as possible for the farmers. This will enable the farmer to take action as soon as possible, hence curbing the outbreak as soon as possible.

Also, to help prevent these popular diseases, a scheduler will be included in the mobile application to enable the farmer to know the vaccination patterns so they can save the date on Google calendar.

1.2 Objectives

1.2.1 Main Objective

- To design a mobile-based machine learning system that will recognize poultry diseases.

1.2.2 Specific Objectives

- To collect chicken fecal images from online sources and available datasets
- To develop the mobile application to be used.
- To test and validate the datasets.
- To deploy the machine learning model to the mobile application.

.

1.3 Justification

With this mobile-based poultry disease recognition system in place, a farmer will have reduced operating costs that would go into investigating which disease is affecting the chicken. It will be just a matter of pulling out his smartphone and opening the mobile application to use it to analyze the chicken waste to identify the disease.

Having a vaccination scheduling feature in the Mobile app will help the poultry farmer counter some of the common poultry diseases.

1.4 Scope

1.4.1 Geographical scope

This system was implemented and is to be used here in Uganda.

1.4.2 Technical Scope

This system was reduced to a mobile application, which scans the chicken waste and compare it with the dataset which is configured using Tensor flow by using google colab.

1.5.3 Time Scope

This project was limited to six months, whereby two months are for proposing and four months for designing and implementing.

CHAPTER TWO: LITERATURE REVIEW

2.1 MAIN CONCEPTS OF THE PROJECT

2.1.1 Poultry Diseases overview

Coccidiosis

Chickens' intestinal systems are infected with parasites of the genus *Eimeria*, which cause coccidiosis. The conventional diagnostic approach includes examining the digestive tract and/or counting the number of oocysts (expressed as oocysts per gram opg) in the stool to calculate the lesion scores.[8]

Salmonella

Salmonella spp are bacterial pathogens of the genus *Salmonella* that cause diseases in chickens, other domestic animals, and humans. The polymerase Chain Reaction (PCR) procedure is used for the detection and identification of various *Salmonella* strains.[9]

Newcastle Disease

Newcastle disease is an acute viral infection in poultry and other bird species caused by avian paramyxovirus serotype 1 (APMV-1) viruses. Newcastle disease virus (NDV), APMV-1 is diagnosed by serology or virus isolation tests or real-time reverse-transcription PCR procedure.[10]

These diseases can be diagnosed early using machine learning and deep learning methods.

2.1.2 Machine learning methods in poultry

(a)Recently, various research has suggested using machine learning to identify diseases in chickens [11]. To categorize fine-grained abnormal broiler droppings photos as normal and abnormal, Wang et al. (2019) suggested disease an automated broiler digestive detector based on deep Convolutional Neural Network models, Faster R-CNN, and YOLO-V3. On the testing data set, faster R-CNN scored 99.1% recall and 93.33% mean average precision, whereas YOLO-V3 attained 88.7% recall and 84.33% mean average precision.

Weakness

The study by Wang et al. (2019) advances the creation of an automatic, non-contact model for recognizing and categorizing irregular droppings in broilers with digestive disorders, but a practical solution is still needed for the early diagnosis of poultry diseases.

(b)In another study, Okinda et al. (2019) proposed a machine vision-based broiler chicken monitoring system in a different study. Based on mobility features and 2D posture shape descriptors, feature variables were derived. Then, two sets of classifiers were created, one using all the feature variables and the other just using the posture shape descriptors. With an accuracy of 0.975, the Support Vector Machine (SVM) with a radial basis kernel function surpassed all other models.

Weakness

Even though the suggested method continually and non-intrusively predicts the occurrence of sickness, it still has to be validated with various chicken breeds and infection types.[12]

(c)Support Vector Machine (SVM) was also used in another study that proposed an early warning algorithm for detecting sick broilers. The posture features of healthy and sick chickens were extracted, the eigenvectors were established, the postures of the broilers were analyzed by machine learning algorithms, and the diseased broilers were predicted.

Weakness

Accuracy rates of 84.2, 60.5, and 91.5% were obtained, but using all the features can yield an accuracy rate of 99.5%. Even though the study proposed a suitable method for small-sample learning and disease diagnosis, the focus was only based on a posture-based algorithm.

2.2 Existing System Comparison Table

S/N	System	Advantage(s)	Drawback(s)
(a)	An automated broiler digestive detector based on deep Convolutional Neural Network models. Wang et al. (2019)	<ul style="list-style-type: none"> Categorize fine-grained abnormal broiler droppings photos as normal and abnormal 	<ul style="list-style-type: none"> No practical Solution suggested at the time
(b)	machine vision-based broiler chicken monitoring system. Okinda et al. (2019)	<ul style="list-style-type: none"> Predicts the occurrence of sickness 	<ul style="list-style-type: none"> Even though the suggested method continually and non-intrusively predicts the occurrence of sickness, it still has to be validated with various chicken breeds and infection types.
(c)	An early warning algorithm for detecting sick broilers.	. The posture features of healthy and sick chickens were extracted, the eigenvectors were established, the postures of the broilers were analyzed by machine learning algorithms, and the diseased broilers were predicted	<ul style="list-style-type: none"> Even though the study proposed a suitable method for small-sample learning and disease diagnosis, the focus was only based on a posture-based algorithm.

2.3 Proposed System

The proposed system aims to use machine learning techniques to recognize poultry diseases by analyzing fecal images. The system will utilize a dataset that is available on Kaggle, which will be used to train the machine learning model to recognize different diseases. In addition to this, the system will also include an additional feature for a chicken vaccination scheduler reminder. This feature will help farmers to ensure that their chickens are properly vaccinated on schedule and will help them to prevent or manage diseases more effectively. Overall, the goal of the system is to assist farmers in recognizing and preventing poultry diseases, which will ultimately lead to a more efficient and profitable poultry farming industry.

CHAPTER THREE: METHODOLOGY

3.1 Requirement Collection (data gathering)

3.1.1 Online Resources

The system utilizes a dataset of poultry diseases obtained from the Kaggle website. This dataset was used to train the machine learning model to recognize different diseases in poultry. The training of the model was done using google colab. Once the model was trained, it was converted into a TensorFlow Lite model using TensorFlow. This allowed the model to be deployed into a mobile application, making it easily accessible to farmers. By using a mobile application, farmers will be able to easily take fecal images of their chickens and have the model analyze the images to determine if any diseases are present. The use of TensorFlow Lite will also ensure that the model can run efficiently on mobile devices with limited resources. Overall, the goal of this system is to provide farmers with an easy to use, accurate, and efficient tool for recognizing and preventing poultry diseases.

3.2 Requirement Analysis

This described the respective system requirements that were captured by the proposed system. Such requirements were classified into:

- Functional requirements.
- Non-functional requirements.

3.2.1 Functional Requirements.

Functional requirements captured the intended functionality of the proposed system. This was expressed as product features, services, tasks, or functions the proposed system would be required to perform.

- The system once implemented would enable farmers to recognize disease outbreaks on the go on the poultry farm.
- The system would also enable a farmer to schedule vaccinations for chickens on the poultry farm.

3.2.2 Non-functional requirements.

Non-functional requirements may also be referred to as system qualities. They captured the desirable properties of the proposed system. Below are some of the non-functional requirements that the proposed system would provide:

- The proposed system would be reliable in the sense that it would provide real-time results of the fecal scan.

- The proposed system would be user-friendly. It would be easy to use for all sorts of poultry farmers.

3.3 System design

The proposed system utilized a machine learning model that was built and updated using python programming. Once the model was trained, it was converted to TFLite format, which is a lightweight format that allows the model to be run on mobile devices. This TFLite model was then integrated into an android app using ML kit and Android Studio.

ML kit is a cloud platform offered by Google that allows developers to easily integrate ML features into their android and iOS apps.

The android app included a feature for a vaccination scheduler that enables farmers to input dates for when they wanted to schedule vaccinations for their chickens. This feature helps farmers to ensure that their chickens were properly vaccinated on schedule and helps them to prevent or manage diseases more effectively. Overall, the goal of the system is to assist farmers in recognizing and preventing poultry diseases, which ultimately leads to a more efficient and profitable poultry farming industry.

3.3.1 Conceptual Design for the System.

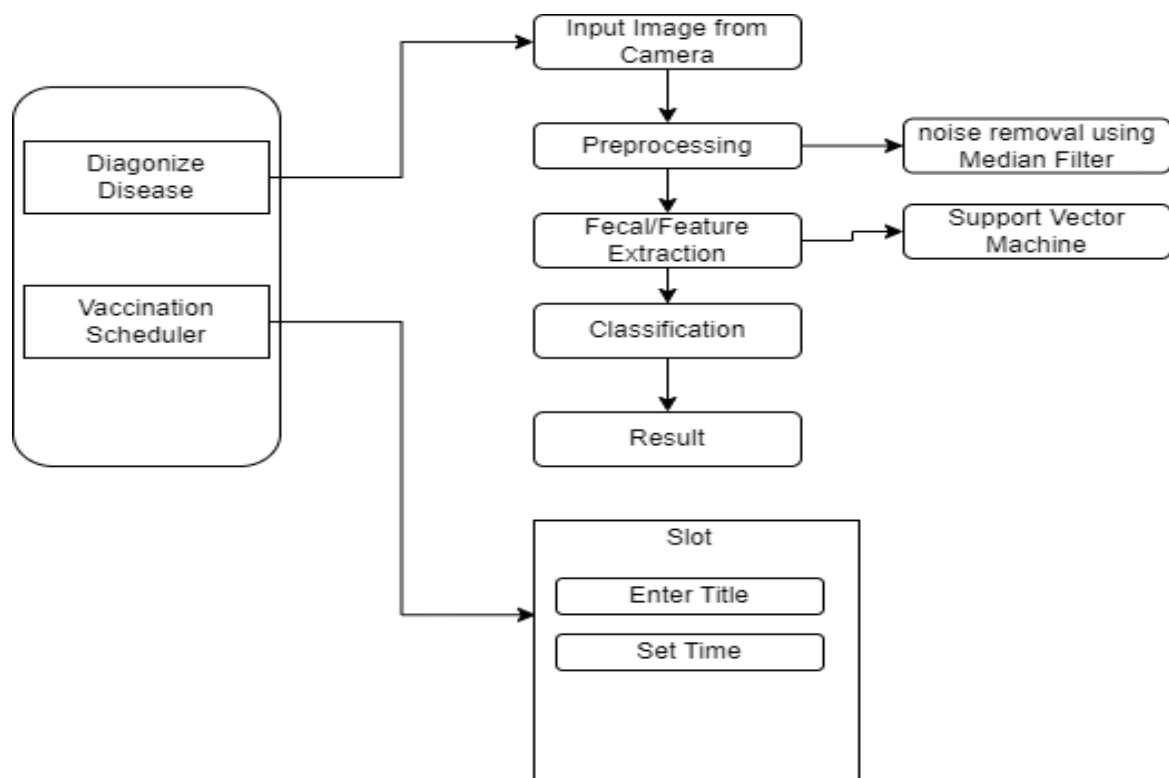


Figure 1. Block Diagram

3.3.2 System Flow Chat

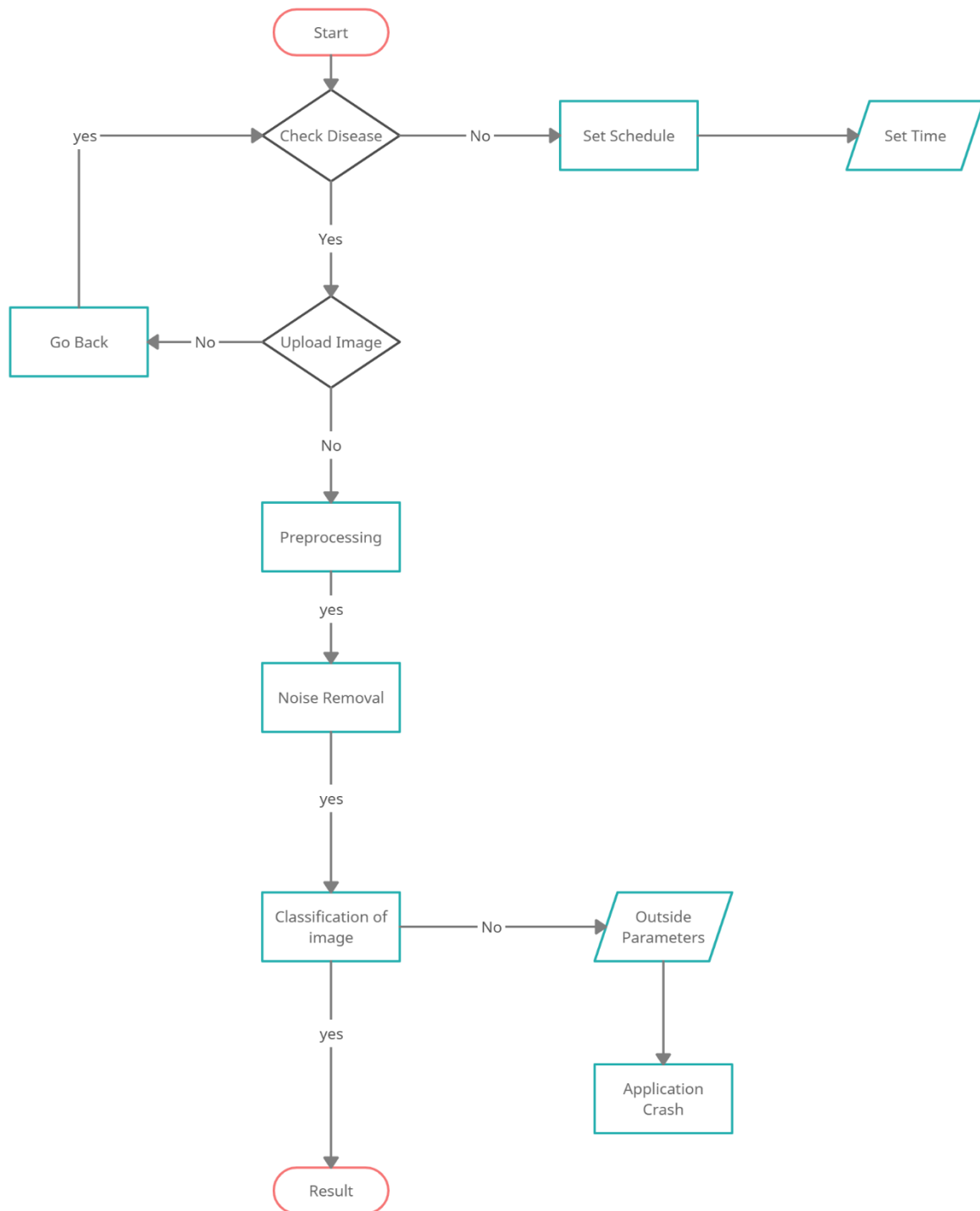


Figure 2. Flow Chart

CHAPTER FIVE: IMPLEMENTATION AND TESTING

5.1 DESIGN AND DEVELOPMENT PLATFORMS

This describes all the platforms which were used to design and develop the system. They include both the simulation and development platforms.

5.1.1 Google Colab

Google Colab is a product from Google Research which allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Google Colab was used for training the dataset and later output the trained model into Tflite format. TensorFlow was used to convert the machine learning model into a TensorFlow Lite model, which was then deployed to the mobile application. This allowed the model to be run on mobile devices, making it more accessible to farmers.

5.1.2 Kaggle

The dataset was obtained from the Kaggle website and was updated and validated before being used in the model.

This process was essential to output a mobile application which recognizes the three poultry diseases we were working on, which ultimately leads to a more efficient and profitable poultry farming industry.

5.1.3 Android Studio

Android Studio is a software which provides an integrated development environment (IDE) optimized for Android apps. The IDE uses Java as an official android development language. The IDE was used to develop the mobile application package that is installed on the user's phone.

5.2 Testing and Validation

Testing was an integral part of the mobile app development process. The mobile application went through various tests which helped me to mitigate the risks associated with any kinds of errors. It focused on ensuring that the functionality, quality, and usability of the mobile application met the user requirements and that the app was ready for launching. The developed system was tested using different techniques and these include:

5.2.1 Unit testing.

This method involved individual testing of the features of the mobile application to ascertain whether it meets the standards required.

5.2.2 Integration Testing.

This involved checking whether the Machine Learning model is deployed into the mobile application successfully.

5.2.3 System Testing.

This was carried out after the development of the full system to ensure that the system meets the stipulated functional and non-functional requirements.

5.2.4 Verification

The system's behaviour was tested by analysing the non-functional requirements of the system. These include;

- **Reliability**

The system is reliable in that it can predict the disease depending on the fecal image loaded.

- **Performance**

The system is fast in its code execution and algorithm implementation that is to say there is no abnormally long delay in performing its purpose.

- **Efficiency**

There is data flow without any inconveniences or limitations once the location is obtained.

- **Usability**

The system is an easy to operate, as it only requires the user to choose what he wants to use the application for detecting the poultry disease if there is any of the three identified.

5.2.5 Validation.

Validation was carried out to assess the operation of the system to ensure that it carries out the intended purpose and needs of the prospective users.

5.3 System validation

System validation is a set of actions used to check the compliance of a system with its purpose and function. Validation will be done to make sure that the modules of the system are performing the desired and proposed goal. The developed mobile application was able to detect the disease depending on the image loaded. This project solves such challenges of diagonalization of disease that can be identified from the fecal images.

5.3.1 Results.

The image was uploaded to the mobile app for disease recognition among the three diseases under study.

Below are the screenshots of the Mobile application Interface on recognizing the disease.

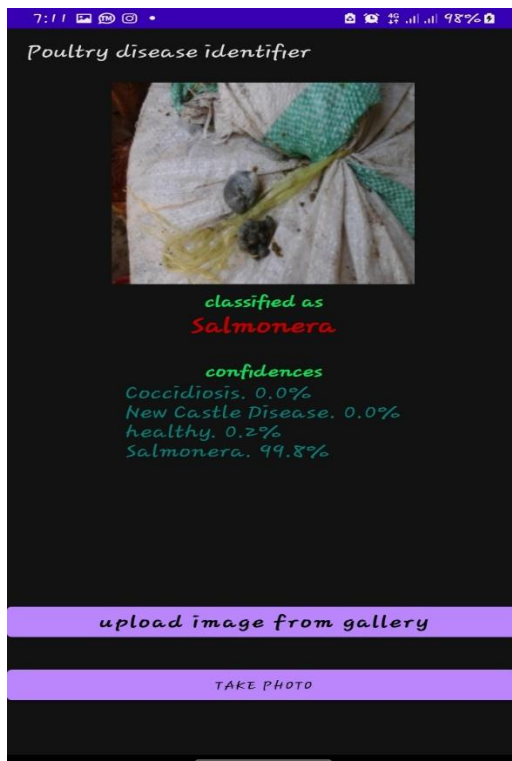


Figure 3. *Salmonera* recognized

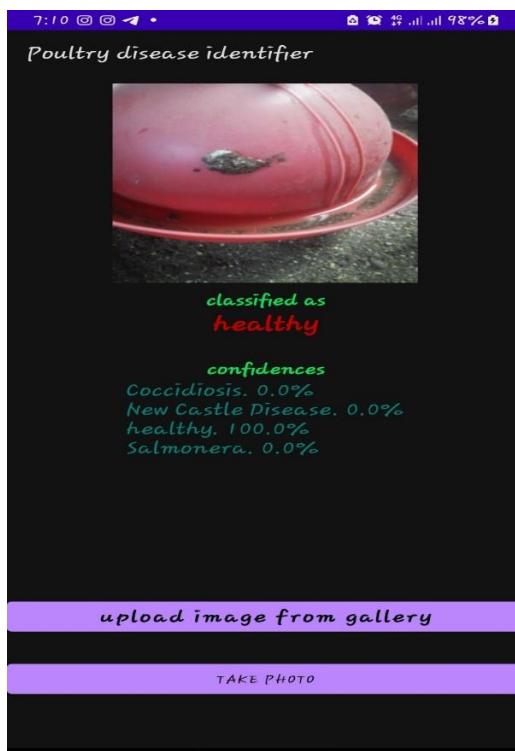


Figure 4. *Healthy*

CHAPTER SIX: DISCUSSIONS AND RECOMMENDATIONS

6.1 Introduction

In this chapter, the summary of the project work done is given, recommendations, challenges and conclusions.

6.2 Summary of the work done

This report describes the design, development, implementation, and testing of a mobile-based system aimed at recognizing if the chicken are infected by any of the three diseases being considered. The diseases are Coccidiosis, Salmonella and Newcastle disease.

Training data which included chicken droppings images (fecal images) were analyzed and used to train a model which was converted to a TensorFlow Lite model which I loaded into the mobile application. I used Google Colab to carry out the analysis, training and reproducing the model that is viable for use in Android Studio to develop the application. The resulting model which is a .h5 file was converted to a TensorFlow Lite model which I imported in Android Studio to run it along with the Java code.

6.3 Critical Analysis and Comparison with other Systems

The main objective of this research was to design a mobile-based machine learning system that would recognize poultry diseases, specifically three of them.

Compared to the existing system which requires gathering farmers to be sensitized, this developed system uses a mobile phone to get all the required information. The existing systems are seasonal and cannot be accessed at any time hence this system being helpful to the farmers. I included a feature for helping the farmer to schedule the vaccination of the chicken.

6.4 Recommendations

On carrying out this research, the following are the future recommendations for the research to be a perfect success.

- The system will be used to carry out poultry disease recognition. This is cited as a recommendation because mobile application will simplify work as the information is easily got by the farmers in case of any updates without including the third party.
- There is need for a different code base written in Swift or Dart to accommodate iPhone users.
- This mobile application can also be expanded to carry out automation of the poultry farm. By automation, I mean, automating water flow and lighting of the poultry farm.

6.5 Challenges.

- This model may not be able to detect new diseases in case they come up.
- This system is limited to only three diseases, this is mainly because those are the only diseases that can be recognized from chicken droppings.
- Acquiring information required to develop the system from farmers was a challenge due to language barrier. Farmers act offended easily.
- Difficulty in getting chicken with the specific disease.
- There is always a percentage error in findings of Artificial Intelligence. This is why in the mobile application, upon diagnosing I included confidence percentage.

6.6 Conclusions.

The developed system ensures that there is minimal wastage of time like agricultural officers coming and first collecting a number of farmers to be talked too hence costs attached. The developed system is easy to access information by the farmers and it is a cheaper method compared to other existing systems. Despite the challenges faced during the development of this project prototype, at least most of the stated specific objectives were achieved.

7.0. References

- [1] “Retrospective study on cattle and poultry diseases in Uganda - PMC.” <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6137842/> (accessed Aug. 01, 2022).
- [2] “973-2769-1-PB.pdf.”
- [4] “Coccidiosis Disease Kills Hundreds of Poultry in Amuru :: Uganda Radio network.” <https://ugandaradionetwork.net/story/coccidiosis-disease-kills-hundreds-of-poultry-in-amuru> (accessed Aug. 02, 2022).
- [5] C. Okinda et al., “A review on computer vision systems in the monitoring of poultry: A welfare perspective,” *Artif. Intell. Agric.*, vol. 4, pp. 184–208, 2020, DOI: 10.1016/j.aiia.2020.09.002.
- [6] J. Johnson, W. R.-E. parasitology, and undefined 1970, “Anticoccidial drugs: lesion scoring techniques in battery and floor-pen experiments with chickens,” Elsevier, Accessed: Aug. 02, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/abs/pii/0014489470900639>
- [7]
- [8] “Wise: Development of a real-time reverse-transcription... - Google Scholar.” https://scholar.google.com/scholar_lookup?author=M.+G.+Wise&author=D.+L.+Suar ez&author=B.+S.+Seal&author=J.+C.+Pedersen&author=D.+A.+Senne&author=D.+J.+King+&publication_year=2004&title=Development+of+a+real-time+reverse-transcription+PCR+for+detection+of (accessed Aug. 02, 2022).
- [9] X. Zhuang, M. Bi, J. Guo, S. Wu, T. Z.-C. and E. in, and undefined 2018, “Development of an early warning algorithm to detect sick broilers,” Elsevier, Accessed: Aug. 02, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0168169917308244>
- [10] C. Okinda et al., “A machine vision system for early detection and prediction of sick birds: A broiler chicken model,” Elsevier, Accessed: Aug. 02, 2022. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S1537511019308438>
- [11] “Chicken Disease Image Classification | Kaggle.” <https://www.kaggle.com/datasets/allandclive/chicken-disease-1> (accessed Aug. 02, 2022).

8.0 APPENDICES

Machine learning code for loading, analysis, and training Model

```
//Import needed modules

import numpy as np

import pandas as pd

import os

import time

import matplotlib.pyplot as plt

import cv2

import seaborn as sns

sns.set_style('darkgrid')

import shutil

from sklearn.metrics import confusion_matrix, classification_report

from sklearn.model_selection import train_test_split

import tensorflow as tf

from tensorflow import keras

from tensorflow.keras.preprocessing.image import ImageDataGenerator

from tensorflow.keras.layers import Dense, Activation, Dropout, Conv2D,

MaxPooling2D, BatchNormalization

from tensorflow.keras.optimizers import Adam, Adamax

from tensorflow.keras.metrics import categorical_crossentropy

from tensorflow.keras import regularizers

from tensorflow.keras.models import Model

# prevent annoying tensorflow warning

import logging

logging.getLogger("tensorflow").setLevel(logging.ERROR)

import warnings

pd.set_option('max_columns', None)

pd.set_option('max_rows', 90)

warnings.simplefilter("ignore")
```

```

//mount google drive to access training data
from google.colab import drive
drive.mount('/content/drive')

//read data
sdir=r'../content/drive/MyDrive/archive/Train/'
csvpath=r'../content/drive/MyDrive/archive/train_data.csv'
df=pd.read_csv(csvpath)
df.columns=['filepaths', 'labels' ]
df['filepaths']=df['filepaths'].apply(lambda x: os.path.join(sdir,x))
print(df.head())

trsplit=.9
vsplit=.05
dsplit =vsplit/(1-trsplit)
strat=df['labels']

train_df, dummy_df=train_test_split(df, train_size=.9, shuffle=True, random_state=123,
stratify=strat)

strat=dummy_df['labels']

test_df, valid_df=train_test_split(dummy_df, train_size=dsplit, shuffle=True,
random_state=123, stratify=strat)

print('train_df lenght: ', len(train_df), ' test_df length: ', len(test_df), ' valid_df length: ',
len(valid_df))

classes=list(train_df['labels'].unique())

class_count = len(classes)

groups=df.groupby('labels')

```

```

print('{0:^30s} {1:^13s}'.format('CLASS', 'IMAGE COUNT'))
for label in train_df['labels'].unique():
    group=groups.get_group(label)
    print('{0:^30s} {1:^13s}'.format(label, str(len(group)))).

```

```

//trim train Data frame
def trim (df, max_size, min_size, column):
    df=df.copy()
    original_class_count= len(list(df[column].unique()))
    print ('Original Number of classes in dataframe: ', original_class_count)
    sample_list=[]
    groups=df.groupby(column)
    for label in df[column].unique():
        group=groups.get_group(label)
        sample_count=len(group)
        if sample_count> max_size :
            strat=group[column]
            samples,_=train_test_split(group, train_size=max_size, shuffle=True,
random_state=123, stratify=strat)
            sample_list.append(samples)

```



```

        elif sample_count >= min_size:
            sample_list.append(group)
    df=pd.concat(sample_list, axis=0).reset_index(drop=True)
    final_class_count= len(list(df[column].unique()))
    if final_class_count != original_class_count:
        print ('*** WARNING*** dataframe has a reduced number of classes' )
    balance=list(df[column].value_counts())
    print (balance)
    return df

max_samples=500
min_samples=0
column = 'labels'
train_df=trim(train_df, max_samples, min_samples, column)
img_size=(224,224)
working_dir=r'./'

// split data to test and train

//Read classes

batch_size=20 # We will use and EfficientNetB5 model, with image size of (224, 224) this size
should not cause resource error

trgen=ImageDataGenerator(horizontal_flip=True,rotation_range=20, width_shift_range=.2,
height_shift_range=.2, zoom_range=.2 )

t_and_v_gen=ImageDataGenerator()

train_gen=train_gen.flow_from_dataframe(train_df, x_col='filepaths', y_col='labels',
target_size=img_size,

class_mode='categorical', color_mode='rgb', shuffle=True, batch_size=batch_size)

valid_gen=t_and_v_gen.flow_from_dataframe(valid_df, x_col='filepaths', y_col='labels',
target_size=img_size,

                                class_mode='categorical', color_mode='rgb', shuffle=False,
batch_size=batch_size)

# for the test_gen we want to calculate the batch size and test steps such that batch_size X
test_steps= number of samples in test set

```

```

# this insures that we go through all the sample in the test set exactly once.

length=len(test_df)

test_batch_size=sorted([int(length/n) for n in range(1,length+1) if length % n ==0 and
length/n<=80],reverse=True)[0]

test_steps=int(length/test_batch_size)

test_gen=t_and_v_gen.flow_from_dataframe(test_df, x_col='filepaths', y_col='labels',
target_size=img_size,

class_mode='categorical', color_mode='rgb', shuffle=False, batch_size=test_batch_size)

# from the generator we can get information we will need later

classes=list(train_gen.class_indices.keys())

class_indices=list(train_gen.class_indices.values())

class_count=len(classes)

labels=test_gen.labels

print ( 'test batch size: ',test_batch_size, ' test steps: ', test_steps, ' number of classes : ',
class_count)

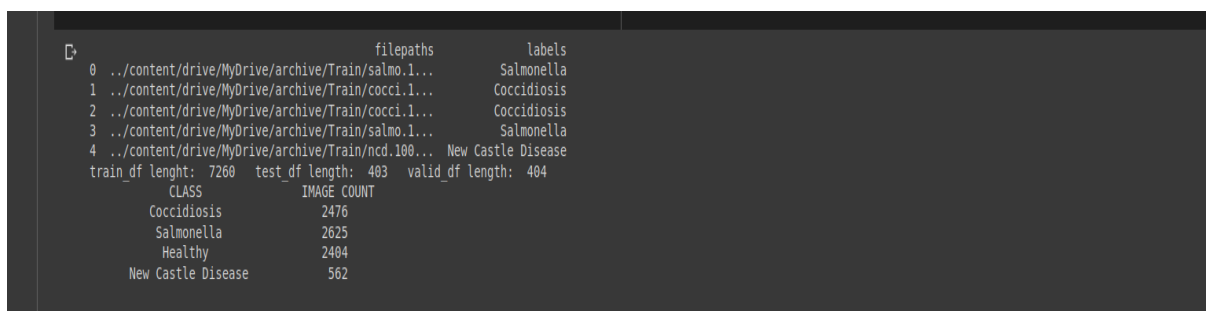
print ('{0:^25s}{1:^12s}'.format('class name', 'class index'))

for class, index in zip(classes, class_indices):

    print(f'{class:^25s}{str(index):^12s}')

//output

```



	filepaths	labels
0	../content/drive/MyDrive/archive/Train/salmo.1...	Salmonella
1	../content/drive/MyDrive/archive/Train/cocci.1...	Coccidiosis
2	../content/drive/MyDrive/archive/Train/cocci.1...	Coccidiosis
3	../content/drive/MyDrive/archive/Train/salmo.1...	Salmonella
4	../content/drive/MyDrive/archive/Train/ncd.100...	New Castle Disease

train_df length:	test_df length:	valid_df length:
7260	483	404

CLASS	IMAGE COUNT
Coccidiosis	2476
Salmonella	2625
Healthy	2494
New Castle Disease	562

```

//create a function to show sample training images

def show_image_samples(gen ):

    t_dict=gen.class_indices

    classes=list(t_dict.keys())

    images,labels=next(gen) # get a sample batch from the generator

    plt.figure(figsize=(20, 20))

```

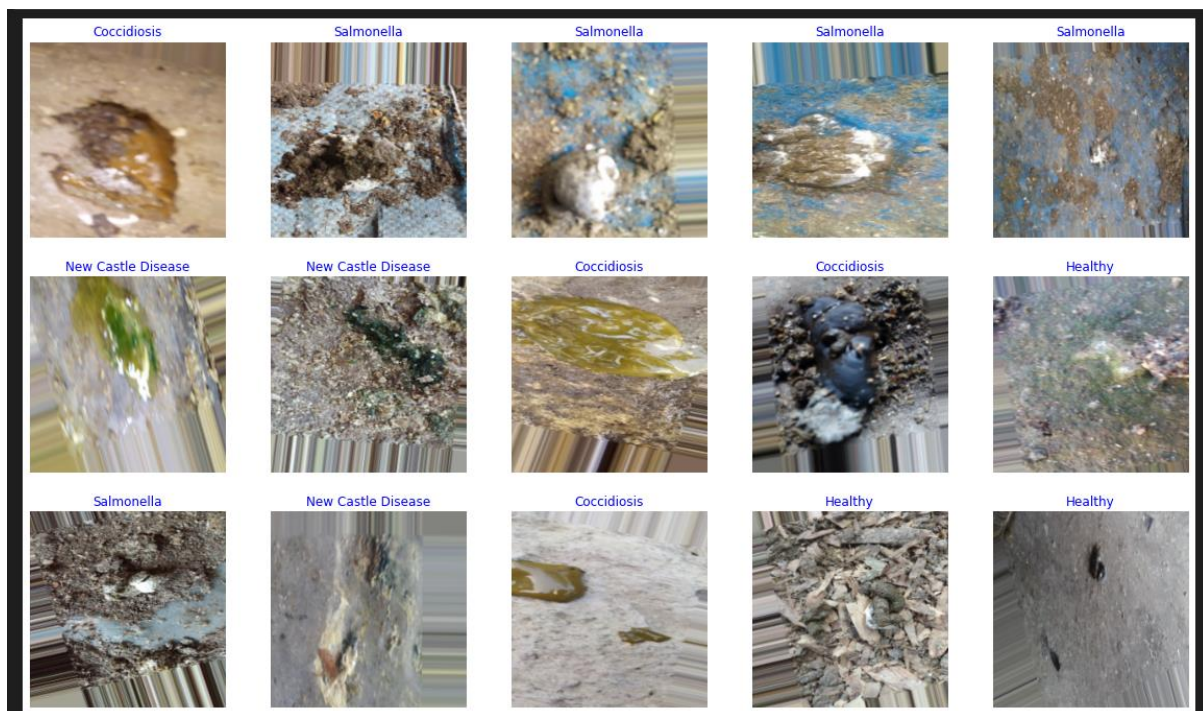
```

length=len(labels)
if length<25: #show maximum of 25 images
    r=length
else:
    r=25
for i in range(r):
    plt.subplot(5, 5, i + 1)
    image=images[i] /255
    plt.imshow(image)
    index=np.argmax(labels[i])
    class_name=classes[index]
    plt.title(class_name, color='blue', fontsize=12)
    plt.axis('off')
plt.show()

show_image_samples(train_gen )

// output

```



```

// Create model using transfer learning
img_shape=(img_size[0], img_size[1], 3)
model_name='EfficientNetB5'

base_model=tf.keras.applications.efficientnet.EfficientNetB5(include_top=False,
weights="imagenet",input_shape=img_shape, pooling='max')

# Note you are always told NOT to make the base model trainable initially- that is WRONG
you get better results leaving it trainable

base_model.trainable=True

x=base_model.output
x=BatchNormalization(axis=-1, momentum=0.99, epsilon=0.001 )(x)
x=Dense(1024,kernel_regularizer=regularizers.l2(10.016),activity_regularizer=regularizers.l1
(0.006), bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.3, seed=123)(x)
x=Dense(128,kernel_regularizer=regularizers.l2(l=0.016),activity_regularizer=regularizers.l1
(0.006), bias_regularizer=regularizers.l1(0.006) ,activation='relu')(x)
x=Dropout(rate=.45, seed=123)(x)
output=Dense(class_count, activation='softmax')(x)
model=Model(inputs=base_model.input, outputs=output)

lr=.001 # start with this learning rate

model.compile(Adamax(learning_rate=lr),loss='categorical_crossentropy',
metrics=['accuracy'])

```

```
// Create keras Callback to continue or halt the training
class ASK(keras.callbacks.Callback):
    def __init__(self, model, epochs, ask_epoch): # initialization of the callback
        super(ASK, self).__init__()
        self.model=model
        self.ask_epoch=ask_epoch
        self.epochs=epochs
        self.ask=True # if True query the user on a specified epoch

    def on_train_begin(self, logs=None): # this runs on the beginning of training
        if self.ask_epoch == 0:
            print('you set ask_epoch = 0, ask_epoch will be set to 1', flush=True)
            self.ask_epoch=1
        if self.ask_epoch >= self.epochs: # you are running for epochs but ask_epoch>epochs
            print('ask_epoch >= epochs, will train for ', epochs, ' epochs', flush=True)
            self.ask=False # do not query the user
        if self.epochs == 1:
            self.ask=False # running only for 1 epoch so do not query user
        else:
            print("Training will proceed until epoch", ask_epoch, ' then you will be asked to')
            print(' enter H to halt training or enter an integer for how many more epochs to run
then be asked again')

        self.start_time= time.time() # set the time at which training started

    def on_train_end(self, logs=None): # runs at the end of training
        tr_duration=time.time() - self.start_time # determine how long the training cycle lasted
        hours = tr_duration // 3600
        minutes = (tr_duration - (hours * 3600)) // 60
        seconds = tr_duration - ((hours * 3600) + (minutes * 60))
```

```

        msg = f'training elapsed time was {str(hours)} hours, {minutes:4.1f} minutes,
{seconds:4.2f} seconds)'

        print (msg, flush=True) # print out training duration time


def on_epoch_end(self, epoch, logs=None): # method runs on the end of each epoch
    if self.ask: # are the conditions right to query the user?
        if epoch + 1 ==self.ask_epoch: # is this epoch the one for quering the user?
            print('\n Enter H to end training or  an integer for the number of additional epochs
to run then ask again')

            ans=input()

            if ans == 'H' or ans =='h' or ans == '0': # quit training for these conditions
                print ('you entered ', ans, ' Training halted on epoch ', epoch+1, ' due to user
input\n', flush=True)

                self.model.stop_training = True # halt training
            else: # user wants to continue training
                self.ask_epoch += int(ans)
                if self.ask_epoch > self.epochs:
                    print('\nYou specified maximum epochs of as ', self.epochs, ' cannot train for ',
self.ask_epoch, flush =True)
                    else:
                        print ('you entered ', ans, ' Training will continue to epoch ', self.ask_epoch, flush=True)


//Instantiate custom callback and create 2 callbacks to control learning rate and early stop

epochs=40
# ask_epoch=40
# ask=ASK(model, epochs, ask_epoch)

rlronp=tf.keras.callbacks.ReduceLROnPlateau(monitor="val_loss",factor=0.5,
patience=2,verbose=1)

```

```

estop=tf.keras.callbacks.EarlyStopping(monitor="val_loss",patience=4,verbose=1,restore_b
st_weights=True)

callbacks=[rlronp, estop]

// Train the model

history=model.fit(x=train_gen, epochs=epochs, verbose=1, callbacks=callbacks,
validation_data=valid_gen,

validation_steps=None, shuffle=False, initial_epoch=0)

```

//Output during and After training

```

Output exceeds the size limit. Open the full output data in a text editor
Epoch 1/40
100/100 [=====] - 1020s 10s/step - loss: 25.0113 - accuracy: 0.7060 - val_loss: 23.6349 - val_accuracy: 0.8886 - lr: 0.0010
Epoch 2/40
100/100 [=====] - 70s 698ms/step - loss: 16.7517 - accuracy: 0.8530 - val_loss: 14.6118 - val_accuracy: 0.9109 - lr: 0.0010
Epoch 3/40
100/100 [=====] - 70s 698ms/step - loss: 11.9049 - accuracy: 0.9005 - val_loss: 10.2511 - val_accuracy: 0.9158 - lr: 0.0010
Epoch 4/40
100/100 [=====] - 72s 713ms/step - loss: 8.5434 - accuracy: 0.9155 - val_loss: 7.1472 - val_accuracy: 0.9505 - lr: 0.0010
Epoch 5/40
100/100 [=====] - 70s 701ms/step - loss: 6.1190 - accuracy: 0.9340 - val_loss: 5.2280 - val_accuracy: 0.9381 - lr: 0.0010
Epoch 6/40
100/100 [=====] - 70s 697ms/step - loss: 4.4370 - accuracy: 0.9570 - val_loss: 3.7301 - val_accuracy: 0.9455 - lr: 0.0010
Epoch 7/40
100/100 [=====] - 70s 699ms/step - loss: 3.2740 - accuracy: 0.9445 - val_loss: 2.8445 - val_accuracy: 0.9257 - lr: 0.0010
Epoch 8/40
100/100 [=====] - 71s 710ms/step - loss: 2.4569 - accuracy: 0.9565 - val_loss: 2.1230 - val_accuracy: 0.9455 - lr: 0.0010
Epoch 9/40
100/100 [=====] - 70s 699ms/step - loss: 1.8689 - accuracy: 0.9650 - val_loss: 1.6573 - val_accuracy: 0.9554 - lr: 0.0010
Epoch 10/40
100/100 [=====] - 70s 698ms/step - loss: 1.4522 - accuracy: 0.9720 - val_loss: 1.3223 - val_accuracy: 0.9455 - lr: 0.0010
Epoch 11/40
100/100 [=====] - 70s 699ms/step - loss: 1.1523 - accuracy: 0.9760 - val_loss: 1.0621 - val_accuracy: 0.9480 - lr: 0.0010
Epoch 12/40
100/100 [=====] - 71s 705ms/step - loss: 0.9606 - accuracy: 0.9735 - val_loss: 0.8876 - val_accuracy: 0.9505 - lr: 0.0010
Epoch 13/40
...
Epoch 39: ReduceLRonPlateau reducing learning rate to 0.0001250000059371814.
Restoring model weights from the end of the best epoch: 35.
100/100 [=====] - 71s 712ms/step - loss: 0.2147 - accuracy: 0.9975 - val_loss: 0.3250 - val_accuracy: 0.9629 - lr: 2.5000e-04
Epoch 39: early stopping

```

Figure 5.Training process of the model

```

//Plot training data

def tr_plot(tr_data, start_epoch):

    #Plot the training and validation data

    tacc=tr_data.history['accuracy']

    tloss=tr_data.history['loss']

    vacc=tr_data.history['val_accuracy']

    vloss=tr_data.history['val_loss']

```

```

Epoch_count=len(tacc)+ start_epoch
Epochs=[]
for i in range (start_epoch ,Epoch_count):
    Epochs.append(i+1)
index_loss=np.argmin(vloss)# this is the epoch with the lowest validation loss
val_lowest=vloss[index_loss]
index_acc=np.argmax(vacc)
acc_highest=vacc[index_acc]
plt.style.use('fivethirtyeight')
sc_label='best epoch= '+ str(index_loss+1 +start_epoch)
vc_label='best epoch= '+ str(index_acc + 1+ start_epoch)
fig,axes=plt.subplots(nrows=1, ncols=2, figsize=(20,8))
axes[0].plot(Epochs,tloss, 'r', label='Training loss')
axes[0].plot(Epochs,vloss,'g',label='Validation loss' )
axes[0].scatter(index_loss+1 +start_epoch,val_lowest, s=150, c= 'blue', label=sc_label)
axes[0].set_title("Training and Validation Loss")
axes[0].set_xlabel('Epochs')
axes[0].set_ylabel('Loss')
axes[0].legend()
axes[1].plot (Epochs,tacc,'r',label= 'Training Accuracy')
axes[1].plot (Epochs,vacc,'g',label= 'Validation Accuracy')
axes[1].scatter(index_acc+1 +start_epoch,acc_highest, s=150, c= 'blue', label=vc_label)
axes[1].set_title("Training and Validation Accuracy")
axes[1].set_xlabel('Epochs')
axes[1].set_ylabel('Accuracy')
axes[1].legend()
plt.tight_layout
plt.show()

tr_plot(history,0)

```


//Output

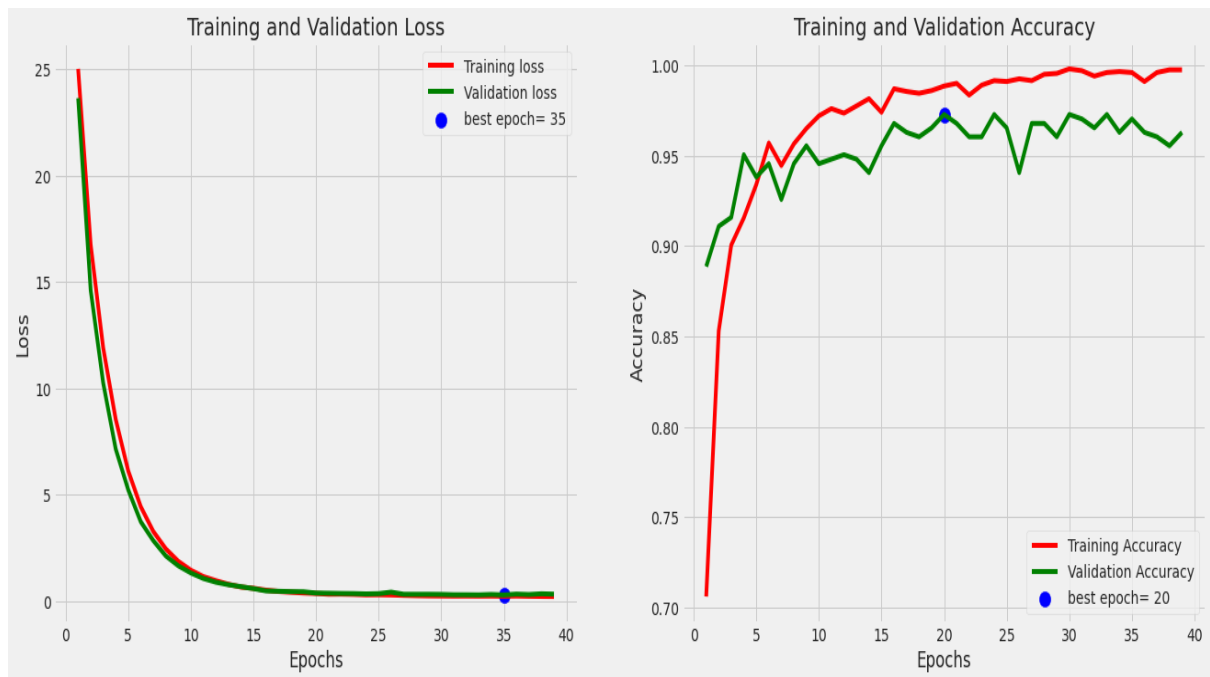


Figure 6. Graph showing both Training Vs Validation Loss and Training Vs Validation Accuracy

// Classification Report

		Confusion Matrix			
Actual	Coccidiosis	117	6	1	0
	Healthy	2	115	1	2
	New Castle Disease	0	0	28	0
	Salmonella	1	2	1	127
		Coccidiosis	Healthy	New Castle Disease	Salmonella
		Predicted			

Figure 7. Confusion Matrix

Classification Report:				
	precision	recall	f1-score	support
Coccidiosis	0.9750	0.9435	0.9590	124
Healthy	0.9350	0.9583	0.9465	120
New Castle Disease	0.9032	1.0000	0.9492	28
Salmonella	0.9845	0.9695	0.9769	131
accuracy			0.9603	403
macro avg	0.9494	0.9678	0.9579	403
weighted avg	0.9612	0.9603	0.9604	403

Figure 8. Classification Report

```

// Save the model
subject='chickens'
acc=str(( 1-errors/tests) * 100)
index=acc.rfind('.')
acc=acc[:index + 3]
save_id= subject + '_' + str(acc) + '.h5'
model_save_loc=os.path.join(working_dir, save_id)
model.save(model_save_loc)
print ('model was saved as ', model_save_loc )

// Covert Model to tflite
from keras.models import load_model
model=load_model ("/content/drive/MyDrive/archive/chickens_93.79.h5")

TF_LITE_MODEL_FILE_NAME = "tflite_model.tflite"
tf_lite_converter=tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model=tf_lite_converter.convert()
tflite_model_name=TF_LITE_MODEL_FILE_NAME
open(tflite_model_name,"wb").write(tflite_model)
# convert_bytes(get_file_size(TF_LITE_MODEL_FILE_NAME),"KB")

#Covert the model
converter=tf.lite.TFLiteConverter.from_keras_model(model)
tflite_model=converter.convert()

#Or using another method
with open('tflite_model_another.tflite','wb') as f:
    f.write(tflite_model)

```

Android Studio Codes used

// Launcher Activity

```
package com.example.poultrydiseaserecognitionapp;
```

```
import androidx.appcompat.app.AppCompatActivity;
```

```
import android.content.Intent;
```

```
import android.os.Bundle;
```

```
import android.view.View;
```

```
import android.widget.Button;
```

```
public class MainActivity3 extends AppCompatActivity {
```

```
    @Override
```

```
    protected void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
```

```
        setContentView(R.layout.activity_main3);
```

```
        Button Diagonize = findViewById(R.id.button1);
```

```
        Button Scheduler = findViewById(R.id.button2);
```

```
        Diagonize.setOnClickListener(new View.OnClickListener() {
```

```
            @Override
```

```
            public void onClick(View view) {
```

```
                switchActivities();
```

```
            }
```

```
        });
```

```
        Scheduler.setOnClickListener(new View.OnClickListener() {
```

```

        public void onClick(View view) {
            gotoscheduler();
        }
    });
}

private void gotoscheduler() {
    Intent switchActivityIntent = new Intent(this, MainActivity2.class);
    startActivity(switchActivityIntent);
}

private void switchActivities() {
    Intent switchActivityIntent = new Intent(this, MainActivity.class);
    startActivity(switchActivityIntent);
}
}

//Xml code for launcher Activity
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    tools:context=".MainActivity3">

    <TextView
        android:layout_width="wrap_content"

```

```

        android:layout_height="wrap_content"
        android:textSize="20sp"

        android:text="@string/poultry_disease_recognition_and_vaccination_scheduling_application"
    "
```

/>

<!-- Button 1 -->

<Button

```

    android:id="@+id/button1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/check_for_chicken_disease" />

```

<!-- Button 2 -->

<Button

```

    android:id="@+id/button2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="@string/schedule_chicken_vaccination" />

```

</LinearLayout>

//Check disease Activity code (kotlin file code)

```

package com.example.poultrydiseaserecognitionapp;

import androidx.appcompat.app.AppCompatActivity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

```

```

public class MainActivity3 extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main3);

        Button Diagonize = findViewById(R.id.button1);
        Button Scheduler = findViewById(R.id.button2);

        Diagonize.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                switchActivities();
            }
        });

        Scheduler.setOnClickListener(new View.OnClickListener() {
            public void onClick(View view) {
                gotoscheduler();
            }
        });
    }

    private void gotoscheduler() {
        Intent switchActivityIntent = new Intent(this, MainActivity2.class);
        startActivity(switchActivityIntent);
    }
}

```

```

private void switchActivities() {
    Intent switchActivityIntent = new Intent(this, MainActivity.class);
    startActivity(switchActivityIntent);
}
}

//Xml code for check disease Activity
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity"
    >

    <Button
        android:id="@+id/button"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_alignParentBottom="true"
        android:layout_centerInParent="true"
        android:text="@string/upload_image_from_gallery"
        android:textAllCaps="false"
        android:textSize="21sp"
        android:textStyle="bold" />

    <ImageView
        android:layout_width="370sp"
        android:layout_height="370sp"
        android:layout_centerHorizontal="true"

```



```

        android:id="@+id/imageView"
        android:layout_marginTop="10sp"
        android:alpha="2"
    />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text="@string/classified_as"
    android:textStyle="bold"
    android:textColor="#0BE15B"
    android:textSize="20sp"
    android:id="@+id/classified"
    android:layout_below="@+id/imageView"
    android:layout_marginTop="10sp"
    />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text=""
    android:textColor="#C30000"
    android:textStyle="bold"
    android:textSize="27sp"
    android:id="@+id/result"
    android:layout_below="@+id/classified"
    />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"

```

```

        android:id="@+id/confidencesText"
        android:text="@string/confidences"
        android:textStyle="bold"
        android:textColor="#0BE15B"
        android:textSize="20sp"
        android:layout_centerHorizontal="true"
        android:layout_below="@+id/result"
        android:layout_marginTop="30sp"
    />
<TextView
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:text=""
    android:textColor="#07807A"
    android:textSize="22sp"
    android:id="@+id/confidence"
    android:layout_below="@+id/confidencesText"
/>

```

</RelativeLayout>

// java code for MainActivity class for the xml file above

```

package com.example.poultrydiseaseidentifier;

import androidx.annotation.Nullable;
import androidx.appcompat.app.AppCompatActivity;
import android.Manifest;
import android.content.Intent;
import android.content.pm.PackageManager;

```

```

import android.graphics.Bitmap;
import android.net.Uri;
import android.os.Bundle;
import android.provider.MediaStore;
import android.view.View;
import android.widget.Button;
import android.widget.ImageView;
import android.widget.TextView;
import android.widget.Toast;


import com.example.poultrydiseaseidentifier.ml.ModelUnquant;


import org.tensorflow.lite.DataType;
import org.tensorflow.lite.support.tensorbuffer.TensorBuffer;


import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.ByteOrder;


public class MainActivity extends AppCompatActivity {

    TextView result, confidence;
    ImageView imageView;
    Button picture,camera;
    int imageSize = 224;


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

```

```

result = findViewById(R.id.result);
confidence = findViewById(R.id.confidence);
imageView = findViewById(R.id.imageView);
picture = findViewById(R.id.button);
camera = findViewById(R.id.button_camera);

picture.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {
        if (checkSelfPermission(Manifest.permission.READ_EXTERNAL_STORAGE)
== PackageManager.PERMISSION_GRANTED) {
            // Launch gallery picker if we have permission
            Intent pickImageIntent = new Intent(Intent.ACTION_PICK);
            pickImageIntent.setType("image/*");
            startActivityForResult(pickImageIntent, 1);
        } else {
            // Request permission to access the gallery if we don't have it
            requestPermissions(new
String[]{Manifest.permission.READ_EXTERNAL_STORAGE}, 100);
        }
    }
});

camera.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View view) {

```

```

        if (checkSelfPermission(Manifest.permission.CAMERA) ==
PackageManager.PERMISSION_GRANTED) {

            // Launch the camera activity if we have permission

            Intent takePictureIntent = new
Intent(MediaStore.ACTION_IMAGE_CAPTURE);

            if (takePictureIntent.resolveActivity(getPackageManager()) != null) {

                startActivityForResult(takePictureIntent, 2);

            }
        } else {

            // Request permission to access the camera if we don't have it

            requestPermissions(new String[]{Manifest.permission.CAMERA}, 101);

        }

    }

});

}

public void classifyImage(Bitmap image){

    try {

        ModelUnquant model = ModelUnquant.newInstance(getApplicationContext());

        // Creates inputs for reference.

        TensorBuffer inputFeature0 = TensorBuffer.createFixedSize(new int[]{1, 224, 224,
3}, DataType.FLOAT32);

        ByteBuffer byteBuffer =ByteBuffer.allocateDirect(4 * imageSize * imageSize * 3);
        byteBuffer.order(ByteOrder.nativeOrder());

        int [] intValues = new int[imageSize * imageSize];

        image.getPixels(intValues, 0,image.getWidth(),0,0, image.getWidth(),
image.getHeight());

        //      int pixel =0;

```

```

for (int i=0; i < imageSize; i++){
    for (int j=0; j < imageSize; j++){
        int val =intValues[j]; //RGB
        byteBuffer.putFloat(((val >> 16) & 0xFF)*(1.f/255.f));
        byteBuffer.putFloat(((val >> 8) & 0xFF)*(1.f/255.f));
        byteBuffer.putFloat((val & 0xFF) *(1.f/255.f));
    }
}
inputFeature0.loadBuffer(byteBuffer);

// Runs model inference and gets result.
ModelUnquant.Outputs outputs = model.process(inputFeature0);
TensorBuffer outputFeature0 = outputs.getOutputFeature0AsTensorBuffer();

float[] confidences = outputFeature0.getFloatArray();
int maxPos =0;
float maxConfidence =0;
for (int i=0; i < confidences.length;i++ ){
    if (confidences[i] > maxConfidence){
        maxConfidence =confidences[i];
        maxPos = i;
    }
}

String[] classes = {"Coccidiosis","New Castle Disease","healthy","Salmonera"};

result.setText(classes[maxPos]);

String s = "";

```

```

        for (int i =0; i< classes.length; i++){
            s += String.format("%s. %.1f%%\n", classes[i],confidences[i] * 100);
        }
        confidence.setText(s);

        // Releases model resources if no longer used.
        model.close();
    } catch (IOException e) {
        // TODO Handle the exception
    }

}

@Override
public void onActivityResult(int requestCode, int resultCode, @Nullable Intent data) {
    super.onActivityResult(requestCode, resultCode, data);

    if (requestCode == 1 && resultCode == RESULT_OK) {

        // Get the selected image from the intent
        Uri selectedImage = data.getData();
        try {
            // Convert the image to a Bitmap
            Bitmap image = MediaStore.Images.Media.getBitmap(this.getContentResolver(),
selectedImage);

            // Set the bitmap in the ImageView preview
            imageView.setImageBitmap(image);

            // Pass the Bitmap to the classifyImage() method for processing
            classifyImage(image);

        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

```

```

    }

    } else if (requestCode == 2) {
        Bundle extras = data.getExtras();
        Bitmap image = (Bitmap) extras.get("data");
        classifyImage(image);
    } else if (requestCode == 101) {
        // Permission granted, launch the camera activity
        Intent takePictureIntent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
        if (takePictureIntent.resolveActivity(getPackageManager()) != null) {
            startActivityForResult(takePictureIntent, 2);
        }
    } else {
        // Permission denied, show a message or do something else
        // ...
    }

    if (result == null) {
        // Display toast message
        Toast.makeText(getApplicationContext(), "No fecal features detected",
        Toast.LENGTH_SHORT).show();
    } else {
        // Continue with processing the result
    }

}

}

//Java code for Setting schedule

```



```

package com.example.poultrydiseaserecognitionapp;

import androidx.appcompat.app.AppCompatActivity;
import android.app.AlarmManager;
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.WindowManager;
import android.widget.DatePicker;
import android.widget.EditText;
import android.widget.TimePicker;
import android.widget.Toast;

import java.util.Calendar;

public class MainActivity2 extends AppCompatActivity implements View.OnClickListener {

    private int notificationId = 1;

    // Declare the pendingIntent as a class field
    private PendingIntent pendingIntent;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);

        // Set onClick Listener
        findViewById(R.id.setBtn).setOnClickListener(this);
        findViewById(R.id.cancelBtn).setOnClickListener(this);
    }
}

```

```

        // Keep the screen on
        getWindow().addFlags(WindowManager.LayoutParams.FLAG_KEEP_SCREEN_ON);

    }

    @Override
    protected void onStop() {
        super.onStop();

        // Cancel the alarm when the activity is stopped
        AlarmManager alarmManager = (AlarmManager)
        getSystemService(ALARM_SERVICE);
        alarmManager.cancel(pendingIntent);
    }

    @Override
    public void onClick(View view) {

        EditText editText = findViewById(R.id.editText);
        DatePicker datePicker = findViewById(R.id.datePicker);
        TimePicker timePicker = findViewById(R.id.timePicker);

        // Intent
        Intent intent = new Intent(MainActivity2.this, AlarmReceiver.class);
        intent.putExtra("notificationId", notificationId);
        intent.putExtra("message", editText.getText().toString());

        // PendingIntent
        PendingIntent pendingIntent = PendingIntent.getBroadcast(
            MainActivity2.this, 0, intent, PendingIntent.FLAG_CANCEL_CURRENT

```

```

);

// AlarmManager
AlarmManager alarmManager =
(AlarmManager)getSystemService(ALARM_SERVICE);

switch (view.getId()) {
    case R.id.setBtn:
        // Retrieve the selected date and time from the date and time pickers
        int year = datePicker.getYear();
        int month = datePicker.getMonth();
        int day = datePicker.getDayOfMonth();
        int hour = timePicker.getCurrentHour();
        int minute = timePicker.getCurrentMinute();

        // Create time.
        Calendar startTime = Calendar.getInstance();
        startTime.set(Calendar.YEAR, year);
        startTime.set(Calendar.MONTH, month);
        startTime.set(Calendar.DAY_OF_MONTH, day);
        startTime.set(Calendar.HOUR_OF_DAY, hour);
        startTime.set(Calendar.MINUTE, minute);
        startTime.set(Calendar.SECOND, 0);
        long alarmStartTime = startTime.getTimeInMillis();

        // Set Alarm
        alarmManager.set(AlarmManager.RTC_WAKEUP, alarmStartTime, pendingIntent);
        Toast.makeText(this, "Done!", Toast.LENGTH_SHORT).show();
        break;

```

```

        case R.id.cancelBtn:
            alarmManager.cancel(pendingIntent);
            Toast.makeText(this, "Canceled.", Toast.LENGTH_SHORT).show();
            break;
    }

}

}

//xml code for setting schedule
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity2"
    android:orientation="vertical"
    android:gravity="center"
    android:padding="5dp"
    >

    <EditText
        android:id="@+id/editText"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:hint="Message"/>

    <DatePicker
        android:id="@+id/datePicker"

```

```
android:layout_width="wrap_content"  
android:layout_height="345dp" />
```

```
<TimePicker
```

```
    android:id="@+id/timePicker"  
    android:layout_width="359dp"  
    android:layout_height="268dp"  
    android:layout_marginTop="15dp" />
```

```
<LinearLayout
```

```
    android:layout_width="wrap_content"  
    android:layout_height="wrap_content"  
    android:layout_marginTop="20dp">
```

```
    <Button
```

```
        android:id="@+id/setBtn"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="SET" />
```

```
    <Button
```

```
        android:id="@+id/cancelBtn"  
        android:layout_width="wrap_content"  
        android:layout_height="wrap_content"  
        android:text="CANCEL"/>
```

```
</LinearLayout>
```

```
</LinearLayout>
```