2021

A Deep normalized neural network model for strawberry fungal leaf disease detection.

Kerre, Deperias Webula
Faculty of Information Technology
Strathmore University

A Deep Normalized Neural Network Model for Strawberry Fungal Leaf Disease Detection

Deperias Webula Kerre

Submitted in Partial Fulfillment of the Requirements for the Award of the Degree of Master of Science in Information Technology at Strathmore University

School of Computing and Engineering Sciences

Strathmore University

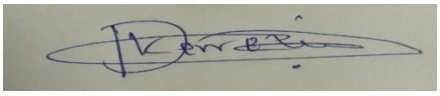
Nairobi, Kenya

December, 2021

Declaration

I declare that this work has not been previously submitted and approved for the award of a degree by this or any other University. To the best of my knowledge and belief, the thesis contains no material previously published or written by another person except where due reference is made in the thesis itself.

Deperias Webula Kerre



7th September 2021

Approval

The thesis of Deperias Webula Kerre was reviewed and approved by the following:

Dr. Henry Muchiri,
Lecturer, School of Computing and Engineering Sciences,
Strathmore University

Dr. Julius Butime,
Dean, School of Computing and Engineering Sciences,
Strathmore University

Dr. Bernard Shibwabo,
Director of Graduate Studies,
Strathmore University

Abstract

Strawberry is one of the cash crops that are being grown in Kenya for both export and local consumption .However, strawberry fungal leaf diseases are threatening the existence of this crop which is an important input in the agricultural production sector. The types of strawberry fungal leaf diseases resulting to greater losses in production include Strawberry Leaf scorch, Strawberry Leaf Spot and Strawberry Leaf Blight. The biggest challenge the farmers face is that of correctly classifying these diseases based on observable leaf features. Famers have incurred losses due to poor/incorrect control measures which results from the misdiagnosis of these diseases. This scenario is more pronounced in rural settings where the farmers have a limited access to expertise in modern agricultural production. As a result of this, automated classification of strawberry plant fungal leaf diseases is highly desired. The literature review found several computer vision techniques that have been leveraged in Strawberry fungal leaf disease detection. Among these solutions are the convolutional neural network-based models. Despite the high detection accuracy, the models do not cover another of strawberry fungal leaf diseases such as leaf spot and fail to generalize well on unseen data. The models also do not consider cases where more than one disease occur on the same part of the plant, in this case the leaf. In this study, a deep learning model was proposed for classifying fungal leaf diseases in strawberry based on an experimental research design. The model generalized well on previously unseen data and considered a scenario where multiple diseases occur on the same leaf(Leaf Scorch and Leaf Blight). The model also covered strawberry Leaf Spot that was not covered by any of the existing deep learning models. Data samples containing a total of 1,134 leaf images, categorized into five classes including healthy leaf images were split into 80% training and 20% validation. The disease classes include strawberry leaf spot, leaf scorch, leaf blight and a class where two diseases (Leaf Blight and leaf Scorch) occur together. The model was trained on 30 epochs from scratch with batch normalization implemented within the convolutions in Keras framework and validated using a confusion matrix. The model achieved an outstanding classification accuracy of 98% , precision of 97% , recall of 95.7% and an F1-score of 96.3%.

Keywords: Batch Normalization (BN), Computer Vision(CV), Convolutional Neural Networks (CNNs), Data augmentation (DA), Deep learning, Image Processing, Strawberry Fungal Leaf Disease Detection.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Algorithms

Definition of Terms

Batch Normalization- This is a technique in used in machine learning to improve the learning rates and generalizability of the models. It involves normalizing the input data to the convolutions of the neural network (Ioffe & Szegedy, 2015).

Data Augmentation-The process of artificially inflating datasets using a series of transformations such as rotation, cropping, flipping (Taylor and Nitschke, 2017).

Deep Learning-This is a subfield of machine learning concerned with the development of neural networks that are deeper or consists of several convolutions. The deeper learning enables them to learn and model complex scenarios (Nielsen, 2015).

Deep Normalized Neural Network Model- A CNN model with batch normalization implemented within the convolution layers(Yin et al., 2017).

Disease Detection- A term that describes the process of establishing the presence of a disease and identifying the disease label.

Strawberry Fungal Leaf Diseases-A plant disease is defined as anything that prevents it from performing to its maximum potential in terms of yield. Strawberry fungal leaf diseases are the ones caused by fungus (plant pathogen) and affect the leaf parts of the plant(Mel'nik, 2000).

Computer Vision-This is the process of modelling and replicating human vision using computers(Zhang et al., 2009).

Machine Learning-A field of computer science that uses statistical techniques to give computers the ability to "learn", without being explicitly programmed(Zhang et al., 2009).

List of Abbreviations/Acronyms

ANN-Artificial Neural Network

BN-Batch Normalization

CNNs - Convolutional Neural Networks

CV-Computer Vision

DA-Data Augmentation

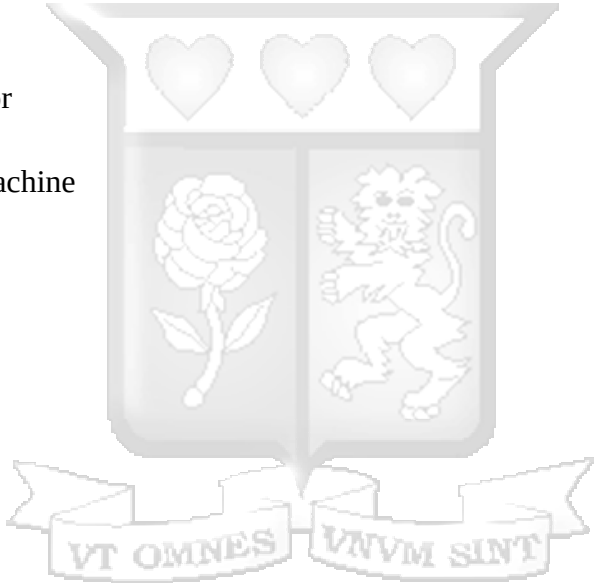
DNA-Deoxyribonucleic Acid

DL- Deep Learning

KNN-K-Nearest Neighbor

SVM- Support Vector Machine

UI-User Interface



Acknowledgement

Sincere gratitude to the Lord God Almighty for giving me good health, strength, time and grace to undertake this study. I would like to thank my immediate supervisor, Dr. Henry Muchiri, for his tireless guidance and support, encouragement and honest critique throughout this study. I would like to also thank Strathmore university, specifically the School of Computing and Engineering Sciences for offering me a partial scholarship to undertake a Masters degree in the school. I cannot forget to thank the thesis coordinator, Dr. Omwenga who assisted with coming up with timely milestones that assisted me to achieve the research objectives. Special thanks also goes to my classmates i.e. the MSIS 2021 and MSIT 2021 classes for the encouragement, knowledge sharing and even the light moments we shared. I cannot also forget Dr. Shibwabo for assisting in the earlier stages of my research. Sincere gratitude to Mr. Wambugu for allowing the taking of sample pictures of strawberry leaves from his farm and Mr. Wafula from KARLO for assisting in assigning disease labels to the raw leaves. Thanks and God Bless you all.

Dedication

I would like to dedicate this study to my parents Mr. Patrick Kerre and Mrs. Winrose Kerre, and my family in general who have been a great inspiration that helped me to always aim higher in my endeavors. God bless you all.

Chapter 1: Introduction

1.1 Background of the study

Agricultural production is an important input in the economic development of many countries around the world including Kenya. Strawberry is one of the cash crops grown in Kenya for both export and local consumption. The crop accounts for more than 50% of the Kenya's agricultural foreign exchange earnings (Korir et al., 2015). The high value of strawberry crop has positively impacted its production, marketing, and consumption hence availing a great potential for income generation and employment creation (Mwangi et al., 2017). The fruits are an important contribution to household nutrition through availability of vitamins. Strawberries also do have substantial foreign market in the European union (Mwangi & Mwaura, 2009). Strawberry fungal leaf diseases are threatening the livelihood of this crop since they have been a bottleneck to the efforts towards increasing production. This has also caused major production and economic losses which has resulted in a negative impact on the country's economic stability and food security (Egesa et al., 2016). According to Egesa et al. (2016), strawberry fungal diseases accounted for 19 % loss in the crop yield. The disease classes of interest in this case include: Leaf Blight, Leaf Spot and Leaf Scorch. Several solutions have been developed to detect fungal strawberry plant leaf diseases; the real time PCR analysis assay for detecting strawberry Leaf spot and Strawberry Leaf Blight requires a lot of time to do carry out the laboratory analysis (Weller et al., 2000). This method also requires special facilities which are not usually available in rural areas where the crops are grown. Kusumandari et al (2019) proposed a model for detecting Strawberry leaf spot based on color segmentation, the model is not suitable for cases where the farmers are facing a problem with the other classes of leaf diseases. Convolutional neural networks (CNNs) have demonstrated great potential in object recognition and image classification tasks (Atabay, 2016).Some of the related works are by Mohanty, Hughes, & Salath´e (2016), Ferentinos (2018), Brahimi et al (2020), Chohan, Khan & Katper (2020), Kusumandari et al (2019), Fang et al (2020) and Xiao et al(2021). The problem of fungal leaf disease detection is more pronounced especially in the cases of Strawberry Leaf Spot and Leaf Scorch; these two classes of diseases are very difficult to distinguish due to the close signs (Mel'nik, 2000). The models developed by Ferentinos (2018) and Mohanty, Hughes, & Salath´e (2016) also failed to generalize well on previously unseen data. Some of the fungal diseases such as leaf blight and leaf scorch occur together on the same leaf. None of the methodologies

has taken this into consideration. Therefore, there is need for a fast, automatic, less expensive, and accurate method of classifying these three classes of strawberry plant leaf diseases hence leading to their detection. There is also need to consider cases where more than one classes of the disease occur on the same part of the leaf, for instance strawberry leaf Blight and leaf scorch.

1.2 Problem Statement

Strawberry farmers in Kenya experience a challenge when it comes to classifying the strawberry fungal leaf diseases. This problem has led to poor yields and production loss amounting to 19 % (Egesa et al, 2016). The close signs exhibited by the diseases on the leaves make it difficult in classifying them correctly; this leads to misdiagnosis hence applying the wrong control measures to the diseases. Several computer vision techniques have been leveraged in the detection of these diseases. Despite the success in the detection accuracy, the techniques are limited in the number of strawberry fungal leaf disease classes covered and none of the methodologies considered the case of more than disease occurring on the same leaf. The methods based on manual feature extraction only considered strawberry leaf spot(Kusumandari et al, 2019; Kiani & Mamedov, 2017) hence not suitable for a multi-class scenario. The deep learning models considered strawberry leaf scorch and leaf Blight (Brahimi et al, 2020; Chohan, Khan & Katper, 2020; Fang et al, 2020;Xiao et al, 2021; Ferentinos, 2018; Mohanty, Hughes & Salath´e, 2016). There is therefore need for a method for the accurate classification of the strawberry fungal leaf diseases, with more classes considered and considering cases where more than one class of fungal diseases happening on the same leaf.

1.3 Research Objectives

1.3.1 Main Objective

The main objective of this study was to solve the problem of strawberry fungal leaf disease classification for the following fungal disease classes: Leaf Blight, Leaf Scorch and Leaf Spot by developing a multiclass deep neural model based on a normalized CNN.

1.3.2 Specific Objectives

1. To analyze strawberry fungal leaf diseases affecting strawberry farming in Kenya.
2. To review the effectiveness of the current methods used for Strawberry fungal leaf disease classification.
3. To develop a model for classifying strawberry fungal leaf diseases using deep learning techniques.
4. To validate the performance of the deep learning model for classifying the strawberry fungal leaf diseases.

1.4 Research Questions

1. Which strawberry fungal leaf diseases are affecting strawberry farming in Kenya?
2. What is the effectiveness of the current methods used for classifying strawberry fungal leaf diseases?
3. How can a model for classifying strawberry fungal leaf diseases based on deep learning techniques be developed?
4. How is the performance of the deep learning model for classifying strawberry fungal leaf diseases validated?

1.5 Justification

The ability to accurately determine the class of a strawberry fungal leaf disease is of prime importance to the farmers. These diseases have been responsible for losses amounting to 19% to the farmers(Egesa et al., 2016). The proposed deep learning model will enable the farmers to accurately classify the three classes of strawberry plant fungal leaf diseases; this will enable them to seek for the appropriate guidance hence avoiding losses resulting from misdiagnosis such as applying wrong control measures.

The study aimed to come up with a model for distinguishing the three classes of fungal strawberry leaf diseases, as opposed to the existing solutions that cover a single class of the diseases at a time. This eliminates the need for multiple solutions to handle the diseases. The deep learning model also eliminates the need for manual feature engineering as it enables automatic learning of the disease features during training.

The study helps researchers and students by contributing to the knowledge base in this area of study. The model created can also be used by other application developers who may wish to enhance their applications with image recognition or to future researchers in computer vision and deep learning to build more on the research.

1.6 Scope and Limitation

This study was limited to three classes of strawberry fungal leaf diseases i.e., Leaf Scorch, Leaf Blight and Leaf Spot and an instance of two classes of the diseases happening together(Leaf blight and leaf scorch). This is due to the prevalence and impact of these diseases on strawberry production in Kenya (Egesa et al., 2016) and the difficulty in distinguishing them due to the close signs (Mel'nik, 2000). The model was not able to give information on the stage of the diseases and a direction on the control measures.

Chapter 2: Literature Review

2.1 Introduction

This chapter gives the empirical literature of the study; this clearly outlines the strawberry fungal leaf disease classes and the challenges they impose on strawberry farming. Furthermore, the distinguishing leaf features for strawberry fungal leaf diseases aid in conceptualizing measures ought to be undertaken for strawberry fungal leaf disease classification.This is followed by a detailed description of the relevant machine learning frameworks and models, the architectures, a proper review of the existing technological solutions for strawberry fungal leaf disease classification, the research gap and finally the conceptual framework.

2.2 Strawberry Fungal Leaf Diseases

A plant disease is defined as a condition that prevents it from performing to its maximum potential in terms of yield. Strawberry fungal leaf diseases are the ones caused by the fungus (plant pathogen) and affect the leaf parts of the strawberry (Mel'nik, 2000).

The survival rate of strawberry plants in Kenya has been threatened by the presence of fungal diseases that have led to losses in the production of this crop. Strawberry plant fungal leaf diseases have been a major threat to farmers especially those in rural areas where there is limited expertise. According to Egesa et al (2016), in a survey conducted in central Kenya on the challenges affecting strawberry farming, these diseases have resulted to a loss contribution of 19% in the total yields of this crop. The classes of strawberry fungal leaf diseases in considered in the study are as discussed below:

2.2.1 Strawberry Leaf Spot

Ramularia tulasnei Sacc is the fungus that causes strawberry leafspot. It is the most harmful strawberry leaf disease (Uselis et al., 2006). The first signs include small, circular spots (purple in color) on the surface of young leaflets. This is followed by the enlargement of the lesion and the center of the spot turning gray to white being surrounded by distinct reddish-brown borders. The other parts infected by the fungus include fruit stalks, fruit calyces and leaf stems which shows the same signs as those on the leaves. The main difference is that the lesions are more elongated on stems and stalks. These lesions usually result to a lower grade and render the fruit unmarketable (Menhood et al., 2018). The disease

has a prevalence range of 43-46 % and disease intensity 14-15.7% on average. Figure 2.1 shows a leaf infected by Leaf Spot.

Figure 2. 1:Strawberry Leaf Spot Disease

2.2.2 Strawberry Leaf Scorch

Strawberry leaf scorch disease is caused by fungus *Diplocarpon carliana* and may be distinguished from Leaf spot by the dark purplish spots about one quarter of an inch (Mel'nik, 2000). Strawberry Leaf spot and leaf scorch usually occur on the same pant and result in the same damage. They are usually mistaken for the different stages of the same disease. Both leaf scotch and Leaf spot usually have a severe damage to the plant; may kill so many leaves such that the whole plant is killed or weakened (Agricultural Research Service US Publication, 1978). Strawberry leaf scorch is as wide as the Leaf spot and occasionally does more damage than the leaf spot. The symptoms on the diseased leaf are as shown in figure 2.2. It does not only attack the leaves, but also does considerable damage to the caps, petioles, stolons and fruit stalks (Mel'nik, 2000).

Figure 2. 2: Strawberry Leaf Scorch Disease

(Chohan et al., 2020)

2.2.3 Strawberry Leaf Blight

Strawberry Leaf Blight is a disease that also affect and result to a great damage to the strawberry plant. *Phomopsis Obscurans* is the fungus that causes this disease. It is responsible for making the plant week hence making it susceptible to other pathogens (Ellis, Wilcox & Madden, 1998). The initial

symptoms include one to several spots which are red or purple in color. The spots then enlarge to v-shaped lesions consisting of a light brown inner zone and a dark brown outer zone. The resultant lesions follow major veins progressing inwards. Figure 2.3 shows a leaf infected by leaf blight. In severe cases, the infected leaves may turn brown and die.

Figure 2. 3: Strawberry Leaf Blight

(Xiao et al., 2021)

There are cases where more than one class of strawberry fungal disease occur on the same leaf. A good example of this is the case of strawberry leaf blight and leaf scorch. Figure 2.4 shows a sample leaf collected exhibiting such a case.

Figure 2. 4: Strawberry Leaf Blight and Leaf Scorch

2.3 Strawberry Fungal Leaf Disease Classification

Plant disease classification is the process of determining the class of a given disease based on certain features. There are several distinguishing features that are used in classification of strawberry plant fungal leaf diseases. They largely depend on the detection method to be used.

2.3.1 The distinguishing features for fungal strawberry leaf diseases

Most farmers especially in the rural areas rely on memory to identify the plant diseases. In such cases, the farmer must have an experience of the diseases in the past although the detection accuracy could not still be validated. Farmers who are planting the crops for the first time and therefore inexperienced usually have a problem in detecting the diseases. There are several distinguishing features that are used in classification of strawberry plant fungal leaf diseases. The features are dependent on the

detection method used. They range from the analysis of the DNA gene strains of the diseased leaves (Weller et al, 2000) to performing laboratory tests on leaf tissue samples (Chandra et al, 2015,). The use of these features may need special materials such as reagents and expertise for high end analysis.

Observable leaf features also play an important role in the detection of these diseases; these features can be analyzed by computer vision algorithms to assist in making inferences regarding the presence of various classes of the diseases(Szandala & Backhouse, 2001). The use of observable leaf features is advantageous as it does not need special facilities such as reagents or expertise to capture and analyze them; the only needed item is a digital camera to capture quality images of the leaf samples.

2.4 Strawberry Fungal Leaf Disease Classification Methods

Several methodologies have been developed for the classification of strawberry fungal leaf diseases. These can be classified into the following categories:

2.4.1 Clinical Methods

Several clinical methodologies have been put in place for detecting strawberry plant leaf diseases. One of these methods is the real-time (TaqMan) PCR assay. This was invented for detecting strawberry angular leafspot pathogen Xanthomonas fragariae (Xf) and the strawberry bacterial blight pathogen Xanthomonas arboricola pv. fragariae (Xaf) (Weller et al, 2000). The method was developed by an analysis of the DNA gene strains of the normal healthy leaves and those affected by the pathogens. The detection of both pathogens to 10^3 cells per strawberry leaf disc was facilitated by a modification of an existing DNA genomic extraction rule. The method achieved the detection by a comparison of the DNA gene sequence of the healthy and affected leaves. Evidently, the method requires expertise in the analysis of the DNA gene sequences and may also need special facilities; these are not usually readily available especially in the rural areas where the strawberry plantations are. It may also take time to source the experts hence leading to more losses.

Other methods developed to detect the leaf diseases are those ones based on laboratory testing of samples such as loop-mediated isothermal amplification (LAMP) and the enzyme-linked immunosorbent assay (ELISA). These methods are also time and labor consuming and require specialized skills and controlled laboratory conditions.

2.4.2 Computer Vision Techniques

Computer vision is an interdisciplinary study that deals with how computers can gain high level understanding of digital images or videos. This understanding has been leveraged in plant disease classification. Machine learning is an application of Artificial Intelligence (AI) that provides machines with the ability to learn and improve with experience without explicitly programming them. It is basically a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data, or to perform other kinds of decision making under uncertainty (Robert, 2014). Machine learning has also been integrated into computer vision to improve the process of disease classification (Ferentinos, 2018).

The overall procedure of a plant disease detection system based on digital image processing comprises of five major steps that aid in the process of disease classification. The steps are as illustrated in figure 2.5.

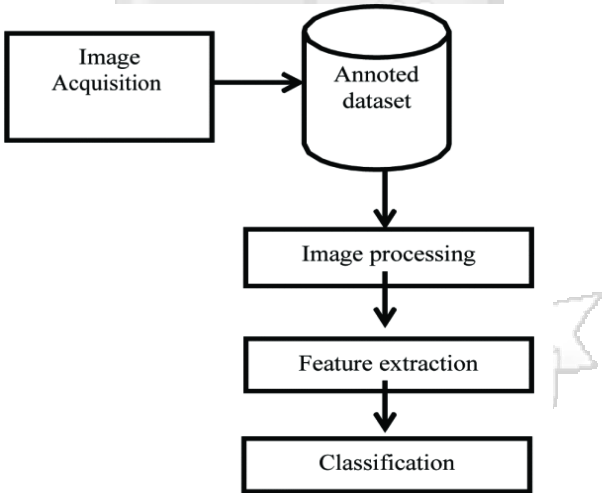


Figure 2. 5: Plant Disease Detection System

(Shruthi, Nagaveni & Raghavendra, 2019)

The description of each step is as follows:

Image Acquisition: This is the initial stage in the plant disease detection system. It involves the acquisition of images from an environment using digital cameras or drones for extensive applications.

Dataset Annotation/Labelling: Different diseases classes are labelled. This will result in a knowledge- based dataset from the collected images. The dataset will now contain images with disease classes and healthy ones.

Image Processing: Various preprocessing techniques are applied under this step to enhance the features to be used for plant disease classification. These include data augmentation, segmentation and standardization.

Feature Extraction: This is an important step in the plant disease classification process. The features to be used for disease classification are extracted at this stage. These features are based on color, shape and texture (Goel et al., 2017). The feature extraction method depends on the features to be extracted.

Classification: This is the final stage where the disease class is inferred based on the extracted features. At this stage, any computer vision model can be used to carry out the classification.

Computer vision techniques consist of traditional computer vision and deep learning techniques. Traditional-feature approaches have been used from time to time to improve performance in computer vision tasks. Features consist of small, "interesting", descriptive patches in a set of input images. The earlier developed feature-descriptors includes Scale Invariant Feature Transform (SIFT), Speeded Up Robust Features (SURF), Features from Accelerated Segment Test (FAST), Hough transforms and Geometric hashing. SIFT and SURF are traditional feature extraction methods which are usually used in combination with traditional machine learning algorithms such as K-Nearest Neighbors and Support Vector Machines to solve Computer Vision problems such as image recognition and classification.

Traditional computer vision algorithms have the following advantages: have a well-known basis or framework, transparent, and well-tuned for greater performance and powerful efficiency in processing tasks. However, they are limited to a lower accuracy and number of objects to be detected (Sean et al, 2019). Related work on strawberry fungal leaf disease classification was by Kusumandari et al (2019), who proposed a model for strawberry plant disease classification using on color segregation and based on the analysis of the HSV color space.

Digital images of strawberry plant leaves were processed and analyzed to determine if the leaves were healthy or diseased. A series of processes were carried out to achieve the detection; these included Image improvements, color segmentation process from RGB color space into HSV color space, regional segmentation. The results of segmentation were used to come up with the parameters that are crucial in determining the quality level of strawberries based on the strawberry leaves, whether the leaf is diseased or not. The proposed model obtained a detection accuracy of 85%. This study proposed

a model that took into consideration a single class of the diseases i.e., leaf Spot and hence not suitable for detecting the other two remaining classes: Leaf blight and Leaf scorch.

Kiani & Mamedov (2017) also proposed a method for the detection of fungal diseases in strawberry based on a fuzzy classifier. The optimized fuzzy parameters resulted an accuracy of 96% for segmented iron-deficiency and 93% for fungal infection. The results by this work presents a good advancement in disease detection. The model can however be fine-tuned to integrate the detection of several more disease classes. Most importantly, enhanced with features that will enable it detect more than a single disease class on the same leaf.

Some of the limitations of traditional computer vision were solved by the introduction of Deep Learning (DL). Deep Learning (DL) has great applications in the field of digital image processing where it is used to automate complex image analysis tasks such as image colorization, classification, segmentation, and detection. More complex tasks whose automation was thought to be unachievable are now being automated at a higher accuracy. A good example of this is image classification; Since being reignited by Krizhevsky, Sutskever & Hinton (2012), the domination of DL in this domain is attributed to its better performance as compared to the classical machine learning algorithms.

The development of Deep Learning came with its limitations; DL learns denser and denser, that is more abstract, representation of the training image as you proceed up the architecture hence requiring more computation power, training time and a very large dataset. Figure 2.6 illustrates the difference between the two techniques:

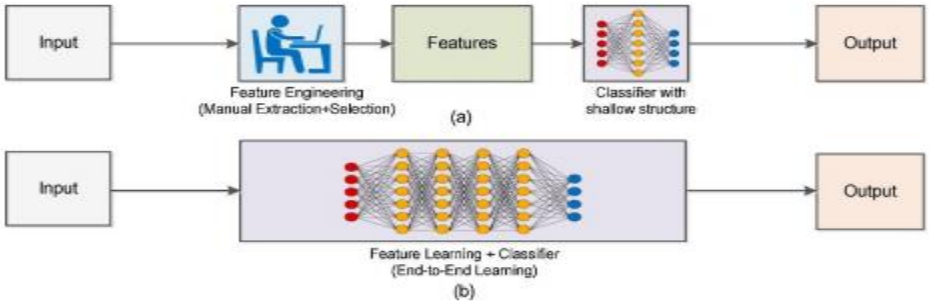


Figure 2. 6: (a) Traditional Computer Vision and (b) Deep Learning

Artificial neural networks (CNNs in particular) are an important component of deep learning that has led to a breakthrough in image recognition. With CNNs, the features are learned directly, and this eliminates the need for manual feature extraction (LeCun, Bengio & Hinton, 2015). ANNs consist of a set of interconnected processing elements known as neurons or nodes. They are inspired by the biological neural networks that constitute the human brain. Neurons are grouped into layers which are then interconnected to each other. The architecture of an Artificial Neural Network is dependent on its topological structure. The topological structure entails the transfer function for each node and the overall connectivity of the ANN.

The organization of a classical neural network constitutes all neurons in one layer being connected to the neurons in the next layer and having associated learnable weights between the neuron layers. The learning process in ANNs is achieved using training examples; the connection weights are iteratively adjusted until the trained ANN can perform some tasks. The iteratively changing weights constitutes an input to the activation function which gives an output only when a certain threshold is achieved. The diagrammatic representation of a classical neural network is as shown in figure 2.7.

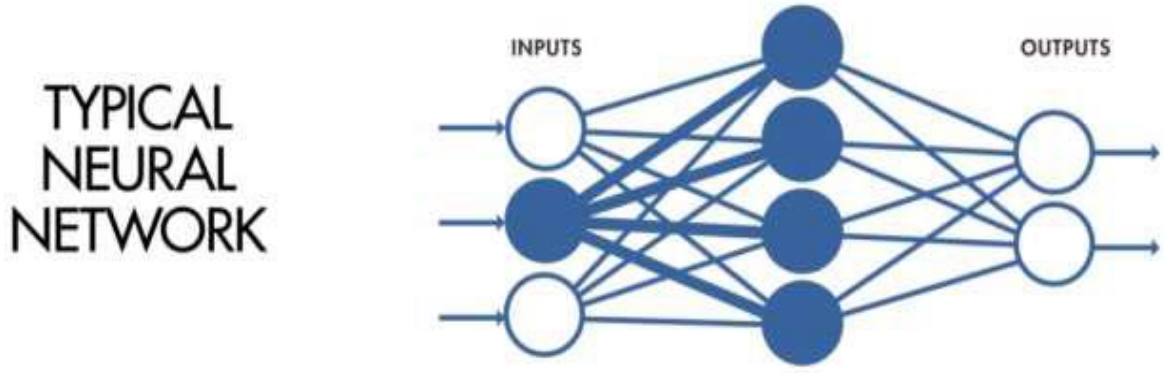


Figure 2. 7: The Structure of an Artificial Neural Network

(MathWorks, 2018)

The back-propagation process (BP) is usually used to train the model. This is a gradient descent-based optimization algorithm normally used in the iterative adjustment of connection weights in the ANN in order ensure the error is minimal. The predictions are usually obtained using the forward-pass process (Filip, Vasar & Prostean, 2018).

Classical neural networks have limitations when used in image processing; they easily experience over-fitting i.e.; the networks fail to generalize well from the training data to correctly classify unseen

data which was not part of the training test. They also have larger number of trainable parameters which requires more computing resources such as processors (IIango & Kumar, 2017). This is not suitable for deployment in constrained resources devices such as smartphones. CNNs are designed to reduce spectral variations in neural networks. They are multi layered ANNs usually suitable for multi-dimensional data; this has resulted to a wide range of applications to image and video processing. CNNs differ from the classical ANNs in the sense that in the CNN, only the last layer is connected while in the classical ANN all the neurons are interconnected (Arel, Rose & Karnowski, 2010). Another distinguishing feature of CNNs is the ability to use many identical copies of the same neuron. This usually enables the CNNs to express computationally large models while at the same time minimizing the number of parameters to be learned. CNNs constitute of an input, output layer and multiple hidden layers. The hidden layers consist of a series of convolutional layers. The structure of a CNN convolutional layers can be illustrated as in figure 2.8.

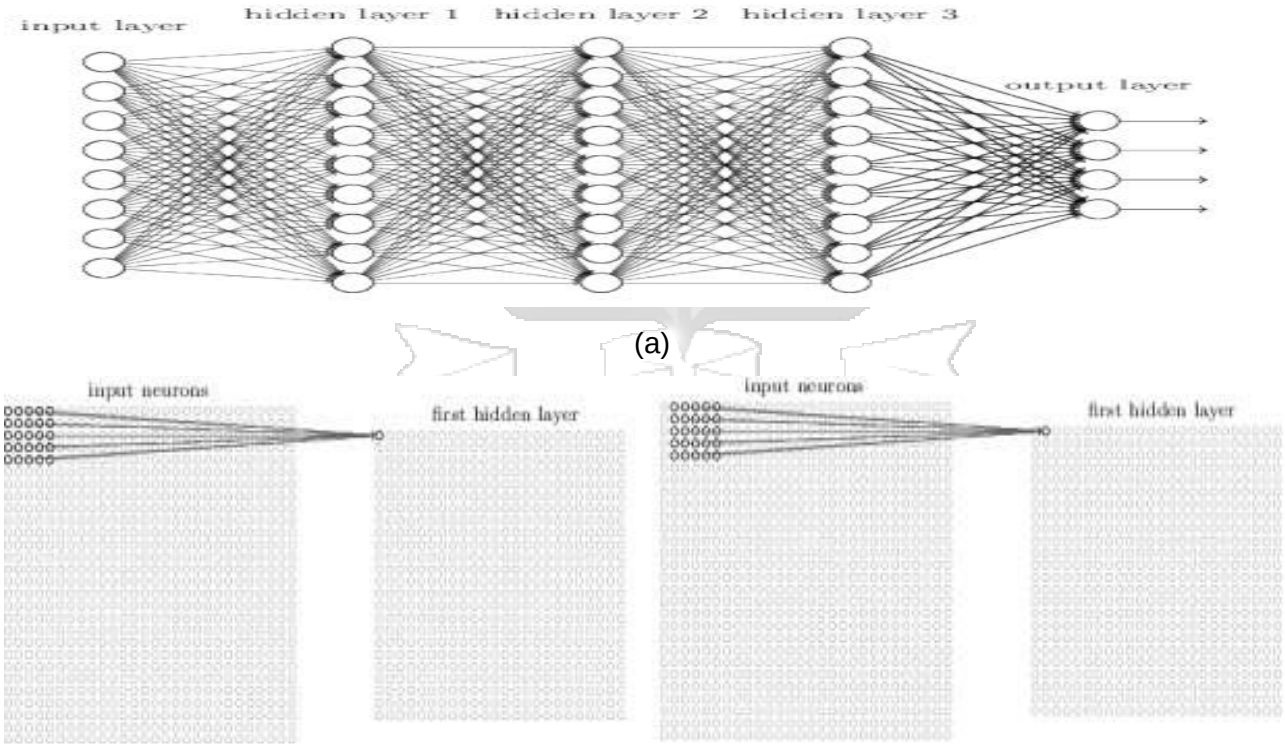


Figure 2. 8:(a) The structure of a CNN and (b) The CNN Convolutional Layers

(Nielsen, 2015)

CNNs have proved to be effective and efficient in image processing tasks. The major inhibition to this advancement has been the requirements of larger datasets and more time taken to train the models. Several solutions have been put in place to overcome this; for instance, transfer learning where a pre-trained model is used to perform classification on image datasets.

CNNs usually consist of set of different layers and these include convolutional, activation function, pooling, and the fully connected layers. The following is a description of each of the layers:

Convolutional Layer- This layer is responsible for feature extraction from the input set of images. The filters within this layer are responsible for this process. The input is usually thought of as a 28 × 28 square of neurons, whose values correspond to the 28 × 28 pixel intensities which are then used as inputs. The input pixels are usually connected to a layer of hidden neurons. In this arrangement, it is worthwhile to note that not every input neuron has a connection to every hidden layer neuron. The connections are made by regions of the input image which are small and localized in nature. More precisely, input neurons forming a small region will have a connection with each hidden layer neuron; for instance, if we consider a small region say 6 x 6, this will correspond to 36 input pixels. This constitutes the local receptive region for the hidden neuron. If we consider a 32 x 32 image, the computer detects a 3-D array of numbers, more precisely 32 x 32 x 3. The 3 represents the RGB values for each of the pixels. A series of steps are followed to achieve the convolution; if we consider a 5 x 5 x 3 filter, the filter takes the first position at the top left corner of the image and then slides, or convolves, across the input image. During this step, a multiplication of the values in the filter with the original pixel values of the image is performed. The results of the multiplication are summed up to create a single figure that denotes the filter's location. A movement of the filter either one stride or unit to the left will then follow and the process iterates several times. The iterations will be carried out until the sliding is done over the whole image; this will result to a 28 x 28 x 1 array of numbers known as the feature map. This implies that six 5 x 5 filters are used, then the result will be a 28 x 28 x 6 array of numbers (Nielsen, 2015). The output reduces in size after each convolution layer and becomes the input for the next one.

Activation Function- Rectified Linear Unit (ReLU) is an activation function that is commonly used. This is attributed to its ability to enable greater acceleration of the convergence of stochastic gradient descent much better than the sigmoid/tanh functions (Krizhevsky et al., 2012). ReLU is an operation

involves replacing all negative pixel values in the feature map by zero. The main aim of this function is to introduce non-linearity since most of the real-world data is non-linear.

Pooling Layer-This layer comes after every convolutional layer. The role of the pooling layers is to simply information in the output from the convolutional layer. The simplification process is achieved by the pooling layer taking each feature map output from the convolutional layer and preparing a condensed feature map. For instance, more precisely, each unit in the pooling layer may summarize a region of neurons in the previous layer. Max-pooling is one of common procedures used in the pooling layer. Max-pooling involves a pooling unit simply outputting the maximum activation in the 2 × 2 input region (Nielsen, 2015). For instance, considering 24 × 24 neurons output from the convolutional layer, after pooling, this will result to 12 × 12 neurons. This process reduces the dimensionality of each feature map while at the same time retaining important information.

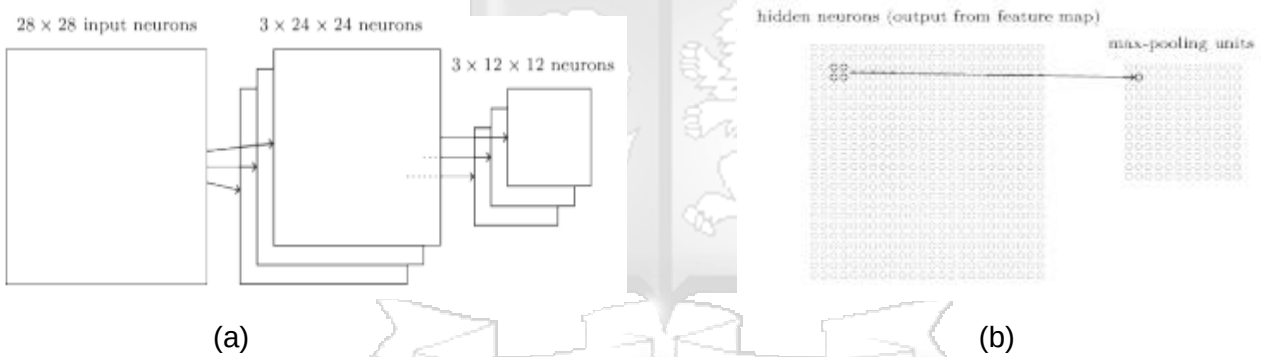


Figure 2.9:(a) Combined Convolutional and Max Pooling and (b) Max Pooling Process

(Nielsen, 2015)

Fully Connected Layer-This is the final layer in a convolutional neural network and follows several iterations of convolutional and pooling layers. This layer connects all the neurons in the previous max-pooled layer to form a fully connected output. This layer is responsible for the actual classification task.

2.4.2.1 CNN Architectures

CNNs have been proved suitable for most computer vision applications, especially image processing. They have achieved a high level of accuracy in image classification tasks; however high accuracy has come along with limitations. Most advancements in CNNs have been based on improving the accuracy

at a cost of speed and size, this has resulted to difficulties when deploying the models in resource-constrained platforms such as mobile phones. The deployed models can also not work well in applications that need real-time processing.

A lot of research has been done to address this problem; this has been achieved by coming up with different approaches that results to high accuracy at a relatively low latency neural network. One of developments in this approach is the MobileNet; this architecture makes use of depth-wise separable convolutions which drastically reduce computation required and network size (Howard et al. 2017). The differences between a standard convolution and depth-wise convolution are illustrated in fig 2.10.

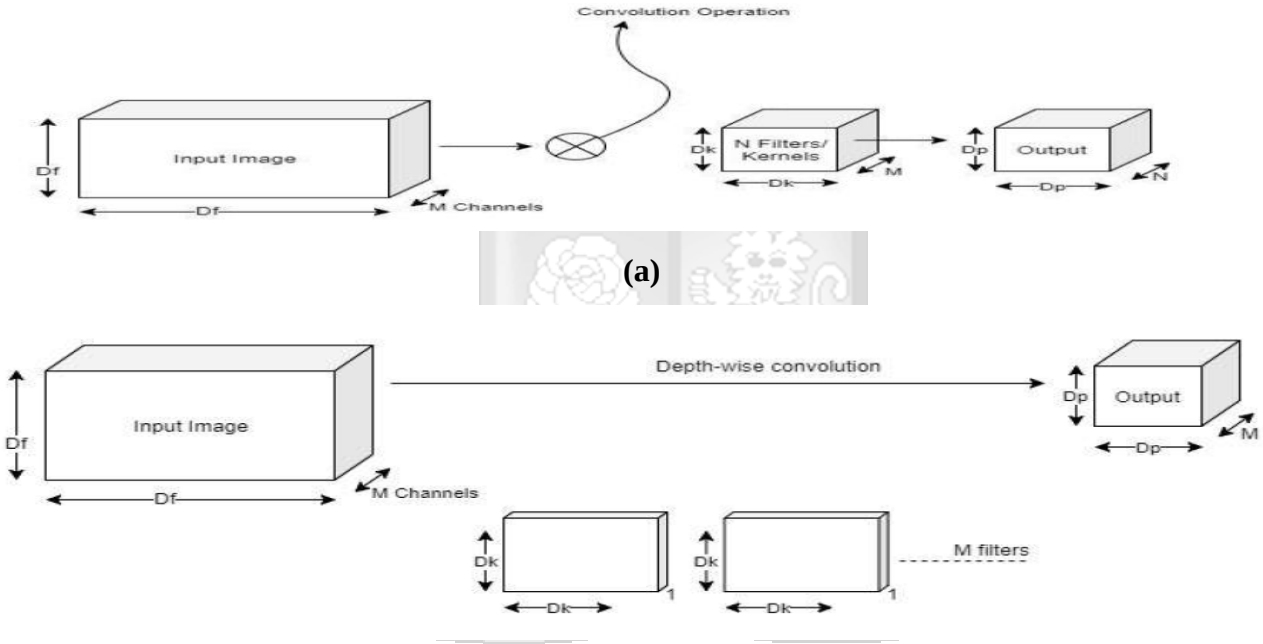


Figure 2.10: (a) Standard Convolution and (b) Depth-wise Convolution

(Howard et al, 2017)

To illustrate the difference in computational costs, consider an input image of Df x Df x M where Df x Df is the image size and M is the number of channels. If there is a total of N filters in all the convolutions, the output of a standard convolution will be Dp x Dp x N hence resulting to a computational cost of Dk x Dk x M x Dp x Dp x N. Depth wise convolutions usually consist of two layers:

Depth-wise convolution-In this layer, convolution is applied to a single layer and therefore the size of the filter will be Dk x Dk x 1. Since there are M input channels the size of the output will be Dp x

Dp x M. This therefore implies that depth-wise convolutions will have a computational cost of Dk x Dk x Dp x Dp x M.

Point-wise Convolution- In this layer, a 1 x 1 convolution is applied to all M channels; the aim of this operation is to combine and generate new features. The filter size will therefore be 1 x 1 x M and the output Dp x Dp x N. This is because initially there was a total of N filters in all the convolutions. Using these two, the computational cost will be M x Dp x Dp x N. The total computation cost for depth-wise convolution is obtained by adding the computational costs for the two layers; the total cost will therefore be (Dk x Dk x Dp x Dp x M) + (M x Dp x Dp x N). A comparison of the two costs i.e., for standard convolution and depth-wise convolution is done using a ratio R, where R=depth-wise separable convolution computation cost / standard convolutions. This will evaluate to 1/N + 1/Dk2 which implies a higher computational cost when using standard convolutions. The MobileNet architecture uses 3 x 3 depth-wise separable convolutions. This architecture saves eight to nine times computational cost used by a standard convolution. MobileNet version 2 makes use of inverted residuals and linear bottlenecks to improve accuracy and performance. When compared with MobileNet version 1, version two is faster for the same accuracy and need about 30% fewer parameters. Therefore, MobileNet version 2 is an effective and efficient feature extractor for computer vision tasks (Howard et al., 2017). The other important component of interest is that of generalization on unseen data. Batch Normalization is another technique that has been proposed in improving the performance of deep neural networks (Ioffe & Szegedy, 2015). This is a technique that involves normalizing activations in intermediate layers of deep neural networks. This technique has resulted to the tendency of improving accuracy and speeding up the training process. According to a study by Bjorck et al (2018), BN primarily enables training with larger learning rates, which enhances faster convergence and better generalization. The better generalization accuracy of the deep neural networks is contributed by the higher learning rates which are enabled by BN. The overfitting effect can either be removed or reduced in strength with batch normalized networks (Ioffe & Szegedy, 2015). To implement batch normalization in a neural network, a set of activations is specified followed by a BN transform for each of them. This is illustrated as in Algorithm 1 below:

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β
Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma\widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

Algorithm 1: Batch Normalizing Transform, applied to activation x over a mini batch (Ioffe & Szegedy, 2015).

With regard to the above algorithm, a layer will now receive BN(x), a normalized transformation from the previous input x. A Batch Normalized model training can be achieved by many means. This includes Stochastic Gradient Descent and/or batch gradient descent. The mini-batch size can be set to m > 1. The other variants such as Adagrad can also be used (Ioffe & Szegedy, 2015). Algorithm 2 illustrates the training process of the normalized network.

Input: Network N with trainable parameters Θ;
subset of activations $\{x^{(k)}\}_{k=1}^{K}$
Output: Batch-normalized network for inference, $N_{\mathrm{BN}}^{\mathrm{inf}}$

1: $N_{\mathrm{BN}}^{\mathrm{tr}} \leftarrow N$ // Training BN network
2: **for** $k = 1 \ldots K$ **do**
3: Add transformation $y^{(k)} = \mathrm{BN}_{\gamma^{(k)},\beta^{(k)}}(x^{(k)})$ to $N_{\mathrm{BN}}^{\mathrm{tr}}$ (Alg. 1)
4: Modify each layer in $N_{\mathrm{BN}}^{\mathrm{tr}}$ with input $x^{(k)}$ to take $y^{(k)}$ instead
5: **end for**
6: Train $N_{\mathrm{BN}}^{\mathrm{tr}}$ to optimize the parameters $\Theta \cup \{\gamma^{(k)}, \beta^{(k)}\}_{k=1}^{K}$
7: $N_{\mathrm{BN}}^{\mathrm{inf}} \leftarrow N_{\mathrm{BN}}^{\mathrm{tr}}$ // Inference BN network with frozen // parameters
8: **for** $k = 1 \ldots K$ **do**
9: // For clarity, $x \equiv x^{(k)}, \gamma \equiv \gamma^{(k)}, \mu_{\mathcal{B}} \equiv \mu_{\mathcal{B}}^{(k)}$, etc.
10: Process multiple training mini-batches $\mathcal{B}$, each of size m, and average over them:

$$\mathrm{E}[x] \leftarrow \mathrm{E}_{\mathcal{B}}[\mu_{\mathcal{B}}]$$
$$\mathrm{Var}[x] \leftarrow \frac{m}{m-1}\mathrm{E}_{\mathcal{B}}[\sigma^2_{\mathcal{B}}]$$

11: In $N_{\mathrm{BN}}^{\mathrm{inf}}$, replace the transform $y = \mathrm{BN}_{\gamma,\beta}(x)$ with $y = \frac{\gamma}{\sqrt{\mathrm{Var}[x]+\epsilon}} \cdot x + \left(\beta - \frac{\gamma\,\mathrm{E}[x]}{\sqrt{\mathrm{Var}[x]+\epsilon}}\right)$
12: **end for**

Algorithm 2: Training a Batch-Normalized Network (Ioffe & Szegedy, 2015)

In CNNs, it is possible to apply batch normalization to any set of activations. One of the approaches proposed by Ioffe & Szegedy (2015) is to focus on a set transform. The set should affine transformations which precedes an element-wise nonlinearity: z = g (Wu + b). W and b represent

learned parameters of the model, g (·) is the nonlinearity. Examples of this can be sigmoid or ReLU. This is implemented both layers i.e. the convolutional and fully connected layers. This is followed by an addition of the BN transform just before the non-linearity. This is achieved by a normalization of x = Wu+b. The effect of the bias b is canceled by the subsequent mean subtraction hence z = g (Wu + b) is substituted with z = g(BN(Wu)). The bias can therefore be safely ignored.

In the convolution layers, the aim is to ensure the normalization obey the convolutional property; this will enable the varying elements in one feature map, at varying locations, to have the same normalization. In order to achieve this aim, all the activations are jointly normalized over all locations in a minibatch. Several modifications are therefore proposed in both algorithms to achieve this: in algorithm 1, the set of all values in a feature map in both mini-batch and spatial locations are represented by β; this implies that a mini-batch of size $m' = |\beta| = m \cdot p\, q$ is applied for feature maps of size p × q and a mini-batch of size m. The parameter pairs $\gamma^{(k)}$ and $\beta^{(k)}$ for every feature map, rather than per activation is learned. Algorithm 2 will also be modified in the same way; this will enable Each activation in a given feature map has the same corresponding linear transformation applied by BN.

2.4.2.2 Machine Learning Frameworks

Machine learning frameworks are a collection of packages and libraries which are important in machine learning; they provide a platform for training and deploying machine learning and deep learning models. Several machine learning frameworks have been developed; the commonly used are TensorFlow and Keras machine learning frameworks. TensorFlow is an end-to-end open-source platform for machine learning developed by researchers and engineers at Google. It is a collection of comprehensive and flexible ecosystem of tools, libraries and other resources that provide high-level APIs workflows. This framework is characterized by the following salient features: Easy Model Building, Robust ML Production Anywhere and Powerful Experimentation for Research. (TensorFlow, 2018).

The Kera's framework is a neural networks library which uses CNTK, Theano, TensorFlow frameworks as a platform and runs on top of them. The use of Keras framework in deep learning offers several advantages; prototyping is easy and faster, it can also be easily deployed and run-on CPU and

GPU.The other advantages of Kera framework include User-Friendly, Modular and Composable, Easy to Extend and Easy to Use (Geron, 2019). Keras framework is specifically designed with capabilities to support deep learning, which makes it suitable for image classification tasks.

Several related works on strawberry fungal leaf disease classification based on deep learning models (CNNs) exist. Brahimi et al (2020) evaluated the performance of multiple state of art CNN architectures using three types of learning strategies (shallow, deep and from scratch). The PlantVillage datasets was used in this evaluation. The dataset consisted of 54323 images. This consisted of 14 classes of crops species. The crop diseases were organized into 38 classes with healthy and diseased leaves. The classifier was trained to differentiate between the plant leaves and the Environment using a background class consisting of 715 images. The background class was created using a set of color images from Stanford public dataset. The integrated dataset including the background class consists of 55038 containing 39 classes. The following state of the art architectures were used: AlexNet, DenseNet-169, Inception v3, ResNet-34, SqueezeNet and VGG13.The accuracy achieved by training from scratch using 80– 20% train-test distribution, includes AlexNet which resulted to an accuracy of 97.82%, GoogleNet with an accuracy of 98.36%. On the other hand, training using transfer learning yielded the following results: AlexNet with an accuracy of 99.24% and GoogleNet with an accuracy of 99.34%. The best accuracy i.e., of 99.72 %, was achieved by Inception v3 using transfer learning. Among the classes of diseases covered, only a single class of strawberry plant fungal leaf diseases was covered i.e., leaf scorch. The model developed did not cover the other classes of strawberry leaf diseases (leaf blight and leaf spot) hence unable to distinguish between the three classes of these diseases.

Another study by Chohan, Khan & Katper (2020) proposed a deep learning-based model named plant disease detector. The model was designed with the ability of detecting plant diseases using leaf images as input. The disease detection model was developed using a CNN. Data augmentation was used to inflate the dataset which is the fed into a CNN with multiple convolution and pooling layers. Two datasets were used to perform plant disease detection. The first dataset consisted of 15 classes of plant species and 38 disease classes. The total number of images in the first dataset were 2952. The final findings of this work were based on PlantVillage dataset which contains 38 classes of different plants. The Plant Village dataset was used to train the model and 15% of data from PlantVillage data was

used for testing. The Proposed model achieved 98.3% testing accuracy. The model achieved an accuracy of 95% when tested on 100 actual environment images. In this study, out of the 38 classes of the plant diseases covered, a single class of strawberry plant leaf disease was covered (leaf scorch). The model is also therefore not suitable for detecting the other two remaining classes which are Leaf Blight and Leaf Spot.

Mohanty, Hughes & Salath´e (2016) also proposed a deep learning model for plant disease classification in which 54,306 images of plant leaves, containing 38 class labels were used. Each class label consisted a crop-disease pair which was to predict the crop-disease pair using image of the plant leaf as input. A comparison was done was done in developing the model in both AlexNet and GoogLeNet architectures. The resultant accuracy ranged from 85.53% to 99.34% with training the AlexNet architecture from scratch using grayscale images in the ratio 80:20 and transfer learning of AlexNet architecture using color images in the ration 80:20, respectively. The model developed in this study failed to generalize well on unseen data. This makes the model unsuitable for use in real-life conditions where the input images are from varying backgrounds. The model is also not suitable for detecting the three classes of strawberry plant fungal leaf diseases as it only featured leaf scorch, a single class of strawberry plant fungal leaf diseases.

In another study by Ferentinos (2018), a deep learning model was developed to automate plant disease detection. An open database of 87,848 images, with 58 distinct classes of [plant, disease] combinations was used to train the model. This dataset consisted of healthy and diseased plant leaves. The training was performed on various architectures. The best performance for the plant disease detection was 99.53%. The model proposed by this study also experienced overfitting effects hence failing to generalize well on unseen data. It also considered a single of strawberry plant leaf diseases (leaf Scorch) hence not a suitable detection tool for farmers facing challenges form the other classes of strawberry plant leaf diseases such as Leaf Blight and Leaf Scorch.

Fang et al (2020) carried out a study in which a leaf disease grade identification method based on a convolutional neural network (CNN) was proposed. 18,347 leaf images selected from the PlantVillage were used as experimental data, including 10 classes of leaf diseases from 8 crop species. First, an adaptive adjustment algorithm based on a two-dimensional (2-D) gamma function was used to process a nonuniform illumination images. Then, binary images were obtained by segmentation of diseased

leaf images using a threshold segmentation method. The ratio of the pixel number in the lesion area to that in the diseased leaf area was calculated; this ratio was regarded as the classification threshold of the disease grades. It was therefore used to determine the disease grade category. A ResNet50-based CNN was additionally proposed to identify disease grades and yielded an accuracy of 95.61%. This study also considered strawberry leaf scorch, a single class of strawberry plant fungal leaf diseases among the 10 classes plant diseases featured. The other classes i.e., Leaf Blight and Leaf Spot were not considered hence rendering the model unsuitable for detecting the remaining two classes of strawberry leaf diseases. Table 2.1 shows a summary of the related works.

Table 2. 1:Summary of Related Works

Author	Technique Used	Classes of Strawberry Leaf Disease Covered	Result	Weaknesses
Xiao et al(2021)	CNN based model	Leaf Blight	Accuracy of 99.60%	Leaf Scorch and Leaf Spot not covered(Xiao et al, 2021)
Brahimi et al (2020)	CNN based model	Leaf Scorch	Accuracy of 99.72 %	Covered a single class of strawberry fungal leaf disease
Chohan, Khan & Katper (2020)	CNN based model	Leaf Scorch	Accuracy of 98.3%	Covered a single class of strawberry leaf disease (Chohan, Khan & Katper, 2020).
Fang et al (2020)	CNN based model	Leaf Scorch	Accuracy of 95.61%	Covered a single class of strawberry leaf disease (Fang et al, 2020)
Kusumandari et al (2019)	Color Segmentation	Accuracy of 85%	Leaf Spot	Covered a single class of strawberry leaf disease. Accuracy level is low as compared to that of CNNs in plant disease detection (Kusumandari et al, 2019).
Ferentinos (2018)	CNN based model	Accuracy of 99.53%	Leaf Scorch	Covered a single class of strawberry leaf disease. Failure to generalize on unseen data (Ferentinos, 2018).
Kiani & Mamedov (2017)	Fuzzy Classifier	Accuracy of 96% for iron deficiency and 93 % fungus infection.	Not Specified	Implemented a binary classification of whether the fungal disease is present or not without specifying the disease class.
Mohanty, Hughes & Salath´e (2016)	CNN based Model	Accuracy 99.34 % (Transfer learning) and 85.53 % (learning from scratch)	Leaf Scorch	Covered a single class of strawberry leaf disease. Failure to generalize on unseen data (Mohanty, Hughes & Salath´e, 2016)

2.5 Research Gap

From the review of the strawberry fungal leaf diseases classification methods provided in section 2.4, there are a number of gaps: the real-time (TaqMan) PCR assay for detecting strawberry requires a lot of time to do the laboratory analysis (Weller et al, 2000). This method is also not suitable for farmers who are far away from the laboratory facilities as leaf samples need to be collected; this will result to wastage of time and sample collection expenses. The method is also only limited to a single class of fungal strawberry leaf disease (strawberry leaf spot). The other clinical methods i.e. laboratory test approaches on plant tissue samples such as enzyme-linked immunosorbent assay (ELISA), and loop-mediated isothermal amplification (LAMP) also require time to do the laboratory analysis and are not easily accessible in rural plantations. The test samples also require special facilities such as reagents and controlled conditions in order to perform the analysis.

The detection of strawberry plant disease based on Leaf Spot using color segmentation by Kusumandari et al (2019) covered only a single class of strawberry fungal leaf diseases. The Model cannot not generalize well in a situation where there are several classes of the diseases. The accuracy given by the proposed methodology is also low as compared to the performance of CNNS in image classification.

The methodology proposed by Kiani &Mamedov (2017), for detecting iron deficiency and fungal infection in strawberry based on a fuzzy classifier did not specify the fungal disease classes. The implementation was a detection of whether there was the presence of a fungal infection or not. The method is also not therefore suitable for classification of various classes of strawberry fungal leaf diseases.

CNNs were also proposed for plant leaf disease detection (Mohanty, Hughes, & Salath´e, 2016; Ferentinos, 2018; Brahimi et al, 2020; Chohan, Khan & Katper, 2020; Fang et al, 2020, Xiao et al, 2021). These models covered two classes of strawberry fungal leaf diseases i.e. Leaf Scorch and Leaf Blight. Strawberry leaf spot disease was not covered by any of the models. The models cannot therefore be readily used by strawberry farmers for classifying all the three classes of the disease.

Even though majority of the deep learning models achieved extremely high accuracy, a number of limitations were also observed and recommended for future work; for instance, the accuracy of the

developed model is reduced to 31% when the model is tested on previously unseen set of data (Mohanty, Hughes & Salath´e, 2016). In another similar study by Ferentinos (2018), it is still clear that the deep learning models can't be used in real conditions due to the overfitting effect. Preliminary experiments of testing the deep learning model with data aside from the one used for training indicated a reduction in the model's performance. This was to a range of about 25-35%. This is not the case in real life as disease detection will involve taking images directly from the field which are not part of the training dataset. What matters really is not the classification accuracy of the model only, but also its ability to generalize to new cases that were not availed during training and the classes of diseases covered. The substantial reduction in the accuracy is a limitation that inhibits the models to be used as a generic tool in leaf disease detection. Another issue of equal importance is the ability of the models to deal with situations where there is more than one class of the fungal disease occurring on the same leaf. None of the reviewed techniques considered this.

To address the aforementioned weaknesses in the existing studies, this study proposed a model for the classification of three classes of strawberry plant leaf diseases based on a normalized CNN and also included an instance where two classes of the diseases occur on the same leaf of the plant.

2.6 Conceptual Framework

The conceptual framework interconnects the reviewed literature with the research problems and the research objectives. The proposed model was trained in order to distinguish three classes of strawberry plant fungal leaf diseases which are Leaf Spot, Leaf Blight and Leaf Scorch. An instance where more than one class of the diseases occur on the same leaf was also considered. Initially before the training phase, the input images were processed in order to prepare them for training. This involved carrying out data augmentation, dividing the images into training and testing sets and labelling the classes. The first part of the proposed disease detection system was training the depth wise CNN from scratch with strawberry leaf images both healthy and diseased. Next, the trained model was tested using the validation dataset. The proposed disease detection model should be able to determine whether or not a strawberry leaf image is diseased. If the leaf image is diseased, then the model should perform recognition of the leaf disease type based on multiple learnt features of leaf diseases from the training dataset. The conceptual framework is shown in Figure 2.11.

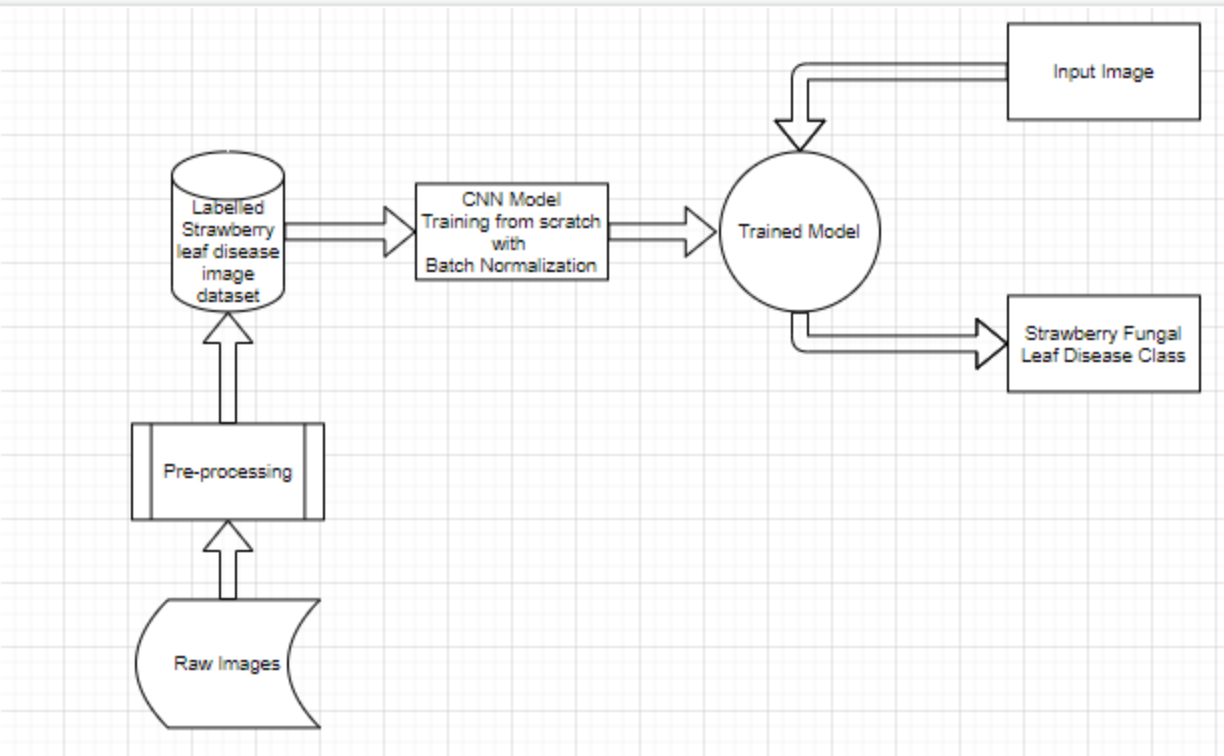


Figure 2.11: The Conceptual Framework

Chapter 3: Research Methodology

3.1 Introduction

The aim of this study was to develop a deep normalized neural network model for detection of strawberry fungal leaf diseases. This chapter describes the research design, data collection procedures and the methodology used in the development of the deep normalized neural network model. This chapter also explored the nature of the research with regard to the research quality and the ethical considerations for the study.

3.2 Research Design

The study adopted the use of the experimental research design. This design involves having a set of variables that are of importance to the experiment be varied while checking the outcome. This has to begin by formulating a hypothesis that will guide the experiment(Dennis &Valacich, 2001). The hypothesis for the study was: Which model training parameters will result to the best model performance in terms of accuracy and the training time? The variables varied constitute the independent variables while the outcome is the dependent variable. This design was suitable for the study as various variables of interest such as the learning rate and the number of training epochs(independent variables) were varied until an optimal level of the model's accuracy was obtained. As a result of this, experimental data was generated from these variables. This included the model's accuracy and the training time. This was of help in determining the optimal model parameters that were to be used in developing a model with the best performance.

3.3 Location of the Study

The study was undertaken at Strathmore university. The location was suitable for the study as the institution provided a conducive environment for the development of the model. There are also the necessary resources for conducting the study and running of the simulations.

3.4 Population and Sampling

3.4.1 Population

The target population for the study was Strawberry leaf images. This constituted both healthy and diseased leaves with appropriate classes of the fungal leaf diseases. The study considered both primary

and secondary data. The secondary data image dataset was obtained from the PlantVillage dataset available on Kaggle and other google websites. The dataset was fit for study as it consisted of images take in a controlled environment and rich in features used for the fungal disease detection. Primary data consisted of 75 images(of leaves infected by both leaf blight and leaf Scorch) obtained from strawberry farms in Kinangop, Nyandarua county in Central Kenya. Primary data was collected using a Sony RX1R II Professional Compact Camera with a 35mm Sensor and 42.4 MP power. This constituted quality images with a dimension of 254x254. The whole dataset constituted a total of 1,134 images. Table 3.1 in section 3.5 shows the distribution of the number of images per disease class for the whole dataset.

3.4.2 Sampling

The study focused on the use of probability sampling to prevent biasing of the sample in order to suit the needs of the study. The study adopted the use of simple random sampling. According to Kothari (2004), this form of sampling agrees with the law of statistical regularity. This implies that if a simple random sample is chosen, it basically carries the same characteristics as the population. With regard to this, 80% of the images collected were used for training, 20% for validation.

3.5 Data Collection

Secondary was obtained from the NewPlantVillage dataset available on Kaggle and other google websites(Ohio online plant disease website, Wisconsin Horticulture, Umass center for Agriculture and Food, Purdue university Vegetable Crops hotline, University of Minnesota extension among others). The images were downloaded using a python script that was based on the google image download modules using specified key words. The data obtained consisted of quality images (dimension 256 x 256) taken in a controlled environment. This constituted a total of 1,059 images. Primary data consisted of images captured from strawberry farms in Kinangop. The images were taken using a Sony RX1R II Professional Compact Camera with a 35mm Sensor and 42.4 MP power. This constituted a total of 75 images. The images were rich in features such as color and texture which were needed for the model training. A sample python script for downloading images for strawberry leaf spot class is as in figure 3.1

```
In [ ]: #Downloading the Leaf Spot Class Images
        import google_image_download
        output_folder="D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Sorted_Research_ Dataset\Strawberry_Leaf_Spot"
        target_folder="C:\\Users\\user\\Documents\\Soft\\chromedriver\\chromedriver.exe" -l 500”
        googleimagesdownload -k "Strawberry Leaf Spot" -o output_folder -t photo -n -nn -cd target_folder
```

Figure 3. 1: A sample python script for downloading google images

In the script in figure 3.1, the google_image_download library was imported to enable the automated downloads. The output folder indicates the path to the folder in which the images were stored in the local disk while the target folder indicates the path to the chrome plug in that was used in downloading the images. The googleimagesdownload-k command takes the search keyword as input that was used to filter the images download. Table 3.1 shows a summary of the dataset used.

Table 3. 1:Summary of the Image Dataset

Disease Class/Label	Number of Images
Strawberry Healthy	450
Strawberry Leaf Scorch	450
Strawberry Leaf Blight	74
Strawberry Leaf Spot	85
Strawberry Leaf Scorch and Leaf Blight	75
Total	1,134

3.6 Data Analysis Methods

This constitutes the process of extracting useful/meaningful information from given data. This process is important in giving a brief analysis of the data structure which helps in the modelling process. It also answers preliminary questions regarding the features of interest in the modelling process. The deep learning models are able to automatically learn/extract the features for image classification during training as opposed to manual feature extraction techniques. The data (images) collected were analyzed using python libraries for features to be used for strawberry fungal leaf disease classification. The libraries (Numpy, Pandas, Skimage.io and Matplotlib) were used in the analysis of the images for color and edge features to determine the difference between the healthy and diseased leaves in terms of these features. The libraries are powerful for image analysis with clear output that can be conceptualized with human eye. The various features used for disease detection include color, texture, and the edges. The edge features represent areas on an image with a pixel transform. The edge features are vital as effects on images are easily detected by analyzing these features(Khadidos & Sachez, 2017). Color and texture also provide a basis for analyzing images for image classification. These

features do not have a larger impact as the analysis does not give a clearer distinction of the images on this basis. The skimage's function called sobel was used in the analysis of the edge features of the leaves. Figure 3.2 shows sample leaf images for strawberry healthy and leaf scorch classes together with their grey scale fashion of the images and their corresponding edge features as a result of the analysis. Figure 3.3 on the other hand shows a sample python script used in implementing the analysis for strawberry leaf scorch class.

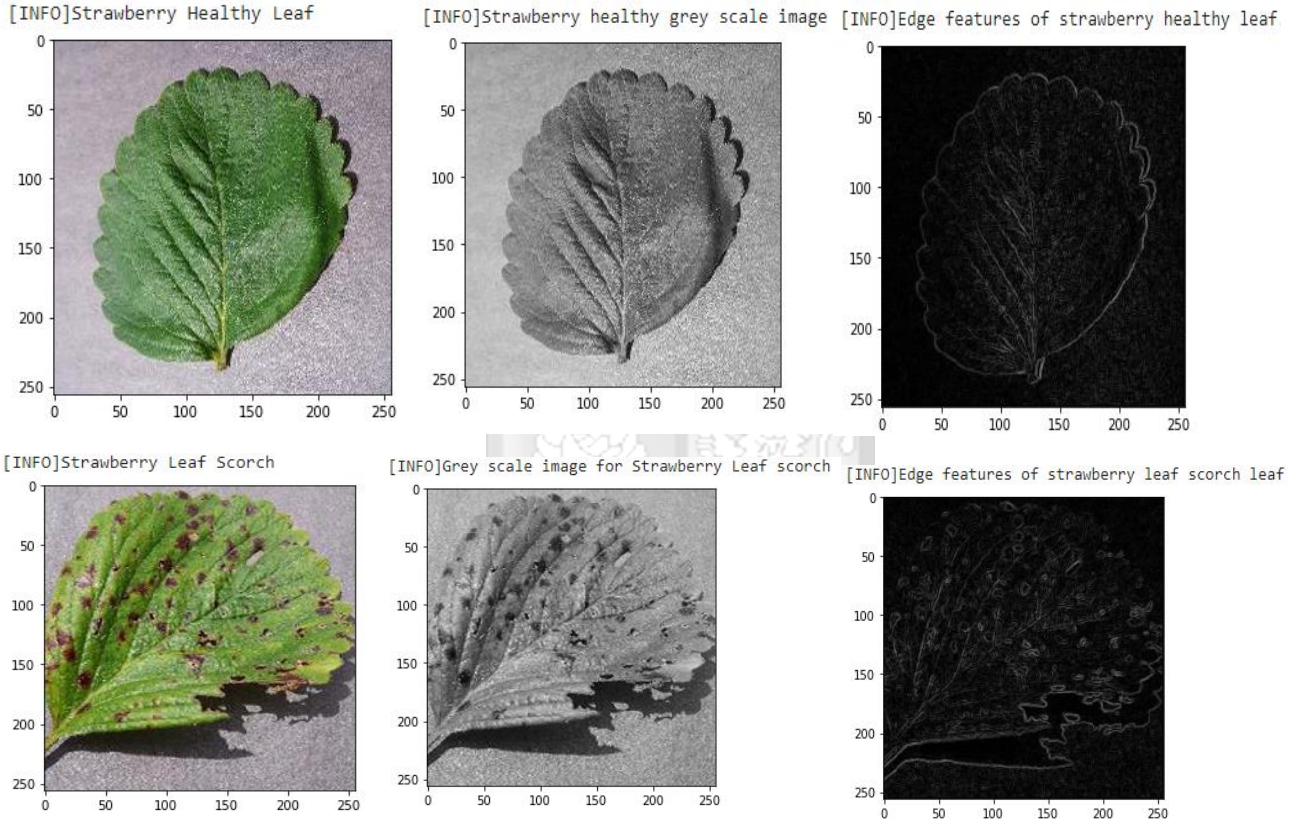


Figure 3:1: Leaf images for strawberry healthy and leaf scorch with the corresponding edge features

```
[15] #Edge features for Strawberry leaf Scorch leaf
     from skimage import filters#Neede library for generating filters
     ed_sobel = filters.sobel(image4)#passing the image instance to the filter function
     imshow(ed_sobel, cmap='gray');#Dispalying the image with filters
     print("[INFO]Edge features of strawberry leaf scorch leaf")
```

Figure 3. 2: Figure 3:2: Python script for analyzing and displaying the edge features of strawberry leaf scorch leaf sample

From the above diagrams, it is clear that the edge features for the healthy and diseased leaves are not the same. The healthy leaf is characterized by smooth edges while the diseased leaf has spotty and rugged edge features. It is not easy to classify the images with naked eyes based on these features. CNNs are able to automatically extract, analyze and learn the features to be used for classification at greater details.

3.7 Deep Normalized Neural Network Model Development

The deep normalized neural model was developed as per the sequence of the following steps:

3.7.1 Image Pre-processing

Image preprocessing was conducted in three steps namely, Data Augmentation, Standardization and data structure organization. The three processes are described below:

3.7.1.1 Data Augmentation

The aim of data augmentation is to artificially inflate the training dataset. The image augmentation techniques used in this study include flipping, rotation, cropping, zoom, shear (Taylor and Nitschke, 2017). This process increased the number of training samples from which assists the model cope with all distortions (such as variations in color, position and lighting intensity) that occur in real-life conditions. The implementation of data augmentation has been discussed in more details in section 5.2.3

3.7.1.2 Standardization

This process involved transforming all the images to the same size of 256 X 256. This was important in minimizing variations within the dataset and to enable the images to fit the input shape of the model. The implementation details are as discussed in section 5.2.3.

3.7.1.3 Dataset Structure Organization

80% of the image dataset was used for training and the remaining 20 % for validation. The images collected were organized into respective folders and subfolders corresponding to the strawberry leaf disease class name. The label for each image was defined by the respective subfolder names. The dataset splitting into training and testing implementation details are as discusses in section 5.2.5. The different class labels in the dataset used are as illustrated in figure 3.3.

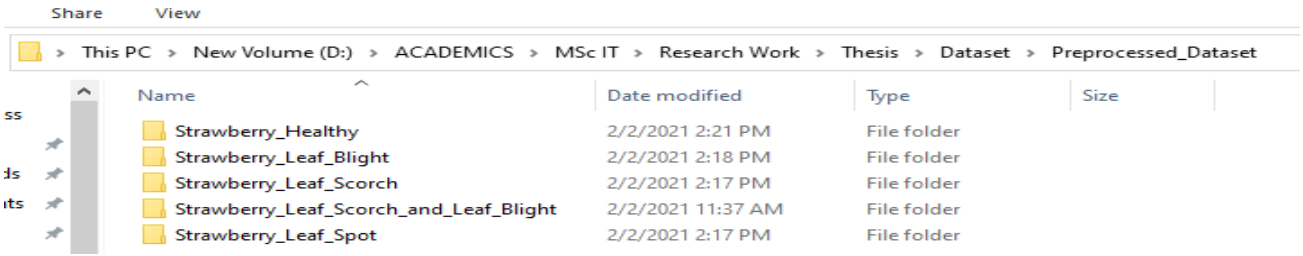


Figure 3. 3: Dataset folder organization

3.7.2 Model Development and Training

The CNN model was developed and trained form scratch using the Keras deep learning framework. Keras framework is specifically designed with capabilities to support deep learning, which makes it suitable for image classification tasks. The other advantages of Kera framework include User-Friendly, Modular and Composable, Easy to Extend and Easy to Use (Geron, 2019). Batch Normalization was implemented in the CNN model convolutions in order to minimize overfitting and improving the model's generalizability. The model development, training and deployment was carried out in the Google Colab platform and the implementation in Python programming language. The Google Colab platform was suitable for the model development as it allows free access to GPU which is provides the required computing capabilities for training the deep learning model. The model.fit_generator command was used in training the model. An object of the ImageDataGenerator(earlier initialized as aug) class was passed as input during model training. The python script used in training the model is as shown in figure 3.4.

```
#Training the model
print("[INFO] training the deep normalized CNN Model...")
history = model.fit_generator(
    aug.flow(x_train, y_train, batch_size=BS),
    validation_data =(x_test, y_test),
    steps_per_epoch=len(x_train) // BS,
    epochs=EPOCHS, callbacks=[tensorboard_callback],verbose=1
    )
```

Figure 3. 4: Python script for Training the model

3.7.3 Model Testing

Functional testing was performed on the model in order to test the model based on the basic functionality parameters. Sample test cases that were used to test the model in this regard as shown is section 5.3.2.

3.7.4 Model Validation

The validation of the model was carried out the 20% validation dataset. This consisted of a total of 226 images. A confusion matrix was used in the implementation phase from which several metrics were derived to evaluate the model. This is as indicated in section 6.2.1.

3.8 System Development Methodology

A prototyping-based methodology was used in this study to develop the CNN model. In particular, evolutionary prototyping was used to guide the different phases of the model's development. In prototyping methodology, a prototype is usually built and continuously refined based on the feedback provided by the stakeholders. Prototyping was necessary in developing the CNN since some parameters such as the number of training steps were experimental hence the need to improve on them in iterations. This methodology allows features to be added or removed progressively and there is real time development via experimentation. The prototyping methodology used in the study is shown in figure 3.5.

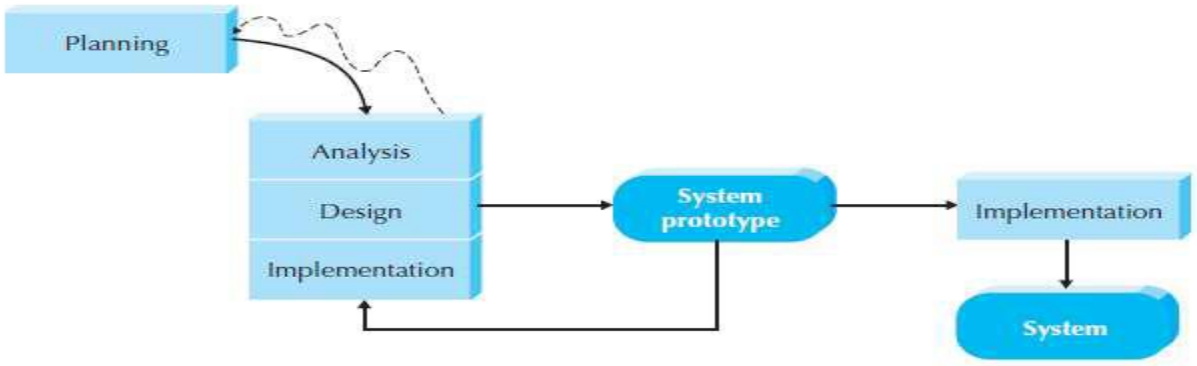


Figure 3. 5:Prototyping-based Methodology

(Dennis, Wixom& Tegarden, 2009)

According to the diagram in figure 3.5, the analysis, design and the implementation phases were carried out concurrently and iterated until the model was complete. This methodology quickly produces a model that is interactive and there is visible progress throughout the study (Dennis, Wixom & Tegarden, 2009).

3.8.1 System Analysis

3.8.1.1 System requirements Specification

The requirements for the fungal disease detection model were developed using the content analysis methodology. Content analysis is a technique used in research for making replicable and valid replicable and valid responses by interpreting and coding textual material(Stemler, 2015). A questionnaire was used to elicit these requirements. A sample questionnaire used is as shown in

appendix E. The functional and non-functional requirements were documented after the analysis phase.

3.8.1.2 System Hardware Analysis

A lot of computational power is required to train and deploy deep models; in most cases a Graphical Processing Unit (GPU) will be required. The model was developed on the Google Colab platform which provides free access to the GPU and other cloud resources such as storage. The specifications of this platform are as indicated in table 5.1 in section 5.2.

3.8.1.3 System Software Analysis

Keras machine learning framework was used to train and deploy the model; this framework is specifically designed for deep neural networks. It also supports seamless running on CPU and GPU. The other advantages of Kera framework include easier to use, Modular composability, Flexible for extension (Geron, 2019). The Google Colab platform also provided access to pre-installed libraries hence eliminating the need to install them before usage.

3.8.2 System Design

In the system design phase, object-oriented programming approach was used. This approach allows one to break the problem at hand into small portions that can be solved at a time. This enhances the general productivity of the system development process(Bracha & Ungar, 2004). The output of this phase was the use case diagram, sequence diagram, system architecture diagram and the database schema diagram. Use case diagrams were used to document the requirements of the system including internal and external influences.

3.8.3 System Implementation

The strawberry fungal leaf disease classification model was built in this phase. The development environment was on the Google Colab, an interactive online environment(provided by google) that allows execution of python scripts. The implementation procedure followed the following series of steps:

i. Install python, check for availability and install the related machine Learning Libraries.

ii. Collect and organize the dataset of images as illustrated in section 3.3.2.3
iii. Develop the model in Keras framework on the Google Colab platform.
iv. Perform hyper parameter tuning.
v. Test and validate the new model.

3.9 Research Quality Aspects

This refers to the degree to which the research will be correctly carried out. It explores both the research validity and reliability.

3.9.1 Validity

Validity refers to the extent to which the data accurately measures what it was intended to. Automated model testing and a confusion matrix was used to ensure the validity of the model. Additionally, the convolution neural network model's loss or learning rate was visually tracked using Tensorboard; this made it easier to debug and optimize the model.

3.9.2 Reliability

Reliability focuses on the extent to which the data collection method will yield the same findings if replicated by others. Proper documentation of system requirements was done to ensure the reliability of the study.

3.11 Ethical Considerations

Secondary data used in the study was obtained from Kaggle open-source datasets and other online resources while primary data was obtained from strawberry farms. To ensure adherence to ethical codes of conduct, the collection process was in line with the permission for educational research. An approval from the Strathmore University Ethics Review Board(in appendix I) was obtained before commencing the study. All previous research works used in developing this study were cited appropriately giving due acknowledgement to the respective authors. A research permit was also obtained from NACOSTI(National Commission for Science, Technology and Innovation) as indicated in appendix J. To ensure the originality of the study, a similarity check was done and the report is as shown in appendix K.

Chapter 4: System Analysis, Design and Architecture

4.1 Introduction

This chapter outlines the design architecture of the proposed deep learning model for strawberry fungal leaf disease detection. The design is based on the conceptual model presented in figure 2.10 in chapter two. The different components of the proposed model were also covered together with the interactions between the different components of the model and between the model and the users. The unified Modelling Language was used to model these interactions and the illustrations done through the system sequence diagram, database schema diagram and use case diagram.

4.2 Requirements Analysis

This section gives a brief description of the requirements that were gathered for use in the development of the deep learning model. The requirements consisted of qualitative data gathered through questionnaires with well guided questions. This captured the expectations of potential users of the model on its functioning in general. A sample of the questionnaire is as shown in appendix E and sample responses in appendix F. The outcome of this process resulted to users' expectations that were broadly classified into two categories: functional and non-functional requirements.

4.2.1 Functional Requirements

i. The model should accept an image from a user.
ii. The model should transform each of the images in the required format and extract the relevant features.
iii. The model should perform the classification of the fungal leaf disease based on the extracted features.
iv. The model should be able to return the classification results (the fungal leaf disease class) to the user instantly.

In order to meet the above requirements, the model was deployed on a web interface that accepts images and with the capability to convert images of various sizes into a standard dimension for classification. The web interface enables one to pass and image to the model and return the fungal leaf disease.

4.2.2 Non-functional Requirements

4.2.2.1 Supportability Requirements

The model should support images from a wide range of input cameras. This will promote the usability of the model in real-world environment.

4.2.2.2 Reliability Requirements

The model should be able to provide consistent results when provided with the same input data.

4.2.2.3 Maintainability

The model should be easy to maintain and support.

4.2.2.4 Security Requirements

The model should be secure such that alteration of the model's configurations should be done by authorized personnel only.

To enhance supportability, the model was deployed on a web interface which accepts images of various sizes and convert them to a standard size. The model's reliability was ensured by training it with the required features to make sure it achieves consistent results with the same input. The model's security was ensured by passing only the saved version of the model to the interface. The model's configuration parameters cannot be accessed here hence making it secure against reconfiguration.

4.3 System Architecture

4.3.1 System Model Architecture

The system architecture is as illustrated in figure 4.1. This was designed based on the system requirements outlined in section 4.2. The training and testing datasets are created from the raw pre-processed leaf images. The model will utilize images captured by cameras as input which are fed into the trained model for inference. The classification results obtained displayed to the user on an interface.

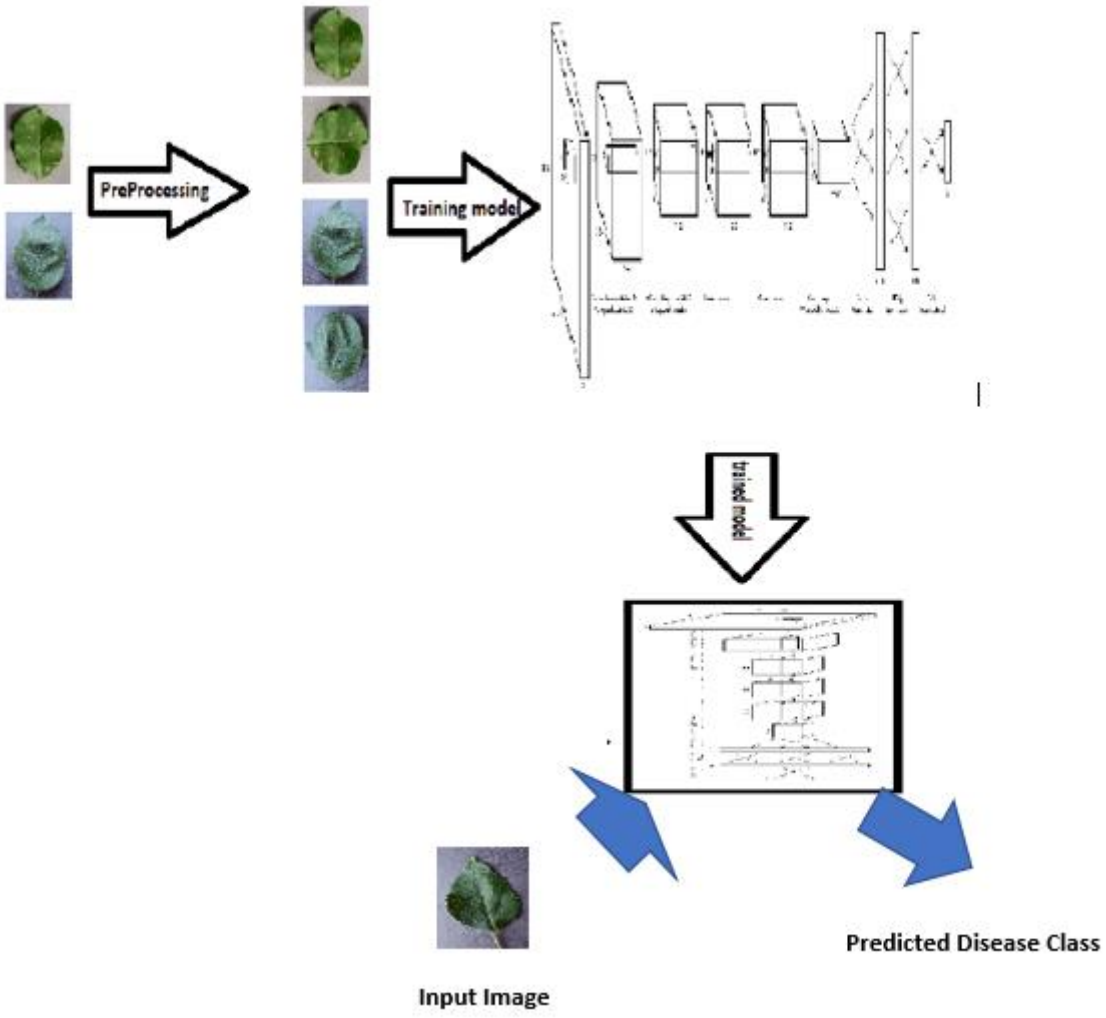


Figure 4. 1: System Architecture

4.4 System Design

Systems design entails the definition of elements of a system like modules, architecture, components and their interfaces and data for a system with regard to the specified requirements. The study adopted object-oriented system analysis and design methodology.

4.4.1 Use Case Diagram

A use case diagram in UML captures the actors' interactions with the system to achieve a certain goal. The actors constitute the general users of the model. In this case, there two major actors who directly interact with the system. The first one is the farmer who uses the model and the admin who is responsible for making model changes. The primary actor is the user who is responsible for uploading

input images into the model. The admin is responsible for the maintenance and constant update of the model. The use case diagram is as shown in figure 4.2.

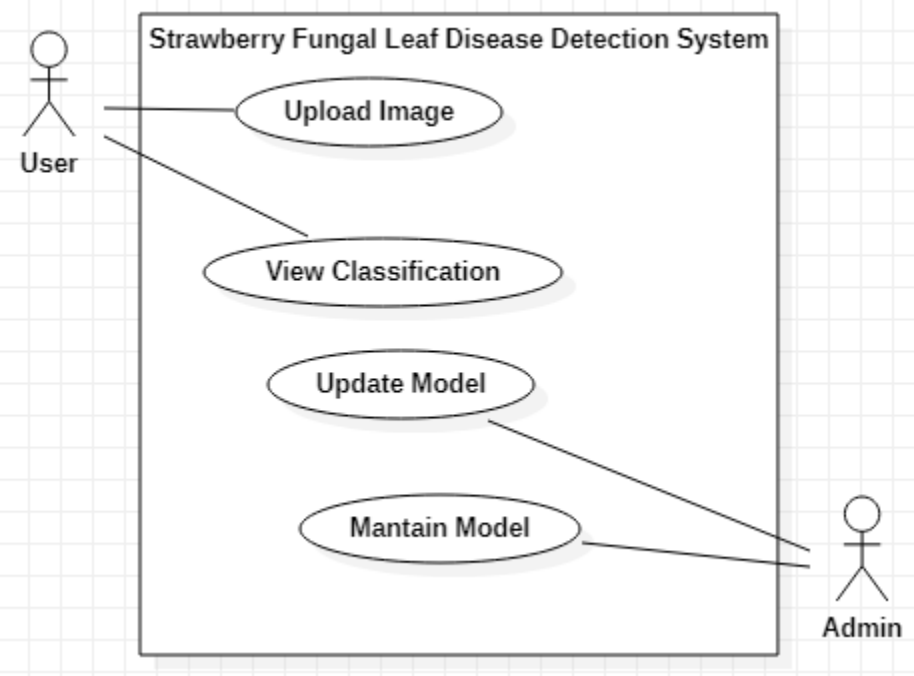


Figure 4. 2: Use Case Diagram

Table 4. 1: Upload Image Description

Use Case	**Upload Image**
Primary Actors	**User**
Brief Description	The use case describes how the user will upload an image into the model for classification
Pre-Condition	Existing Image
Post-condition	New Image Uploaded
Major Steps Performed	
Actor	**System**
User selects an image from the file directory	
User uploads the image	
	System loads the image
	System saves the uploaded image

Table 4. 2:View Classification Description

Use Case	**Load Image**
Primary Actors	**User**
Brief Description	The user views the results of the disease class prediction made
Pre-Condition	Inferences carried out by the model
Post-condition	Strawberry Fungal Leaf Disease class prediction
Major Steps Performed	
Actor	**System**
The user runs the classification command	
	System performs inferences regarding the disease class based on the features learned
	System returns the classification results
User Views the Results	

Table 4. 3:Update Model Description

Use Case	**Load Image**
Primary Actors	**Admin**
Brief Description	The admin updates the model with new features added
Pre-Condition	Trained model
Post-condition	Up to date model
Major Steps Performed	
Actor	**System**
The Admin runs the update command	
	System returns the classification results

Table 4. 4:Maintain Model Description

Use Case	**Load Image**
Primary Actors	**Admin**
Brief Description	The admin maintains the model by getting rid of any errors.
Pre-Condition	Trained model
Post-condition	Well-functioning model
Major Steps Performed	
Actor	**System**
The admin goes through the model after every update to check for any errors	
The admin fixes any errors present	
	System saves the new state of the model

4.4.3 Sequence Diagram

The sequence diagram is used to capture the interactions of objects within a system. It clearly outlines the flow of activities within the disease detection model hence enhancing a better understanding of the model working. Figure 4.3 shows the sequence diagram for strawberry fungal disease detection.

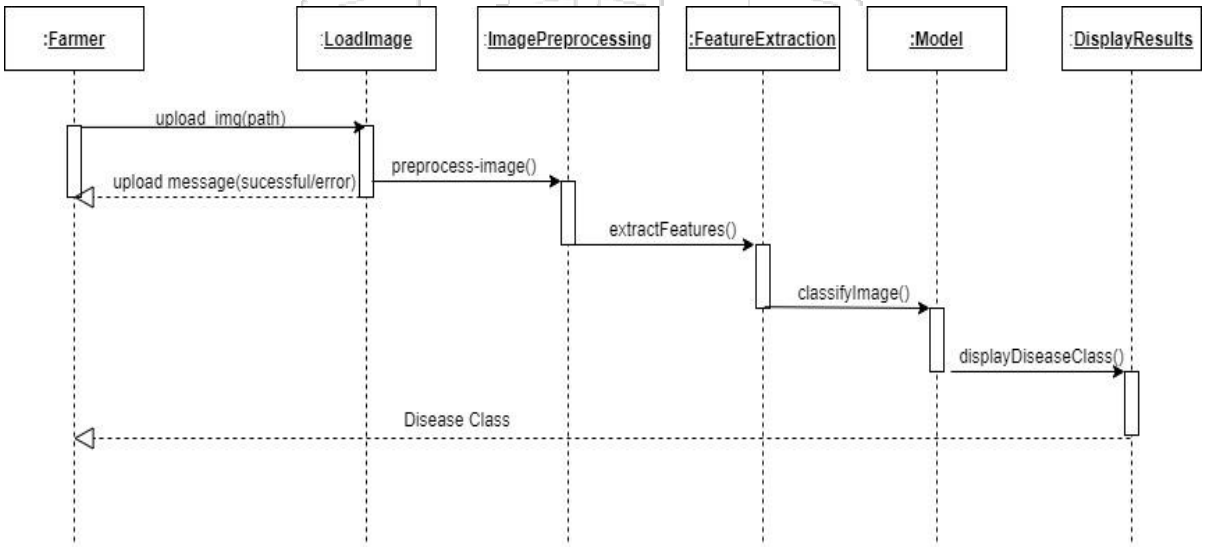


Figure 4. 3: Sequence Diagram for Strawberry Fungal leaf disease detection

4.3.4 Class Diagram

The class diagram provides a general overview of the system. It clearly depicts the classes within the system and the interactions among them hence enhancing a better understanding of the whole system by developers.

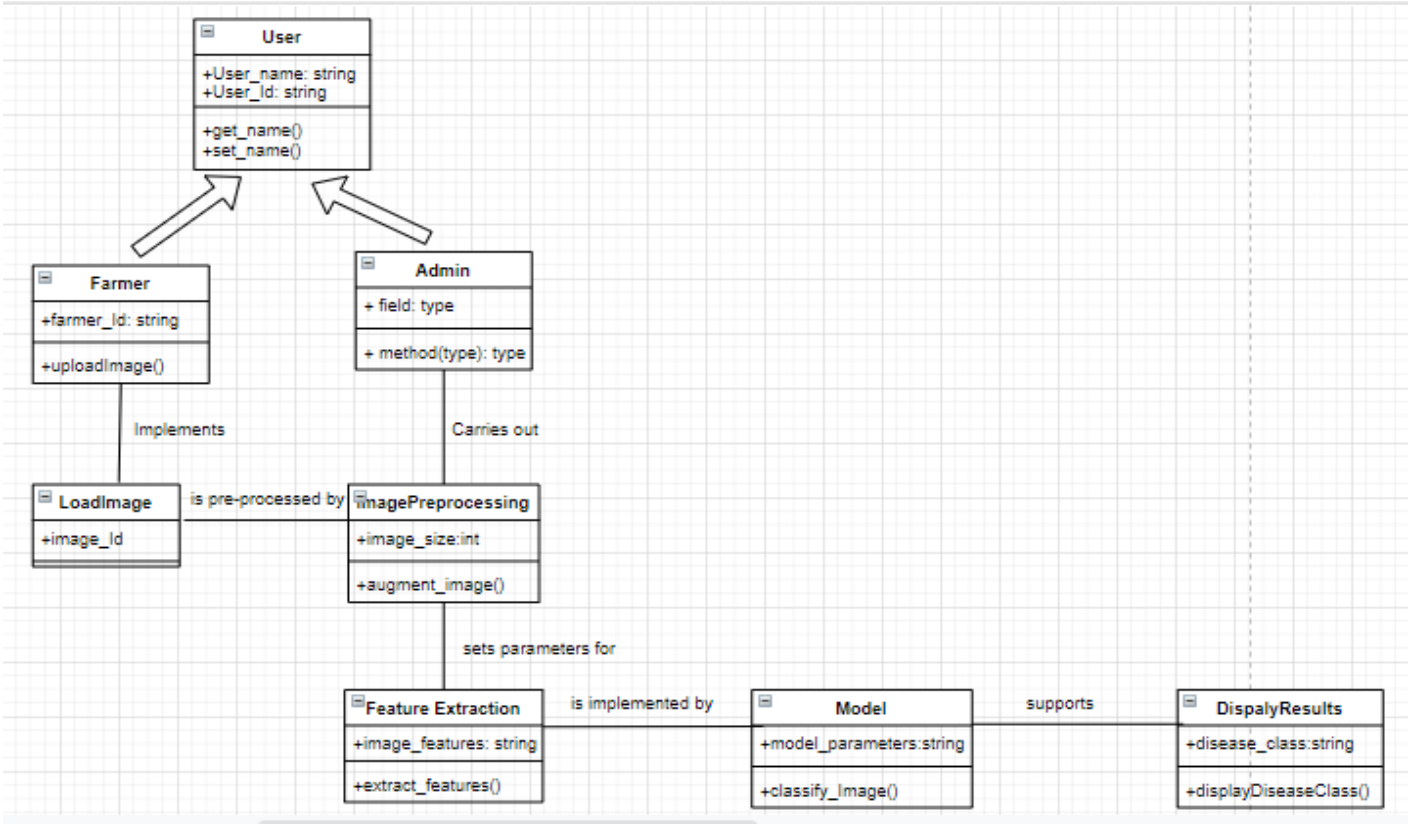


Figure 4. 4: Class Diagram

Chapter 5: System Implementation and Testing

5.1 Introduction

The aim of this study was to develop a deep learning model for strawberry fungal leaf disease classification which in turn will help the detection of these disease classes. This chapter gives details on the implementation and testing of the deep learning model for strawberry fungal leaf disease classification. Under implementation, the focus is on data preparation (pre-processing), the various modules of the model, their implementation and functionality and the tools used in the implementation. Testing on the other hand details the verification of the model's ability to satisfy the functional and usability requirements. The study employed the post-train testing approaches.

5.2 System Implementation

5.2.1 Development Environment

The model development was carried out on the Google Colab (Google Colaboratory) platform. This is a free cloud service offered by Google. The platform was preferred as it offers access to important software and hardware resources such as the GPU (Graphical Processing Unit) which were required in developing the model. It is quite expensive to acquire such hardware resources and newer versions are always availed creating the need to update or download the newer versions from time to time. The Google Colab platform also provides access to powerful libraries such OpenCV, Keras and TensorFlow which were essential in developing the model. It was also possible to automatically back up the notebooks containing the code and the image datasets the google drive. The hardware resources consisted of the hardware components used on the Google Colab development environment. This consisted of the GPU, CPU, RAM and the Disk. There was limited control on these resources as they were being accessed on a Google Colab which is a cloud-based platform. Table 5.1 shows the details of the hardware resources for the model.

Table 5. 1: Hardware Resources

Hardware	Details	
Google Cloud	GPU	1xTesla K80, having 2496 CUDA cores, compute 3.7, 12GB (11.439GB Usable) GDDR5 VRAM
	CPU	1xsingle core hyper threaded i.e. (1 core, 2 threads) Xeon Processors @2.3Ghz
	RAM	12.6 GB
	Disk	320 GB

The software resources on the other hand included the main programming language used and the respective libraries used in the model development process. The programming language used was python and this was due to the availability of the required libraries. Different computer vision and machine learning libraries were used: NumPy, Keras, TensorFlow, Augmentor, and PIL. These libraries provided different functionalities required for the model development. Google Colab notebooks were used for code writing and editing. Table 5.2 shows the software resources that were used for developing the model.

Table 5. 2:Software Resources

Software	Details	
Python 3.8	Libraries	Version
	NumPy	Version 1
	TensorFlow	Version 2
	Keras	1.16.2
	Augmentor	8.0.1
	PIL	8.0.1
	Scikit Learn	0.24
	Cv2	4.5.1
	Matplotlib	3.3.4
	Split_Folder	0.4.3

5.2.2 The CNN Model Components

The graphical representation of the deep learning model is as shown in figure 5.1. This shows the input, hidden and the output layers of the model. This was obtained using the Tensorboard visualization tool i.e. “tensorboard --logdir ” and accessing port 6006 in the web browser. The graph presents the various components of the model that were tracked during training.

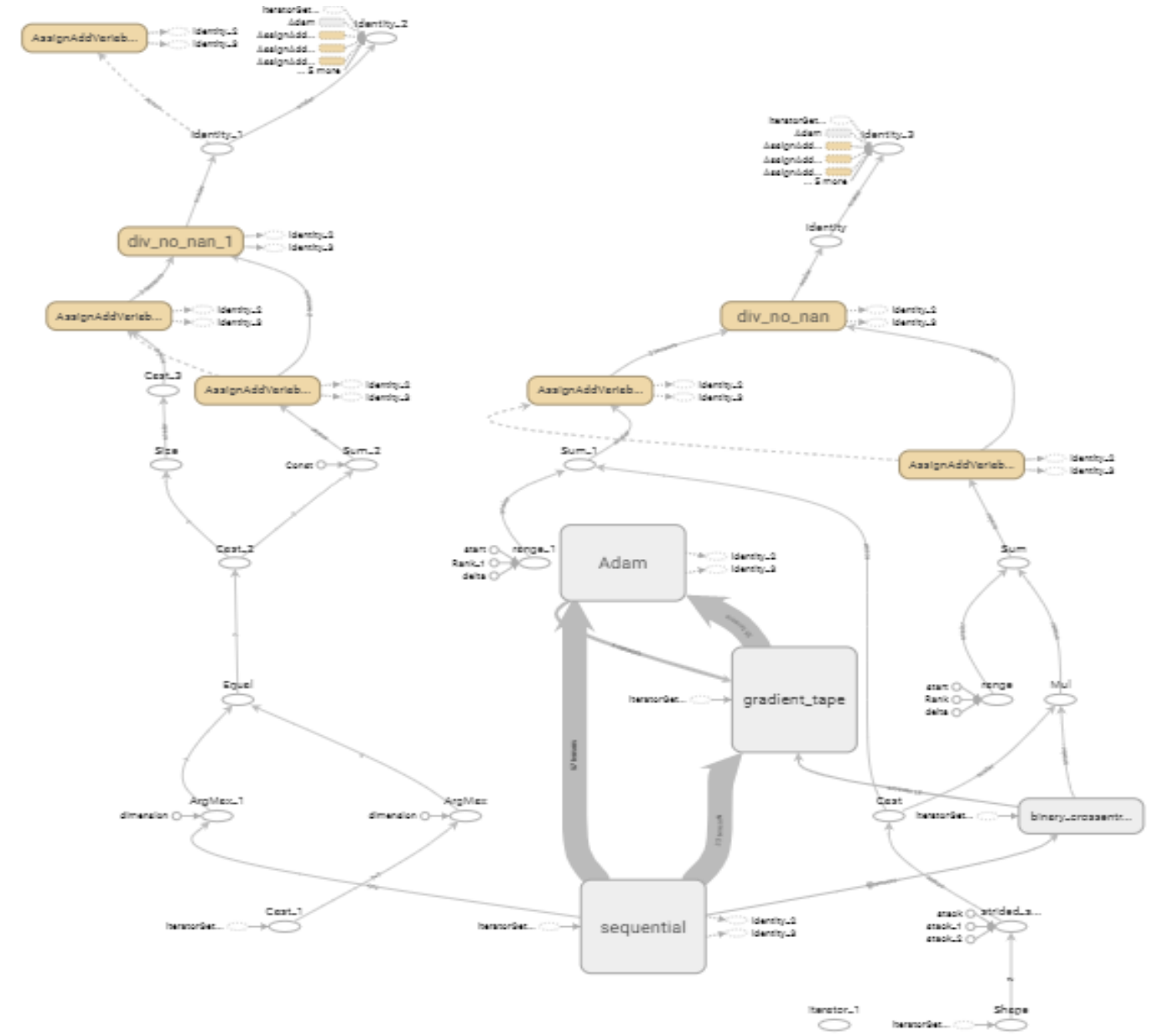


Figure 5. 1: Graphical Representation of the Deep Learning Model

5.2.2.1: Storage

The collected images were stored on Google drive, a cloud based storage platform offered by Google. This platform enabled the larger image dataset to be loaded in a fast and efficient manner. The notebooks used for writing and editing code were also stored on the platform enabling efficient synchronization with the dataset. The choice of storage also offered a cheaper platform as compared to acquiring hardware devices for data storage.

5.2.2.2: Input Layer

This is the first layer of the CNN model and it interacts with the external environment. This layer consists of the input nodes, place holder and module which provided input images to the network. This was implemented to accept standardized input images (256X256) in the form of 2D arrays and pass this information to the hidden layers. The graphical representation of the input layer is as shown in figure 5.2.

Figure 5. 2: Graphical Representation of the input layer

5.2.2.3 Hidden Layer

This refers to a series of network nodes that are not visible from the outside world. They are responsible for performing the necessary computations for feature extraction from the input images. This layer passes information from the input layer to the output layer having extracted the relevant features. The following processes were implemented within the hidden layer: Convolution, pooling and activation. The graphical representation of this layer is as shown in figure 5.3. The model made use of the SoftMax activation as shown in figure 5.3.

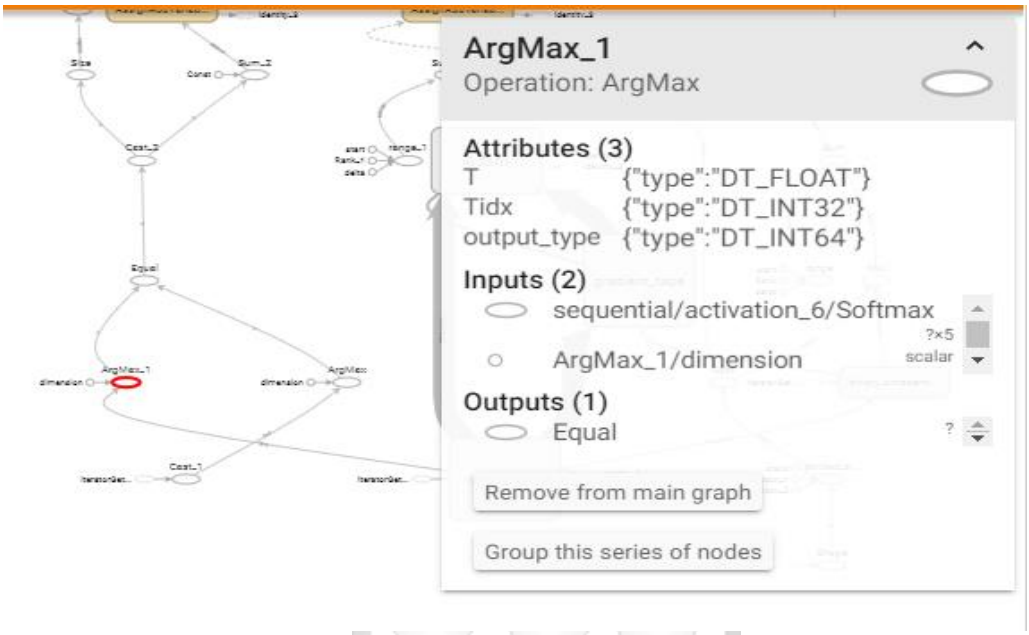


Figure 5. 3: Graphical Representation of the hidden layer

5.2.2.4 Output Layer

This is the last layer of the CNN model and is responsible for performing the classification hence giving final output of the model. SoftMax activation function was used for image classification. The cross entropy function was used to determine how close the predictions were to the actual labels. This was suitable as the model was a multiclass model. The graphical representation of the output/classification layer is as shown in figure 5.4.

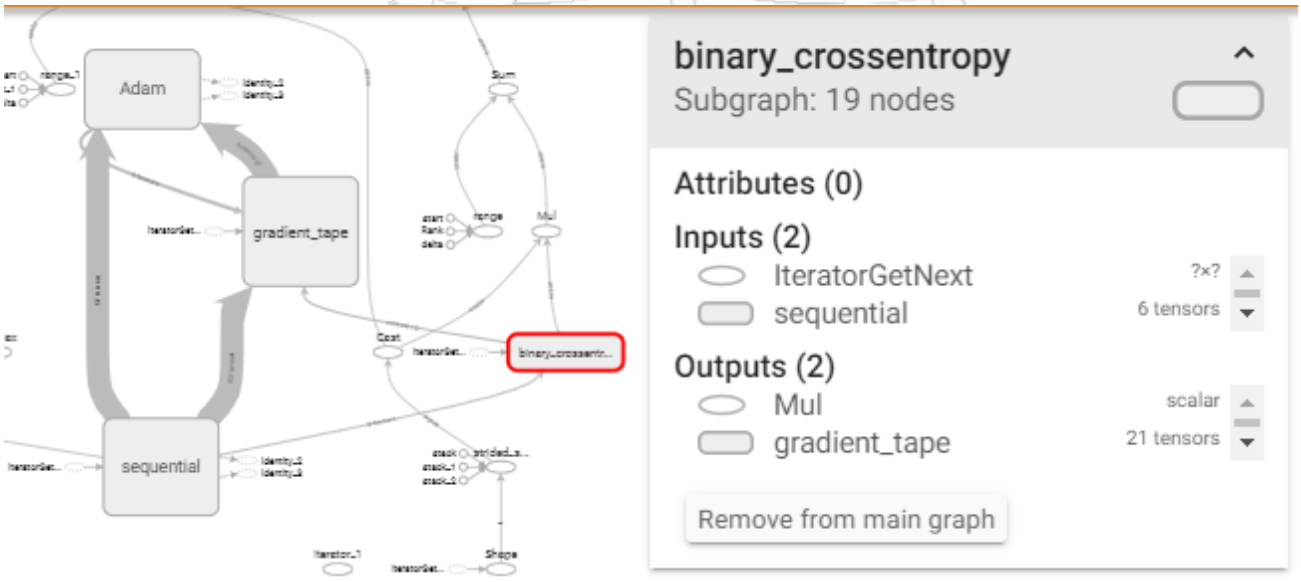


Figure 5. 4: Graphical Representation of the classification layer

5.2.3 Dataset Pre-processing

The data-pre-processing activities carried out include data augmentation and standardization. Data augmentation was implemented on the training set of images. This process was important for the model as it introduced variability in the dataset and increased the number of training examples hence availing more features to be learned by the model. An object of the ImageDataGenerator() class was created with the specified transformations for data augmentation. The ImageDataGenerator class inflates input images with an infinite number of transformations hence availing more features for training(Chollet, 2016). This was later passed to the model as input during training. The augmentation techniques used include rotation, zoom, flipping and shearing. The python script that was used to implement the augmentation is as shown in figure 5.5.

```
#Creating an object of the image generator for data augmentation. This will be passed to the model  during training
aug = ImageDataGenerator(
    rotation_range=25, width_shift_range=0.1,
    height_shift_range=0.1, shear_range=0.2,
    zoom_range=0.2,horizontal_flip=True,
    fill_mode="nearest")
```

Figure 5. 5: Python Script for Data Augmentation

Data standardization is another process that was carried under the pre-processing stage. This involved conversion/resizing of the images into a quality standard dimension of 256 X 256. This was important in enhancing efficiency in feature extraction. The conversion was also important in fitting the input format of the deep learning model. The images collected were already in a consistent format (JPEG) hence there was no need for conversion. Figure 5.6 shows a sample python script that was used to implement the conversion to a standard dimension of 256 x 256 of the strawberry healthy class. Sample code snippet showing standardization of images in the other classes are as shown in appendix A.

```
In [2]: #Standardization of the strawberry healthy leaves class
        #Importing the necessary Libraries
        import PIL
        import os
        import os.path
        from PIL import Image

In [4]: Imageset1=r"D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Augmented Dataset\Strawberry_Healthy" # Passing the image directory
        for file in os.listdir(Imageset1):# Iterating through the image files in the folder
            stdimg1=Imageset1+"/"+file
            img=Image.open(stdimg1)
            img=img.resize((256,256))# Resizing the images
            img.save(stdimg1) #Saving the images
```

Figure 5. 6 Python script for Image conversion to a standard dimension

5.2.4 Loading the Dataset

The images dataset was loaded into google drive as a folder in order to access and use them on the Google Colab platform. This was achieved by mounting the drive on Colab using drive.mount() function. The path to this folder passed was then passed to a variable root_directory on Colab. This was achieved as shown by the python script in figure 5.7.

```
[40] #Mounting the drive and passing path to the image dataset
     from google.colab import drive
     drive.mount('/content/drive/')
     root_directory = '/content/drive/MyDrive/THESIS DATA/Research Data'#Path to  the image dataset
```

Figure 5. 7: Python script for mounting the drive and passing the image path to a variable to google drive

The images were loaded and converted into a numpy array using a defined function which iterated through the specific folders. The definition of the function and its output are as shown in appendix B.

5.2.5 Transforming the Image Classes into Labels

The class labels for each fungal disease class were mapped to unique binary values for the training task. This was achieved using the scikit learn label binarizer library. The output was an array of integers dumbed/saved to a pickle file to be used later for prediction. The array consists of a series of multiple integers for machine interpretation during prediction. This was implemented as in the python script shown in figure 5.8 and the output shown in figure 5.9.

```
[ ]  #Transforming image labels using scikits label binarizer
     label_binarizer = LabelBinarizer()
     image_labels = label_binarizer.fit_transform(label_list)
     pickle.dump(label_binarizer,open('fungal_disease_label_transform.pkl', 'wb'))
     n_classes = len(label_binarizer.classes_)
     #Printing  the classes
     print(label_binarizer.classes_)
     print("Total Number of Classes:", n_classes)
     print(image_labels)
```

Figure 5. 8: The label binarizer python script

```
['Strawberry_Healthy' 'Strawberry_Leaf_Blight' 'Strawberry_Leaf_Scorch'
 'Strawberry_Leaf_Scorch_and_Leaf_Blight' 'Strawberry_Leaf_Spot']
Total Number of Classes: 5
[[1 0 0 0 0]
 [1 0 0 0 0]
 [1 0 0 0 0]
 ...
 [0 0 1 0 0]
 [0 0 1 0 0]
 [0 0 1 0 0]]
```

Figure 5. 9: The label binarizer output

After the label transformation, the images loaded were then split into training and validation sets(80% training and 20% validation). The choice of this split was based on the acceptable results indicated in reviewed works in section 2.4. The splitting was implemented using the train_test_split() function from the sklearn library as shown in figure 5.10.

```
[ ]  #Splitting the image dataset into training and testing
     print("[INFO] Spliting data to train and test portions")
     x_train, x_test, y_train, y_test = train_test_split(np_image_list, image_labels, test_size=0.2, random_state = 42)

     [INFO] Spliting data to train and test portions
```

Figure 5. 10: Splitting the dataset into training and testing

5.2.5 Model Implementation

The deep learning model was implemented as a sequential CNN using the following parameters: 30 Epochs, a batch size of 32, the number of training steps set to 100, a learning rate of 1e-3, a depth of 3 and the image width and height set to a size of 256. These parameters were chosen on an experimental basis as the study was based on an experimental design. This is as indicated in section 5.4 for hyper parameter tuning. The image height and width were set to 256 in order to fit the input shape of the model. The model was not only defaulted to the " channel_first" architecture but also a switch for backends that support "channel_last" was also created. The model initially consisted of 2D Convolutional layer with 32 filters of 3 x 3 kernel and a ReLU (Rectified Linear Unit) activation. Batch normalization, max pooling, and 25% (0.25) dropout operation was then performed in the subsequent layers. This was followed by two blocks of 2D Convolutional layer with 64 filters and ReLU activation followed by a pooling and dropout layer. This step was repeated for the last set of fully connected layers with 128 filters in the Conv2D layer being the only difference. The architecture of the model was implemented as indicated in appendix C with its summary in appendix D.

5.3 Model Training and Testing

5.3.1 Model Training

Before training the model, the optimizer was initialized with the learning rate and decay parameters that were already defined as indicated in section 5.2.5. The Adam optimizer was chosen for the study as it is characterized by faster performance and a better global minimum convergence as compared to the other optimization techniques. The model was then compiled on the binary cross entropy as the loss function. This loss function was suitable as the class labels being used in the training were

represented in a binary form. The model was trained using the model.fit_generator function which accepts the image generator defined earlier as an image input. The performance of the model during training is as indicated in figure 5. 11. The training and validation accuracy of the model throughout the training process is as shown in figure 5.12 with the x-axis indicating the number of epochs and the y-axis the respective accuracies. Figure 5.13 indicates the training and validation loss.

```
[INFO] training the deep normalized CNN Model...
/usr/local/lib/python3.7/dist-packages/tensorflow/python/keras/engine/training.py:1844: UserWarning: `Model.fit_generator` is deprec
  warnings.warn('`Model.fit_generator` is deprecated and '
Epoch 1/30
56/56 [==============================] - 33s 442ms/step - loss: 0.7441 - accuracy: 0.5697 - val_loss: 0.8534 - val_accuracy: 0.2578
Epoch 2/30
56/56 [==============================] - 24s 436ms/step - loss: 0.3005 - accuracy: 0.7878 - val_loss: 0.8432 - val_accuracy: 0.2089
Epoch 3/30
56/56 [==============================] - 25s 438ms/step - loss: 0.1720 - accuracy: 0.8791 - val_loss: 3.8616 - val_accuracy: 0.2156
Epoch 4/30
56/56 [==============================] - 25s 437ms/step - loss: 0.1464 - accuracy: 0.8871 - val_loss: 4.5696 - val_accuracy: 0.2067
Epoch 5/30
56/56 [==============================] - 24s 437ms/step - loss: 0.1267 - accuracy: 0.9053 - val_loss: 1.2418 - val_accuracy: 0.3044
Epoch 6/30
56/56 [==============================] - 24s 436ms/step - loss: 0.1010 - accuracy: 0.9379 - val_loss: 1.9131 - val_accuracy: 0.3022
Epoch 7/30
56/56 [==============================] - 25s 437ms/step - loss: 0.0855 - accuracy: 0.9404 - val_loss: 1.4806 - val_accuracy: 0.3733
Epoch 8/30
56/56 [==============================] - 25s 442ms/step - loss: 0.0870 - accuracy: 0.9336 - val_loss: 0.6557 - val_accuracy: 0.5489
Epoch 9/30
56/56 [==============================] - 25s 437ms/step - loss: 0.0874 - accuracy: 0.9432 - val_loss: 1.3542 - val_accuracy: 0.5622
Epoch 10/30
56/56 [==============================] - 25s 439ms/step - loss: 0.1047 - accuracy: 0.9215 - val_loss: 0.6880 - val_accuracy: 0.7289
Epoch 11/30
56/56 [==============================] - 25s 441ms/step - loss: 0.0903 - accuracy: 0.9362 - val_loss: 0.9899 - val_accuracy: 0.7956
Epoch 12/30
56/56 [==============================] - 25s 440ms/step - loss: 0.0801 - accuracy: 0.9443 - val_loss: 0.1712 - val_accuracy: 0.9333
Epoch 13/30
56/56 [==============================] - 25s 441ms/step - loss: 0.0723 - accuracy: 0.9563 - val_loss: 0.8020 - val_accuracy: 0.5933
Epoch 14/30
56/56 [==============================] - 25s 444ms/step - loss: 0.0588 - accuracy: 0.9563 - val_loss: 0.2867 - val_accuracy: 0.7822
Epoch 15/30
56/56 [==============================] - 25s 442ms/step - loss: 0.0701 - accuracy: 0.9581 - val_loss: 0.1643 - val_accuracy: 0.9244
Epoch 16/30
56/56 [==============================] - 25s 451ms/step - loss: 0.0722 - accuracy: 0.9548 - val_loss: 0.1578 - val_accuracy: 0.8844
Epoch 17/30
56/56 [==============================] - 25s 446ms/step - loss: 0.0487 - accuracy: 0.9664 - val_loss: 0.3474 - val_accuracy: 0.7822
Epoch 18/30
56/56 [==============================] - 25s 447ms/step - loss: 0.0659 - accuracy: 0.9616 - val_loss: 0.3684 - val_accuracy: 0.7444
Epoch 19/30
56/56 [==============================] - 25s 445ms/step - loss: 0.0647 - accuracy: 0.9554 - val_loss: 0.4159 - val_accuracy: 0.7156
Epoch 20/30
56/56 [==============================] - 25s 450ms/step - loss: 0.0509 - accuracy: 0.9675 - val_loss: 0.1232 - val_accuracy: 0.9289
Epoch 21/30
56/56 [==============================] - 26s 455ms/step - loss: 0.0314 - accuracy: 0.9804 - val_loss: 0.0792 - val_accuracy: 0.9333
Epoch 22/30
56/56 [==============================] - 25s 449ms/step - loss: 0.0268 - accuracy: 0.9850 - val_loss: 0.0996 - val_accuracy: 0.9022
Epoch 23/30
56/56 [==============================] - 25s 448ms/step - loss: 0.0370 - accuracy: 0.9686 - val_loss: 0.0903 - val_accuracy: 0.9178
Epoch 24/30
56/56 [==============================] - 25s 451ms/step - loss: 0.0429 - accuracy: 0.9743 - val_loss: 0.0672 - val_accuracy: 0.9578
Epoch 25/30
56/56 [==============================] - 25s 452ms/step - loss: 0.0371 - accuracy: 0.9769 - val_loss: 0.0366 - val_accuracy: 0.9622
Epoch 26/30
56/56 [==============================] - 25s 450ms/step - loss: 0.0243 - accuracy: 0.9849 - val_loss: 0.0687 - val_accuracy: 0.9578
Epoch 27/30
56/56 [==============================] - 26s 456ms/step - loss: 0.0299 - accuracy: 0.9851 - val_loss: 0.0616 - val_accuracy: 0.9644
Epoch 28/30
56/56 [==============================] - 25s 451ms/step - loss: 0.0302 - accuracy: 0.9787 - val_loss: 0.1262 - val_accuracy: 0.9267
Epoch 29/30
56/56 [==============================] - 26s 455ms/step - loss: 0.0325 - accuracy: 0.9828 - val_loss: 0.1604 - val_accuracy: 0.8778
Epoch 30/30
56/56 [==============================] - 25s 454ms/step - loss: 0.0283 - accuracy: 0.9770 - val_loss: 0.0292 - val_accuracy: 0.9844
```

Figure 5. 11: The Deep Learning Model Performance During Training

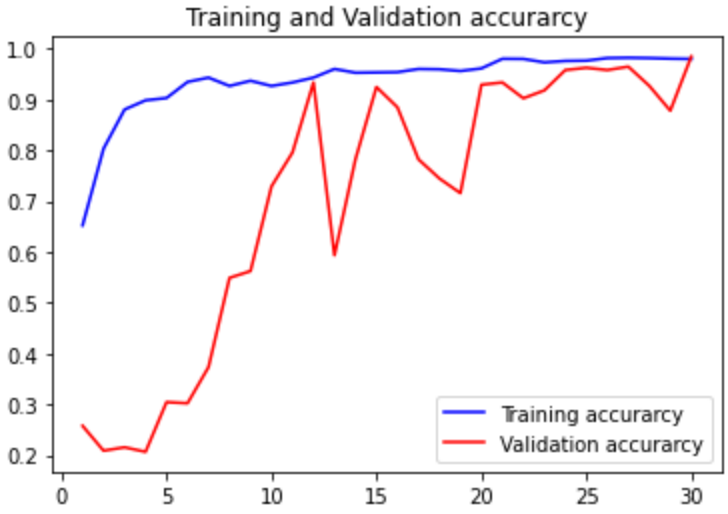


Figure 5. 12: Training and validation accuracy

From figure 5.12, it is evident that the validation accuracy of the model was lower than the training accuracy at the beginning and increased with model training. The training accuracy indicates the performance of the model on the training dataset while validation accuracy the performance of the model on the validation dataset. A large gap between the two accuracies implies that the model is overfitting. The validation accuracy had several dips which stabilized a little towards the end. The dips are attributed to the varying performance of the model during learning as the number of features learned increased throughout the training process. The minimal gap between the two accuracies in the 30^{th} epoch indicates minimal overfitting effect by the model.

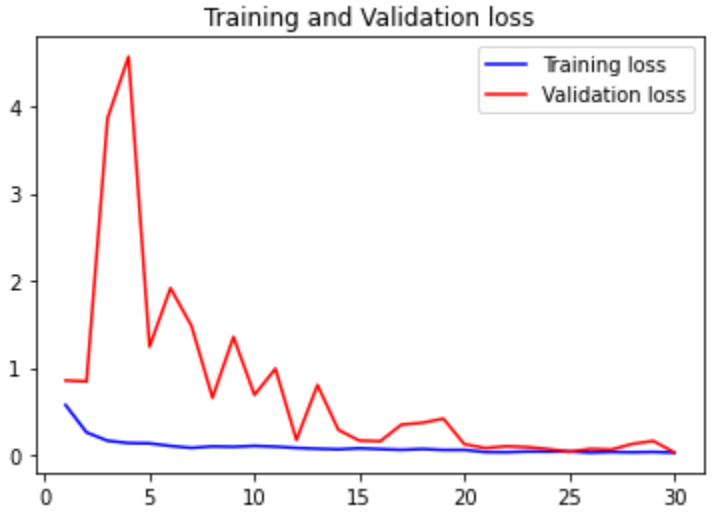


Figure 5. 13: Training and validation loss

Figure 5.13 indicates that the training loss was less than the validation loss with the validation loss being characterized by several dips. Loss is a calculated on training and validation sets. It represents the sum of errors made for each sample data in the training and validation sets. The training and validation losses decreases to a point of stability between the 15th and the 20th epochs with no gap between the two losses at the last epoch. The dips are attributed to lower generalization capability in the initial phases which increased throughout the training process. This indicates that the model was a good fit on the validation dataset or generalized well.

5.3.2 Model Testing

Functional testing was carried out on the model in order to certify the model's ability to meet the basic functional requirements. Non-functional testing test cases such as reliability and supportability testing were also carried on the model. Table 5.3 shows the test cases used in testing the model while table 5.4 shows the results of model testing.

Table 5. 3: Model Testing Checklist

Test Case	Inspection Check	Priority
Functional	Does the model accept an Image from the user?	High
Functional	Does the model transform each of the images in the required format and extract the relevant features?	High
Functional	Does the model perform the classification of the fungal leaf disease based on the extracted features?	High
Functional	Does the model return the classification results (the fungal leaf disease class) to the user instantly?	High
Supportability	Does the model support an image from a wide range of cameras?	Moderate
Reliability	Does the model provide consistent results with the same input?	High
Reliability	Does the model provide warning messages to the users?	Moderate

Table 5. 4: Model Testing Results

Test Case	Inspection Check	Results
Functional	Does the model accept an Image from the user?	Pass
Functional	Does the model transform each of the images in the required format and extract the relevant features?	Pass
Functional	Does the model perform the classification of the fungal leaf disease based on the extracted features?	Pass
Functional	Does the model return the classification results (the fungal leaf disease class) to the user instantly?	Pass
Supportability	Does the model support an image from a wide range of cameras?	Pass
Reliability	Does the model provide consistent results with the same input?	Pass

5.4 Model Deployment and Validation

The model was deployed on a simple web-based user interface(UI) using the python Flask API(Python Web Application Framework). The backend of the interface constitutes a python function named as predict_fungal_disease() which has two parameters: image path and the path to the saved model. The image labels transform file was also passed during the function call. This was basically achieved by passing the path to the image label transform pickle file. The function takes the image path as input and returns the strawberry fungal disease label. The image is first converted to the standard dimension(256X256) that fits the input shape of the model by another function called imag_to_array() before is passed to the model for prediction. Predicting the disease label basically involved calling the function and passing the image path. The upload image button on the interface enables the user to upload an image and make a prediction using the predict disease label button. The model deployment using flask implementation code is as shown in appendix G with the sample the sample web interface shown in appendix H. Sample output of the prediction is as shown in figure 5.14.

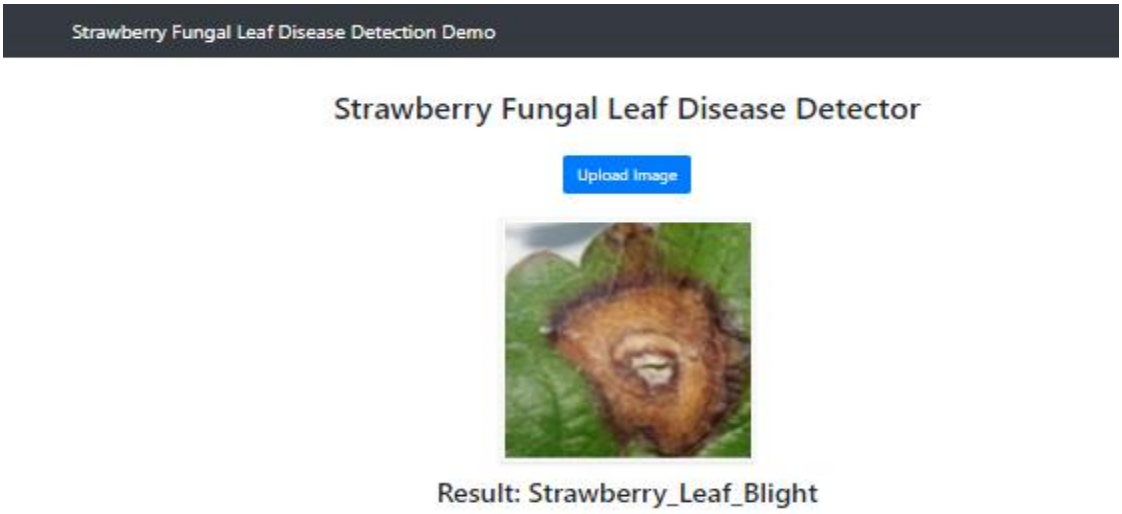


Figure 5. 14:Sample prediction of the disease class

The python function at the backend of the interface was used to validate the model. The validation was implemented on the validation dataset. This constituted a total of 226 images. The distribution of the images per disease class were as follows: Healthy-90 images, Leaf Scorch-90 images, Leaf Spot-17 images, Leaf Blight- 14 images , Leaf Scorch and Leaf Blight- 15 images.

5.5: Hyperparameter Tuning

The development of the deep learning model was based on an experimental research design. Several parameters were varied to monitor the performance of the model. The parameters adjusted included: the learning rate and the number of epochs. The hyper parameters were important in determining the model structure and in controlling system requirements such as the cost of memory. The variation of the parameters was made possible by the use of the Google Colab platform which allowed access to the GPU. The learning rate is important in controlling the speed or the rate at which the model learns the features to be used in classification. An optimal learning rate will enable the model's weight to be updated in an acceptable time and resulting to optimal final weights of the model. Lower learning rates results to the model taking significantly longer time to train. Smaller learning rates may also never converge or converge to a sub-optimal solution. Extremely large learning rates are also not desirable; this has an effect of increasing the gradient descent. It is not possible to calculate the optimal learning rate for a particular dataset. It was also observed that the higher the number of epochs used(with other

factors remaining constant), the higher the probability of the model in overfitting. The parameters and the corresponding results are as shown in table 5.5.

Table 5. 5:Hyperparameter Tuning

Hyper Parameter	Results
Learning Rate	
0.00001	Longer training period of 2465 seconds Accuracy of 66.86%
0.0001	Training period of 1872 seconds Converged to a non-optimal accuracy of 65.34 %
0.001	Optimal accuracy of 98% with an optimal training time of 1400 seconds
Number of Epochs	
20	Accuracy of 89.15%
30	Accuracy of 98.4%
40	Accuracy of 87.16%

Chapter 6: Results and Discussions

6.1 Introduction

The main objective of the study was to develop deep learning model for classifying the 3 classes of strawberry fungal leaf diseases which are Leaf Scorch, Leaf Blight and Leaf Spot. The study also considered a case where more than one class of the diseases occur on the same leaf(Leaf Scorch and Leaf Blight). This constituted a total of 5 classes including the healthy class. This chapter discusses the model evaluation results, a comparison of the model's performance with other machine learning models together with detailed discussions on them.

6.2 Model Evaluation

The following metrics were used in the evaluation of the model's performance: accuracy, precision, recall and the F1-score. They are obtained as follows:

(6.1) $$\text{Accuracy} = \frac{\text{TP+TN}}{\text{Total}}$$ (Géron, 2019)

Accuracy alone does not provide an in depth analysis of the performance of the model on new data. Other performance metrics such as precision, recall and F1-score are useful in model evaluation(Géron, 2019). Precision is given by equation 6.2.

(6.2) $$\text{precision} = \frac{\text{TP}}{\text{TP+FP}}$$ (Géron, 2019)

TP stands for True positive and FP stands for False positive. Recall on the other hand gives the proportion of actual positives that is correctly classified. It measures the ability of the model in identifying all the relevant instances. Recall is obtained as shown in equation 6.3 where FN stands for false negative.

(6.3) $$\text{recall} = \frac{\text{TP}}{\text{TP+FN}}$$ (Aurélien, 2019)

The F1-score gives a summary of the recall and precision into a single metric value. It is obtained by computing the harmonic mean of the recall and precision as shown in equation 6.4.

(6.4) $$\text{F1}=2X\frac{precision\ x\ recall}{precision+recall}$$ (Géron, 2019)

The definition of true positive, true negative, false positive and false negative in the context of the study is as follows:

i. **True Positive (TP):** Correctly identified prediction for each disease class.

ii. **True Negative (TN):**Correctly rejected prediction for certain disease classes.

iii. **False Positive (TP):**Incorrectly identified predictions for certain disease classes.

iv. **False Negative(TP):**Incorrectly rejected predictions for the disease classes.

6.2.1 Accuracy Results

Out of a total of 226 images used for validation, 222 images were correctly classified by the model into the respective disease classes resulting to an accuracy of 98%. This was a good performance from the model. The confusion matrix generated is as shown in table 6.1.The confusion matrix represents the performance measure of the model across the five labelled classes.

Table 6. 1:The Confusion Matrix

Actual Label/Predicted Label	Healthy	Leaf Blight	Leaf Scorch	Leaf Scorch and Leaf Blight	Leaf Spot	Accuracy per class
Healthy	89	1	0	0	0	98.89%
Leaf Blight	0	14	0	0	0	100%
Leaf Scorch	0	0	90	0	0	100%
Leaf Scorch and Leaf Blight	0	0	2	13	0	86.67%
Leaf Spot	0	1	0	0	16	94.12%

From the confusion matrix table in figure 6.1, the correct predictions are indicated along the diagonal(shaded) while the performance of the model per class shown in the last column. One instance of the healthy class was incorrectly classified as Leaf Blight. The model correctly classified each of instances of Leaf Scorch and Leaf Blight classes presented to it. This is a good performance in these classes attributed to the ability of the model being able to learn well the features used for prediction. Two instances of Leaf Scorch and Leaf Blight wrongly classified as Leaf Scorch. Lastly, one instance of Leaf Spot wrongly classified as leaf blight. The erroneous classification in the Leaf Scorch and Leaf Blight class is due to the two diseases affecting the same leaf hence having overlapping disease features which vary in magnitude. The features consist of an overlap of black pots and a continuous brownish surface on the leaves.

6.2.2 Precision, Recall and F1-Score Results

The other metrics(precision, recall and the f1-score) were generated from the confusion matrix using the sklearn.metrics.classification_report function from the sklearn library. The result of this is as shown in the classification report in figure 6.1.

	precision	recall	f1-score	support
Strawberry_Healthy	1.000	0.989	0.994	90
Strawberry_Leaf_Blight	0.875	1.000	0.933	14
Strawberry_Leaf_Scorch	0.978	1.000	0.989	90
Strawberry_Leaf_Scorch_and_Leaf_Blight	1.000	0.867	0.929	15
Strawberry_Leaf_Spot	1.000	0.941	0.970	17
accuracy			0.982	226
macro avg	0.971	0.959	0.963	226
weighted avg	0.984	0.982	0.982	226

Figure 6. 1:The classification Report

With reference to the classification report in figure 6.1, the model achieved an aggregated precision of 0.971, a recall of 0.959 and an f1-score of 0.963. Under the support column, we have the number of images per class used for validation. The figure 226 denotes the total number of images used for validation. A precision of 0.971 implies that 97% of the prediction results returned by the model were relevant/correct. This indicates a good proportion of the relevant results by the model. The aggregate recall of 0.959 indicates a good performance of the model in correctly identifying the relevant instances of the disease classes. An aggregate f1-score of 0.963 achieved by the model implies that the model achieved both higher recall and precision as f1-score is the harmonic mean of the two. This results to a good accuracy in making the predictions.

With regard to these results, the model achieved a good classification accuracy of 98% and generalized well on the validation data. The ability of the model to generalize well on unseen data is indicated by the narrow gap between the training and the validation accuracy as indicated in figure 5.12. This in turn indicates a low overfitting effect by the model. The model also performed well in strawberry leaf spot class(94.12% accuracy) and an instance where both Leaf Blight and Leaf Scorch occur(86.67% accuracy). This is as opposed to the existing models where these classes were not considered.

6.3 CNN Model Results Comparison

The developed CNN model was compared with other machine learning methodologies. These included the K-Nearest Neighbor(KNN), Support Vector Machine(SVM) and the Decision tree.

6.3.1 K-Nearest Neighbor(KNN)

KNN classifier was developed and trained for the classification task for comparison with the developed model. This algorithm relies on the distance between the feature vectors. It classifies the images using the most common class among the available training class features. The model was trained and evaluated as shown in figure 6.2.

```
#k-NN classifer Traing and evaluation  on the raw pixel intensities
print("[INFO] evaluating raw pixel accuracy...")
model = KNeighborsClassifier(n_neighbors=args["neighbors"],
  n_jobs=args["jobs"])
model.fit(trainRI, trainRL)
acc = model.score(testRI, testRL)
print("[INFO] raw pixel accuracy: {:.2f}%".format(acc * 100))
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print('Precision: %.3f' % precision_score(y_test, y_pred))
print('Recall: %.3f' % recall_score(y_test, y_pred))
print('F1 Score: %.3f' % f1_score(y_test, y_pred))
```

Figure 6. 2: Implementation of the KNN Model

Table 6.2 shows the results of the KNN model

Metric	Value(%)
Accuracy (%)	57.58
Precision (%)	55.36
Recall (%)	53.84
F1-Score (%)	54.59

Table 6. 2: KNN Results

6.3.2 Support Vector Machine(SVM)

The support vector machine learning model was developed using the Scikit-learn API which was used for re-training and validating the pre-trained model. The model training and evaluation was done as shown in figure 6.3 with the results shown in table 6.3.

```
#Defining the svm model
from sklearn import svm
from sklearn.model_selection import GridSearchCV
param_grid={'C':[0.1,1,10,100],'gamma':[0.0001,0.001,0.1,1],'kernel':['rbf','poly']}
svc=svm.SVC(probability=True)
model=GridSearchCV(svc,param_grid)
#loading the split library
from sklearn.model_selection import train_test_split
#splitting the dataset
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.20,random_state=77,stratify=y)
print('Splitted Successfully')
#traing/fitting the model on the training dataset
model.fit(x_train,y_train)
#model evaluation
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print('Precision: %.3f' % precision_score(y_test, y_pred))
print('Recall: %.3f' % recall_score(y_test, y_pred))
print('F1 Score: %.3f' % f1_score(y_test, y_pred))
```

Figure 6. 3: SVM Implementation

Metric	Value(%)
Accuracy (%)	72.64
Precision (%)	83.46
Recall (%)	67.81
F1-Score (%)	74.83

Table 6. 3: SVM Results

6.3.3 Random Forest

Random forest is a collection of Decision Trees. The trees vote together to give the best set of results. This model was developed using Scikit-learn library. The models was fitted to the training dataset and evaluated as shown in figure 6.4. Table 6.4 shows the results for the random forest based model.

```
#Importig Random Forest Model
from sklearn.ensemble import RandomForestClassifier
#Creatig a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100)
#Train the model using the training sets
clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
#Import scikit-learn metrics module for calculating the accuracy
from sklearn import metrics
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
print('Precision: %.3f' % precision_score(y_test, y_pred))
print('Recall: %.3f' % recall_score(y_test, y_pred))
print('F1 Score: %.3f' % f1_score(y_test, y_pred))
```

Figure 6. 4: Random Forest Training and Evaluation

Metric	
Accuracy (%)	78.14
Precision (%)	82.67
Recall (%)	76.16
F1-Score (%)	79.28

Table 6. 4: Random Forest Results

The comparison of the performance of the CNN model with the other machine learning algorithms clearly depicts the greater performance the CNN model. This is attributed to the mode's ability to learn the complex features used in the classification of the fungal diseases.

Chapter 7: Conclusions, Recommendations and Future Work

7.1 Conclusion

The objective of this study was to develop a deep learning model for the classification of strawberry fungal leaf diseases. This was important in enabling the farmers to efficiently detect these diseases hence minimizing the losses incurred by applying the appropriate control. The disease classes covered include Leaf Scorch, Leaf Blight, Leaf Spot a case where leaf scorch and Leaf Blight occur simultaneously. The study was based on a set of milestones that enabled it to meet the set objectives(in section 1.3.2) summarized as follows:

The first objective was to analyze strawberry fungal leaf diseases affecting strawberry farming in Kenya. This objective was achieved by a theoretical review of the relevant literature indicating the prevalent disease classes followed by an analysis of the disease characteristics and the various features used in their identification. As illustrated in section 2.2 and 2.3, the study found out that the most prevalent classes of these disease are Strawberry Leaf Scorch, Leaf Blight and Leaf Spot. There are also cases where more than a single class of the diseases occur together for instance strawberry Leaf Blight and Leaf Scorch. It was also evident that observable leaf features play an important role in the detection of these diseases as they exhibit different features. The use of these features is advantageous as it only requires a device for capturing the images for analysis and prediction by the model. There is no need of for special reagents or Lab conditions for carrying out the detection.

The second objective was to review the effectiveness of the current methods used for Strawberry fungal leaf disease classification. The study reviewed both the traditional and the deep learning based computer vision techniques for strawberry fungal leaf disease classification. The review formed a basis of the study as it assisted in formulating a clear research gap on which the study was based. This is as shown in section 2.5. The study found out the detection methods did not consider all the prevalent classes of the fungal diseases. They did not also consider a case where several disease classes occur simultaneously.

The third objective was to develop a model for classifying strawberry fungal leaf diseases using deep learning techniques. The deep learning model was developed based on the Agile methodology and on an experimental research design. The development environment was on google Colab, a google

interactive python platform that offers a free access to the GPU. The model development parameters were as follows: Number of epochs-30, Learning Rate of 0.001 and a default image size of 256 by 256. The system modelling process is illustrated in chapter 4, where several designs such the use case diagram, sequence diagram and the class diagrams were developed. The actual implementations of the various components of the model are outlined in chapter 5.

The last objective was to validate the performance of the deep learning model for classifying the strawberry fungal leaf diseases. A define python function called predict_fungal_disease()with two parameters(the model and the image path/URL) was used to implement the validation. The function takes an image path as input and returns the fungal disease label. The model achieved an accuracy of 98%. Other metrics used to validate the model include precision, recall and the F1 -score. The results were as follows: Precision-97%, Recall-95.9% and F1-score-96.3%. A comparison of the CNN model's performance with that of other machine learning models clearly shows the model's ability to model complex image features for classification.

7.2 Research Contribution

The research made the following Contributions:

The developed deep learning model provides a tool for farmers for efficient classification of the various strawberry fungal leaf diseases. This is important in minimizing the losses incurred as a result of these diseases. It also minimizes the time and expenses incurred in seeking assistance from the experts in these diseases.

The model's ability in detecting multiple diseases in a single leaf(Leaf Blight and Leaf Scorch) is also a great contribution especially for instances where these diseases occur simultaneously. This is important in applying the correct measure to both of the disease classes to avoid a situation where only one disease is controlled while the other one damages the crops.

The primary data collected in Kinangop also provides additional data to the available data repository(i.e. PlantVillage repository available on this link: https://github.com/spMohanty/PlantVillage-Dataset) for future research in the same field.

7.3 Recommendations

Based on the research findings, the researcher recommends combining the results of this study with the existing solutions in order to enhance stability in strawberry fungal leaf disease detection. The model can also be deployed in a mobile application for easier access by the farmers to use in their farms.

7.4 Limitations of the Research

i. The model did not consider other parts of the plant where the diseases would occur. The focus was on the leaf as the diseases are most prevalent on this part and most of the data available consisted of leaf images. The model cannot therefore be used to detect diseases that affect other parts of strawberry.

ii. The model does not provide solutions to manage the strawberry fungal diseases. The solutions to manage the diseases are easily available. The major problem lies in correctly classifying the fungal diseases.

iii. Some classes of the image dataset such as Leaf Blight, Leaf Spot and the case where both Leaf Scorch and Leaf Blight occur did not have many images as earlier anticipated. Data augmentation was used to generate more training samples for the model.

iv. The detection of more than one disease class in the same image depends on the positioning of the leaf. The model performs well if the leaf is well exposed i.e. the infected areas are both well exposed to the camera.

v. The model does not recognize other images presented to it such as images of people, places or images from plants other than strawberry.

7.5 Challenges Encountered

It was a challenge to get more quality images within the stipulated research time frame. This is especially for the case where two or more diseases occur simultaneously on the same leaf. There was no control over the cloud resources offered on google Colab such as the GPU. In most instances, loss of access to the GPU would always force the model to be retrained especially if it occurs during model training.

7.6 Future Work

The limitations of this study forms the basis for future work. The researcher suggests the following for future work:

i. Train the model with a robust dataset especially for the class with more than one disease. This will enable the model to learn more on the overlapping features of these diseases hence perform better in classifying them.

ii. The model should be enhanced with the best practices to manage the strawberry fungal leaf diseases. This entails a recommendation on the best control measures for the diseases. This will increase accessibility of recommend solutions after disease detection.

iii. More data should be collected especially on the instances where more than one disease occur on the same part. This will enhance research on this area as there are few data repositories with such instances.

iv. The severity of strawberry fungal leaf diseases changes with the passage of time. It is important to classify the strawberry fungal leaf diseases at earlier stages for efficient control. There is therefore need to modify/improve the training of these models on such basis.

v. Train the model with an additional class of images other than those of strawberry. This will enable the model to identify these images when presented to it.

REFERENCES

Atabay, H. A. (2016). A convolutional neural network with a new architecture applied on leaf classification. *IIOAB J*, *7*(5), 226-331.

Arel, I., Rose, D. C., & Karnowski, T. P. (2010). Deep machine learning-a new frontier in artificial intelligence research [research frontier]. *IEEE computational intelligence magazine*, *5*(4), 13-18.

Barbedo, J. G. A. (2013). Digital image processing techniques for detecting, quantifying and classifying plant diseases. *SpringerPlus*, *2*(1), 660.

Banerjee, A., Roy, S., Sharma, S. K., Dutta, S. K., Chandra, S., & Ngachan, S. V. (2016). Reverse transcription loop-mediated isothermal amplification (RT-LAMP) assay for rapid diagnosis of chilli veinal mottle virus. *Archives of virology*, *161*(7), 1957-1961.

Brahimi, M., Arsenovic, M., Laraba, S., Sladojevic, S., Boukhalfa, K., & Moussaoui, A. (2018). Deep learning for plant diseases: detection and saliency map visualisation. In *Human and machine learning* (pp. 93-117). Springer, Cham.Brynjolfsson, E., & McAfee, A. (2017). What's driving the machine learning explosion. *Harvard Business Review*, *18*.

Bracha, G., & Ungar, D. (2004). Mirrors: design principles for meta-level facilities of object-oriented programming languages. *ACM SIGPLAN Notices*, *39*(10), 331-344.

Bjorck, N., Gomes, C. P., Selman, B., & Weinberger, K. Q. (2018). Understanding batch normalization. In *Advances in Neural Information Processing Systems* (pp. 7694-7705).

Chohan, M., Khan, A., Chohan, R., Katpar, S. H., & Mahar, M. S. Plant Disease Detection using Deep Learning.

Dennis, A., Wixom, B., & Tegarden, D. (2015). *Systems analysis and design: An object-oriented approach with UML*. John wiley & sons.

Dennis, A. R., & Valacich, J. S. (2001). Conducting experimental research in information systems. *Communications of the association for information systems*, *7*(1), 5.

Devikar, P. (2016). Transfer Learning for Image Classification of various dog breeds. *International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)*, *5*(12), 2707-2715.

Dye, D. W., & Wilkie, J. P. (1973). Angular leafspot of strawberry in New Zealand. *New Zealand Journal of Agricultural Research*, *16*(3), 311-314.

Egesa, A. O., Njeri, N., Matheri, F., Mwirigi, P., & Mwangi, M. 3.3 Challenges facing strawberry farming in central Kenya. In *2nd biennial international conference on enhancing sustainable agricultural production and marketing systems* (vol. 38, no. 1, p. 133).

Ellis, M. A., Wilcox, W. F., & Madden, L. V. (1998). Efficacy of metalaxyl, fosetyl-aluminum, and straw mulch for control of strawberry leather rot caused by Phytophthora cactorum. *Plant disease*, *82*(3), 329-332.

Fang, T., Chen, P., Zhang, J., & Wang, B. (2020). Crop leaf disease grade identification based on an improved convolutional neural network. *Journal of Electronic Imaging*, *29*(1), 013004.

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

Fulton, R. H. (1958). Studies on strawberry leaf spot in Michigan. *Michigan Expt Sta Quart Bull*, *40*, 581-8.

Géron, A. (2019). *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems*. O'Reilly Media.

Goel, R., Kumar, V., Srivastava, S., & Sinha, A. K. (2017). A review of feature extraction techniques for image analysis. *International Journal of Advanced Research in Computer and Communication Engineering*, *6*(2), 153-155.

LEGER, R. G., & University of Southern California. (1979). Research Findings and Theory as a Function of Operationalization of Variables: A Comparison of Four Identification Techniques for the Construct, 'Inmate Social Type'. *Sociology and Social Research*, *63*, 346-363.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*.

Ilango, M., & Kumar, A. S. (2017, February). Deterministic multicast link based energy optimized routing in MANET. In *2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT)* (pp. 1-10). IEEE.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

Khadidos, A., Sanchez, V., & Li, C. T. (2017). Weighted level set evolution based on local edge features for medical image segmentation. *IEEE Transactions on Image Processing*, *26*(4), 1979-1991.

Korir, J. K., Affognon, H. D., Ritho, C. N., Kingori, W. S., Irungu, P., Mohamed, S. A., & Ekesi, S. (2015). Grower adoption of an integrated pest management package for management of mango-infesting fruit flies (Diptera: Tephritidae) in Embu, Kenya. *International Journal of Tropical Insect Science*, *35*(2), 80-89.

Kothari, C. R. (2004). *Research methodology: Methods and techniques*. New Age International.

Kiani, E., & Mamedov, T. (2017). Identification of plant disease infection using soft-computing: Application to modern botany. *Procedia computer science*, *120*, 893-900.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, *60*(6), 84-90.

Kusumandari, D. E., Adzkia, M., Gultom, S. P., Turnip, M., & Turnip, A. (2019, July). Detection of Strawberry Plant Disease Based on Leaf Spot Using Color Segmentation. In *Journal of Physics: Conference Series* (Vol. 1230, No. 1, p. 012092). IOP Publishing.

Langley, P., & Simon, H. A. (1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*(11), 54-64.

Mel'nik, V. A. (2000). *Key to the fungi of the genus Ascochyta Lib. (Coelomycetes)*. Parey.

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

Mwangi, M., & Mwaura, S. (2009). Challenges in strawberry seedling production in Kenya. *Journal of Applied Biosciences*, *23*, 1403-1405.

Mwangi, M., Egesa, A., & Matheri, F. (2016, August). Strategies to increase strawberry competitiveness among fruit growers, marketers and consumers in Kenya. In *VIII International Strawberry Symposium 1156* (pp. 921-928).

Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 2018). San Francisco, CA: Determination press.

O'Mahony, N., Campbell, S., Carvalho, A., Harapanahalli, S., Hernandez, G. V., Krpalkova, L., ... & Walsh, J. (2019, April). Deep learning vs. traditional computer vision. In *Science and Information Conference* (pp. 128-144). Springer, Cham.

Pavlenko, I., Trojanowska, J., Ivanov, V. and Liaposhchenko, O., 2018, June. Scientific and methodological approach for the identification of mathematical models of mechanical systems by using artificial neural networks. In *International Conference on Innovation, Engineering and Entrepreneurship* (pp. 299-306). Springer, Cham.

Piquerez, S. J., Harvey, S. E., Beynon, J. L., & Ntoukakis, V. (2014). Improving crop disease resistance: lessons from research on Arabidopsis and tomato. *Frontiers in Plant Science*, *5*, 671.

Robert, C. (2014). Machine learning, a probabilistic perspective.

Sergeev, A., & Del Balso, M. (2018). Horovod: fast and easy distributed deep learning in TensorFlow. *arXiv preprint arXiv:1802.05799*.

Shahid, S. A., & Al-Shankiti, A. (2013). Sustainable food production in marginal lands—Case of GDLA member countries. *International soil and water conservation research*, *1*(1), 24-38.

Shruthi, U., Nagaveni, V., & Raghavendra, B. K. (2019, March). A review on machine learning classification techniques for plant disease detection. In *2019 5th International Conference on Advanced Computing & Communication Systems (ICACCS)* (pp. 281-284). IEEE.

Szandala, E. S., & Backhouse, D. (2001). Suppression of sporulation of Botrytis cinerea by antagonists applied after infection. *Australasian Plant Pathology*, *30*(2), 165-170.

Stemler, S. E. (2015). Content analysis. *Emerging trends in the social and behavioral sciences: An Interdisciplinary, Searchable, and Linkable Resource*, 1-14.

Taylor, L., & Nitschke, G. (2017). Improving deep learning using generic data augmentation. *arXiv preprint arXiv:1708.06020*.

Uselis, N., Valiuškaitė, A., & Raudonis, L. (2006). Incidence of fungal leaf diseases and phytophagous mites in different strawberry cultivars. *Agron. Res*, *4*, 421-425.

Vasar, C., Szeidert, I., Filip, I., & Prostean, G. (2018). Short Term Electric Load Forecast with Artificial Neural Networks. *arXiv preprint arXiv:1804.06660*.

Weller, S. A., Beresford-Jones, N. J., Hall, J., Thwaites, R., Parkinson, N., & Elphinstone, J. G. (2007). Detection of Xanthomonas fragariae and presumptive detection of Xanthomonas arboricola pv. fragariae, from strawberry leaves, by real-time PCR. *Journal of microbiological methods*, *70*(2), 379-383.

Yin, Z., Wan, B., Yuan, F., Xia, X., & Shi, J. (2017). A deep normalization and convolutional neural network for image smoke detection. *Ieee Access*, *5*, 18429-18438.

Appendices

Appendix A: Data Standardization

In [1]:
```
#The process of data standardization entails the conversion of the sample images to a standard dimnesion of 256 X 256
```

In [2]:
```
#Standardization of the strawberry healthy leaves class
#Importing the necessary Libraries
import PIL
import os
import os.path
from PIL import Image
```

In [4]:
```
Imageset1=r"D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Augmented Dataset\Strawberry_Healthy" # Passing the image directory
for file in os.listdir(Imageset1):# Iterating through the image files in the folder
    stdimg1=Imageset1+"/"+file
    img=Image.open(stdimg1)
    img=img.resize((256,256))# Resizing the images
    img.save(stdimg1) #Saving the images
```

In [5]:
```
#Standardization of the strawberry Leaf Blight class
Imageset2=r"D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Augmented Dataset\Strawberry_Leaf_Blight"# Passing the image directo
for file in os.listdir(Imageset2):# Iterating through the image files in the folder
    stdimg2=Imageset2+"/"+file
    img=Image.open(stdimg2)
    img=img.resize((256,256))# Resizing the images
    img.save(stdimg2) #Saving the images
```

In [6]:
```
#Standardization of the strawberry Leaf Scorch class
Imageset3=r"D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Augmented Dataset\Strawberry_Leaf_Scorch"# Passing the image directo
for file in os.listdir(Imageset3):# Iterating through the image files in the folder
    stdimg3=Imageset3+"/"+file
    img=Image.open(stdimg3)
    img=img.resize((256,256))# Resizing the images
    img.save(stdimg2) #Saving the images
```

In [10]:
```
#Standardization of the strawberry Leaf Scorch and Leaf Blight class
Imageset4=r"D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Augmented Dataset\Strawberry_Leaf_Scorch_and_Leaf_Blight"# Passing t
for file in os.listdir(Imageset4):# Iterating through the image files in the folder
    stdimg4=Imageset4+"/"+file
    img=Image.open(stdimg4)
    img=img.resize((256,256))# Resizing the images
    img.save(stdimg2) #Saving the images
```

In [11]:
```
#Standardization of the strawberry Leaf Spot class
Imageset5=r"D:\ACADEMICS\MSc IT\Research Work\Thesis\Dataset\Augmented Dataset\Strawberry_Leaf_Spot"# Passing the image directory
for file in os.listdir(Imageset5):# Iterating through the image files in the folder
    stdimg5=Imageset5+"/"+file
    img=Image.open(stdimg5)
    img=img.resize((256,256))# Resizing the images
    img.save(stdimg2) #Saving the images
```

Appendix B: Python script for loading dataset and conversion to numpy array

```
image_list, label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(root_directory)
    for directory in root_dir :
        # remove .DS_Store from list
        if directory == ".DS_Store" :
            root_dir.remove(directory)

    for plant_folder in root_dir :
        plant_disease_folder_list = listdir(f"{root_directory}/{plant_folder}")

        for disease_folder in plant_disease_folder_list :
            # remove .DS_Store from list
            if disease_folder == ".DS_Store" :
                plant_disease_folder_list.remove(disease_folder)

        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            plant_disease_image_list = listdir(f"{root_directory}/{plant_folder}/{plant_disease_folder}/")

            for single_plant_disease_image in plant_disease_image_list :
                if single_plant_disease_image == ".DS_Store" :
                    plant_disease_image_list.remove(single_plant_disease_image)

            for image in plant_disease_image_list[:Default_Number_of_Images]:
                image_directory = f"{root_directory}/{plant_folder}/{plant_disease_folder}/{image}"
                if image_directory.endswith(".jpg") == True or image_directory.endswith(".JPG") == True:
                    image_list.append(convert_image_to_array(image_directory))
                    label_list.append(plant_disease_folder)
#Converting the loaded images into a numpy array
np_image_list = np.array(image_list, dtype=np.float16) / 225.0
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")
```

```
[INFO] Loading images ...
[INFO] Processing Strawberry_Healthy ...
[INFO] Processing Strawberry_Leaf_Blight ...
[INFO] Processing Strawberry_Leaf_Scorch_and_Leaf_Blight ...
[INFO] Processing Strawberry_Leaf_Spot ...
[INFO] Processing Strawberry_Leaf_Scorch ...
[INFO] Image loading completed
```

Appendix C: CNN Model Configuration

```
#The model configurartion
model = Sequential()
inputShape = (height, width, depth)
chanDim = -1
if K.image_data_format() == "channels_first":
    inputShape = (depth, height, width)
    chanDim = 1
model.add(Conv2D(32, (3, 3), padding="same",input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(3, 3)))
model.add(Dropout(0.25))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(64, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(Conv2D(128, (3, 3), padding="same"))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=chanDim))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(1024))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(n_classes))
model.add(Activation("softmax"))
```

Appendix D: The CNN Model Summary

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 256, 256, 32)      896
_________________________________________________________________
activation (Activation)      (None, 256, 256, 32)      0
_________________________________________________________________
batch_normalization (BatchNo (None, 256, 256, 32)      128
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 85, 85, 32)        0
_________________________________________________________________
dropout (Dropout)            (None, 85, 85, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 85, 85, 64)        18496
_________________________________________________________________
activation_1 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_1 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 85, 85, 64)        36928
_________________________________________________________________
activation_2 (Activation)    (None, 85, 85, 64)        0
_________________________________________________________________
batch_normalization_2 (Batch (None, 85, 85, 64)        256
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 42, 42, 64)        0
_________________________________________________________________
dropout_1 (Dropout)          (None, 42, 42, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 42, 42, 128)       73856
_________________________________________________________________
activation_3 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_3 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 42, 42, 128)       147584
_________________________________________________________________
activation_4 (Activation)    (None, 42, 42, 128)       0
_________________________________________________________________
batch_normalization_4 (Batch (None, 42, 42, 128)       512
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 21, 21, 128)       0
dropout_2 (Dropout)          (None, 21, 21, 128)       0
_________________________________________________________________
flatten (Flatten)            (None, 56448)             0
_________________________________________________________________
dense (Dense)                (None, 1024)              57803776
_________________________________________________________________
activation_5 (Activation)    (None, 1024)              0
_________________________________________________________________
batch_normalization_5 (Batch (None, 1024)              4096
_________________________________________________________________
dropout_3 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 5)                 5125
_________________________________________________________________
activation_6 (Activation)    (None, 5)                 0
=================================================================
Total params: 58,092,421
Trainable params: 58,089,541
Non-trainable params: 2,880
_________________________________________________________________
```

Appendix E: Interview Guide

Introduction

Dear Respondent,

This interview questionnaire is part of study conducted by Deperias Kerre as part of the requirements for the award of a degree of Master of Science in Information Technology at Strathmore University. The main aim of this study is to come up with a model for the detection of strawberry fungal leaf diseases. The knowledge that we will gain from your responses will aid in the development of model in such a way that it will be of more use to the farmers and other users. The information requested will only be used for academic purposes and will be treated with at most confidentiality.

N/B: Feel free to handwrite or type in the Responses.

Kind Regards,

Deperias Kerre

PART A: Basic Functionalities

a. What do you expect to supply to the model as input for prediction and what is the expected outcome?

PART B: Other Functionalities

b. List any other preferences that will improve the experience of using the model or make the model more convenient to use.

Your assistance will be highly appreciated.

Appendix F: Sample Responses

Introduction

Dear Respondent,

This interview questionnaire is part of study conducted by Deperias Kerre as part of the requirements for the award of a degree of Master of Science in Information Technology at Strathmore University. The main aim of this study is to come up with a model for the detection of strawberry fungal leaf diseases. The knowledge that we will gain from your responses will aid in the development of model in such a way that it will be of more use to the farmers and other users. The information requested will only be used for academic purposes and will be treated with at most confidentiality.

N/B: Feel free to handwrite or type in the Responses.

Kind Regards,

Deperias Kerre

PART A: Basic Functionalities

a. What do you expect to supply to the model as input for prediction and what is the expected outcome?

-I expect the model to ask or prompt me for an image as input. I also expect a notification if the image is successfully uploaded.

-I also expect the model to return the strawberry fungal disease type in simple statement e.g its name.

PART B: Other Functionalities

b. List any other preferences that will improve the experience of using the model or make the model more convenient to use.

-The model should be easy to use especially for some of us who are not IT savvy.

-The model should have minimal cost implications in terms of storage space.

-It should be secure to use e.g not interfere with personal data.

-The model should be easy to maintain or support in case of errors.

-The model should accept images from a wide range of cameras and convert them to the required format.

Your assistance will be highly appreciated.

Appendix G: Model Deployment to a Simple UI Using the Flask API

D:\ACADEMICS\MSc IT\Research Work\Thesis\Thesis Demo App\web_app.py - Notepad++ [Administrator]

File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?

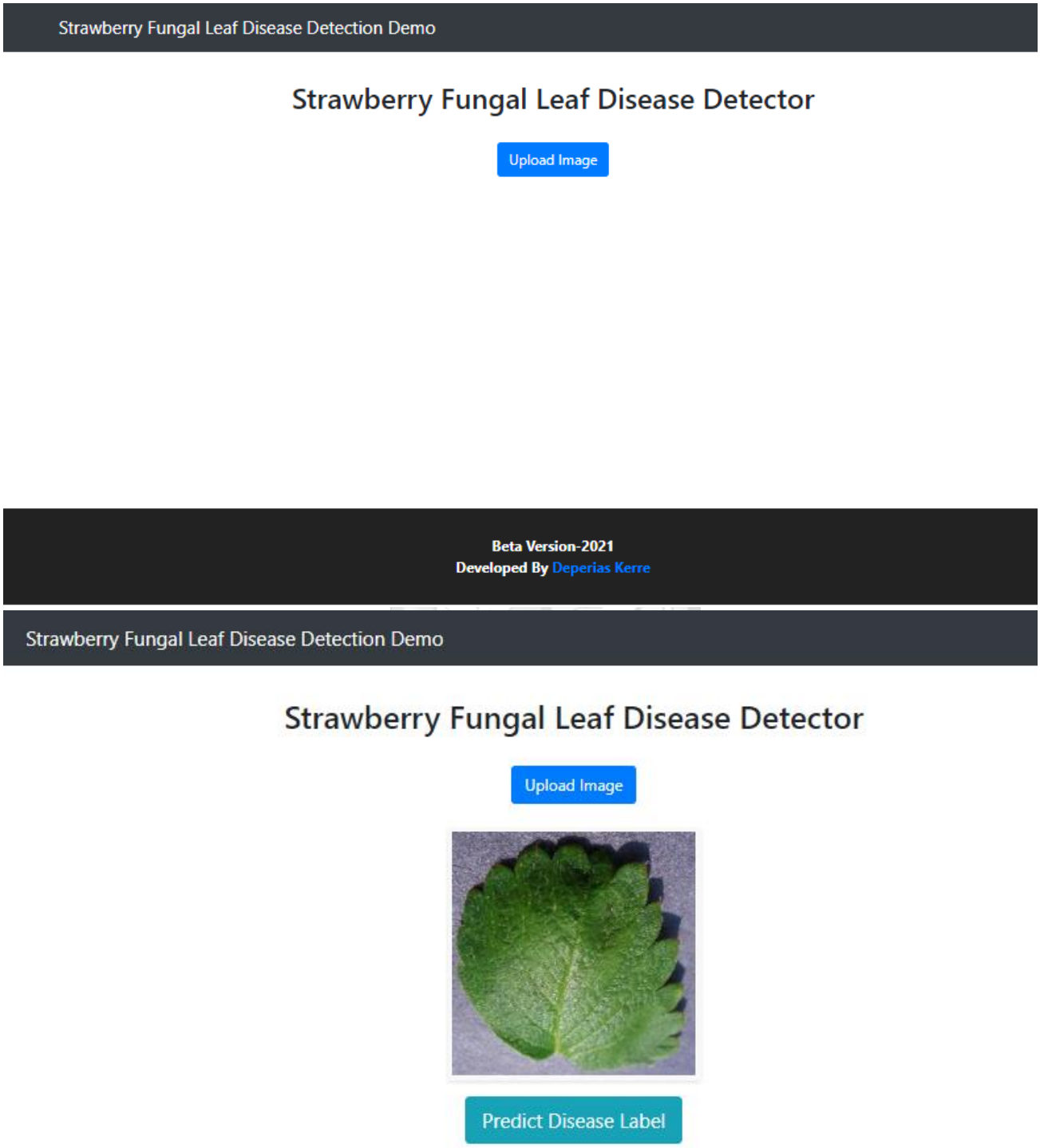
web_app.py | import.html | index.html

```
from __future__ import division, print_function
# coding=utf-8
import sys
import os
import glob
import re
import numpy as np
# Keras
from keras.applications.imagenet_utils import preprocess_input, decode_predictions
from keras.models import load_model
from keras.preprocessing import image
import pickle
import cv2
import tensorflow as tf
# Flask utils
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
from gevent.pywsgi import WSGIServer
# Define a flask app
app = Flask(__name__)
# Model saved with Keras model.save()
MODEL_PATH = "D:/ACADEMICS/MSc IT/Research Work/Thesis/Strawberry_Fungal_Disease_Classification_Model.h5"
#Loading the labels transform
filename = 'D:/ACADEMICS/MSc IT/Research Work/Thesis/fungal_disease_label_transform.pkl'
image_labels=pickle.load(open(filename, 'rb'))
#Load your trained model
model = tf.keras.models.load_model(MODEL_PATH)
```

```
#Function to Predict the disease label
def predict_fungal_disease(img_path,model):
    img = image.load_img(img_path, target_size=(256, 256))
    image_array = image.img_to_array(img)
    np_image = np.array(image_array, dtype=np.float16) / 225.0
    np_image = np.expand_dims(np_image,0)
    result = model.predict_classes(np_image)
    disease_label=(image_labels.classes_[result][0])
    return disease_label
@app.route('/', methods=['GET'])
def index():
    # Main page
    return render_template('index.html')
@app.route('/predict', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        # Get the file from post request
        f = request.files['file']

        # Save the file to ./uploads
        basepath = os.path.dirname(__file__)
        file_path = os.path.join(
            basepath, 'uploads', secure_filename(f.filename))
        f.save(file_path)
        #Making Prediction
        prediction=predict_fungal_disease(file_path,model)
        return prediction
    return None
if __name__ == '__main__':
    app.run(port=5002, debug=True)
```

Appendix H: Sample Web-Based UI



Appendix I: Strathmore University Ethical Review Board Approval

Strathmore UNIVERSITY

21st January 2021

Mr Kerre, Deperias
deperias.kerre@strathmore.edu

Dear Mr Kerre,

RE: A Deep Normalized Neural Network Model for Strawberry Fungal Leaf Disease Detection

This is to inform you that SU-IERC has reviewed and **approved** your above research proposal. Your application reference number is **SU-IERC0938/20.** The approval period is **21st January 2021 to 20th January 2022.**

This approval is subject to compliance with the following requirements:

i. Only approved documents including (informed consents, study instruments, MTA) will be used
ii. All changes including (amendments, deviations, and violations) are submitted for review and approval by SU-IERC.
iii. Death and life-threatening problems and serious adverse events or unexpected adverse events whether related or unrelated to the study must be reported to SU-IERC within 48 hours of notification
iv. Any changes, anticipated or otherwise that may increase the risks or affected safety or welfare of study participants and others or affect the integrity of the research must be reported to SU-IERC within 48 hours
v. Clearance for export of biological specimens must be obtained from relevant institutions.
vi. Submission of a request for renewal of approval at least 60 days prior to expiry of the approval period. Attach a comprehensive progress report to support the renewal.
vii. Submission of an executive summary report within 90 days upon completion of the study to SU-IERC.

Prior to commencing your study, you will be expected to obtain a research license from National Commission for Science, Technology and Innovation (NACOSTI) https://oris.nacosti.go.ke and also obtain other clearances needed.

Yours sincerely,

Dr Virginia Gichuru,
Secretary; SU-IERC

Cc: Prof Fred Were,
Chairperson; SU-IERC

STRATHMORE UNIVERSITY INSTITUTIONAL ETHICS REVIEW COMMITTEE
21 JAN 2021
TEL: +254 (0)703 034 000
P. O. Box 59857 - 00200
NAIROBI - KENYA

Appendix J: NACOSTI Research Permit/License

REPUBLIC OF KENYA

NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Ref No: **289663**

Date of Issue: **23/April/2021**

RESEARCH LICENSE

This is to Certify that Mr.. Deperias Webula Kerre of Strathmore University, has been licensed to conduct research in Nairobi, Nyandarua on the topic: A Deep Normalized Neural Network Model for Strawberry Fungal Leaf Disease Detection for the period ending : 23/April/2022.

License No: **NACOSTI/P/21/10111**

289663

Applicant Identification Number

Director General
NATIONAL COMMISSION FOR SCIENCE,TECHNOLOGY & INNOVATION

Verification QR Code

NOTE: This is a computer generated License. To verify the authenticity of this document, Scan the QR Code using QR scanner application.

Appendix K: Similarity Report

Curiginal

Document Information

Analyzed document	A Deep Normalized Neural Network Model for Strawberry Fungal Leaf Disease Detection.docx (D109940825)
Submitted	6/30/2021 11:21:00 AM
Submitted by	
Submitter email	Deperias.Kerre@strathmore.edu
Similarity	3%
Analysis address	library.strath@analysis.urkund.com