|  |
| --- |
|  |
| Project Module Code: 7048CEM  Module Title: Computing Individual Research Project |
| Plant Disease Detection, A Cross-Platform Mobile Application Using Artificial Neural Networks |
| Author: Michael Ajao-Olarinoye |
| SID: 10118047 |
| Supervisor: Supervisor's Name |
| Submitted in partial fulfilment of the requirements for the Degree of Master of Science in Data science and computational intelligence |
| Academic Year: 2020/21 |

**Declaration of Originality**

I declare that this project is all my work and has not been copied in part or whole from any other source except where duly acknowledged. As such, all use of previously published work (from books, journals, magazines, internet etc.) has been acknowledged by citation within the main report to an item in the References or Bibliography lists. I also agree that an electronic copy of this project may be stored and used for plagiarism prevention and detection.

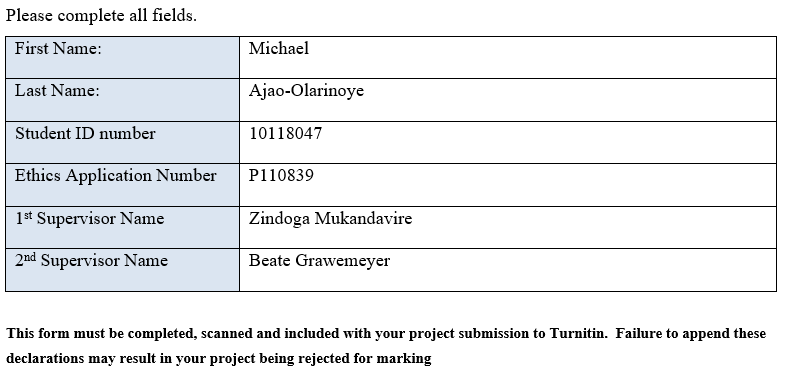
**Statement of copyright**

I acknowledge that the copyright of this project report, and any product developed as part of the project, belong to Coventry University. Support, including funding, is available to commercialize products and services developed by staff and students. Any revenue that is generated is split with the inventor/s of the product or service. For further information please see [www.coventry.ac.uk/ipr](http://www.coventry.ac.uk/ipr) or contact [ipr@coventry.ac.uk](mailto:ipr@coventry.ac.uk).

**Statement of ethical engagement**

I declare that a proposal for this project has been submitted to the Coventry University ethics monitoring website (https://ethics.coventry.ac.uk/) and that the application number is listed below (Note: Projects without an ethical application number would be rejected for marking)

Signed: Michael Ajao-olarinoye Date: 11/12/2020

****

Abstract

Table of Contents

[Abstract 3](#_Toc58587867)

[Table of Contents 4](#_Toc58587868)

[Acknowledgements 6](#_Toc58587869)

[1 INTRODUCTION 7](#_Toc58587870)

[1.1 Background to the Project 8](#_Toc58587871)

[1.2 Project Objectives 8](#_Toc58587872)

[1.3 Overview of This Report 9](#_Toc58587873)

[2 LITERATURE REVIEW 10](#_Toc58587874)

[2.1 Effect of plant disease on farmers 10](#_Toc58587875)

[2.2 Computer vision as a problem 11](#_Toc58587876)

[2.3 Plant Disease Review 13](#_Toc58587877)

[3 METHODOLOGY 19](#_Toc58587878)

[3.1 Different Datasets 19](#_Toc58587879)

[3.1.1 PlantVillage 19](#_Toc58587880)

[3.1.2 PlantDoc 19](#_Toc58587881)

[3.1.3 Digipathos 20](#_Toc58587882)

[3.2 Models Architecture 21](#_Toc58587883)

[3.2.1 Convolutional neural network (CNN) 21](#_Toc58587884)

[3.2.1.1 Convolution 21](#_Toc58587885)

[3.2.1.2 Pooling 21](#_Toc58587886)

[3.2.1.3 Relu 21](#_Toc58587887)

[3.2.1.4 Flattening 22](#_Toc58587888)

[3.2.1.5 Fully Connected Layer 22](#_Toc58587889)

[3.2.2 MobileNetV2 22](#_Toc58587890)

[3.2.3 Inception ResNet V2 24](#_Toc58587891)

[3.3 Transfer learning 24](#_Toc58587892)

[3.4 Flutter 25](#_Toc58587893)

[4 Implementation of the mobile application 26](#_Toc58587894)

[4.1 Data Augmentation 27](#_Toc58587895)

[4.2 Model architecture 27](#_Toc58587896)

[4.2.1 CNN Model Training 27](#_Toc58587897)

[4.2.2 mobileNetv2 and inception resnetV2 model training using transfer learning 29](#_Toc58587898)

[4.3 Flutter Mobile Application 29](#_Toc58587899)

[5 Analysis of the Models 31](#_Toc58587900)

[5.1 Visualization of feature Maps 31](#_Toc58587901)

[5.1.1 CNN model validation and Tflite conversion 32](#_Toc58587902)

[5.1.2 MobileNetv2 model validation 33](#_Toc58587903)

[5.1.3 Inception ResNetv2 34](#_Toc58587904)

[6 Testing of the Mobile Application 36](#_Toc58587905)

[7 PROJECT MANAGEMENT 37](#_Toc58587906)

[7.1 Project Schedule 37](#_Toc58587907)

[8 CRITICAL APPRAISAL 38](#_Toc58587908)

[9 CONCLUSIONS 39](#_Toc58587909)

[9.1 Achievements 39](#_Toc58587910)

[9.2 Future Work 39](#_Toc58587911)

[10 STUDENT REFLECTIONS 40](#_Toc58587912)

[11 BIBLIOGRAPHY AND REFERENCES 41](#_Toc58587913)

[Appendix B – Interim Progress Report and Meeting Record iii](#_Toc58587914)

[Appendix E – Project Presentation iv](#_Toc58587915)

[Appendix F – Certificate of Ethics Approval vi](#_Toc58587916)

[Appendix X – As required vii](#_Toc58587917)

Acknowledgements

# INTRODUCTION

Agriculture has been one of the oldest occupations known to man. Most countries use agriculture as a way of generating income by exporting different agricultural products and by-products. This research project shows how computer vision algorithm can be used in solving one of agriculture major problem. This is the detection of plant disease that ravages peoples farm. According to Britannica, plant disease is said to be the impairment of a good and healthy plant, which destroys or changes the organic composition of the plant. All plants are vulnerable to diseases, both indigenous and exotic. Any disease and form of plant or crop changes between seasons. Certain plants are infected frequently very rapidly depending on the presence, environmental conditions, crops and planted varieties – an outstanding example of how vegetable disease devastates farmers' earnings if they are not detected and recognised easily.

According to the United Nations Food and Agriculture Organisation (UN), food crops are impacted by cross-border plant pests and diseases, causing significant losses for farmers and threatening food protection. Plant diseases contribute 10–16 per cent of the annual global crop harvest losses measured at US$ 220 billion. According to the Food and Agriculture Organization (FAO) study, our world population is projected to exceed 9.1 billion by 2050. Agricultural demand must also be raised to 70% to satisfy the food needs of a rapidly growing population. In the other hand, the abundant use of chemicals such as bactericides, fungicides and nematicides to combat plant diseases has caused detrimental effects in the agro-ecosystem. There is currently a need for successful early warning strategies to monitor plant diseases to ensure food security and the survival of the agro-ecosystem. In recent years, the distribution of plant pests and diseases has increased significantly. Globalization, trade and climate change, as well as reduced flexibility in production processes due to decades of intensification of agriculture, have all played a part.

Detection of plant diseases plays an essential role in agriculture, as farmers often must determine if the crop, they are cultivating is healthy enough. It is of the utmost importance to take this seriously, as it can lead to severe problems in plants that impact the consistency, quantity or competitiveness of the commodity.

Plant pests and diseases can spread quickly to other countries and achieve epidemic proportions. Outbreaks and upheavals would cause tremendous losses to crop and pastures, endanger the livelihoods of poor farmers and the food and nutrition welfare of millions at a time. A clear example of how plant diseases devastate food crops in African countries (Animation: How Plant Diseases Devastate Food Crops in African Countries-Connected, 2019) is published in a blog that presents an animated video about how plant diseases devastate food crops in African countries, insect spread diseases, and plant viruses. The crops are mainly contaminated with cassava mosaic viruses and cassava brown streak viruses, which are spread via whiteflies destroying through the fields. It is used to produce all manner of various products. More research carried out by the Associated Virus Network has led to the loss of $1 billion in poverty, food shortages and hunger in crops, and thereby to economic and social development in the affected countries (Isabel, 2019).

Farmers in parts of the world could also benefit from the lack of the appropriate infrastructure to provide agronomic and phytopathological advice if a useful and conveniently available mobile application tool could be created. Crop analysis has historically been carried out visually by persons with some expertise or experience in the diagnosis of plant disorders. As for all human behaviour, this approach is prone to psychological and perceptual processes that can lead to bias, visual illusions, and inevitably errors (Bock et al., 2010). More specifically, qualified plant pathologists are not often available, especially in poor and isolated areas. It is also worth noting that many agricultural areas are too vast to be adequately tracked during seasons (Arnal Barbedo 2013). Image-based techniques may also play an essential role in the diagnosis and identification of plant diseases where human assessment is insufficient, inaccurate or inaccessible.

In today's apps, Machine Learning capabilities have been a must-have. They will boost your loyalty and give your audience the exceptional app experience they deserve. Thanks to recent hardware and software optimizations, the execution of Machine Learning models is transitioning from the cloud to mobile devices themselves.

## Background to the Project

My concept was to use computer vision techniques to incorporate the capability to identify distinct plant diseases from an image. The app will run our classification model solely on the user's mobile device, thanks to TensorFlow Lite on-device efficiency optimizations.

The demographic of people this project is designed for are small and medium scale farmer, third world farmer and farmer's that will also want to implement drone technology with the use of the neural network. The mobile application will be available to everyone and every farmer that has access to a smartphone.

## Project Objectives

This project has aims and objective that has to be answered during development. The aims that I would want to determine are:

* Finding the best neural network model that will be optimized for a mobile application.
* Build a mobile application that farmers can have on their smartphone to help with:
  + Identifying the type of plant specie
  + Identify if the plant leaf image is healthy or not
  + Show the type of disease and the prediction confidence
* Make sure the mobile application is both android and iOS compactable.
* Make use of flutter for the development of the cross-platform mobile application.

## Overview of This Report

This report will show the discussion of different chapters ranging from the

# LITERATURE REVIEW

This research focuses on building and testing different models to be used working with image detection and classification of plant diseases. This thesis looks at plant diseases affecting small scale farmers, different methods of implementing classification techniques. This work would involve comparing deep learning architecture and building a user interface with Flutter. There is the various dataset that could be used in the training of a model. There have been different methods used in building a mobile application that is used for plant disease detection. This writeup focuses on the review of the effect of plant disease on farmers, computer vision as a domain problem, and plant disease review. Barbedo (2013) discussed vital factors impacting the architecture and efficiency of deep neural network applied to plant pathology. The in-depth study of this subject, which highlights the benefits and disadvantages, can contribute to more realistic conclusions on the subject. Different studies and experiments were done on how to optimally design and build a mobile phone application that would be used for plant diseases detection and classification. Different themes are going to be reviewed in this chapter, like discussing the effect of diseases affecting crops or plants—literature on computer vision and deep learning with neural networks.

## Effect of plant disease on farmers

A variety of factors, including population growth, income levels, urbanization, habits, and tastes, affect the food market. Over the next quarter-century, nearly 80 million individuals are likely to be added to the global population every year, growing the world population by 35 % from 5.7 billion in 1995 to 7.7 billion by 2020 (United Nations 2020). In developed countries, whose share of the global population could estimate to rise from 79% - 84% in 1995 – 2020 respectively, more than 95% of the population is predicted to increase. The actual population growth would be the largest in Asia during this time. However, the relative increase would be the highest in Sub-Saharan Africa, where the population projection is to increase by 80% by 2020. For example, the demand for cereals to feed livestock would increase considerably in the coming decades, particularly in developed countries, in response to strong demand for livestock products. The demand for cereals for animal feed by developed countries was predicted to double between 1993 and 2020, while the demand for cereals for food for direct human consumption projection is to rise by 47% (Pinstrup-Andersen, 2001). Food importation would not be option overtime for third world countries. Relying on the growth and productivity of a small-medium scale farmer would be essential to fill the gap on food deficit.

The question why Diagnostics Matter would be answered as follows. It is known that threats to plants by introduced pathogens, including seeds, plants goods, or representatives of natural ecosystems such as forests and grasslands, are growing as a result of globalization, expanded human migration, climate change, and pathogens and vector evolution (Anderson *et al.*, 2004). Infestations of pests also coincide with climatic changes, such as erratic rainfall, increased humidity or drought, which in themselves can decrease crop yields. Pest outbreaks can have a catastrophic effect each year but only cause marginal losses in other years (Yudelman *et al.*, 1998). About 80% of small-scale South African farmers included in the research by (Ghimire 2017) that had insufficient knowledge of vegetable crop diseases. Although some farmers had a general knowledge of the chemicals are ideal for a particular disease, few (15%) had the money to purchase them. Various attempts have been made to eliminate crop losses due to disease. Historical approaches to the systematic use of pesticides have gradually been complemented by integrated pest management (IPM) approaches over the last decade (Ehler, 2006).

Regarding the prevalence of diseases, according to (Ghimire 2017), the findings showed that more than 50% of crop losses were caused by diseases, mainly where no control measures were applied. Early and accurate detection and surveillance of pathogens at local, regional and global scales are required to anticipate outbreaks and provide time for the formulation and deployment of mitigation strategies. Other direct effects of plant disease on the health of humans and animals are caused by mycotoxins produced by plant pathogenic fungi, such as Aspergillus and Fusarium, which can contaminate food and feed, resulting in several diseases and disorders.

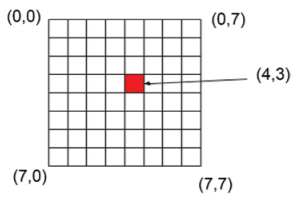
Improving the ability of farmers to control crop diseases requires knowledge and innovation. Farmers are unable to comprehend the basis for disease control techniques without recognizing fundamental issues such as the pathogen as a causative agent, inoculum origins, and the notion of latent duration. The result by (Nelson et al., 2001) supports the notion that to increase their agricultural production, farmers need access to knowledge and technologies. Training of farmer teachers, enhancement of the reach and consistency of the training curriculum, creation of necessary but practical experiments addressing a variety of issues in the field of disease and crop management, use of multimedia electronic technologies to provide technical knowledge to facilitators, and closer ties between breeding programmes and FPR-FFS (Nelson et al., 2001). "Farmer Field Schools" (FFS) and "Farmer Participatory Testing" (FPR) which are experimentally approached built to collaborate with resource-poor farms on plant disease management.

## Computer vision as a problem

Computer vision is a sub-field in artificial intelligence that focuses on interpreting the quality of visual images, such as photos and videos. Computer vision is an automatic extraction of image information. Knowledge can mean everything from 3D models, camera orientation, object detection and identification to grouping and searching for image content (Solem, 2012). The challenge of computer vision seems to be necessary since humans' vision could solve the issue. However, the issue remains mostly unresolved, based on both the restricted knowledge of biological vision and the difficulty of experience of vision in a diverse and almost infinitely different physical environment. Transforming images or related image patches into a compact summary using features has become common. The number of various features proposed in the literature has risen dramatically over the last decades. However, features that visually display specific unique characteristics are usually picked. These components are also called visual attributes and are essential for machine vision and pattern recognition as they reduce essential tasks such as image retrieval, object recognition, object tracking, autonomous navigation, scenic reconstruction or scenic interpretation (Farinella et al., 2013). These components reduce the essential tasks of imagery, imagery acquisition or the identification of objects. A remarkable study by Alan Turing presented the philosophy of computing, spelling out the principles behind all modern computers (Bernhardt, 2016). The cornerstone of computer science is now this revolutionary and reliable principle.

From the beginning, there has been a growing need to understand how computation takes place in all devices, looking at how overtime computer vision has developed. Davies wrote about how in the 1920s (Davies, 2017) knowledge of the need to explain visual characteristics accurately arose in the field of visual perception, and basic ideas were developed that can be traced back to almost any approach to feature extraction. Much analysis has been carried out since the transition of fundamental ideas to the field of computer vision, including the development of new principles and methods. Computer vision has many problems that make it hard for a machine to implement with computation and easy for the natural human eyes. From the process of image recognition to object location to scene analysis, several studies (Lecun et al., 1998). (Ramcharan et al., 2019) (Sladojevic et al., 2016) have been conducted. Image recognition or character recognition is a highly constrained problem in computer vision domain. Looking at the position of objects, instead of only interpreting a small region of an image is also a problem in computer vision, it is essential to scan images for objects of different kinds. Computer vision has a few issues which research has been trying to solve over time (Davies, 2017).

Image processing is a method used to manipulate a digital image, either to get an improved image or to remove any valuable information from it. In image processing, the input is an image, and the output may be an image or any of the characteristics or features associated with the image. A video is a series of pictures or frames. The encoding of images still to video processing. The digital image is an interactive representation of an object/scene or a text that has been scanned. The digitalization of an image means that it is translated into a set of numbers and stored in a data retrieval device. A grid of rows and columns is set in an example of pixels. Imagine an eight-row, eight-column square grid. The image consists of 8 x 8 or 64 pixels. The image in Figure 1shows a 2D coordinate system where (0,0) is the upper-left corner.



**Figure 1:** A 2D coordinate system of a pixel

In the figure above the left-top corner of the image coordinate system starts from (0, 0) In Figure 1, the pixel at the top right-hand edge is (7,0), the lower left-hand edge (7,0) and the lower right-hand pixel (7,7). This can be represented as (x, y). The position is x from the left side of the image, and y from the top of the image, the vertical position.

The camera functions as a sensing system, much like our eyes detect pictures of objects. Photos are sent to an interpreter machine, such as a computer, where the incoming signals are interpreted similarly to a neuron. Examples of other sensing instruments include X-ray, CT-scan and MRI machines; satellite imagery systems; and paper scanners. Interpreting instruments, such as computers, ensure that the data collected by the camera is interpreted. Many machine vision-related computations are done within the computer, such as attribute extraction and pattern determination.

## Plant Disease Review

Deep learning is another term for a multilayer, artificial neural network or multilayer perception. We have various types of deep learning systems based on the design of the neural network and its operational concepts. For example, neural feed-forward networks, convolutional networks, recurrent neural networks, self-encoders, and deep-seated beliefs are various forms of deep learning systems (Ansari, 2020). A regular neural network (NN) consists of several primary, interconnected processors called neurons, each generating a series of real-value activations. Environmental sensors activate input neurons. Other neurons are activated by weighted associations from previously active neurons (Schmidhuber, 2015).

Several methods have been devised to solve the issue of plant disease identification and classification; one of the methods is the classical numerical visual method which has been commonly used. Pioneering research, in this context, from (Sannakki et al., 2010), diseases found on plant leaves are automatically classified to evaluate the colour-specific information on poisonous plants, their algorithm used image processing techniques. For each image pixel, a first k-means-based clustering was carried out to separate the infected spots, and subsequent grading based on fuzzy logic techniques were carried out. According to (Johannes et al., 2017) they designed a system that automatically diagnosis plant disease using current image recognition algorithm in conjunction with machine learning inference methods. It was tested with seven different mobile devices over images taken over three seasons. The study by (Siricharoen et al., 2016) presented a technique that merged texture, colour and form to detect the presence of a particular disease on a wheat plant. In a study by (Ramcharan et al., 2019), a CNN object detection model was built to learn to recognize foliar disease symptoms in cassava (Manihot esculenta Crantz) in a smartphone application. The model was deployed and tested on mobile images and video of 720 diseased leaflets in a Tanzanian agricultural area. Two degrees of severity of symptoms were evaluated within each disease type, mild and pronounced, to determine model success for early identification of symptoms. According to (Ramcharan et al., 2019), if the promise of mobile CNN models is to be realized, it is essential to consider tuning recall.

Furthermore, for architecture in real-world systems, the varied output related to various input data (Image or video) is a significant factor. An investigation of the critical factors impacting the architecture and efficacy of deep neural nets applied to plant pathology was studied by (Barbedo, 2018). An image-based classification scheme for plant disease detection was introduced in (Gandhi et al., 2018). To enhance the limited number of local images available, it uses Generative Adversarial Networks (GANs). A Convolutional Neural Network (CNN) model implemented in a smartphone app carries out the classification. There is a significant advancement in the development of deep neural networks (Lecun et al., 1998).

An analysis involving the PlantVillage dataset was conducted (Argüeso et al., 2020) which contains 54,303 classified images, also consisting of 38 plant leaf and disease forms (classes). The data was separated into a source (32 classes) and a target (6 classes) domain. The Inception V3 network was fine-tuned in the root domain to understand the general characteristics of the plant leaf. FSL (Few-Shot Learning) using Siamese networks and Triplet losses is used and compared to classical fine-tuning transfer learning. The source and goal domain sets were divided into a training set (80 %) to improve the methods and a test set (20 %) to obtain the data. The efficiency of the algorithm was assessed using absolute Accuracy and precision and recall per class. For FSL experiments, algorithms were trained with a different number of images per class, and experiments were replicated 20 times to classify the effects statistically. The result from the experiment shows that New plant leaf and disease forms with very limited datasets can be learned by deep learning. Siamese networks, achieving almost 90% reduction in training data requirements and surpassing conventional learning methods with small training sets (Argüeso et al., 2020).

According to Liu 2019, who conducted a study on pestNet proposes a regionally based end-to-end method for large-scale multi-class pest identification and classification based on deep learning. PestNet consists of three primary sections of it. Next, a new channel-space attention module (CSA) is proposed to be implemented into the convolutional neural network (CNN) backbone for the extraction and enhancement of functionality. The second is called the Regional Proposal Network (RPN) which is adapted to include regional proposals as possible pest positions dependent on the derived picture features maps. Position-sensitive score map (PSSM), the third part, is used to substitute ultimately linked layers for pest classification and bound box regression. Also, spatial regions of interest (ROIs) are used as contextual information on pest features to enhance identification accuracy (Liu et al., 2019).

Another study using CNN programmed a model built to diagnose and identify plant diseases using apple and tomato leaf images of stable and diseased plants from the plant disease dataset. The model consists of four convolutional layers, each accompanied by a pooling layer. Two completely integrated thick layers and sigmoid feature were used to detect the possibility of illness arising or not. Model training was conducted on apple and tomato leaf image dataset containing 3663 images, achieving an accuracy of 87%. The over-fitting problem is defined and eliminated by setting the dropout value to 0.2. Since the model requires parallel processing, GPU Tesla is also used to test its output speed and Accuracy (Francis et al., 2019).

In (Sibiya et al., 2019), CNN was used to identify diseases in maize plants and histogram techniques to demonstrate the importance of the model. Basic CNN architectures such as AlexNet, GoogLeNet and ResNet have been introduced in (Zhang et al., 2018) to recognize tomato leaf diseases. Training/validation precision was plotted to demonstrate the efficiency of the model; ResNet was deemed to be the strongest of all CNN architectures. In order to identify diseases in banana leaf, LeNet architecture was introduced, F1-score was used to test the model in colour and Gray Scale modes (Amara et al., 2017). In (Ferentinos, 2018), five CNN architectures, namely, AlexNet, AlexNetOWTbn, GoogLeNet, Overfeat, and VGG architectures were used in which all the other versions were outclassed by VGG. In (TÜRKO'LU & Hanbay, 2019), three classifiers, Support Vector Machines (SVM), Extreme Learning Machine (ELM), and K-Nearest Neighbour (KNN)), used with state-of-the-art deep learning models such as GoogLeNet, ResNet-50, ResNet-101, Inception-v3, InceptionResNetv2, and SqueezeNet, recognized eight distinct plant diseases. A contrast was made between those models, and in terms of performance metrics such as sensitivity, precision, and F1-score, ResNet-50 with SVM classifier got the best results. A new DEEP LEARNING model-Inception-v3-was used for cassava disease detection, according to (Ramcharan et al., 2017).

A new deep learning model, Inception-v3, has been used for cassava disease detection. In (Fujita et al., 2016), the two simple versions of CNN listed plant diseases in cucumber and got the maximum precision, equivalent to 0.823. The conventional identification and classification system for plant diseases was replaced by the Super-Resolution Convolutional Neural Network (SRCNN) in (Yamamoto et al., 2017). AlexNet and SqueezeNet v1.1 model were used for the classification of tomato plant disease, of which AlexNet was considered to be the superior DEEP LEARNING model in terms of Accuracy (Durmuş et al., 2017). A comparative review (Too et al., 2019) was proposed to pick the right deep learning architecture for plant disease detection. Besides, six tomato plant diseases were identified in (Rangarajan et al., 2018) using AlexNet and VGG-16 deep learning architectures, and a thorough comparison was given with the aid of Accuracy of classification.

Deep learning models/architectures and even visualization techniques were used in the following approaches, which were introduced for a better view of plant diseases. E.g. (Brahimi et al., 2018) developed the saliency map to visualize the symptoms of plant disease; (Sladojevic et al., 2016) using CaffeNet CNN architecture, detected 13 different forms of plant disease and achieved CA equivalent to 96.30 percent, which was better than the previous method such as SVM. Besides, to denote the disease spots, multiple philtres were used. Similarly, using the freely accessible PlantVillage dataset, AlexNet and GoogLeNet CNN architectures (Mohanty et al., 2016) were used. Output was measured by accuracy (P), recall (R), F1 ranking, and overall Accuracy. The novelty of this paper was the implication of three possibilities for analyzing the efficiency metrics and contrast of the two popular CNN architectures (colour, grayscale, and segmented). It has been concluded that AlexNet outperformed GoogLeNet. Besides, the spots of diseases were clearly seen by visualization activation in the first layers. In (Cruz et al. 2017), for the identification of olive plant diseases, an updated LeNet model was used. To spot the diseases in the plants, segmentation and edge maps were used. Four cucumber diseases were observed (Ma et al., 2018) and were compared with Random Forest, Help Vector Machines and AlexNet models for Accuracy. In addition, to view the signs of diseases in the plants, the picture segmentation approach was used. In the (Brahimi et al., 2019) named teacher/student network, a new DEEP LEARNING model was implemented, and a new visualization approach was proposed to define the plan spots.

|  |  |  |
| --- | --- | --- |
| **Deep Learning Models** | **Parameters** | **Features of models** |
| LeNet | 60k | CNN's first model. Few parameters are relative to the other CNN models. Limited computational capability |
| AlexNet | 60M | Established as the first modern CNN ever. Best image recognition results at the time. Used by ReLU to achieve improved efficiency. Dropout strategy was used to prevent over-fitting |
| OverFeat | 145M | The first model used by a single CNN for the identification, localization and classification of objects. A wide number of parameters relative to AlexNet |
| ZFNet | 42.6M | By considering 7 x 7 kernels and increased precision, reduced weights (as opposed to AlexNet) |
| VGG | 133M–144M | 3 × 3 Receptive Fields were known to have more non-linearity functions that discriminated against decision-making. Due to a massive number of parameters, a computationally costly model |
| GoogLeNet | 7M | Fewer parameters are relative to the AlexNet model. Improved precision at its time |
| ResNet | 25.5M | Vanishing gradient dilemma, Higher Accuracy than the VGG and GoogLeNet versions |
| DenseNet | 7.1M | Dense relation between the layers. Reduced number of parameters with higher Accuracy |
| SqueezeNet | 1.25M | Similar precision to AlexNet with 50 times lower parameters. Considered 1 × 1 filter instead of 3 × 3 filters. The input channels have been limited. Wide activation maps of the convolution layer |
| Xception | 22.8M | A deep-wise, removable approach to convolution. Performed more than VGG, ResNet, and Inception v3 versions. |
| MobileNet | 4.2M | depth-wise separable convolution Significantly reduced parameters. Achieved accuracy close VGG and GoogLeNet |
| Modified/Reduced MobileNet | 0.5/0.54M | Less number of parameters relative to MobileNet. Related Accuracy compared to MobileNet |
| VGG-Inception | 132M | A cascaded variant of the VGG and the launch module. The number of parameters was decreased by replacing 5 × 5 convolution layers with two 3 × 3 layers. The testing performance was improved compared to several well-known DEEP LEARNING models such as AlexNet, GoogLeNet, Inception-v3, ResNet, and VGG-16. |

1. Studies were employing deep learning for plant disease recognition.

Since the implementation of AlexNet (Krizhevsky et al., 2017) for image recognition, segmentation, and classification, a variety of state-of-the-art deep-learning models and architectures have grown. When it won the ImageNet Large Scale Visual Recognition Challenge (ILSSVRC 2012 competition), AlexNet (Krizhevsky et al., 2017) was first used in the public environment. It was at this competition that AlexNet proved that the deep convolutional neural network could be used to solve the classification of images. AlexNet won by a majority in the ILSVRC 2012 contest. The AlexNet neural network consists of 5 convolutional layers, some of which are accompanied by max-pooling layers and three fully linked layers and a final 1000-way SoftMax, which have 60 million parameters and 650,000 neurons. It was trained on 1000 different classes on 1.2 million high-resolution images in the ImageNet LSVRC-2010 contest.

Irrespective of the approach, the correct identification of the disease when it first appears is a crucial step in the effective management of the disease. Historically, agricultural extension associations or other agencies, such as local plant clinics, have sponsored disease recognition. More recently, such efforts have also been assisted by the online availability of information for disease detection, exploiting the worldwide rise in Internet penetration. More recently, cell phone-based applications have proliferated, benefiting from the globally unprecedented widespread adoption of mobile phone technologies in all areas of the world (ITU, 2015).

# METHODOLOGY

The research approach will concentrate on developing and evaluating various models to be used for image recognition and classification, and using other pre-trained models using the transformation process. The Google Flutter Platform will be used to create a web user interface programme. The TensorFlow library will be used to process image data and to train all models. The best method and model would be utilized to develop a mobile application for plant disease detection and classification. This methodology chapter will look at the different dataset that is available to use in developing this project, different pieces of literature where people used a different variation of the plant village dataset for different analysis. The model architecture of the CNN model that will be designed in this project, how to use transfer learning and why I decided to use the method to implement some of the models in this project. We will also look at the model architecture of both mobilenetV2 and inception resnet V2. This project will make use of different libraries and framework to develop the mobile application.

## Different Datasets

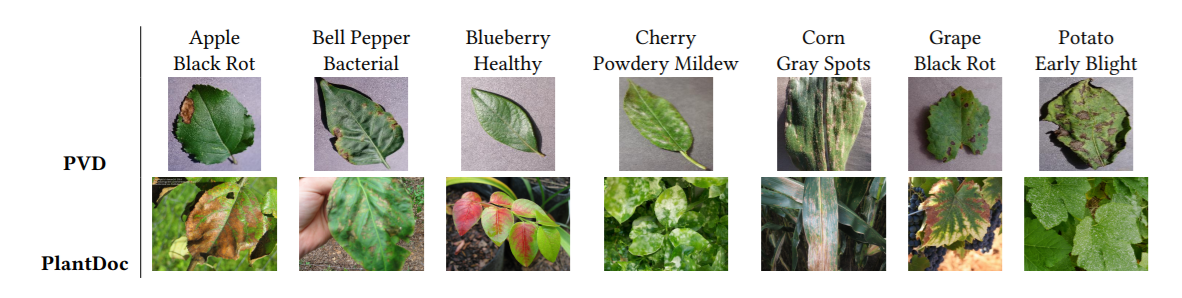
To complete this Dissertation, I ended up discovering a couple of datasets for plant disease and different ways people used them. In table 1 above, it can be found that plant village dataset was used in the research done by the researchers above. People made a different variation of the dataset by taking out different plant types or reducing the number of classes to generate a new dataset for training. The inadequate available broad non-lab data sets continue to be an essential obstacle for the vision-based identification of plant diseases.

### PlantVillage

A public dataset containing 54,305 photographs of diseased and stable plant leaves obtained under controlled conditions is given. The pictures cover 14 varieties of crops, including apple, two blueberries, plum, grape, citrus, peach, pepper, cabbage, mango, soya, squash, strawberry and tomato. It includes images of 17 essential diseases, 4 bacterial diseases, 2 mould-related diseases, 2 viral diseases and 1 mite-related disease. Every class mark is a crop-disease pair, and we are trying to predict the crop-disease pair, provided only the picture of the plant leaf (Mohanty et al., 2016).

### PlantDoc

In the fall of 2019, researchers at the Indian Institute of Technology published PlantDoc, a dataset of 2,598 images spanning 13 plant species and 27 groups (17 diseases; 10 healthy) for image classification and object detection. Researchers note that the dataset has taken more than 300 human hours to compile and annotate. Unlike related datasets such as CropDeep and DeepWeeds, this dataset is open to the public for free download to deep learning researchers.



1. sample for PlantVillage and PlantDoc dataset different between lab controlled and real-life images (Singh et al., 2019)

The dataset of PlantVillage includes photographs taken under supervised conditions. This dataset restricts the usefulness of disease detection since in fact, plant photos can include several leaves with various types of background conditions with different lighting conditions (shown in Fig 1).

### Digipathos

Fully annotated dataset for plant crops and diseases provided by the Brazilian Agricultural Research Corporation (EMBRAPA). A publicly accessible archive containing approximately 50,000 photographs of 171 diseases involving 21 species of plant is another example of plant disease dataset, could be found in <https://www.digipathos-rep.cnptia.embrapa.br/> (Barbedo, 2018).

The dataset I choose is a secondary dataset which can be downloaded from the plant disease dataset that can be found on Kaggle which consists of approximately of 50 thousand RGB photographs of normal and diseased crop leaves grouped into 38 distinct groups. As part of my research also I would test three neural network architecture using transfer learning and building out a neural network layer to train on the data, a user interface would be built while using flutter framework and dart programming language to create the logical aspect of the mobile application.

During the research process, I also made use of another annotated plant disease dataset that contains 15 classes, crop species: bell pepper, potatoes and tomatoes. The dataset contains about 20 thousand images, and I split them into ration 80 and ration 20 training and testing data respectively. The datasets will be analysed and trained using the python programming language and google colab as an environment to be tested by the GPU (Graphics Processing Unit) to be used for training and evaluating the model. Use the TensorFlow Lite converter to cover the model so that it can be viewed from a cell phone application that can be developed using the Flutter platform.

## Models Architecture

The first model created was the CNN (convolutional neural network), the model is built and analysis performed on the model shown in the analysis section shows how an image appears in each convoluted block that was created in the code. The second model used was the mobile-optimized neural network model called MobileNetv2, while the third neural network chooses for this project was the inceptionResnetv2. The second and third model would be trained using the Transfer Learning technique.

### Convolutional neural network (CNN)

CNN was created for image analysis as a deep neural network. It has been noticed that CNN has an outstanding ability in pursuing data interpretation, for example, natural language processing (Zhang, 2016) indeed. It is designed to work with 2-dimensional image data. Two simple processes, namely convolution and pooling, are still part of CNN. Using filters, the convolution operation would remove characteristics (feature map) from the data collection, from which their corresponding spatial details can be retained. To minimize the dimensionality of function maps from the convolution operation, the pooling operation, also called subsampling, is used. The most famous CNN pooling practises used Max pooling and average pooling. Relu is the common preference for the activation feature to pass gradient in training by backpropagation due to the complicity of CNN.

#### Convolution

A convolution, like a conventional neural network, is a linear procedure involving the multiplication of a set of weights with the data. The multiplication is carried out between an array of input data and a two-dimensional array of weights, called a kernel, provided that the procedure was constructed for two-dimensional input.

#### Pooling

The second operation that is important for CNNs is pooling. Such an operation is much better to comprehend than convolution. It is used by downsampling to decrease the number of parameters and preserve only useful information to be further analyzed. There are different types of pooling which can be used in building up the CNN model.

#### Relu

For a non-linear operation, Relu stands for a Rectified Linear Unit. Output is equal to f(x) = max (0, x). We do this to implement CNN's non-linearity.

#### Flattening

We are flattening our whole matrix into a vertical-like vector. Therefore, it would be moved on to the input layer. In Keras, a flattening layer reshapes the tensor to have a form proportional to the number of features present in the tensor.

#### Fully Connected Layer

We transfer our flatten vector to the input layer. To construct a model, we merged these features. Finally, to identify the outputs, we have an activation feature such as SoftMax or sigmoid.

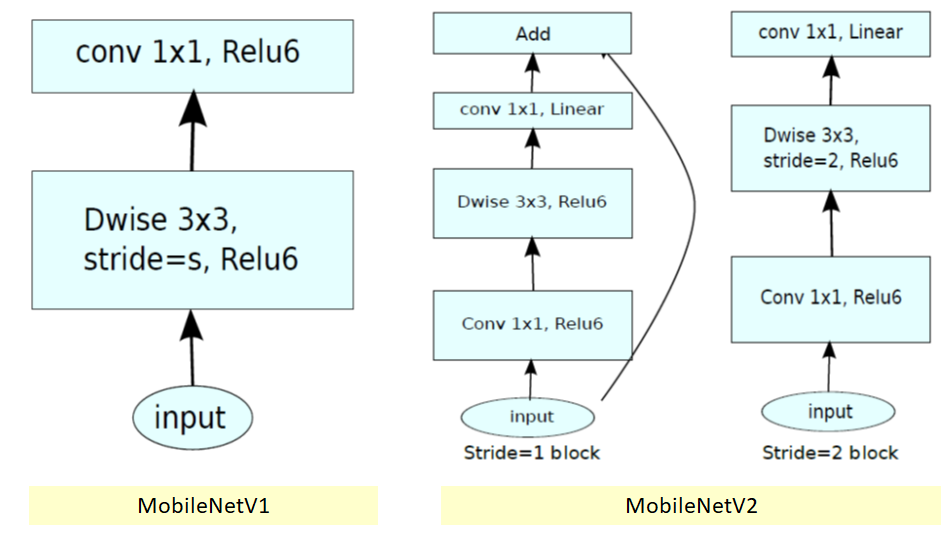
A picture containing diagram

Description automatically generated

1. The internal block design of a simple CNN using image data

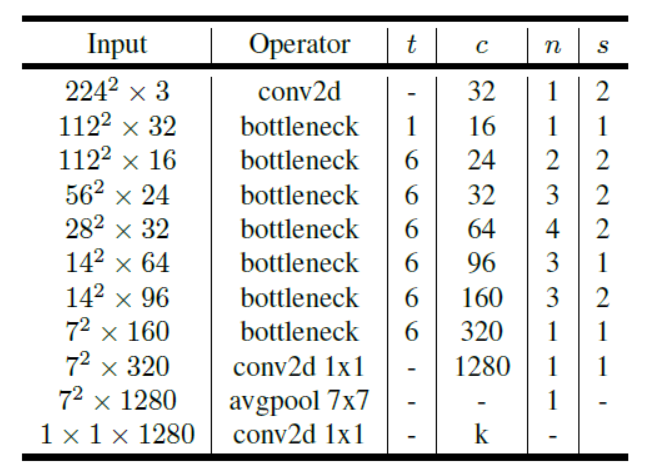
### MobileNetV2

The MobileNetV2 architecture relies on an inverted residual system where the residual block input and output are thin bottling lenses, as opposed to standard residual models using symbolic representations in the MobileNetV2 input, using lightweight depth convolutions to process interim expansion layer characteristics. Depthwise Separable Convolution was implemented in the previous version was called MobileNetV1, which significantly decreases the complexity cost and model size of the network that is ideal for handheld devices or low processing capacity devices (Sandler et al., 2018).



1. Block diagram of both mobileNet V1 and V2

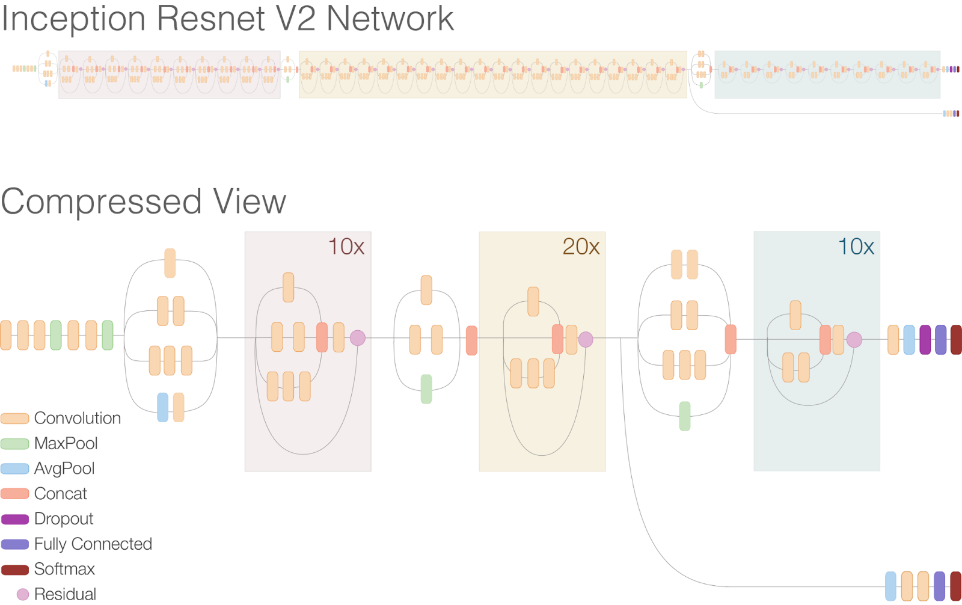
There are two kinds of blocks in MobileNetV2. One is a residual block with a 1. Another one is a two-step block for downsizing. On all forms of bricks, there are three layers. This time, the first layer is the ReLU6 1×1 convolution. The second layer is the convolution of the depth. The third layer is another 1×1, but without any non-linearity. It is stated that if ReLU is used again, deep networks can only have the power of a linear classifier on the non-zero volume portion of the output domain.



1. MobileNetV2 model architecture

### Inception ResNet V2

Driven by the success of the ResNet, a hybrid inception module has been proposed. There are two sub-versions of Inception ResNet, v1 and v2 (Szegedy et al., 2016).

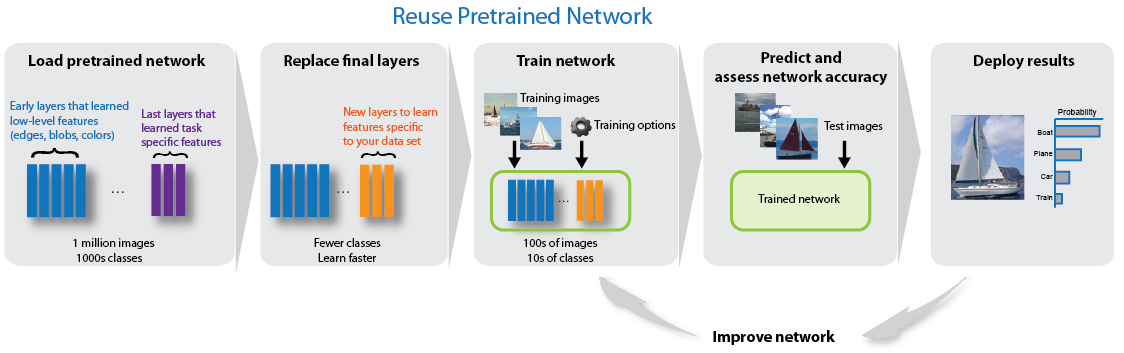


1. Inception ResNet model architecture

Inception v4 and inception resnetv4 architecture are mostly alike, and inception ResNet v2 is a costlier version of family architecture inception. The image above shows the base model architectural design of the inception Resnet mode.

## Transfer learning

transfer learning has been a successful method using in solving a classification problem. Transfer learning is a process in which a model learned to solve a particular issue is re-proposed (Yosinski et al., 2014). Usually, the first layers would train to detect standard features. The last layers would be able to detect more complex features. The last layer would contain an N number of SoftMax neurons. In transfer learning, the primary neural network learned to remove generic image features adequately. Therefore we would need to leverage this information and prevent the need to relearn it. However, to enhance the forecasts, we want to improve this case with our network predictions to refine how our target neural network derives specific features relevant to the problem. Typically, the changes are made in the last layers of a network. Transfer learning would be implemented using a pre-trained model



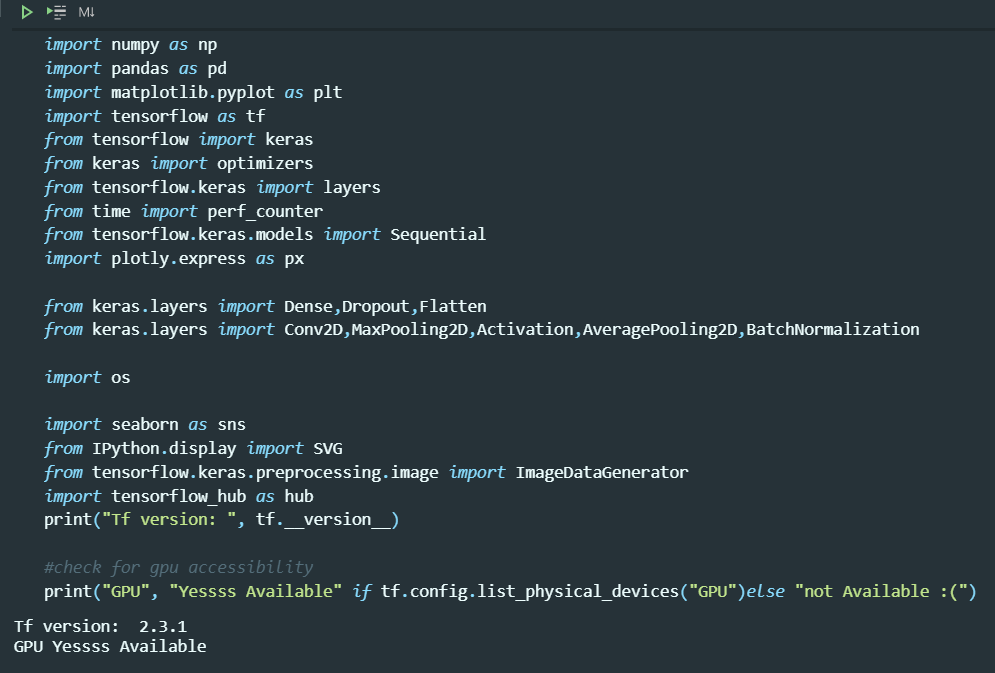
1. Image of transfer learning (Transfer Learning Using AlexNet, 2015)

## Flutter

Flutter is a Google-owned UI toolkit. It has been developed to develop mobile, web, and desktop native applications in a single codebase. What I love about Flutter is that you can create a real native mobile application without much coding. Developing a smartphone application with Flutter is cheaper because you do not need to create and manage two mobile applications. There would not be any notable difference between a native application and a Flutter app if you can easily use and configure widgets offered by Flutter to create a useful UI. I decided to make use of the models listed above and the flutter framework to use in building out this project

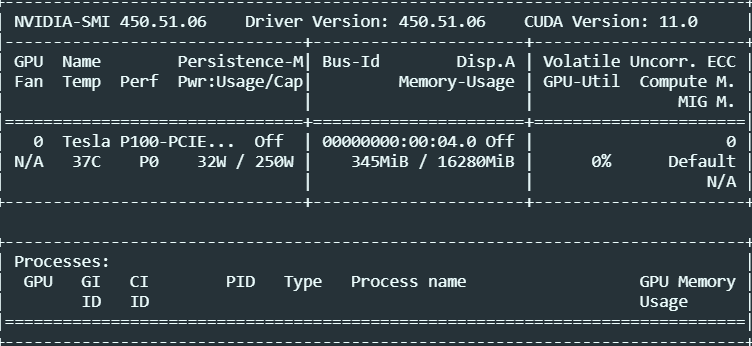
# Implementation of the mobile application

In this project, the following requirements will be needed, most importantly will be the use of Flutter, TensorFlow, TensorFlow-Lite, and Keras, the programming language that will be used in this project is both python and dart. The training environment used in building out the models will be Kaggle's kernel because there is access to a faster GPU. In contrast, Vscode and an android emulator were used in building out the UI and logic of the Flutter application. The image below shows the libraries imported for the analysis and training of the plant Village dataset.



1. Code for libraries imported

Kaggle is a data science and machine learning expert group, one of the world's best. The platform has over 1 million registered users, thousands of public libraries, and code snippets, particularly notebooks. Some of the world's best scientists are actively using this platform. Kaggle offers the use of a GPU for 30 hours a week. The figure below shows the GPU provided, which is an Nvidia tesla p100



1. The image that Nvidia Tesla P100 GPU from Kaggle kernel.

## Data Augmentation

Keras's go-to class for preparing image data for deep learning is ImageDataGenerator. It offers quick access to the local file system and some diverse loading methods from various structures. It has some pre-processing and growing capability for compelling data. The flow\_from\_directory is used to load in the data because the image data classes and label names are on the folders.

1. Validation split - Assigns a train-test-split to the passed data based on the given float. 0.2 means the data has been split into training and testing data ratio of 80/20.
2. Rescale – this function is used to adjust the pixel value of the images passed 1/255
3. Rotation\_range – this function is used to randomly rotate the image data 360 value was used.
4. Filled\_mode – when the image data is rotated, some pixels move outside of the image and leave an empty area filled in with the value "nearest," which fills in space with the nearest pixel value.
5. Random\_shift - The object will not always be in the middle of the frame. To address this problem, we can either horizontally or vertically adjust the image pixels; this is achieved by applying a particular constant value to all pixels. Width\_shift\_range and height\_shift\_range were set to 0.2, which indict by the number of pixels.
6. Flipping images is also a brilliant augmentation procedure. It makes sense to do it on a lot of different pieces—Horizontal-flip and lateral-flip with vertical or horizontal axis tossing.
7. Zoom\_range - The zoom augmentation either zooms in on the image arbitrarily or zooms out of the image.

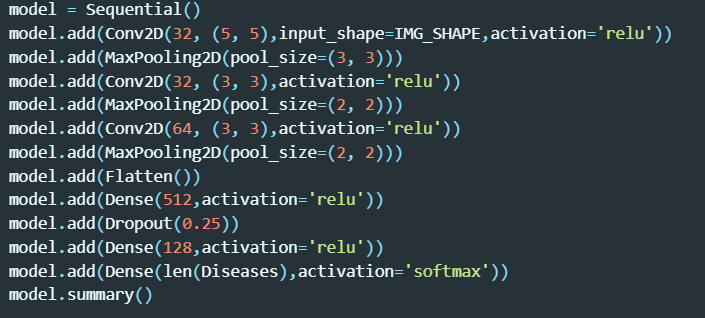
The classes of the saved into a text file by using the class indices essential function. Which will be used as the label for the model and in the flutter mobile application.

## Model architecture

After the data has passed through data augmentation, the next step would be to define the model, the stopping function, the number of training epochs. For the transfer, the same parameters will be used to build and train the model on the training and validation data.

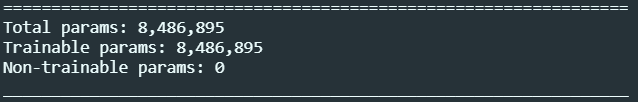
### CNN Model Training

The block of code shown in the image below is the code written to build the convolutional neural network. The model is built the sequential Api class of Keras, Keras been a library used to build a neural network. Sequential API is used to build a simple neural network. The convolutional 2D function is used to build a convolution block. The number 32 shows that 32 features should be extracted from the input shape that was passed that takes in (224, 224, 3), which represents the height, width, and the RGB colour scheme of the input training image. The same code of code uses the activation function called Relu (Rectified linear activation) to fire up neurons. Max pooling 2D is used to build the pooling block. Max pooling in the lock was used to decrease the number of parameters and preserve only the useful information. Three different convoluted and pooling blocks were created, three entirely convoluted layers were created. The last line of code used to build the last convoluted block extracts 64 features from the input image. The dense layer converts a 4D array to a 1D array.



1. The layers of code written to build the CNN neural network

Between the convolution layer and the fully connected layer, a flatten layer is added to converts a two-dimensional feature matrix into a vector that can be fed into a neural network classifier that is connected. The dropout layer is added between the fully connected convolution layer to make some of the neurons relax, and it is also used to prevent overfitting. Both Relu and SoftMax activation is used to build an entirely convoluted layer. The SoftMax activation used in the last layer of the model is used for probability output, the trainable parameters, and the total parameter for training the model 8,486,895.

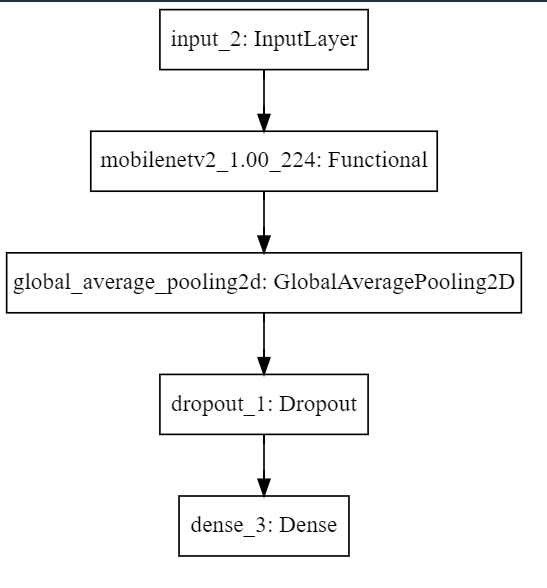
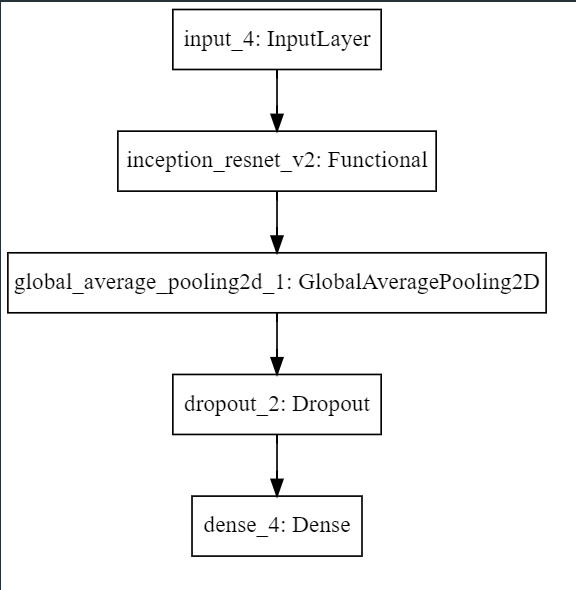


1. Trainable and total parameters of the CNN neural network

The model trained and compiled for 39 epochs using Adam optimizer. The loss function of categorical cross-entropy, the validation metrics is Accuracy.

### mobileNetv2 and inception resnetV2 model training using transfer learning

using TensorFlow, the base model was imported; the input layer is defined by the dimension of our image, and the number of colour channels been the first layer. This is a pre-trained model with no final layer. The model was set to training value to false to prevent the weights from being changed during training. An additional layer is added to the base model using a functional API, calling the fit function to train the model. Using Adam optimizer with learning rate 0.001, loss function with categorical cross-entropy and validation metrics been Accuracy. The number of epochs been 50. The steps per epoch chosen are the number of training data images divided by the batch size, 64, and the validation step function was the validation data divided by the batch size. Callbacks were also created called early stopping, which would stop the model from training when the data's validation accuracy after five rounds of no improving because the patient value passes were for five passes. The inception resnetv2 also was built and fit using almost the same parameter. The only difference in the parameter is in compile function is loss function of categorical cross-entropy with label smoothing = 0.5

1. The transfer learning model architecture of both mobilenetV2 and inception ResNet v2

## Flutter Mobile Application

Created a new project file in Vscode to build the mobile application, there is going to be a main. dart file that comes with the base file of the flutter application, so I created a splashscreen.dart file and importing the material.dart file. In the splash screen file, a stateless widget was created to build the splashscreen and a couple of variables were set which includes the load time, where to navigate to after loading the text on the screen and the image path(which is located in the assets folder). The styles were built to have a colour gradient of green and blue. The pubspec.yaml file is where the assets files will link the image for the application, the inputted labe.txt file and the model.tflite file.

A file was created called home.dart, which is where all the logic of the application will be built. To activate the Tflite library in the flutter, the Gradle file in the android folder will be edited to include the minimum sdk file to 19, and an aaptoption setting in the file that will not make the Tflite file to compress. Two stateful widgets where build to use the functionality of been able to access the camera and been able to access the gallery. Using the scaffold widget, the container widget and column widget to build out the beautiful UI using material Application. The full code of the mobile application can be found on Github and would be located in the appendix.

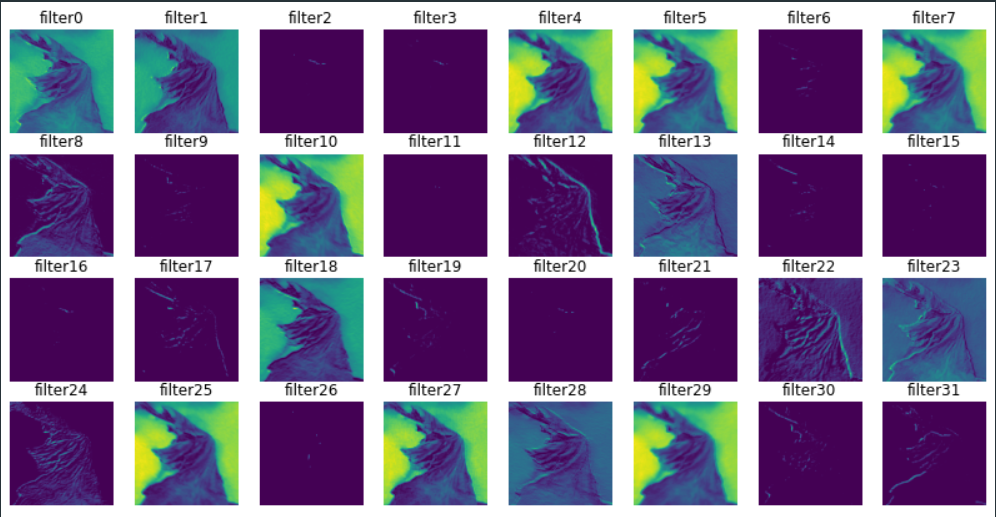
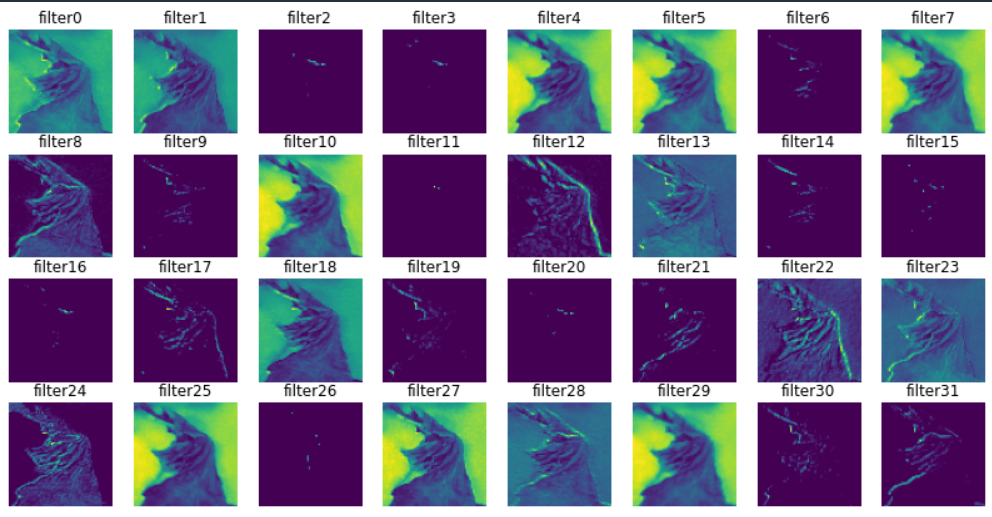
For the logic, an ontap library function was used to create the button that was linked to the gesture detector widget. I will be easier to look through the code that constructed the mobile application interface which will be found on Github.

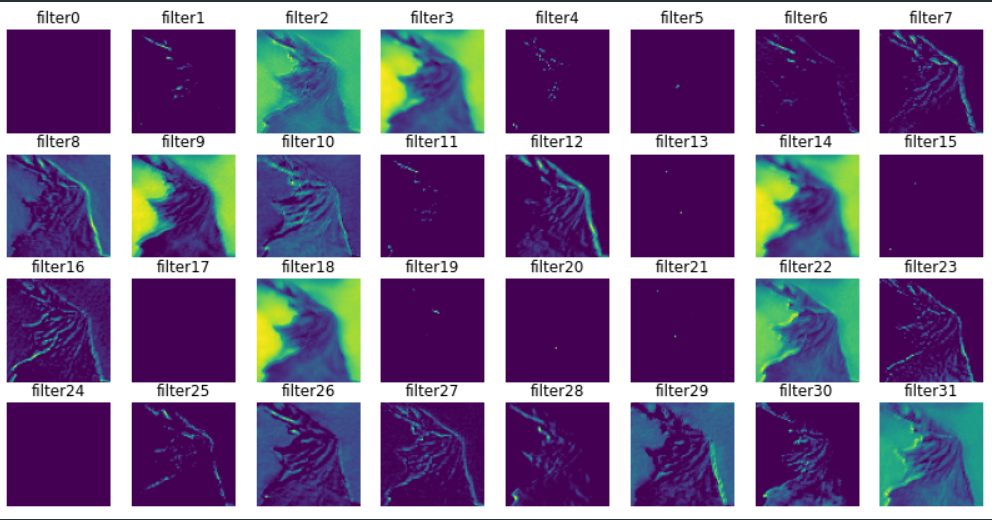
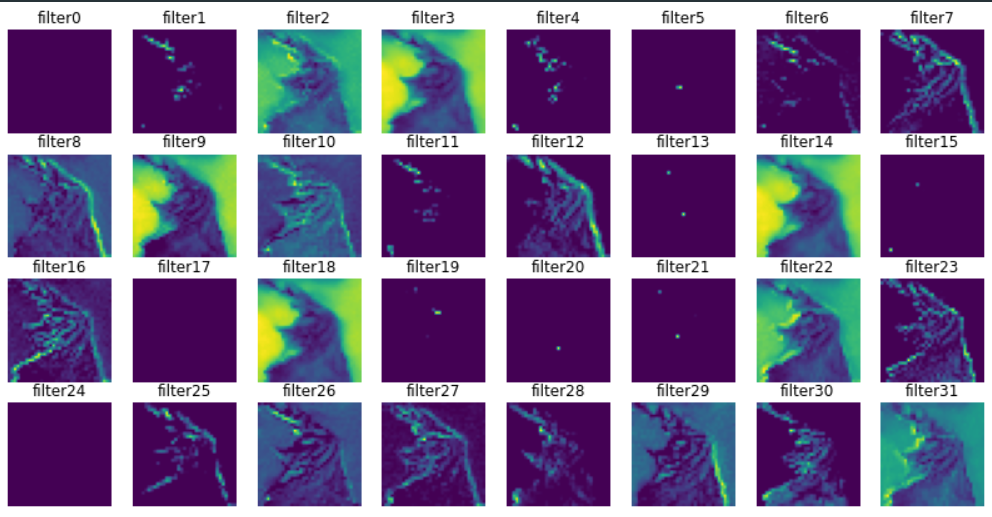
# Analysis of the Models

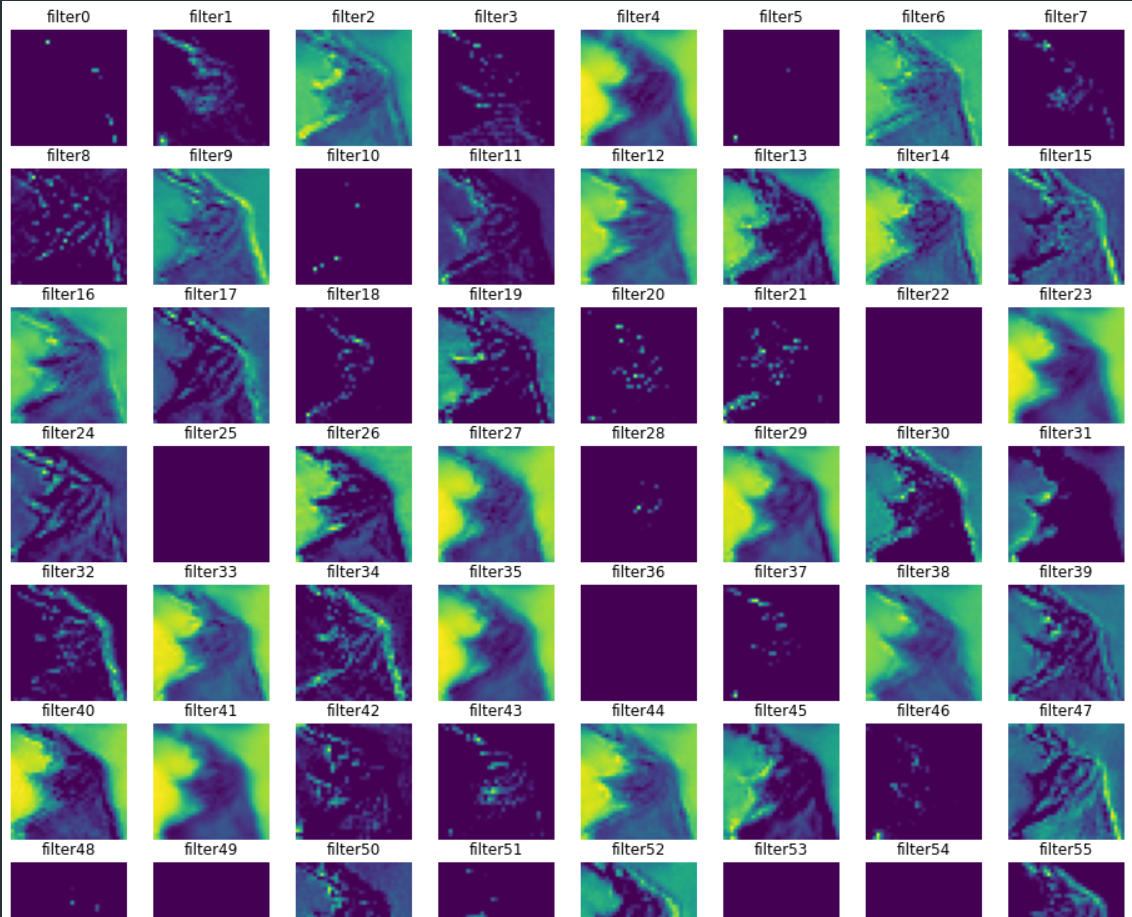
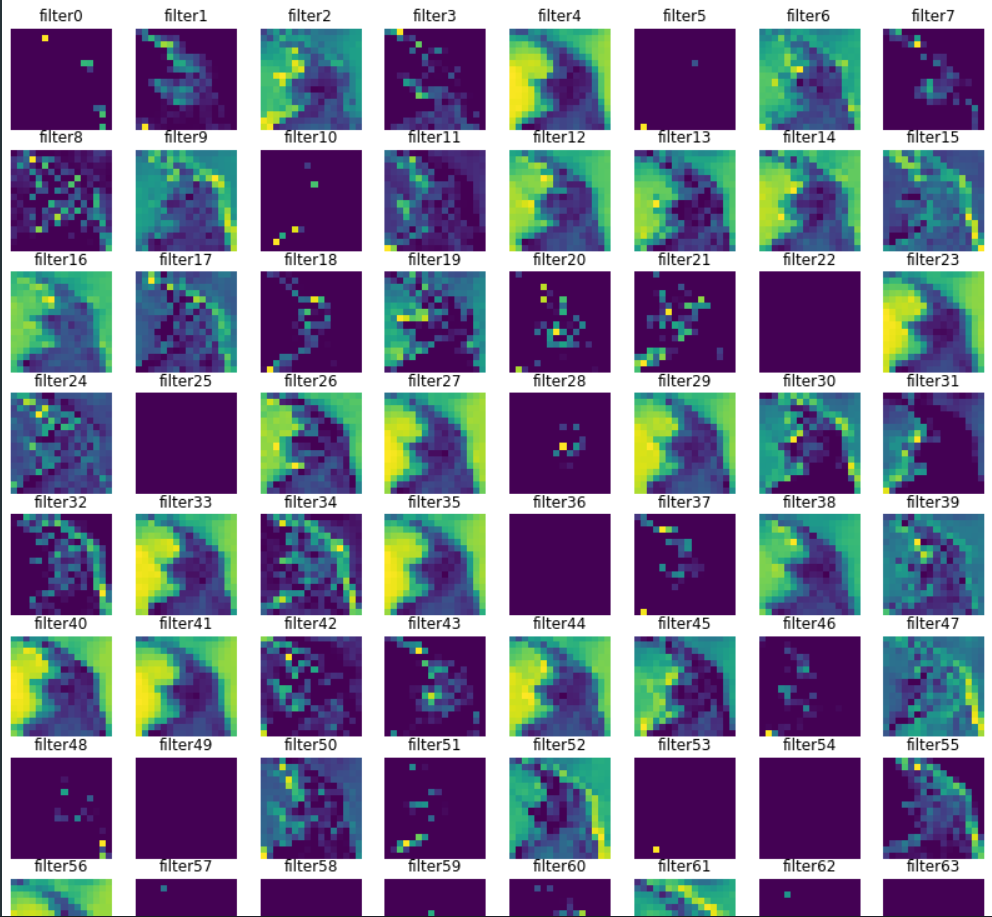
This chapter discusses the analysis done on the plant village dataset. In this chapter, the visualization of the feature maps of the convoluted and pooling layers of the CNN neural network that was created will be discussed. The testing of the model performance and how well the model did training on each epoch against the accuracy and loss functions.

## Visualization of feature Maps

The function maps help to illustrate on any layer what the model is studying. If the depth increases, more spatial data can be learned from the model. This illustrated the neural networks' functionality and predicted behaviour. This further extends and improves the overall model design as we have experience of the actual design, including how it works.

1. Images of the convoluted layers and pooling layers of the neural network

### CNN model validation and Tflite conversion

After the model was fit, the following result of the analysis was gotten and in the CNN model, we can see from the image below that the true label and the predicted label are the same, with the test accuracy been at 95%. This shows that the model performs well on the test data and can also be used in the mobile application.



The images below show the training history of both the loss and accuracy, of the training and testing data respectively. When the model started training we can see from the first image that the analysis show how the training data and validation data fit through the epochs trained on.

Chart, line chart

Description automatically generated

Looking at the accuracy we can see how the model quickly learned the features in the images and improved very quickly after the first epoch and seen to me stable trying to improve after the 25 epoch, both the validation accuracy and validation loss fluctuate a bit while trying to train the model. The model loss shown in the figure below is the lowest gotten in all the experimentation gotten from this project which is 0.16.

Chart, line chart

Description automatically generated

After saving the model, it was then converted into TensorFlow lite format (tflite). The original saved models were 33155kbs in size which is about 33Mbs. After using the Tflite library the quantized model was 8249kbs which is about 8Mbs, which is about 25% of the original model size.

### MobileNetv2 model validation

The image below shows that the trained mobileNet model train well, the true label and predicted label are the same for the test image that was passed, looking at this model tho it didn’t perform as better as the CNN model in both its accuracy and loss, but it can be seen that the size was small and would mean that this model, when utilized in a mobile application, will perform faster in term of the speed required to be utilized on the mobile application CPU.



The images below show the Accuracy and loss that was seen in the training and validation data respectively. There is a difference in how the loss and accuracy of this model differ from all the other models that are in this model. The test accuracy was 85% and the test loss was at 1.01. This is a good prediction on this model and it can be said that if there were more data and classes the model would perform better than required. The slop in the loss shows how the trained data slowly drops in its loss for both the training and testing data.

Chart, line chart

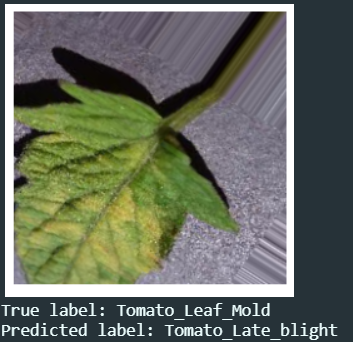
Description automatically generated

Chart, line chart

Description automatically generated

After saving the original model the, the Tflite library was used to convert the original model to a Tflite model. the float model was 8736Kbs in size which is 8Mbs in size, after using the Tflite library the model was quantized in 2279Kbs which is 2Mbs, which is the smallest model in this experimentation.

### Inception ResNetv2



Chart, line chart

Description automatically generated

Chart, line chart

Description automatically generated

|  |  |  |
| --- | --- | --- |
| MODELS | ACCURACY | LOSS |
| CNN model | 0.95 | 0.16 |
| MobileNetv2 | 0.85 | 1.01 |
| Inception resnet v2 |  |  |

# Testing of the Mobile Application

The model and mobile application built with flutter were tested using An android emulator that can be created on android studio. The emulator is named pixel 4XL that uses google API 30, the emulator Os is the android 11 framework, and it uses 2Gb of Ram data. The images below show the splash screen, and how the device gallery was accessed, a picture of the home screen and the process where a random plant disease image was downloaded from the internet and last image is which shows what the prediction on the image was and the model confidence.

To use the Mobile Application, it has two functions, for now, you can use your mobile phone camera to take a photo and the model will try to predict what is wrong with the photo of the plant diseases that were taken. The second functionality is if you already have the image stored in the device local file, you only need to use the device media query to input the image for detection.Graphical user interface, application

Description automatically generated

The mobile application would also run the same way on an IOS mobile phone, the mobile UI was designed to be as simple as it would help farmers to easily understand what the application has to offer. The colour scheme is easier on the eyes and has a feeling of it been an agricultural application. The model prediction time is fast as expected because we made use of the smallest model available in this project for testing.

Graphical user interface, application

Description automatically generatedA picture containing graphical user interface

Description automatically generatedGraphical user interface, application

Description automatically generated

Graphical user interface, application

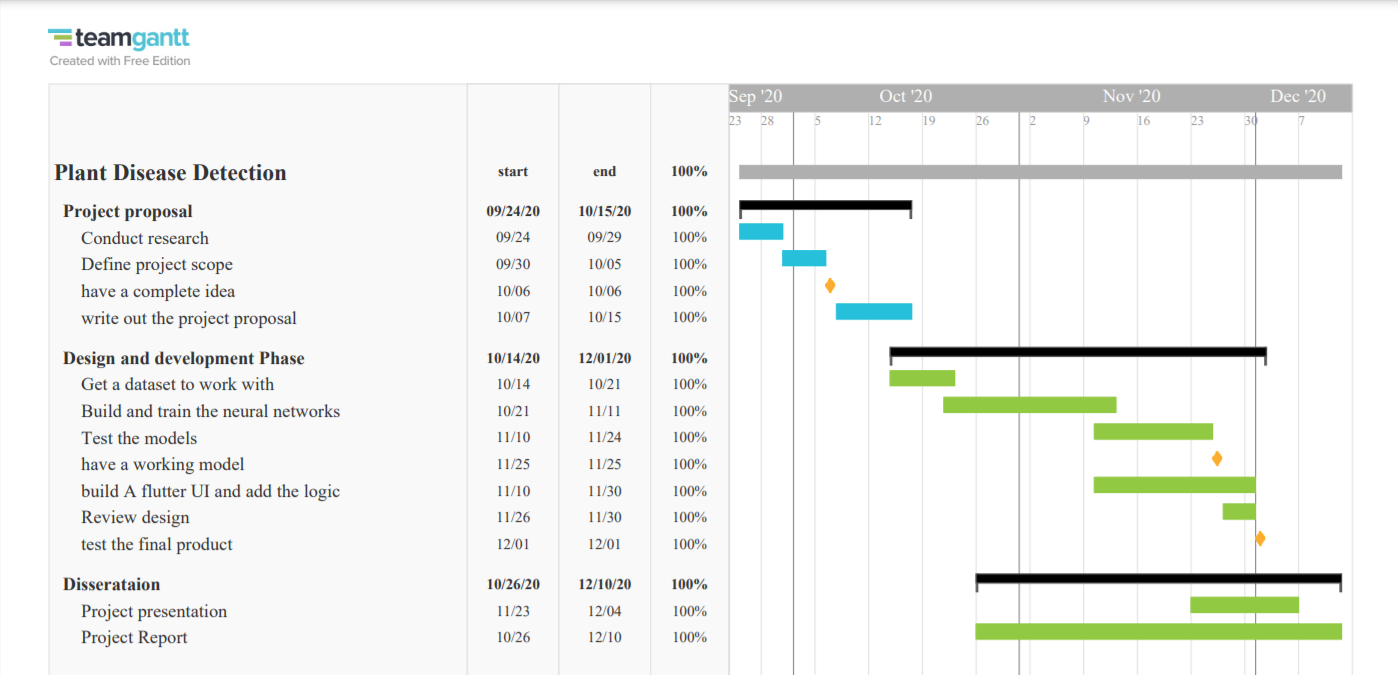
Description automatically generatedText

Description automatically generated

# PROJECT MANAGEMENT

## Project Schedule

[ This could include the work breakdown structure, Gantt chart, and comments about how well you managed to keep to the original plan, or what adjustments were necessary. ]



# CRITICAL APPRAISAL

This research project answered the c to use the question that was asked in the context of how a mobile application that would be built to detect plant disease could solve the problem of a simple farmer that has access to a smartphone. Looking at the validity of this research it can be said that the method used was the right and appropriate method to build a cross-platform mobile application, that will run a neural network as it’s logic to use in detecting plant disease. The mobile application was tested using both ios and an android smartphone. Other literature shows the use of different models and different platform in which the test or application was used in presenting the finished product. My research project shows the use of how the growth in the number of smartphones and the easy assess to a smartphone could be utilized in a major development move in Agriculture. This research concludes that a mobile-optimized neural network trained on an image data could be used together with googles created UI design framework called flutter, to build a mobile application that will be used in mobile application development.

My result finding shows that the questions asked could be justified, and the software development protocol was followed when developing the project. The result gotten from the analysis when compared together shows that the convolution block built in different ways has a different effect on any training data that was inputted. My result is precise as it was tested with two different programming languages and development environment.

# CONCLUSIONS

Concluding this project it can be said that the models that were used to train the dataset would improve more if there were more data images. With a large enough dataset, a more comprehensive list of plant diseases and plant species could be used to create a more robust application. The results gotten from the analysis shows that mobileNetV2 performs well enough to be used in the mobile application. The CNN model that was built performed the best score of 95% on the accuracy level. The use of flutter to build the mobile application makes the project to be developed fast in time by making use of building blocks to construct the interface and connecting the logic for the mobile application with the neural network that was built using TensorFlow library, and flutter Machine learning libraries. The projects answer the question that a mobile application can be built for third world farmer to help them in detecting plant disease which will help in mitigating and reducing the cost of losses to the crops. The project shows the use of transfer learning as a fast way of utilizing a model in machine learning. I figured out that if I have access to a paid cloud GPU I could have trained with large data over a long period without the kernel timing out. Although it might result in more money spent on training the model.

## Achievements

The mobile application achieved a good result on the number of test that was performed on it, the model that was used was the smallest in size making it easier for utilization in different smartphones. the user interface is easy for anyone to understand.

Developing a mobile application for both an Ios and an android smartphone was achieved in this project

## Future Work

Other features could be added to the mobile application, the neural network model that was built could be used as a framework for realtime plant disease detection in drones. Climate data

* The model could be use for real time detection using drones
* Climate data could be integrated for crop yield prediction and recommendation
* A chatbot could be implemented in the App to help farmers navigate solutions
* Farmers social media community and ecommerce function could also be integrated into the mobile application

[ Outline possible enhancements or extensions to the product, or further work needed to address outstanding issues, etc. ]

# STUDENT REFLECTIONS

From the beginning development of this project from deciding the type of dataset to use, to picking out the required models, environment to train and test the model and coding out the user interface for the mobile application, there were a lot of choices to go with. In the case of the dataset, I found 4 datasets online that I could work with, all having a different number of classes. I had to find out how to unzip a file on google colab, as that was the first development environment I wanted to use. The Digipatho took 24hours to unzip but the environment kernel keeps timing out because

[ A reflective and critical appraisal of your personal performance, problems encountered and how they were resolved, lessons learnt, what could have been done better or differently, etc. ]

# BIBLIOGRAPHY AND REFERENCES

Ramcharan, A., McCloskey, P., Baranowski, K., Mbilinyi, N., Mrisho, L., Ndalahwa, M., Legg, J., & Hughes, D. P. (2019). A mobile-based deep learning model for cassava disease diagnosis.*Frontiers in Plant Science; Front Plant Sci, 10*, 272.



Arnal Barbedo, J.G. (2013) "Digital Image Processing Techniques for Detecting, Quantifying and Classifying Plant Diseases". SpringerPlus [online] 2 (1). [12 November 2020]

*Advanced topics in computer vision* (2013). In Farinella G. M., Battiato S. and Cipolla R.(Eds.), . Springer London, Limited.

ITU (2015). *ICT Facts and Figures – the World in* 2015. Geneva: International Telecommunication Union

Solem, J. E. (2012). In Oram A., Hendrickson M. and Muller L.(Eds.), *Programming computer vision with python* (First edition.. ed.)

Bernhardt, C. author, (2016). *Turing's Vision: The birth of computer science*

Davies, E. R. (2017). *Computer vision: Principles, algorithms, applications, learning*. Saint Louis: Elsevier Science & Technology.

United Nations (2020) *United Nations | Peace, Dignity and Equality <BR> on a Healthy Planet* [online] Available from <https://www.un.org/en/> [13 November 2020]

Ehler, L. E. (2006). Integrated pest management (IPM): Definition, historical development and implementation, and the other IPM. Pest Management Science, 62(9), 787-789.



Ghimire, B. (2017) "Analysis of Yield and Yield Attributing Traits of Maise Genotypes in Chitwan, Nepal". *Journal of Food Processing & Technology* [online] 08 (01). available from <https://www.longdom.org/proceedings/developing-plant-disease-management-strategies-for-smallholder-vegetable-farmers-in-the-limpopo-province-south-africa-36394.html> [13 November 2020]

Nelson, R., Orrego, R., Ortiz, O., Tenorio, J., Mundt, C., Fredrix, M., and Vien, N.V. (2001) "Working with Resource-Poor Farmers to Manage Plant Diseases". Plant Disease 85 (7), 684–695

Anderson, P. K., Cunningham, A. A., Patel, N. G., Morales, F. J., Epstein, P. R., & Daszak, P. (2004). Emerging infectious diseases of plants: Pathogen pollution, climate change and agrotechnology drivers.*Trends in Ecology & Evolution, 19*(10), 535-544.

Ansari, S. a. (2020). In SpringerLink (Online service) (Ed.), *Building computer vision applications using artificial neural networks with step-by-step examples in OpenCV and TensorFlow with python* (1st ed. 2020. ed.)

Schmidhuber, J. (2015). Deep learning in neural networks: An overview.*Neural Networks, 61*, 85-117.

Argüeso, D., Picon, A., Irusta, U., Medela, A., San-Emeterio, M. G., Bereciartua, A., & Alvarez-Gila, A. (2020). Few-shot learning approach for plant disease classification using images taken in the field.*Computers and Electronics in Agriculture, 175*, 105542.

Liu, L., Wang, R., Xie, C., Yang, P., Wang, F., Sudirman, S., & Liu, W. (2019). PestNet : an end-to-end deep learning approach for large-scale multi-class pest detection and classification  - White Rose Research Online. *Whiterose.Ac.Uk*.

M. Francis, & C. Deisy. (2019). Disease detection and classification in agricultural plants using convolutional neural networks — A visual understanding. Paper presented at the *- 2019 6th International Conference on Signal Processing and Integrated Networks (SPIN),*1063-1068.

*Animation: How Plant Diseases Devastate Food Crops in African Countries – Connected* (2019) available from <https://www.connectedvirus.net/animation-how-plant-diseases-devastate-food-crops-in-african-countries/> [11 November 2020]

Isabel. (2019, 26 September). How plant diseases devastate food crops in African countries. Video animation - The Global Plant Council. The Global Plant Council. <https://globalplantcouncil.org/how-plant-diseases-devastate-food-crops-in-african-countries-video-animation/>

Bock, C. H., Poole, G. H., Parker, P. E., & Gottwald, T. R. (2010). Plant disease severity estimated visually, by digital photography and image analysis, and by hyperspectral imaging.*Null, 29*(2), 59-107. <https://doi.org/10.1080/07352681003617285>

Amara, J., Bouaziz, B., & Algergawy, A. (2017). A deep learning-based approach for banana leaf diseases classification.*Datenbanksysteme Für Business, Technologie Und Web (BTW 2017)-Workshopband,*

Brahimi, M., Boukhalfa, K., & Moussaoui, A. (2017). Deep learning for tomato diseases: Classification and symptoms visualization.*Applied Artificial Intelligence, 31*(4), 299-315.

Cruz, A. C., Luvisi, A., De Bellis, L., & Ampatzidis, Y. (2017). (2017). Vision-based plant disease detection system using transfer and deep learning. Paper presented at the *2017 Asabe Annual International Meeting,*1.

Cruz, A. C., Luvisi, A., De Bellis, L., & Ampatzidis, Y. (2017). X-FIDO: An effective application for detecting olive quick decline syndrome with deep learning and data fusion.*Frontiers in Plant Science, 8*, 1741.

DeChant, C., Wiesner-Hanks, T., Chen, S., Stewart, E. L., Yosinski, J., Gore, M. A., Nelson, R. J., & Lipson, H. (2017). Automated identification of northern leaf blight-infected maize plants from field imagery using deep learning.*Phytopathology, 107*(11), 1426-1432.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis.*Computers and Electronics in Agriculture, 145*, 311-318.

Liu, B., Zhang, Y., He, D., & Li, Y. (2018). Identification of apple leaf diseases based on deep convolutional neural networks.*Symmetry, 10*(1), 11.

Lu, Y., Yi, S., Zeng, N., Liu, Y., & Zhang, Y. (2017). Identification of rice diseases using deep convolutional neural networks.*Neurocomputing, 267*, 378-384.

Mohanty, S. P., Hughes, D. P., & Salathé, M. (2016). Using deep learning for image-based plant disease detection.*Frontiers in Plant Science, 7*, 1419.

Oppenheim, D., & Shani, G. (2017). Potato disease classification using convolution neural networks.*Advances in Animal Biosciences, 8*(2), 244.

Ferentinos, K. P. (2018). Deep learning models for plant disease detection and diagnosis.*Computers and Electronics in Agriculture, 145*, 311-318.

Sibiya, M., & Sumbwanyambe, M. (2019). A computational procedure for the recognition and classification of maize leaf diseases out of healthy leaves using convolutional neural networks.*AgriEngineering, 1*(1), 119-131.

TÜRKOĞLU, M., & Hanbay, D. (2019). Plant disease and pest detection using deep learning-based features.*Turkish Journal of Electrical Engineering & Computer Sciences, 27*(3), 1636-1651.

Zhang, K., Wu, Q., Liu, A., & Meng, X. (2018). Can deep learning identify tomato leaf disease?*Advances in Multimedia, 2018*

Ramcharan, A., Baranowski, K., McCloskey, P., Ahmed, B., Legg, J., & Hughes, D. P. (2017). Deep Learning for Image-Based Cassava Disease Detection. Frontiers in Plant Science, 8. <https://doi.org/10.3389/fpls.2017.01852>

Yamamoto, K., Togami, T., & Yamaguchi, N. (2017). Super-Resolution of Plant Disease Images for the Acceleration of Image-based Phenotyping and Vigor Diagnosis in Agriculture. *Sensors*, *17*(11), 2557. <https://doi.org/10.3390/s17112557>

Durmuş, H., Güneş, E. O., & Kırcı, M. (2017). (2017). Disease detection on the leaves of the tomato plants by using deep learning. Paper presented at the *2017 6th International Conference on Agro-Geoinformatics,*1-5.

Fujita, E., Kawasaki, Y., Uga, H., Kagiwada, S., & Iyatomi, H. (2016). (2016). Basic investigation on a robust and practical plant diagnostic system. Paper presented at the *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA),*989-992.

Rangarajan, A. K., Purushothaman, R., & Ramesh, A. (2018). Tomato crop disease classification using pre-trained deep learning algorithm.*Procedia Computer Science, 133*, 1040-1047.

Too, E. C., Yujian, L., Njuki, S., & Yingchun, L. (2019). A comparative study of fine-tuning deep learning models for plant disease identification.*Computers and Electronics in Agriculture, 161*, 272-279.

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*, 1–11.

Ma, J., Du, K., Zheng, F., Zhang, L., Gong, Z., & Sun, Z. (2018). A recognition method for cucumber diseases using leaf symptom images based on deep convolutional neural network.*Computers and Electronics in Agriculture, 154*, 18-24.

Brahimi, M., Mahmoudi, S., Boukhalfa, K., & Moussaoui, A. (2019). (2019). Deep interpretable architecture for plant diseases classification. Paper presented at the *2019 Signal Processing: Algorithms, Architectures, Arrangements, and Applications (SPA),*111-116.

Singh, D., Jain, N., Jain, P., Kayal, P., Kumawat, S., & Batra, N. (2019). *PlantDoc: A Dataset for Visual Plant Disease Detection*. <https://arxiv.org/pdf/1911.10317.pdf>

Transfer Learning Using AlexNet. (2015). Mathworks.com. <https://uk.mathworks.com/help/deeplearning/ref/alexnet.html#bvn44n6>

Barbedo, Jayme Garcia Arnal, et al. "Annotated Plant Pathology Databases for Image-Based Detection and Recognition of Diseases." *IEEE Latin America Transactions* 16.6 (2018): 1749-1757

Zhang, Y., & Wallace, B. C. (2016). A sensitivity analysis of convolutional neural networks for sentence classification.*arXiv Preprint arXiv:1510.03820,*

‌Yosinski, J., Clune, J., Bengio, Y., & Lipson, H. (2014). How transferable are features in deep neural networks? ArXiv.org. <https://arxiv.org/abs/1411.1792>

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

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. ArXiv.org. https://arxiv.org/abs/1801.04381v

Appendix B – Interim Progress Report and Meeting Record

Appendix E – Project Presentation

Appendix F – Certificate of Ethics Approval



Appendix X – As required

https://www.kaggle.com/michaelajao/ewe-plant-disease-detection