SUNYANI TECHNICAL UNIVERSITY
FACULTY OF APPLIED SCIENCE AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE

DEVELOPING SELECTED CABBAGE CROP DISEASE DETECTION MODEL USING CONVOLUTIONAL NEURAL NETWORK

BY

DESMOND DEVEH
Co-Authors
AKUKA IDDRISU AZIMBE
ALOLBILA SIXTUS AKURUGU
APEWE SULEMANA

SEPTEMBER, 2022

DEVELOPING SELECTED CABBAGE CROP DISEASE DETECTION MODEL USING CONVOLUTIONAL NEURAL NETWORK

BY

DESMOND DEVEH

AKUKA IDDRISU AZIMBE

ALOLBILA SIXTUS AKURUGU

APEWE SULEMANA

PROJECT SUBMITTED TO THE DEPARTMENT OF COMPUTER SCICENCE, FACULTY OF APPLIED SCIENCE AND TECHNOLOGY IN PARTIAL FULFILMENT FOR THE REQUIREMENTS OF THE AWARD OF HIGHER NATIONAL DIPLOMA IN (HND) COMPUTER SCIENCE

SEPTEMBER, 2022

DECLARATION

STUDENT'S DECLARATION

We hereby declare that this submission is our own work towards the award of Higher National Diploma and that to the best of our knowledge, it contains no material previously published by another person nor material which has been accepted for the award of any other. Degree/Diploma of the University except where the acknowledgement has been made in the text.

Name of Student	Index Number
DEVEH DESMOND	06190315
AKUKA IDDRISU AZIMBE	06190302
ALOLBILA SIXTUS AKURUGU	06190306
APEWE SULEMANA	06190307

CERTIFICATION

Supervisor's Certification

I hereby declare that the preparation and presentation of this research work were supervised per the guidelines on supervision of research work laid down by Sunyani Technical University.

Signature………………………………………………..Date…………………………

DR. BEN BEKLISI KWAME AYAWLI

(Supervisor)

Head of Department's Certification

I hereby certify that a copy of this project work has been submitted to the Department of Computer Science for keeping for record and reference purposes.

Signature…………………………………………….Date………………………………

DR. BEN BEKLISI KWAME AYAWLI

(Head of Department)

DEDICATION

This work is dedicated to God Almighty for seeing us through a successful academic journey. We are much grateful.

ACKNOWLEDGEMENT

Foremost, we would like to express our deepest and sincere gratitude to our supervisor Dr. Ben B. K. Ayawli for the continuous support of our project, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped us in all the time of research and writing of this project. We could not have imagined finishing our project without his support and encouragement.

We are highly indebted to our Head of Department Dr. Ben B. K. Ayawli for providing us this opportunity to prepare this project. We would also like to thank the entire staff of Computer Science Department. Being final year students, making of the project adds a final touch to our career. We express our sincere thanks to all.

Last but not the least, we would like to express our deepest gratitude to our parents for their financial support and prayers.

ABSTRACT

Developing countries such as Ghana have resources that are ideal to producing crops of various varieties. Cabbage is one of the crops that is grown in large scale. Nevertheless, the yield per unit area of cabbage in Ghana is very small compared to other countries. Some major diseases that affect the cabbage crops in Ghana is Alternaria leaf spot and Black rot. Most researchers used traditional machine learning algorithms which require a handcrafted feature extraction to classify a given image that leads to poor classification accuracy results. In this project, a model was developed using CNN algorithm, and the algorithm is capable of automatically extracting features, unlike other algorithms. Dataset made up of images of diseased and healthy cabbage crop were collected from Kaggle database and cabbage crop field. InceptionV3 was used in the pretrained model for the experimentation. Evaluation of the proposed model was conducted using Python. Evaluation results demonstrates accurate detection rate of 99.85%. The proposed model is significant in detecting cabbage crop diseases to aid in providing the appropriate solution leading to reduced losses to farmers.

Keywords: Convolutional Neural Network; Plant Disease Detection; Cabbage crop; Alternaria leaf spot; Black rot.

TABLE OF CONTENTS

STUDENT'S DECLARATION i
CERTIFICATION ii
DEDICATION iii
ACKNOWLEDGEMENT iv
ABSTRACT v
LIST OF FIGURES viii
LIST OF TABLES ix
ABBREVIATIONS x
CHAPTER ONE 1
INTRODUCTION 1
1.1. Background of Study 1
1.2 Statement of the Problem 2
1.3 Objective of the Study 3
1.3.1 General Objectives 3
1.3.2 Specific Objectives 3
1.4 Significance of the Study 3
1.5 Scope of Study 4
1.6 Limitation of the Study 4
1.7 Chapter Organization 4
CHAPTER TWO 5
LITERITURE REVIEW 5
2.1 Types of Cabbage Diseases 5
2.2 Pests of Cabbage Crops 7
2.3 Other Techniques used in Machine Learning 8
2.4 Digital Image Processing 9
2.5 Convolutional Neural Network (CNN) 9
2.5.1 Convolutional Layer 10
2.5.2 Pooling Layer 11
2.5.3 Fully Connected Layer (Dense layer) 11
2.6 Activation Function 13
2.7 Conclusion 15

CHAPTER THREE 16
METHODOLOGY 16
3.1 The proposed research model 16
3.1.1 Image Acquisition 17
3.1.2 Image Pre-Processing 17
3.1.3 Training Data 19
3.1.4 Image Segmentation 19
3.1.5 Feature Extractions 20
3.1.6 CNN Algorithm 22
3.1.7 Classification 25
3.1.8 Crop Disease Detection 25
CHAPTER FOUR 26
RESULTS AND DISCUSSION 26
4.1 Simulation Results 26
4.1.1 Training and validation of Accuracy and Loss 26
4.3 Experimental results of the proposed model 29
4.4 Comparison of other researcher's results obtained 31
CHAPTER FIVE 32
SUMMARY, CONCLUSION AND RECONMENTDATION 32
5.1 Summary 32
5.2 Conclusion 32
5.3 Recommendation 33
References 34
APPENDIX 37
SOURCE CODE 37

LIST OF FIGURES

Figure 2.1: Black Rot affected crop 6

Figure 2.2: Affected Alternaria Leaf Spot (Source: George, 2020) 6

Figure 2.3: Aphide Affected Cabbage Crop (Source: George, 2020)................................ 7

Figure 2.4: Diamonback Moth Affected Cabbage Crop (Source: George, 2020) 7

Figure 2.5: Convolution Operation (Source: Yamashita et al., 2018) 10

Figure 2.6: Max Pooling with A Stride of 2 with 2x2 Filter (Arora et al., 2019)............ 11

Figure 2.7: Fully Connected Layer 12

Figure 3.1: Workflow.. 16

Figure 3.2: Region splitting and merging method used in segmentation 20

Figure 3.4 Convolution Operations.. 23

Figure 3.5: Pooling Layer with A Stride of 2 with 2x2 Filter... 24

Figure 3.6: Fully Connected Layer 24

Figure 4.1: Training and validation Accuracy/Loss Epoch 5 27

Figure 4.2: Training and validation Accuracy/Loss Epoch 15 27

Figure 4.3: Training and validation Accuracy and Loss Epoch 30................................... 28

Figure 4.5: Training and validation Accuracy and Loss Epoch 40................................... 28

Figure 4.6: Interface of the model.. 29

Figure 4.7: Alternaria spot cabbage crop 29

Figure 4.8: Black rot cabbage crop affected 30

Figure 4.9: Healthy cabbage crop 30

LIST OF TABLES

Table 4.1 Shows the comparison of others research works 31

ABBREVIATIONS

CNN: Convolutional Neural Network.

ANN: Artificial Neural Network

SDGs: Sustainable Development Goals

SVM: Support Vector Machine

XC: Xanthomonas Campestris

API: Application Programming Interface

CPU: Central Processing Unit

GPU: Graphical Processing Unit

ReLU: Rectify Linear Unit

CE: Cross-Entropy

FC: Fully Connected

IP: Internet Protocol

CHAPTER ONE

INTRODUCTION

1.1. Background of Study

In Ghana, which is mostly an agricultural nation, agriculture supports the majority of the population. Product quality control is fundamentally required in order to get more useful products. Numerous studies show that crop diseases have caused a decline in the quality of agricultural products. The most vital agricultural goods are fruits and vegetables Gavhale & Gawande, (2014)

One of the most popular crops in Ghana is the cabbage crop, which provides income. Only 80% of the overall production in the rural areas is sold in the market, with approximately 20% going to household use. Ghana is also regarded as a crucial commodity for guaranteeing food security due to its nutrients by Lupescu & Zimmerman, (2019).

Diseases are impairments to the normal state of the crop that modify or interrupt its vital functions such as photosynthesis, transpiration, pollination, fertilization, germination by Gavhale & Gawande, (2014)

The main culprits behind cabbage leaf diseases are bacteria, fungi, viruses, etc. To address this, the Convolutional Neural Networks technique is used to classify leaf diseases according to the types of diseases so that farmers can act quickly to minimize the output loss. Affected plants may exhibit colored streaks or patches on their leaves or stems, among other obvious symptoms. These visual signs could be located using image processing methods by Barbedo and Garcia, (2013)

1.2 Statement of the Problem

Cabbage, or Brassica oleraeea Var. capitata, is one of the most widely consumed crops in Ghana and one of the most important vegetables after tomatoes, onions, carrots, and garden eggs. Ghana has a good climate and soil cultivation for many different types of crops, including cabbage, and it grows in different regions with various agro-economic conditions. However, commercial cabbage farming is mainly done in Southern Ghana, specifically in the Akwapem and Kwahu areas. Cabbage is also cultivated in the moist high elevations around Tarkwa Samisco, (2021).

The production of cabbage crops is affected by biotic and abiotic factors. Among those, the major factor is the biotic factor, which is caused by microorganisms like bacteria, fungus, and viruses. Many cabbage diseases affect the growth and production of the crop. To mention some of the diseases, there are Alternaria Leaf Spot, Bacterial Soft Rot, Black Leg, Black Rot, and Virus.

In addition, abiotic factors, which include a number of other variables, influence cabbage yield. To mention some, lack of well-performing cultivars, poor vegetable setting due to heavy rains and excessively high temperatures, and pets. In contrast to current plant disease classification techniques, this model proposes a useful and flexible way of identifying the kinds of diseases in the cabbage crops.

This research aimed at addressing some major challenges in cabbage crop disease detection:

- The poor classification accuracy rate and time, which is a key concern in crop disease detection.
- To detect Alternaria Leaf Spot and Black Rot diseases in cabbage crops.

1.3 Objective of the Study

This section of the report presents objectives of the study. They are grouped into general and specific objectives which are discussed in turn.

1.3.1 General Objectives

The general objective of this research is to address the detection of some popular cabbage crop diseases using convolutional neural network model.

1.3.2 Specific Objectives

To achieve the general objective, the following specific objectives were formulated.

- To utilize Convolutional Neural Network (CNN) method to classify healthy and disease-affected cabbage crop.
- To propose a model to detect cabbage crop diseases using CNN to improve the accuracy rate and execution time.
- To implement a convolutional neural network model using a Python programming language equipped with computer vision model.
- To create a web-based platform to determine if a cabbage crop is affected with Alternaria Leaf Spot and Black Rot diseases in cabbage crops.

1.4 Significance of the Study

This will also bring money into the country, which will help to stimulate the economy

The development of this model will aid in the detection and identification of diseases such as Alternaria Leaf Spot and Black Rot, which plague cabbage farms. It will also assist farmers in reducing production costs and losses. The farmer will have complete control over his or her farm and be able to produce vast quantities of food for human use. This research aims to contribute to the sustainable development goals (SDG's) to end hunger, achieve food security and improved nutrition and promote sustainable agriculture.

1.5 Scope of Study

The study focused primarily on developing an Alternaria Leaf Spot and Black Rot disease detection model using CNN. A healthy cabbage crop, Alternaria Leaf Spot and Black Rot affected leaves images are collected from Kaggle database for the evaluation of the proposed model.

The study does not involve proposing agro-chemicals, effective treatment, or estimating the severity of the disease after it is detected. Moreover, this study is restricted to Alternaria Leaf Spot and Black Rot disease detection.

1.6 Limitation of the Study

It was discovered during the research that low resolutions and sharpness of images of cabbage crops for the evaluation may affect the detection rate of the crop diseases. Detection accuracy may therefore be affected with poor image resolutions.

1.7 Chapter Organization

The project is divided into five chapters. Chapter 1, the introduction, and Chapter 2 is devoted to a survey of the literature on disease detection using deep learning convolutional neural network algorithms. Chapter 3 presents the methodology of the project, which deals with the proposed convolutional neural network model. Chapter 4 presents results and a discussion. Chapter 5 presents a summary, conclusion, and reference of the work.

CHAPTER TWO

LITERITURE REVIEW

This chapter presents an overview of some common cabbage crop disease detection techniques, as well as some of the most recent developments in the field of machine learning.

2.1 Types of Cabbage Diseases

There are numerous varieties of diseases that affect cabbage; however, only a few are covered here:

Black Rot, a disease of cabbage and other crucifer crops around the world, is brought on by the bacterium Xanthomonas campestris pv. campestris (Xcc). Since its discovery on turnips in New York in 1893, the disease has been a frequent issue for growers. The disease flourishes in warm, humid climates and spreads from plant to plant through water splashes, wind-blown water droplets, and personnel or animals traveling from infected areas to healthy ones. When growing transplants in greenhouses or seed beds, Xcc can spread quickly and may already be present long before any symptoms are noticed. The bacterium can invade seeds and infect early seedlings as well. George, (2020). **Figure 2.1** shows a cabbage crop with Black rot disease.

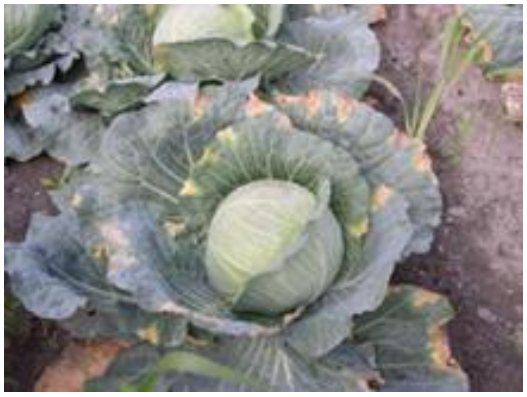


Figure 2.1: Black Rot affected crop Source: George,(2020)

According to George, (2020) Alternaria leaf spot is a type of disease that is a common foliar disease of brassica crops caused by the fungal pathogen Alternaria brassicicola. The disease can be a problem for many crops including cabbage, cauliflower, kale, Brussels sprouts, and broccoli. Even small infections can lead to an unmarketable crop. Severe foliar infections can lead to a reduction in yield due to leaf loss or a reduction in weight.

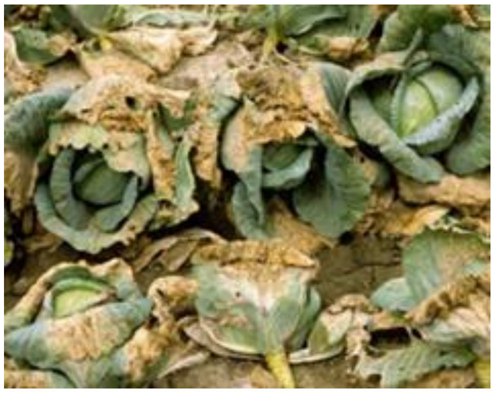


Figure 2.2: Affected Alternaria Leaf Spot Source: George, (2020)

2.2 Pests of Cabbage Crops

Aphids are a pest on cabbage plants that can suck the vital sap from the leaves, leaving the plant weakened, damaged, and susceptible to disease and bug invasions. Growing companion plants nearby will attract ladybugs and other natural predators of aphids.

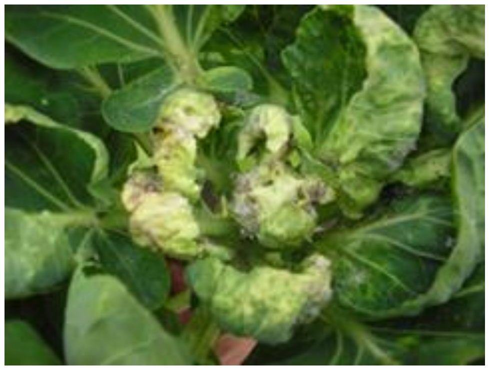


Figure 2.3: Aphide Affected Cabbage Crop Source: George, (2020)

Diamondback moth (dbm). This is one of the most serious pests of cabbage plants grown under comparatively high-temperature conditions. The caterpillars are green or brownish-green. The adult female lays eggs on the leaves, singly or in groups. Small slender green caterpillars on emergence feed on the leaf epidermis and later make holes in the plant leaves.

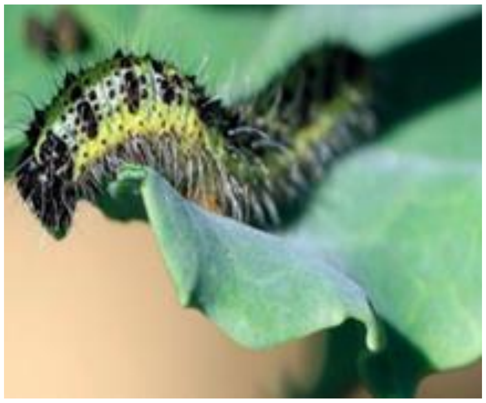


Figure 2.4: Diamonback Moth Affected Cabbage Crop Source: George, (2020)

2.3 Other Techniques used in Machine Learning

According to Heaton, (2018) automated systems for identifying plant diseases utilizing pictures and machine learning, particularly CNNs, have made tremendous progress in increasing the accuracy of the right diagnosis. Convolutional Neural Networks are a type of machine learning approach that has proven to be a useful tool for assimilating vast volumes of heterogeneous data and making accurate predictions about complicated and unpredictable occurrences.

Yamashita et al., (2018) proposed a technique that detects the disease that the paddy crop is suffering from by using CNN and SVM classifiers. Their model uses feature extraction and SVM classifiers for image processing. They have taken 250 images. They have used 50 images for testing and the rest of the 200 images for training the model. They have also developed a mobile app that clicks the image of the infected plant, zooms it, and crops it, then uploads the image, and the person receives a notification.

Badage, (2018) presented a model for the detection of the disease that the plant is suffering from. The author has used a canny edge detection algorithm. The author has used the canny edge detection algorithm to track the edge and get the histogram value to predict the disease that the plant is suffering from. The model also periodically monitors the cultivated field. The model detects the disease in the early stages. Then machine learning is used for training. Then the model takes a proper decision and predicts the disease that the plant is suffering from.

Barik, (2018) proposed a technique for region identification of the rice disease using image processing. The author has proposed a model that not only identifies the disease that the rice plant is suffering from but also identifies the affected region. The author has used image processing and for classification. The author has used machine learning techniques such as

Support Vector machines (SVM). After the prediction is done, the severity of the disease is determined, and then it is classified into different categories.

Kulkarni, et al., (2012) proposed a method for detecting plant diseases early and accurately, using diverse image processing techniques and an artificial neural network (ANN). In their approach, the network is trained on 140 samples, of which 8 samples are Alternaria, 26 samples are BBD, and 89 samples are Anthractnose. These samples are used for training and testing. They discovered that Alternaria recognized samples of 6 diseases with a recognition rate of 75%, BBD recognized samples of 21 diseases with a recognition rate of 80.76%, and Anthractnose recognized samples of 86 diseases with a recognition rate of 96%.

2.4 Digital Image Processing

Today's world of technology is filled with digital images. Simply said, an image is a representation of an object, a person, or a scene. Digital images are represented by the 2D function f(x, y), where x and y stand for the positions of individual pixels with intensity values and indicate the projection of a 3D scene into a 2D projection plane. Vector notation can be used to represent pixel coordinates. Each vector is typically horizontally oriented while its transpose is vertically oriented. Equation (1) by Tyagi, (2018) was used in digital image processing.

$$X = \begin{bmatrix} X \\ Y \end{bmatrix} = [X\ Y]^T = (X, Y) \qquad (1)$$

2.5 Convolutional Neural Network (CNN)

A common neural network used to handle issues with image classification, object detection, face recognition, image recognition, and other issues is the convolutional neural network (ConvNets). The convolutional neural network is widely utilized in a variety of applications, including speech recognition, natural language processing, image processing, picture restoration, and others. When examining objects, CNN can detect and categorize items with little to no pre-processing, resulting in high-quality results, and feature separation with multilayered data is straightforward Tilahun, (2020). CNN has four main layers: convolutional layer, pooling layer, activation function, and fully connected layer.

2.5.1 Convolutional Layer

The convolution layer served as the inspiration for the CNN algorithm. A matrix operation is used in this layer's set of mathematical processes to extract a feature map from the input image Arora et al. (2019). The mathematical operation is depicted in Figure 2.5. First, the filter is shifted step by step starting from the upper left corner of the image. At each step, the values in the image are multiplied by the values of the filter (kernel) and the result is summed. A new matrix with a smaller size is created from the input image.

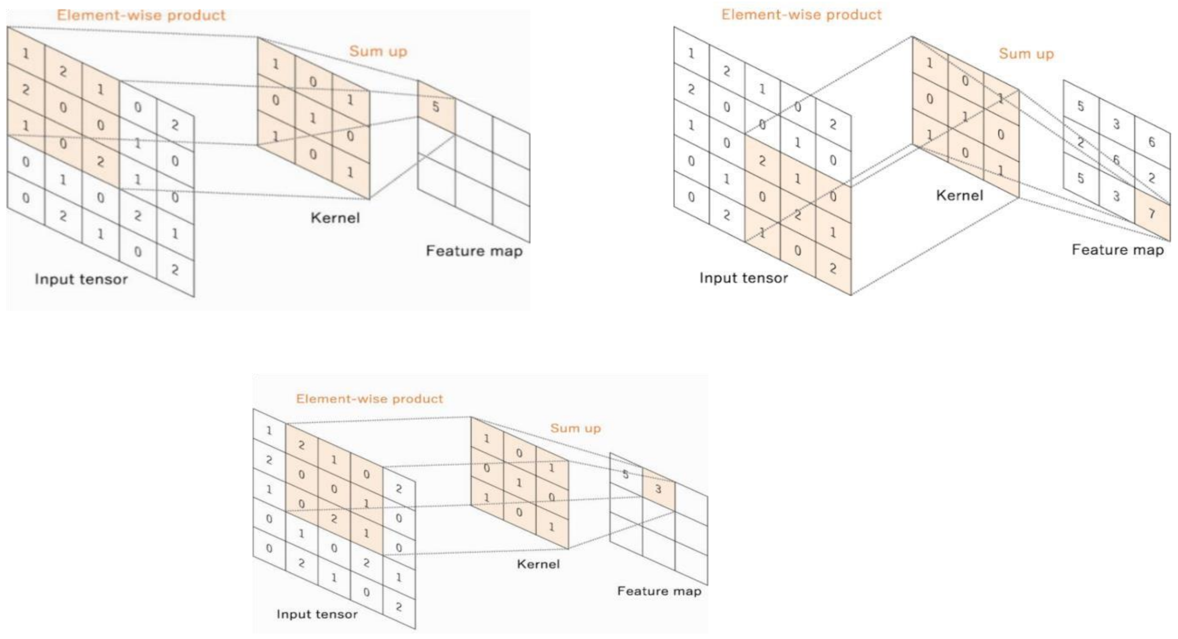


Figure 2.5: Convolution Operation (Source: Yamashita et al., 2018)

2.5.2 Pooling Layer

The pooling layer is the next layer after the convolutional layer operation. It lowers the in-plane dimensionality of feature maps, introduces translation invariance with a slight shift and distortion, and lowers the number of ensuing learnable parameters. A conventional down sampling procedure is carried out before the final layer is added Arora et al., (2019). After an operation is performed in the convolution layer, a feature map is generated as an output. The output of the convolution layer's feature map will be reduced in the pooling layer. Different filter sizes are used in the pooling layer, but usually a 2x2 size filter is used, and also several functions are used to list some max pooling, average pooling, and sum pooling, which are some of the functions used. A max pooling operation is depicted in Figure 2.6. Max pooling is performed by selecting the largest element from the input matrix concerning the filter.

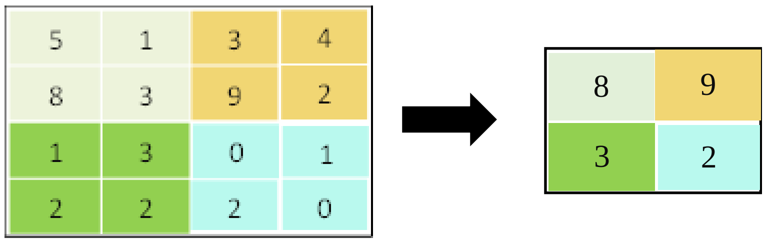


Figure 2.6: Max Pooling with A Stride of 2 with 2x2 Filter Source: Arora et al., (2019)

2.5.3 Fully Connected Layer (Dense layer)

Is a crucial part of CNN that allows a computer vision system to correctly identify and classify an image. The picture breakdown is converted into a feature and then independently examined once the input image has been fed into the CNN algorithm and has passed the convolution and pooling layers. To classify an image, the FC layer flattens the output of the pooling layer Arora et al., (2019).

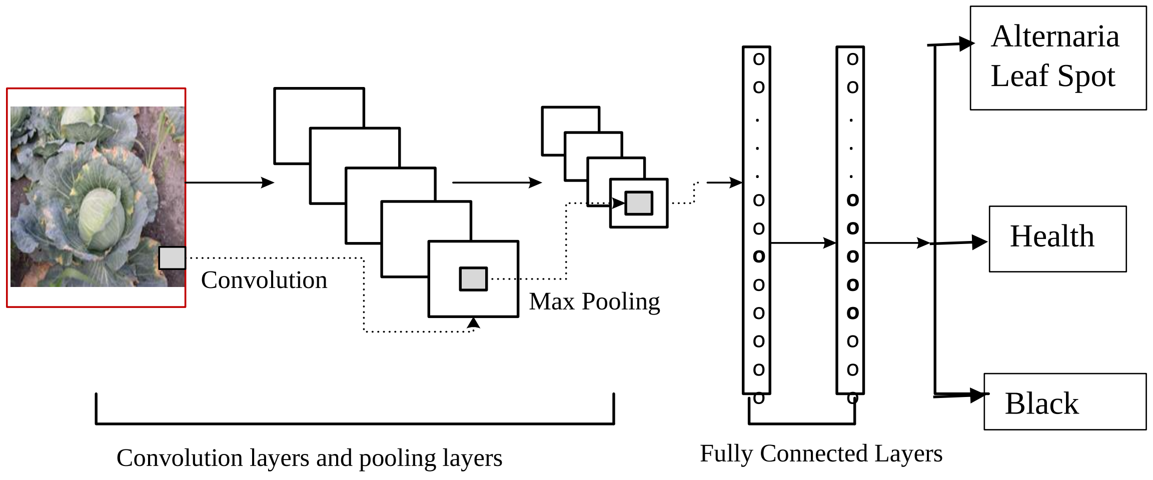


Figure 2.7: Fully Connected Layer

- Keras: Is an API used for different machine learning and deep learning researches. The API is written in Python and capable of running on top of CNTK, TensorFlow, and Theano. Keras helps machine learning researchers transform an idea into results with minimum effort. In this study, Keras' API was used on top of TensorFlow. Some of the advantages of the API are:

1. Easier to use and extensible
2. Support both CPU and GPU
3. Support CNN and RCNN

- TensorFlow: Is one of the most well-known frameworks for data science and machine learning. The API provides a lot of tools, resources, and libraries. These characteristics make it easier for researchers to create and use machine learning applications. Additionally, the API's intuitiveness helps developers to easily develop and train models. Besides, the simple and flexible architecture of TensorFlow helps for fast experimentation and simple transformation of ideas into implementation.

2.6 Activation Function

Is one of the CNN architecture's parameters, and the neural network is used to solve various complex issues like object detection, object recognition, and image categorization by Tilahun, (2020). The function's job is to determine whether or not to fire a function by calculating a weighted sum and adding bias. The learning process of the model can be improved by choosing the appropriate activation function from among the many available types. Among these activation features are: Softmax, ReLU, Dropout, and InceptiveV3.

- **SoftMax:** A vector of K real values is transformed into a vector with a sum of 1 by the SoftMax function. The SoftMax converts the input values, which may be positive, negative, zero, or higher than one, into values between 0 and 1, allowing them to be understood as probabilities. The SoftMax converts small or negative inputs into small probabilities, and big or positive inputs into large probabilities, although it will always fall between 0 and 1. Equation (2) by (Jaso, 2019) was used in SoftMax function.

$$\sigma(z^{\rightarrow})i = \frac{e^{z}i}{\sum_{j=1}^{k} e^{z}i} \qquad (2)$$

- **Rectify Linear Unit (ReLU):** ReLU, or rectified linear activation function, is a piecewise linear function that outputs zero if the input is negative and the input directly if the input is positive. It has become the norm for many different types of neural networks since it is easier to train and frequently yields better performance. Equation (3) by Jaso, (2019) was used in ReLU function.

$$\mathrm{f(x)} = \mathrm{ma\,x}(0, \mathrm{x}) \qquad (3)$$

- **Dropout activation function:** The dropout layer uses a technique that randomly turns off some sections of the neurons to stop the model from overfitting. Incoming and outgoing connections from the neurons are also turned off when specific sections of the neurons are turned off, which improves the model's learning and prevents the model from generalizing to the test dataset Gavhale and Gawande, (2014). Once these layers are applied the feature map is then flattened in Single dimension and a fully connected neural network is formed.

- **InceptiveV3:** InceptionV3 is an image recognition model that has been shown to attain greater accuracy on the ImageNet dataset. The model is that culmination of many ideas developed by multiple researchers over the years by Szegedy et al., (2015). The model itself is made up of symmetric and asymmetric building blocks, including convolutions, average pooling, max pooling, concatenations, dropouts, and fully connected layers. Batch normalization is used extensively throughout the model and applied to activation inputs. Loss is computed using SoftMax.
- **Loss function:** Also known as a cost function. Let y= actual output, $\hat{y}$=predicted output and K= number of classes. Then, $y - \hat{y}$ calculated utilizing a cost function called cross-entropy (CE). For the binary classification, problem binary cross-entropy is used. The experiment conducted using binary CE and categorical CE. Equation (4) and Equation (5) was used in binary cross-entropy.

$$L_{\text{Cross-entropy}}(\hat{y}, y) = -y(\log(\hat{y}) + (1-y)\log(1-\hat{y})) \quad (4)$$

$$L_{\text{cross-entrophy}}(\hat{y}, y) = -\sum_{j=0}^{k} y_j \log(\hat{y}_j) \text{ for } j=1,2\ldots k \quad (5)$$

2.7 Conclusion

During the review and summaries of various techniques of plant disease detection using image processing that have been used by researchers in the past few years, despite some major techniques used to detect if the leaves are healthy or affected, various challenges such as poor classifications rate and execution time.

CHAPTER THREE

METHODOLOGY

This chapter discusses the method used in the design and implementation of the proposed cabbage disease detection model using CNN.

3.1 The proposed research model

This section presents the workflow diagram of the proposed model. Refer to Figure 3.1 shows the proposed workflow diagram.

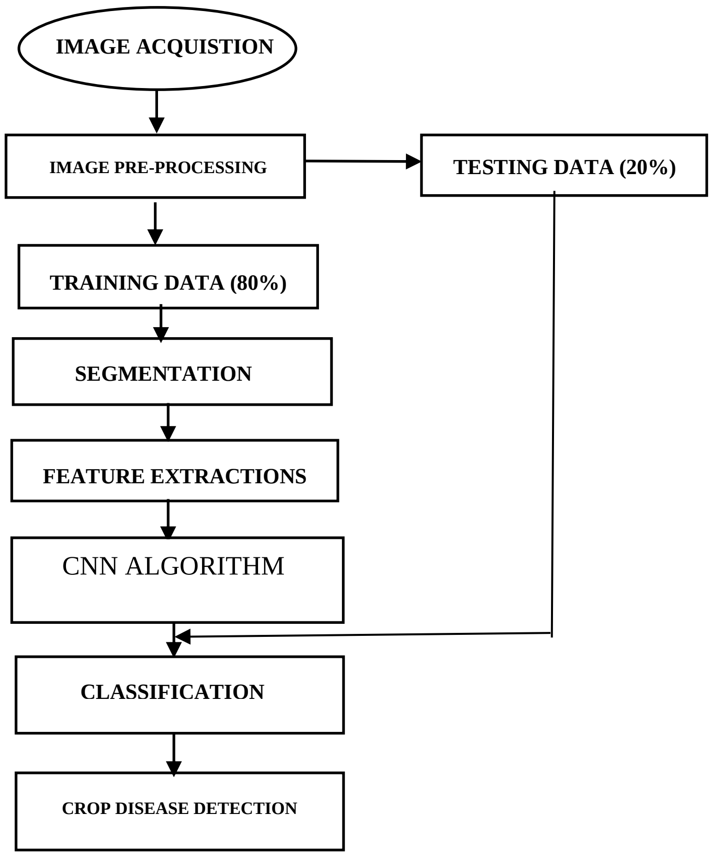


Figure 3.1: Workflow

3.1.1 Image Acquisition

Image acquisition generally entails converting an optical image (Real World Data) into a collection of numerical data that can then be processed by a computer. An image must first be acquired by a camera and transformed into a controllable entity before any image processing can start Mishra et al., (2017). An image is a two-dimensional (2-D) function or a visual representation of something. It is written as f(x, y), where x, y are two-dimensional coordinates and f is their intensity. Each coordinate is referred to as a pixel Mishra et al., (2017). In this research 20% images of cabbage crops were taken using a high pixel's lens camera in cabbage crop field, and the remaining 80% came from Kaggle database.

Prior to the CNN algorithm's training, validation, and testing, we obtained these images for preprocessing, segmentation, and feature extraction.

3.1.2 Image Pre-Processing

Image preprocessing is to rescale, clean images data from the image acquisition and format images by resizing, orienting and color corrections before used for training, validation and testing in the model, the model considers the nature of the images in a variety of situations. Equation (6) by Tete et al., (2017) was used to convert to RGB images to grayscale.

$$f(x) = r * 0.2989 + g * 0.5870 + b * 0.11 \qquad (6)$$

where **r**= Red, **g**= Green, **b**= Blue.

We used a data generator built in the model to divide the images by (150 x 150), this is to resize the images into equal dimensions and channels. The algorithm for the image preprocessing is demonstrated in algorithm 1.

Algorithm 1:

```
Input   Captured image (x)
Output  Preprocessed (y) (testing and training data)
        1   Import NumPy and Keras libraries
        2   Fetch x
        3   Set batch size = 32
        4   Set x dimension (h, w) = 150 x 150
        5   If x = (150 x 150) {
        6           z = convert x to RGB grid pixels
        7           fp(i) = convert(z)
        8           rescale i pixel values
        9           y = preprocessed RGB image (i)
        10          rand y:( 20% testing, 80% training)
        11      }
```

Algorithm 1 shows the image preprocessing method. Firstly, we use pip to install the python libraries: pip install NumPy and pip install Keras. The Numpy and Keras are python libraries that handle all the mathematical calculations. Also, we used x to denote the captured images (dataset). Where we set the batch size to 32 for the image data generator to fetch x in 32 batches. The image dimensions which is the height = h and width = w (150 x 150). Secondly, if the image size (x) is 150 * 150 then x is converted to z, where z is the converted images in RGB grid pixels. It converts the z images into a floating point as fp(i). Also, we rescale the i pixel values. Finally, line 9 -11 the preprocessed RGB image is generated which is denoted by y. It randomly selects 20% as testing data and 80% as training data. Lastly the algorithm end.

3.1.3 Training Data

The training set is used to train the model to learn image features like curve, line, texture and other features. In this project, 80% of dataset was used for training. The weight of inceptionV3 and imageNet was used to train the dataset. InceptionV3 was used because of its high image recognition.

3.1.4 Image Segmentation

Image segmentation is a technique that helps to reduce the complexity of the image for further treatment by dividing the computerized image into various subgroups called segments. Region-based segmentation is one of the many segmentation techniques used in image segmentation. In segmentation that is done on the basis of regions, a region is defined as a collection of related pixels that are connected to one another. It's possible for pixels to be comparable in terms of brightness and hue. To be grouped into similar pixel regions in this sort of segmentation, a pixel must abide by a set of predetermined rules. The two methods for region-based segmentation are as follows:

- Region growing method and
- Region splitting and merging method.

Region splitting and merging technique was used in this research. During region splitting, the entire image is first regarded as one region. If the region doesn't adhere to the specified rules, it is divided into further regions (often four quadrants), and the predefined rules are then used to those regions to determine whether to further subdivide or to designate them as a region. When every region complies with the established guidelines, the process above is repeated until no further separation of regions is necessary. Every pixel is treated as a separate zone when using the region merging approach. We choose a region as the seed region to see if nearby regions are comparable based on specified rules. If they resemble one another, it combines them into a single zone and continue to create the segmented parts of the entire image. An image is split into as many areas as possible using region splitting, and the resulting segments are then combined to create a high-quality segmented version of the original image. Refer to Figure 3.2. The CNN algorithm needed less time to process the data because of this. Equation (7) by Zhang et al.(n.d.) was used to segment the images.

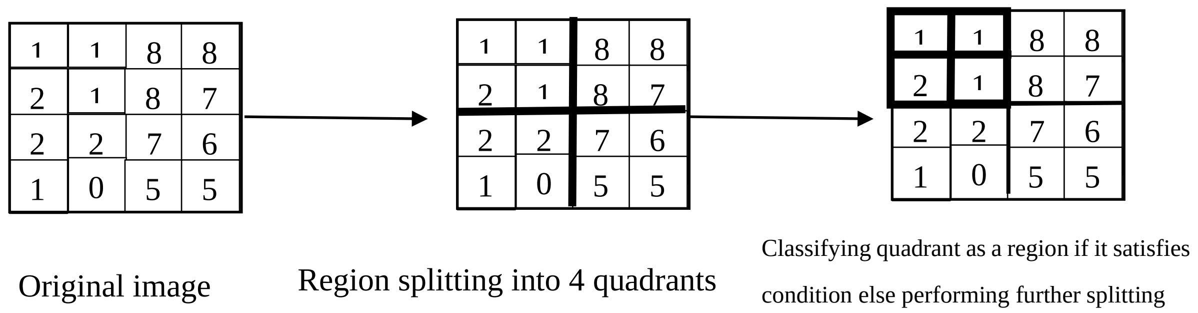

1 1 8 8
2 1 8 7
2 2 7 6
1 0 5 5
Original image
Region splitting into 4 quadrants
Classifying quadrant as a region if it satisfies condition else performing further splitting

Figure 3.2: Region splitting and merging method used in segmentation

$$Z_{max} - Z_{min}| <= threshold \qquad (7)$$

where $\mathbf{Z_{max}}$ = maximum pixel intensity values in a region and

$\mathbf{Z_{min}}$ =minimum pixel intensity values in a region. Where threshold is the segmented image as output.

3.1.5 Feature Extractions

The aim of this stage is to find and extract features that can be used to determine the meaning of a given cabbage crop image. Image processing and image features usually include color, shape, and texture features. The proposed model considers Gabor filters to calculate feature sets from the cabbage crop image. In image processing, a Gabor filter is a linear filter used for texture analysis, which essentially means that it analyzes whether there is any specific frequency content in the image in specific directions in a localized region around the point or region of analysis. The Gabor wavelet was developed based on a Gaussian function, which has good localization in both the frequency and spatial domains and has proved to be a powerful tool for extracting image texture features by Wang et al. (2017).

Gabor filter is adapted in this research to change the texture features in the spatial domain, because of its variable direction, bandwidth, and center frequency. By comparing the response of filter amplitude in each channel, boundary information between textures can be detected, to realize texture detection and segmentation in the image by Wang et al. (2017).

The accuracy of cabbage disease detection depends on the feature expression of the cabbage crop image. Therefore, it is important to mine the relevant feature information from the cabbage image region. Because of that, we added some efforts to enrich the feature expression of the cabbage crop images.

In this research, 2D Gabor convolution nucleus which has excellent spatial locality and directional selectivity and can capture the spatial frequency and local structural characteristics of multiple directions in the local area of the cabbage crop image. Its functional expression by Wang et al. (2017) was used. Refer to Equation (3), Equation (4), and Equation (5) represents the real part of the 2D Gabor function, which has stronger filtering ability and feature extraction ability.

$$G(x, y, \lambda, \theta_1, \psi, \sigma, \gamma) = exp\left(-\frac{x^2+\gamma^2 y^2}{2\sigma^2}\right) exp\left(i\ \left(2\pi \frac{x'}{\lambda_1} + \psi\right)\right) \tag{8}$$

$$G(x, y, \lambda, \theta_1, \psi, \sigma, \gamma) = exp\left(-\frac{x^2+\gamma^2\, y^2}{2\sigma^2}\right) \tag{9}$$

$$G(x, y, \lambda, \theta_1, \psi, \sigma, \gamma) = exp\left(-\frac{x^{\cdot 2}+\gamma'^2 y'^2}{2\sigma^2}\right) sin\left(2\pi \frac{x'}{\lambda_1} + \psi\right) \tag{10}$$

where x'=xcosθ + ysinθ and y'= xsinθ + ycosθ

In the equation $\boldsymbol{\lambda}$ represent the wavelength, $\boldsymbol{\theta}$ represents the orientation of the normal to parrel strips of a Gabor function, $\boldsymbol{\psi}$ represents the phase offset, $\boldsymbol{\sigma}$ represents sigma, $\boldsymbol{\gamma}$ represents the spatial aspect ratio.

3.1.6 CNN Algorithm

In this research the four main CNN algorithm layers was used. These layers are; convolutional layer, pooling layer, and fully connected layer which are explained in details.

I. **Convolutional Layer:** In this research, the convolutional layer is used to computes the convolutional operation of the input images using kernel filters to extract fundamental features and pass the features to the next layers. The kernel filters are of the same dimension but with smaller constant parameters as compared to the input images. Also, the filter mask slides over the entire input image step by step and estimates the dot product between the weights of the kernel filters with the value of the input image, which results in producing a 2D activation map. Where CNN start to learn the visual features. Also, in generating the activation map, the process is repeated on every element of the input image, then finally the output volume of the convolutional layer is generated by stacking the activation maps of every filter along the depth dimension. The convolution layer served as the inspiration for the CNN algorithm. A matrix operation is used in this layer to set of mathematical operations to extract a feature map from the input image. First, the filter is shifted step by step starting from the upper left corner of the image. At each step, the values in the image are multiplied by the values of the filter (kernel) and the result is sum. A new matrix with a smaller size is created from the input image. Equation (11) by Xray, (2020) was used in convolutional layer.

$$y[n] = (x * h)[n] = \sum_k x[k]h[n-k] \tag{11}$$

where **n** is an integer, **h** is zero phase filter, **x[n]** represent the input time series, **h[n]** is the filter weight, and **y[n]** is the output time series.

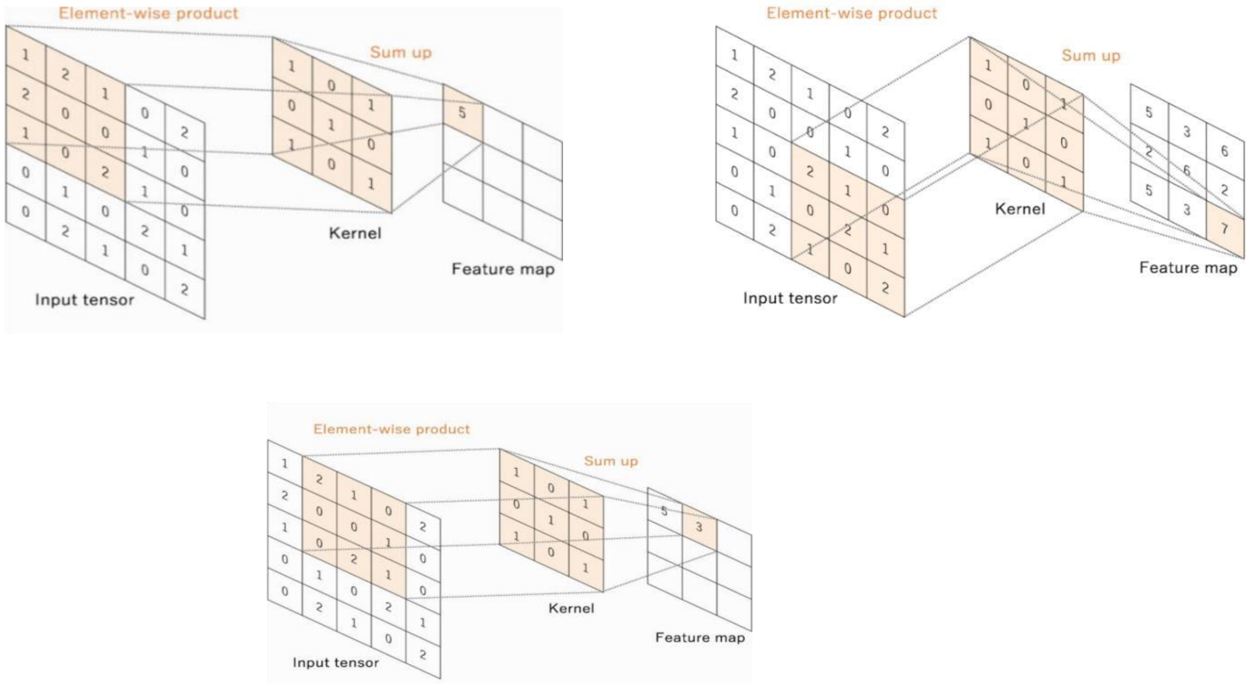


Figure 3.4 Convolution Operations

II. **Pooling layer:** The pooling layer is the next layer after the convolutional layer operation. It lowers the in-plane dimensionality of feature maps, introduces translation invariance with a slight shift and distortion, and lowers the number of ensuing learnable parameters. A conventional down sampling procedure is carried out before the final layer is added Arora et al. (2019). After an operation is performed in the convolution layer, a feature map is generated as an output. The output of the Convolutional layer feature map will be reduced in the pooling layer. Different filter sizes are used in the pooling layer, but usually a 2x2 size filter is used, and also several functions are used to list some max pooling, average pooling, and sum pooling, which are some of the functions used. A max pooling operation is depicted in Figure 3.5. Max pooling is performed by selecting the largest element from the input matrix concerning the filter. Max pooling on a 4*4 channel using 2*2 kernel and a stride of 2, in this research convolving with a 2*2 Kernel. If we observe the first 2*2 set on which the kernel is focusing the channels are four values 5,1,8,3. Max pooling picks the maximum value from that set which is 8.

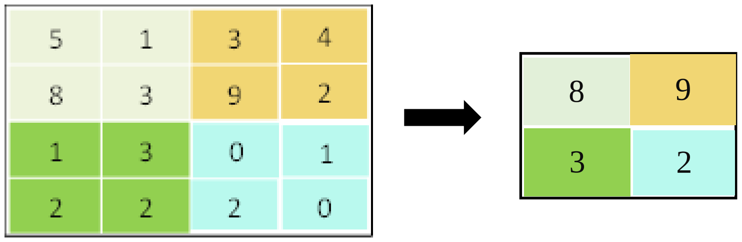


Figure 3.5: Pooling Layer with A Stride of 2 with 2x2 Filter

III. **Fully Connected Layer (Dense layer):** Fully connected layer is a crucial part in CNN that allows a computer vision system to correctly identify and classify an image. The image breakdown is converted into a feature and then independently examined once the input image has been fed into the CNN algorithm and has passed the convolution and pooling layers. To classify an image, the FC layer flattens the output of the pooling layer. Figure 3.6 shows fully connected layer.

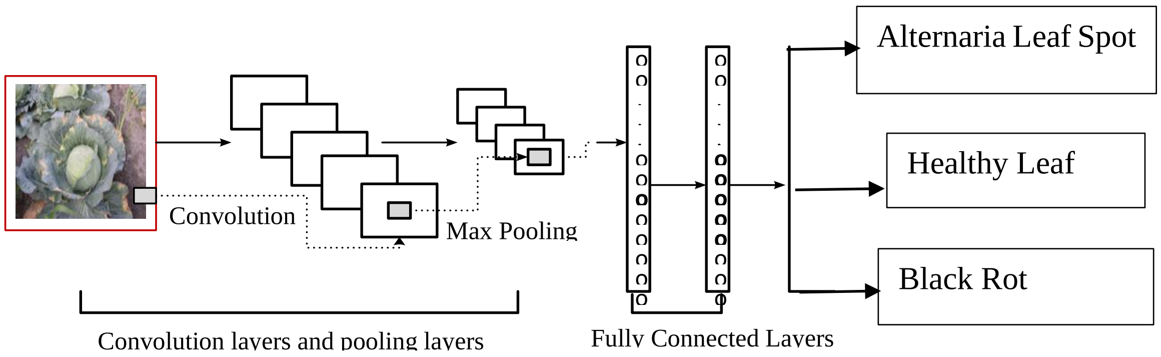


Figure 3.6: Fully Connected Layer

3.1.7 Classification

Classification is the process of categorizing the given dataset into classes, it can perform on both structured or unstructured data. In the proposed model, a classification function was used to process structured data. This starts with predicting the class of the given data points. Our model has three classes which are Alternaria leaf spot disease, Black rot disease, and healthy leaf from cabbage crops. The classifier, in this case, takes the training data to understand how the given input variables are related to the class by comparing to tested data. Once the classifier is trained accurately, it can be used to detect whether the cabbage crop is affected by the two disease or healthy.

The classifier compares the trained data to the testing data before predicting.

3.1.8 Crop Disease Detection

At this stage of the proposed CNN model, the prediction is represented in a browser.

Flask is a micro web framework in a python programming language that provides useful tools and features that makes creating a web application in Python easier. Flask was designed and licensed under a three-clause BSD License according to Mufid et al., (2019).

We used the flask to create a server side using the cabbage disease model as backend database and applied jinja2 templets as frontend for the user specifically the farmer to interact with the cabbage disease detection model.

The backend: The backend of this projects is referred as the server-side of the web application. We use the cabbage model developed from the CNN to ensure that the user response is accurate and yield the proper predictions.

The front end: In this project, the area where the user immediately interacts with the proposed model.

CHAPTER FOUR

RESULTS AND DISCUSSION

This chapter presents the results of a cabbage crop disease detection model using the CNN algorithm. Simulation and experimental results are represented in this section. Comparison of the proposed model to other similar models are represented in a table in this section.

4.1 Simulation Results

To test and demonstrate the effectiveness of our proposed model, a computer simulation has been performed using Python and a Laptop computer with the following features:

- Computer Type: Laptop, HP Pavilion 15-cdoxx.
- Operating System: Microsoft Windows 11 Home.
- Processor: AMD A12-9720P Core (TM) i5 CPU@2.70GHz
- System Type: 64-bit operating system, x64-based processor
- Installed RAM: 8.00 GB (7.87 GB usable)
- Storage Disks: 1TB

The experiments carried out by the proposed model build are divided into two scenarios. The first part is the model compilation accuracy and loss in training and validation. The second part presents the prediction of cabbage crop results from the build model.

4.1.1 Training and validation of Accuracy and Loss

After the model had been trained with cabbage image data and validated, in this research, validation rate of 97.5% and an accuracy rate of 99.85%. Four different experiments were carried out by setting up four different epochs. Figure 4.1 shows training and validation, accuracy, and loss using Epoch 5.

In epoch 5, the proposed model obtained training rate of 83.11% and validation rate of 78.65% respectively. It also obtained training loss of 41.68% and validation loss 72.53% respectively.

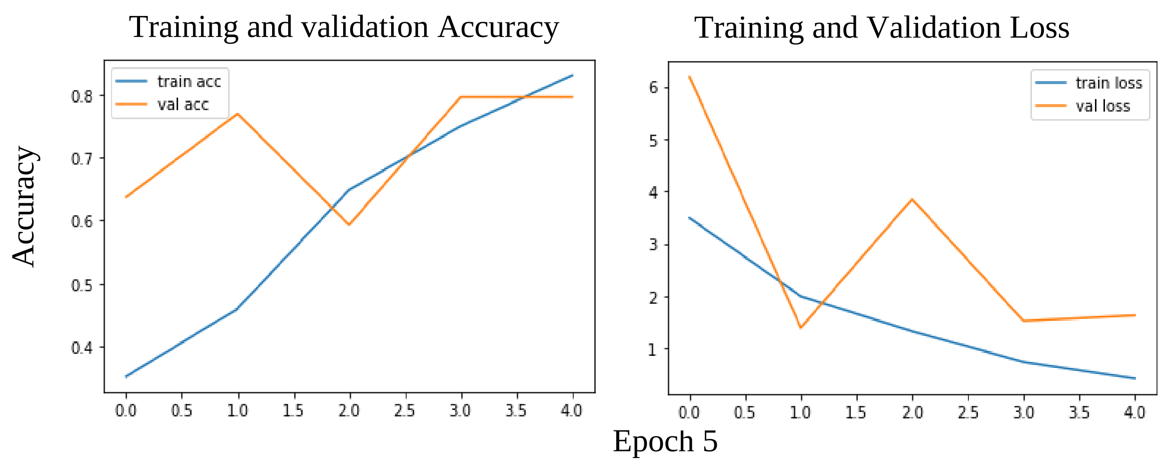


Figure 4.1: Training and validation Accuracy and Loss Epoch 5

In Figure 4.2. Epoch was set to15 using the proposed model, training accuracy rate was 95.95% and validation rate of 92.92% respectively. Also, training loss was 12.07% and validation loss of 62.53% respectively.

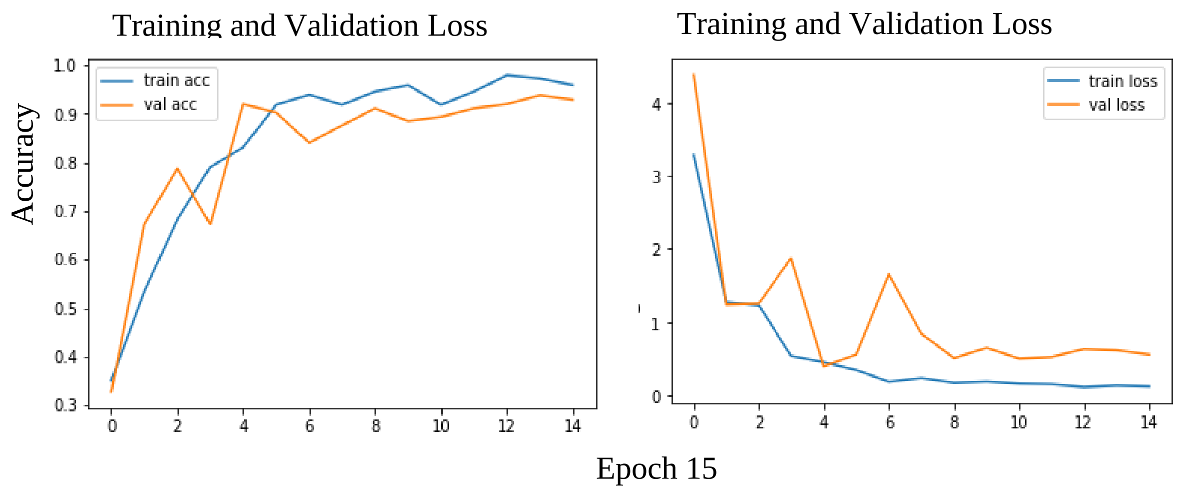
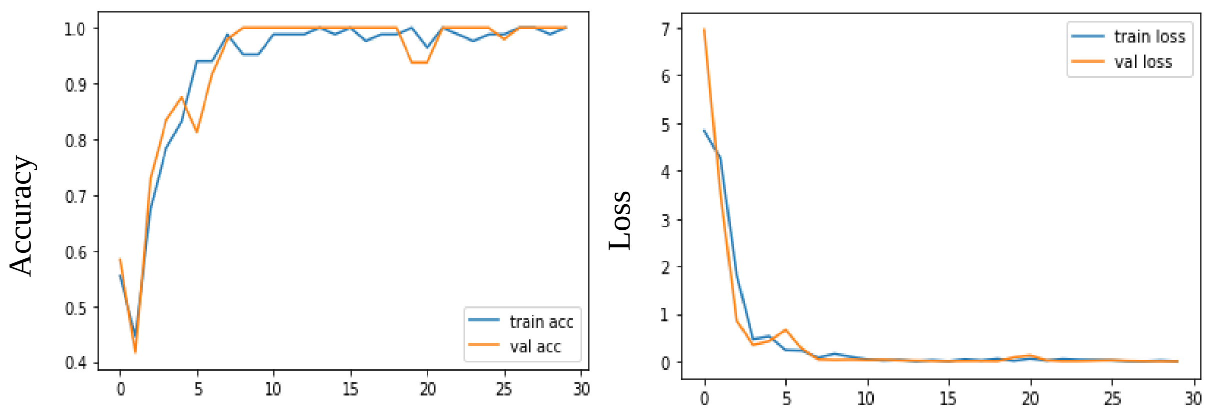


Figure 4.2: Training and validation Accuracy and Loss Epoch 15

In Figure 4.3 Epoch was set to 30, using the proposed model, training accuracy was 97.30% and validation rate 87.61% respectively. And also training loss was 11.33% and validation loss of 55.61% respectively.

Training and validation Accuracy Training and Validation Loss

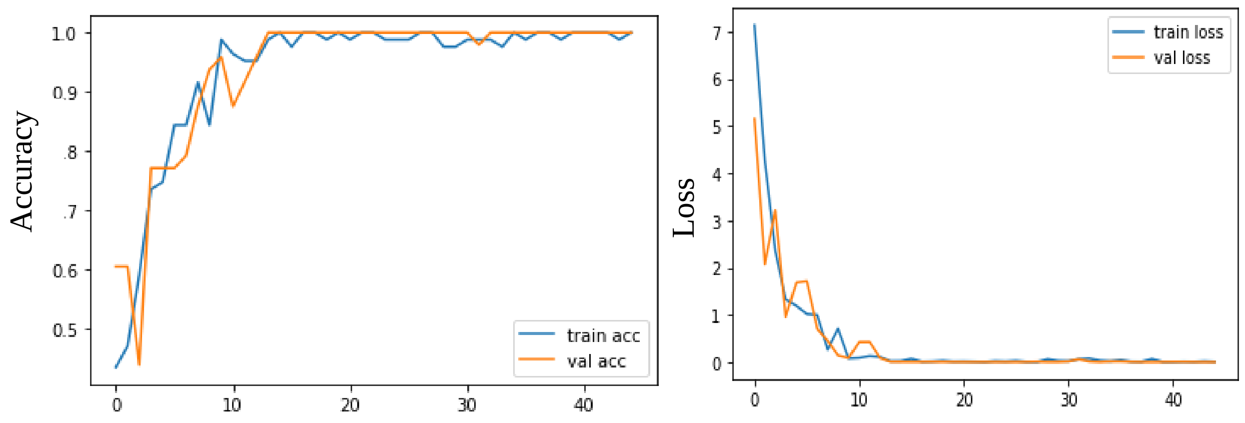


Epoch 30

Figure 4.3: Training and validation Accuracy and Loss Epoch 30

In Figure 4.4, Epoch was set to 40, using the proposed model, training accuracy rate was 99.32% and validation rate of 87.73% respectively. Also training loss was 0.08% and validation loss of 1.63% respectively.

Training and validation Accuracy Training and validation loss

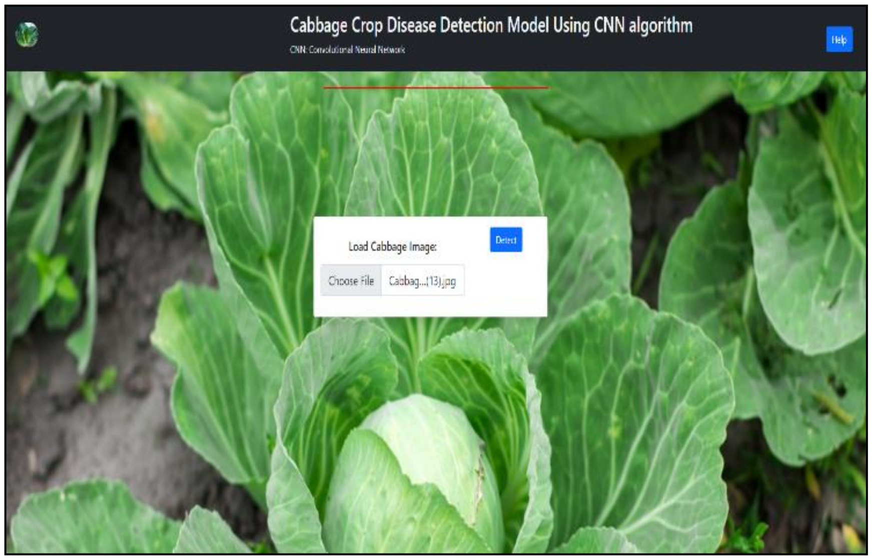


Epoch 40

Figure 4.5: Training and validation Accuracy and Loss Epoch 40

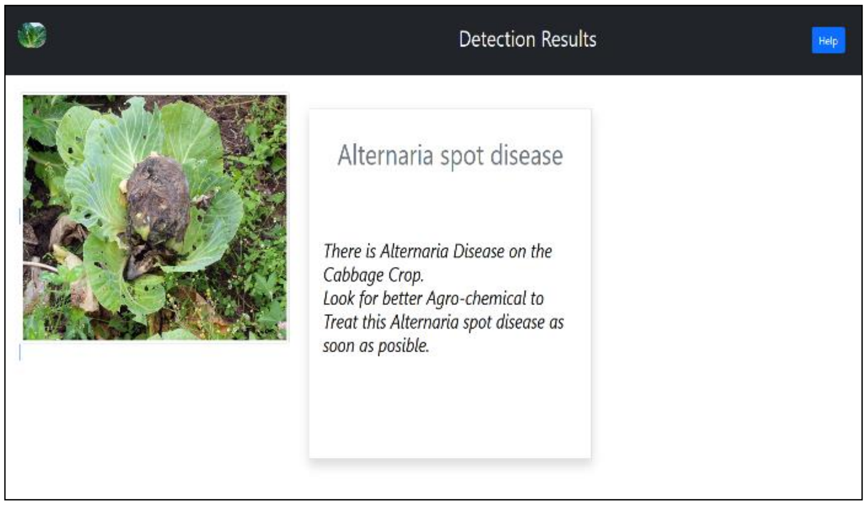
4.3 Experimental results of the proposed model

The proposed model was tested using a Web interface to present the prediction of a cabbage crop image. In the experiment, the proposed model was fed with different images of cabbage crops and predicted them all. Figure 4.6 shows the web interface, Figure 4.7 shows a healthy cabbage crop, Figure 4.8 shows Alternaria leaf spot, and Figure 4.9 shows a black rot cabbage image.



Cabbage Crop Disease Detection Model Using CNN algorithm
CNN: Convolutional Neural Network
Help
Load Cabbage Image:
Detect
Choose File
Cabbag...(13).jpg

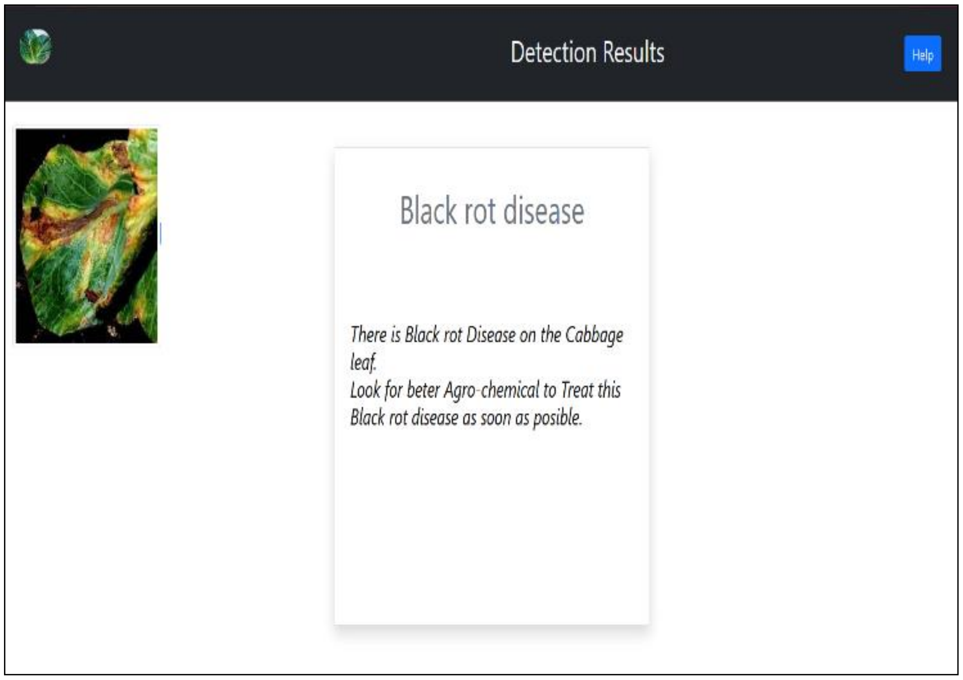
Figure 4.6: Interface of the model



Alternaria spot disease

There is Alternaria Disease on the Cabbage Crop.
Look for better Agro-chemical to Treat this Alternaria spot disease as soon as posible.

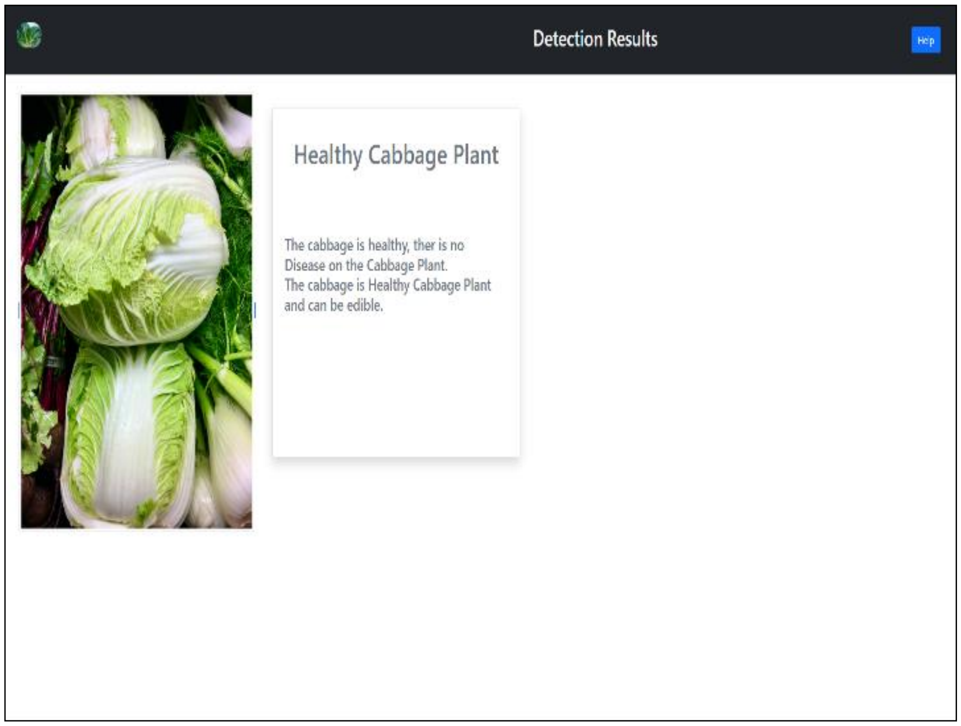
Figure 4.7: Alternaria spot cabbage crop



Black rot disease

There is Black rot Disease on the Cabbage leaf.
Look for beter Agro-chemical to Treat this Black rot disease as soon as posible.

Figure 4.8: Black rot cabbage crop affected



Healthy Cabbage Plant

The cabbage is healthy, ther is no Disease on the Cabbage Plant.
The cabbage is Healthy Cabbage Plant and can be edible.

Figure 4.9: Healthy cabbage crop

4.4 Comparison of other researcher’s results obtained

The comparison results between the proposed algorithm and other research results are shown in Table 4.1. This research paper evaluates the classification performance comparison with existing methods. In Table 4.1, the first column shows the table list numbers, the second column denotes the Author names and years of publication, the third column shows the Algorithm used by the corresponding Author, and the fourth column represents the real data taken from the field, the last column refers to the classification accuracies which correspond to the algorithm used. The symbol in the table “–” implies the researcher did not use real data from field while symbol “√” implies the researcher used real data from the field. The best classification accuracy is selected from the highest classification accuracy rate, which is displayed in Table 4.1.

S/N	Authors	Algorithm	Real data	Accuracy
1	Kulkarni & K, (2012)	ANN	-	80.76%
2	D. Oppenheim et al. (2018)	CNN	√	95.8%
3	M. Sardogan, A. Tuncer and Y. Ozen (2018)	CNN	-	86%
4	K. P. Ferentinos (2018)	CNN	-	99.53%
5	(Lin and Shen 2018)	CNN	-	86.96%
6	**Proposed model**	**CNN**	**√**	**99.85%**

Table 4.1: Shows the comparison of others research works

CHAPTER FIVE

SUMMARY, CONCLUSION AND RECONMENTDATION

5.1 Summary

Enhancing agricultural product yields is a priority for developing nations like Ghana. This project proposed two major cabbage crop disease detection model using CNN. These diseases are Alternaria leaf spot and Black rot in the cabbage crop. The simulation results verify the proposed model for cabbage crop disease detection using CNN. The proposed model used the inceptionV3 pre-trained model which helped to achieve a high accuracy rate in the detection of the two diseases and has also improved execution time. The CNN proposed model was equipped using Python programming language, which helped the model to achieve the detection of the two diseases. The proposed model was evaluated using a web-based for the experiment. The proposed model has two parts, the backend which keeps the model running whiles the web-based serves as the user interface. Finally, the proposed model scored 99.85% accuracy in detecting the cabbage leaf image as Healthy and Alternaria leaf spot or Black rot affected.

5.2 Conclusion

In this research, the CNN model was developed to detect Healthy cabbage crops, Alternaria leaf spot, and Black rot diseases, which affect the leaf section.

The proposed model was intended to improve the accuracy rate and execution time. The proposed model using CNN to detect some selected cabbage crop diseases has been successfully applied to predict Healthy cabbage crops, Alternaria leaf spot, and Black rot cabbage crop diseases.

The proposed model was built using Python programming language equipped with a computer vision model. Using several layers in the CNN algorithm to compile the model effectively. Simulations and experiments demonstrated how capable the CNN model has improved in accuracy rate and execution time when compared to other models.

The model uses a web-based interface for operations and is required to operate on only two cabbage crop diseases which are Alternaria leaf spot and Black rot diseases including Healthy cabbage crops.

5.3 Recommendation

The major cabbage crop diseases that affect the cabbage crop are Alternaria leaf spot and Black rot disease. In this research, the cabbage crop disease detection model was developed.

According to the issue domain, a small dataset was generated and utilized in comparison, however, it is advised to gather and construct a large dataset because the machine learning model gets better at identifying the disease as the number of images increases. The heads and stems of the cabbage crop, in addition to the leaf portion, contribute to the entire detection and classification of the crop's disease.

Estimating the disease's severity can be helpful for farmers and other people in the agricultural industry since it tells them how much the plant is harmed by the disease and what steps need to be done. This enables the industry to have precise knowledge about the disease's stage and the procedures required to address it successfully.

There are other plant disease detection algorithms besides CNN. As a result, experiments should be conducted to identify and categorize plant disease using a limited dataset employing a variety of approaches and algorithms.

References

Garbin, C., & Marques, Z. (2020). Dropout vs. batch normalization:an empirical study of their impact to deep learning. pp. 12777–12815. doi:10.1007/s11042-019-08453-9

Badage, A. (2018, September). Crop disease detection using machine learning. *International Research Journal of Engineering and Technology (IRJET), 5*(09). Retrieved from www.irjet.net

George, H. (2020, August). *Identity,Prevent and Treat Common Cabbage Diseases*. Retrieved from Gardenerspath.com: http://google.com

Jaso, B. (2019, January 09). Deep Learning Performance. *Deep learning for computer vision: image classification, object detection, and face recognition in python.Machine Learning Mastery*. Retrieved from Googlescholar.com

Lupescu, M., & Zimmerman, J. (2019). Strategic analysis and intervention plan for potatoes and potato products e Agro-Commodities Procurement Zone of the pilot Integrated Agro-IndustrialPark in Central-Eastern Oromia. Retrieved from www.ijcsmc.com

Mufid, M. R., Basofid, A., AL Rasyid, M. H., Rochimansyah, I. F., & Rokhim, A. (2019). Design an MVC Model using Python for Flask Framework Development. *International Electronics Symposium(IES)*.

Samisco, E. (2021, January 3). *The Farmers Dream*. (Cabbage farming in Ghana) Retrieved from thefarmdreams.com

Sugimura, D., Mikami, T., Yamabashita, H., & Hamamoto, T. (2015, November). Transactions on Image Processing. *IEEE, 24*. Retrieved from researchgate.net

Szegedy, C., Vanhoucke, V., Sergey, L., Shlens, J., & Wojna, Z. (2015). Rethinking The inception Architecture For Computer Vision. *The Simons Foundation and Members Institutions*.

Tyagi, V. (2018, October 01). Understanding Digital Image Processing. *Engineerng and Technology , 1*, 374. doi:https://doi.org/10.1201/9781315123905

Xray. (2020, January 4). *Parked Photon*. Retrieved from www.parkedphoton.com/author/xray

Arnal Barbedo, Jayme Garcia. 2013. "Digital Image Processing Techniques for Detecting, Quantifying and Classifying Plant Diseases." *SpringerPlus* 2(1):1–12. doi: 10.1186/2193-1801-2-660.

Arora, Sanjeev, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. 2019. "Fine-Grained Analysis of Optimization and Generalization for Overparameterized Two-Layer Neural Networks."

Barik, Lipsa. 2018. *A Survey on Region Identification of Rice Diseases Using Image Processing*. Vol. V.

Gavhale, Ms. Kiran R., and Prof. Ujwalla Gawande. 2014. "An Overview of the Research on Plant Leaves Disease Detection Using Image Processing Techniques." *IOSR Journal of Computer Engineering* 16(1):10–16. doi: 10.9790/0661-16151016.

Heaton, Jeff. 2018. "Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep Learning." *Genetic Programming and Evolvable Machines* 19(1–2):305–7. doi: 10.1007/s10710-017-9314-z.

Kulkarni, Anand, and Ashwin K. 2012. "Applying Image Processing Technique to Detect Plant Diseases." 2:3661–64.

Lin, Guifang, and Wei Shen. 2018. "Research on Convolutional Neural Network Based on Improved Relu Piecewise Activation Function." Pp. 977–84 in *Procedia Computer Science*. Vol. 131. Elsevier B.V.

Mishra, Vikas Kumar, Shobhit Kumar, and Neeraj Shukla. 2017. "Image Acquisition and Techniques to Perform Image Acquisition." *SAMRIDDHI : A Journal of Physical Sciences, Engineering and Technology* 9(01):21–24. doi: 10.18090/samriddhi.v9i01.8333.

Mufid, Mohammad Robihul, Arif Basofi, M. Udin Harun al Rasyid, Indhi Farhandika Rochimansyah, and Abdul rokhim. 2019. "Design an MVC Model Using Python for Flask Framework Development." Pp. 214–19 in *2019 International Electronics Symposium (IES)*.

Szegedy, Christian, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2015. "Rethinking the Inception Architecture for Computer Vision."

Tete, Trimi Neha, and Sushma Kamlu. 2017. "Plant Disease Detection Using Different Algorithms." Pp. 103–6 in *Proceedings of the Second International Conference on Research in Intelligent and Computing in Engineering*. Vol. 10. IEEE.

Tilahun, Natnael. 2020. *ADDIS ABABA SCIENCE AND TECHNOLOGY UNIVERSITY DEVELOPING POTATO'S LEAVES DISEASE DETECTION MODEL USING CONVOLUTIONAL NEURAL NETWORK*.

Wang, Jindong, Yiqiang Chen, Shuji Hao, Xiaohui Peng, and Lisha Hu. 2017. *Pattern Recognition Letters Deep Learning for Sensor-Based Activity Recognition: A Survey*.

Yamashita, Rikiya, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. 2018. "Convolutional Neural Networks: An Overview and Application in Radiology." *Insights into Imaging* 9(4):611–29.

Zhang, Hang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. n.d. *Context Encoding for Semantic Segmentation*.

APPENDIX

SOURCE CODE

Name file: *Training.py*

```
################ importing necessary libraries
from tensorflow.keras.layers import Input, Lambda, Dense, Flatten
from tensorflow.keras.models import Model
from tensorflow.keras.applications.in
ception_v3 import InceptionV3
import tensorflow as tf
from sklearn.model_selection import train_test_split
import pandas
from tqdm import tqdm
from tensorflow.keras.models import load_model
from tensorflow. keras.applications.inception_v3 import preprocess_input
from sklearn.preprocessing import label_binarize
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator,load_img
from tensorflow.keras.models import Sequential
import matplotlib.pyplot as plt
import numpy as np
from glob import glob
tf.config.list_physical_devices('GPU')
##################### re-sizing all the images to this
```

```
IMAGE_SIZE = [150, 150]
####### set the training path and validation path
train_path = 'Dataset/train'
valid_path = 'Dataset/val'
##################### Apply Transfer learning
inception = InceptionV3(input_shape=IMAGE_SIZE + [3], weights='imagenet',
                        include_top=False)
## don't train existing weights
for layer in inception.layers:
    layer.trainable = False
    ## our layer
    x = Flatten()(inception.output)
    # custom output layer
    prediction = Dense(len(folders), activation='softmax')(x)
    # creating the cabbage disesase detection a model object
    model = Model(inputs=inception.input, outputs=prediction)

    # viewing  the structure of the model and save as text file
    with open('modelsummary.txt', 'w') as f:
        model.summary(print_fn=lambda x: f.write(x + '\n'))

    #### Compile the model using optimization method Adams
    model.compile(
                    loss='categorical_crossentropy',
                     optimizer='adam',
                    metrics=['accuracy']
    )
    ## Using Image Data Generator to import the training data and the validation dataset
    ## Data augmentation starts here
    ##import the image preprocessing library
    from tensorflow.keras.preprocessing.image import ImageDataGenerator
    train_datagen = ImageDataGenerator(rescale = 1./255,
                                       shear_range = 0.2,
                                       zoom_range = 0.2,
                                       horizontal_flip = True)
    validat_datagen = ImageDataGenerator(rescale = 1./255)
```
### providing the target size and batch size for the dataset as image size
```
    training_set = train_datagen.flow_from_directory('Dataset/train',
                                                     target_size = (150, 150),
                                                     batch_size = 32,
                                                     class_mode = 'categorical')
    val_set = validat_datagen.flow_from_directory('Dataset/val',
                                                  target_size = (150, 150),
                                                  batch_size = 32,
    # fitting the model to the CNN
    r = model.fit_generator(
                          training_set,
                          validation_data=val_set,
                         epochs=20,
                          steps_per_epoch=len(training_set),
                         validation_steps=len(val_set)
    )
```

## saving model as a h5 file to disk
```
model.save('model/cabbage_model_inception.h5')
print("model save to disk")
```
## ploting grpah for loss and accuracy

# plot the loss
```
plt.plot(r.history['loss'], label='train loss')
plt.plot(r.history['val_loss'], label='val loss')
plt.legend()
plt.show()
plt.savefig('LossVal_loss')
# plot the accuracy
plt.plot(r.history['accuracy'], label='train acc')
plt.plot(r.history['val_accuracy'], label='val acc')
plt.legend()
plt.show()
plt.savefig('AccVal_acc')
```

name file: App.py

```
### import necessary libraries
from __future__ import division, print_function
# import Flask and Werkzeug server for the flask
from flask import Flask, redirect, url_for, request, render_template
from werkzeug.utils import secure_filename
# import CNN libraries
from tensorflow.compat.v1 import ConfigProto
from tensorflow.compat.v1 import InteractiveSession
from keras.preprocessing.image import load_img
from keras.preprocessing.image import img_to_array
from keras.models import load_model
from keras.models import load_model
from keras.preprocessing import image
# coding=utf-8
import sys
import os
import glob
import re
import numpy as np
import tensorflow as tf
# configure the user GPU
gpus = tf.config.experimental.list_physical_devices('GPU')
```

```
for gpu in gpus:
    tf.confi.experimental.set_memory_growth(gpu, True)
tf.config.list_physical_devices('GPU')

##load the cabbage model which will be used as flask database in the server
model = load_model('model/cabbage_model_inception.h5')
print(model)
print('@@ model loaded successfull')
##define prediction parameters
def pred_cab_diseas(cabbage_plant, model):
    test_image = image.load_img(cabbage_plant, target_size = (150,150)) # load image
    print("@@ Got image for prediction")

    test_image = img_to_array(test_image)/255 # convert image to np array and normalize
    test_image = np.expand_dims(test_image, axis= 0) # change dimention 3D TO 4D

    result = model.predict(test_image) # predict diseased cabbage or not
    print ('@@ Raw result = ', result)

    pred = model.predict(test_image)
    pred = np.argmax(result, axis = 1) #get the index of max value
    print (pred)
    if pred == 0:
        pred= " Alternaria spot disease", 'Cabbage-Altenaria-spot.html' # if index 0 burned leaf
    elif pred == 1:
        pred = " Black rot disease", 'Cabbage-Black-rot.html' ## if index 1
    elif pred == 2:
        pred = "Healthy Cabbage Crop", 'Cabbage-healthy.html' #if index 2 fresh leaf
    elif pred == 3:
        pred = "Unidentify Object", 'no_disease.html' # if index 3
    return pred
#------------------>> pred_cabbage_disease <<-- end
## Creating Flask Instances

app = Flask(__name__)

#render index.html page
@app.route("/", methods=['GET', 'POST'])
def home():
    return render_template('index.html')

# get input image from client then predict base on the  respective .html page for solution
@app.route("/predict",methods = ['GET', 'POST'])
def predict():
    if request.method == 'POST':
        file = request.files['image'] # fet imput

        filename = file.filename
        print("@@ Input Posted = ", filename)

        file_path = os.path.join('static/user uploads',secure_filename(file.filename))
        file.save(file_path)

        print("@@ Predicting class......")

    #----------->>>>>>>>>--------<<<<<<<<
    ## Make prediction
        pred, output_page = pred_cab_diseas(file_path, model)
        result=pred
        return render_template(output_page, pred_output=result, user_image = file_path)
    return None
```

```
   ##for local host server
if __name__ == "__main__":
        app.run(threaded= False, port=9999)
```

Jinja2 Application interface

Name: index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>{% block title %} Flask App {% endblock title %} </title>
   <link rel="stylesheet" href="{{ url_for('static', filename= 'style.css') }}">
   <link rel="stylesheet" href="{{ url_for('static', filename= 'bootstrap.min.css') }}">
</head>
<body style="background: url({{url_for('static', filename='background.jpg')}}); background-
repeat: no-repeat; background-size: cover;">
<!--   ---------------------<<<<< header sections start here >>>>> -->
   <header class="navbar bg-dark p-2 text-light">
      {% block header %}
      <div class="logo mx-3 my-1">
         <img src="{{url_for('static', filename='logo.jpg')}}" alt="">
      </div>
      <div class="link-div">
      </div>
      <div class="nav-text">
         <h2>Cabbage Crop Disease Detection Model Using CNN algorithm</h2>
         <p>CNN: Convolutional Neural Network</p>
      </div>
      <div class="help-link mx-4">
         <button class="help btn btn-primary">Help</button>
      </div>
      {% endblock header %}
```

```
 </header>
<!--   ---------------------<<<<< Main content section start here >>>>> -->
     {% block content %}
         <div class="container-fluid mt-4 bg-ight text-dark "id="wrap" >
           <center>
                <div class="row position-absolute top-50 start-50 translate-middle mt-1 ">
                     <div class="shadow p-3 mb-5 bg-body rounded">
                      <div class="col-12">
                        <div class="input-group mb-3">

                          <form action="/predict" method="post" class="row g-3"
enctype="multipart/form-data" onsubmit="showloading()">
                              <div class="col-md-8">
                              <h5 class="text-dark m-3 form-label">Load Cabbage Image:</h5>
                                <input class="form-control form-control-lg" name="image"
id="formFileLg" type="file">
                              </div>
                              <div class="col-md-4 center">
                                <button type="submit" value="predict" class="btn btn-primary mb-
3">Detect</button>
                             </div>
                          </form>
                         </div>
                      </div>
                   </div>
                </div>
           </center>
         </div>
     {% endblock content %}
  <script>
```

```
      window.alert("System Alert: This page act as an Interface for the Cabbage Disease
Detetion Model and Does not predict Cure or Treatment");   </script>

   <script src=" {{url_for('static', filename='bootstrap.min.js')}} " ></script>

</body>
</html>
```

Atlternarial spot disease jinja templates

Name: Cabbage-Altenaria-spot.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %} Flask App {% endblock title %} </title>
    <link rel="stylesheet" href="{{ url_for('static', filename= 'style.css') }}">
    <link rel="stylesheet" href="{{ url_for('static', filename= 'bootstrap.min.css') }}">
</head>
<body>

    <header class="navbar bg-dark p-2 text-light">
        {% block header %}
        <div class="logo mx-3 my-1">
            <img src="{{url_for('static', filename='logo.jpg')}}" alt="">
        </div>
        <div class="link-div">

        </div>
        <div class="nav-text">
            <h2> Detection Results </h2>
        </div>
        <div class="help-link mx-4">
           <button class="help btn btn-primary">Help</button>
        </div>
        {% endblock header %}
    </header>

        {% block content %}
          <!--   ---------------------<<<<<get user input image here >>>>> -->
          <div class="container-fluid mt-4 text-secondary" style="width: 100%; height:
100vh;">
           <div class="row">
              <div class="col-6" style="width: 500px; height:300px; float: left; margin-left:
15px;">
                <span class="border border-primary">
                                                   <img src="{{ user_image }}" alt="user image"
class="img-thumbnail">
                                         </span>
              </div>
              <!--  -----------------<<<<< predicted results from Python model >>>>> -->
              <div class="col-6  border shadow p-4 m-4" style="width: 500px;height:500px;">
                 <div class="col" style="width: 100%; height: 200px; float: right;">
                    <div>
                       <h1 style="padding:15px; background-color:rgb(153,156,150), color:
white;" class="text-center mb-5 content-h1 rounded">
                          {{pred_output}}
                       </h1>
                    </div>
                                         <div class="details"  style="width: 100%; height: 100px;" >

                              <!--   -------<<<<< Predicted instructions is here>>>>> -->
```

```
                                             <h3 style="color: #0b0b0b; font-style: italic">
                                                      <br> There is Alternaria leaf spot Disease on
the Cabbage Crop. <br>
                                                      Look for better Agro-chemical to Treat this
{{pred_output}} as soon as posible.
                                             </h3>
                                    </div>
                           </div>
            </div>
         </div>
      </div>
  <!-- Main-conatiner ends here << -->
      {% endblock content %}
  <scripts src=" {{url_for('static', filename='bootstrap.min.js')}} " ></scripts>
</body>
</html>
```

Black Rot Disease Jinja templates

Name file: Cabbage-Black-rot.html

```
<!DOCTYPE html>
<html lang="en">
<head>
   <meta charset="UTF-8">
   <title>{% block title %} Flask App {% endblock title %} </title>
   <link rel="stylesheet" href="{{ url_for('static', filename= 'style.css') }}">
   <link rel="stylesheet" href="{{ url_for('static', filename= 'bootstrap.min.css') }}">
</head>
<body>

   <header class="navbar bg-dark p-2 text-light">
      {% block header %}
```

```
        <div class="logo mx-3 my-1">
            <img src="{{url_for('static', filename='logo.jpg')}}" alt="">
        </div>
        <div class="link-div">
              </div>
        <div class="nav-text">
            <h2> Detection Results </h2>

        </div>
        <div class="help-link mx-4">
            <button class="help btn btn-primary">Help</button>
        </div>
        {% endblock header %}
    </header>

        {% block content %}
          <!--   ---------------------<<<<<get user input image here >>>>> -->
          <div class="container-fluid mt-4 text-secondary" style="width: 100%; height:
100vh;">
           <div class="row">
               <div class="col-6" style="width: 500px; height:300px; float: left;">
                  <span class="border border-primary">
                                              <img src="{{ user_image }}" alt="user image"
class="img-thumbnail">
                                    </span>
               </div>
               <!--   ---------------------<<<<< predicted results from Python model >>>>> -->
               <div class="col-6  border shadow p-4 m-4" style="width: 500px;height:500px;">
                   <div class="col" style="width: 100%; height: 200px; float: right;margin-left:
15px">
                      <div>
```

```
                    <h1 style="padding:15px; background-color:rgb(153,156,150), color:
white;" class="text-center mb-5 content-h1 rounded">
                        {{pred_output}}
                    </h1>
                 </div>
                                 <div class="details"  style="width: 100%; height: 100px;" >

                                 <!--   ---------------------<<<<< Predicted instructions is
here>>>>> -->
                                         <h4 style="color: #0b0b0b; font-style: italic">
                                                 <br> There is Black rot Disease on the Cabbage
leaf. <br>
                                                 Look for beter Agro-chemical to Treat this
{{pred_output}} as soon as posible.
                                         </h4>
                                 </div>
                         </div>
            </div>
        </div>
     </div>



  <!-- Main-conatiner ends here << -->
     {% endblock content %}

  <scripts src=" {{url_for('static', filename='bootstrap.min.js')}} " ></scripts>
</body>
</html>
```

Healthy Cabbage jinja template

Name file: Cabbage-healthy.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>{% block title %} Flask App {% endblock title %} </title>
    <link rel="stylesheet" href="{{ url_for('static', filename= 'style.css') }}">
    <link rel="stylesheet" href="{{ url_for('static', filename= 'bootstrap.min.css') }}">
</head>
<body>

    <header class="navbar bg-dark p-2 text-light">
        {% block header %}
        <div class="logo mx-3 my-1">
            <img src="{{url_for('static', filename='logo.jpg')}}" alt="">
        </div>
        <div class="link-div">

        </div>
        <div class="nav-text">
            <h2> Detection Results </h2>

        </div>
        <div class="help-link mx-4">
            <button class="help btn btn-primary">Help</button>
        </div>
        {% endblock header %}
    </header>

        {% block content %}
          <!--   --------------------<<<<<get user input image here >>>>> -->
        <div class="container-fluid mt-4 text-secondary" style="width: 100%; height:
100vh;">
         <div class="row">
            <div class="col-6" style="width: 500px; height:300px; float: left;margin-left:
15px">
              <span class="border border-primary">
                                          <img src="{{ user_image }}" alt="user image"
class="img-thumbnail">
                                 </span>
            </div>
            <!--   --------------------<<<<< predicted results from Python model >>>>> -->
            <div class="col-6  border shadow p-4 m-4" style="width: 500px;height:500px;">
               <div class="col" style="width: 100%; height: 200px; float: right;">
                  <div>
                     <h1 style="padding:15px; background-color:rgb(153,156,150), color:
white;" class="text-center mb-5 content-h1 rounded">
                        {{pred_output}}
                     </h1>
                  </div>
                                 <div class="details"  style="width: 100%; height: 100px;" >


                                 <!--   --------------------<<<<< Predicted instructions is
here>>>>> -->
                                          <h4 style="color: #0b0b0b; font-style: italic">
                                                   <br> The cabbage is healthy, there is no
Disease on the Cabbage. <br>
                                                   The cabbage is {{pred_output}} and can be
edible.
                                          </h4>
                                 </div>
                        </div>
```

```
                </div>
            </div>
        </div>
   <!-- Main-conatiner ends here << -->
        {% endblock content %}

   <scripts src=" {{url_for('static', filename='bootstrap.min.js')}} " ></scripts>
</body>
</html>
```

Cascading Style Sheet (CSS)

Name file: style.css

```
* {
    margin: 0;
    padding: 0;

}
body {

    background: #fff;
}
.navbar {
    display: flex;
    justify-content: space-between;
    align-items: center;
/*    background: #c7c7c7;*/
    height: 100px;
}
.logo {
    height: 50px;
    width: 50px;
    border-radius: 50%;
}
.logo img{
    width: 50px;
    border-radius: 30px;
}
#wrap {
    width: 500px;
    height: 100hv;
    border: 1px solid red;
    background-image: url(background.jpg)

}
```