Applying a Deep Learning Approach for Wheat Rust Disease Detection

By: Mosisa Dessalegn

A Thesis Submitted to the Department of Computer Science and Engineering, School of Electrical Engineering and Computing in partial fulfillment of Masters degree in Computer Science and Engineering

Office of Graduate Studies Adama Science and Technology University
Adama, Oromia, Ethiopia

September 2019

Applying a Deep Learning Approach for Wheat Rust Disease Detection

By: Mosisa Dessalegn

Name of Advisor: Rajeesh Sharma R (Ph.D.)

A Thesis Submitted to the Department of Computer Science and Engineering, School of Electrical Engineering and Computing in partial fulfillment of Masters degree in Computer Science and Engineering

Office of Graduate Studies Adama Science and Technology University
Adama, Oromia, Ethiopia

September 2019

Statement of Declaration

I, the undersigned Mosisa Dessalegn Olana, Student of Adama Science and Technology University, under School of Electrical Engineering and Computing, Department of Computer Science and Engineering, declare that this study is my original work using referenced sources, without unauthorized assist with every part written word by word or in a rephrased manner clearly citing the references. I haven't submitted this study in any other training program for a partial fulfillment.

Name: Mosisa Dessalegn

Signature: _____

This study work has been submitted for examination with my approval as a study Advisor

Name: Rajeesh Sharma R (PhD.)

Signature: _____

Date of submission: _____

Approval of Board of Examiners

We, the members of the Board of Examiners of the final open defense by Mosisa Dessalegn have read and evaluated his/her thesis entitled "Applying Deep Learning Approach for Wheat Rust Disease Detection" and examined the candidate. This is, therefore, to certify that the thesis has been accepted in partial fulfillment of the requirement of the Degree of Masters in Computer Science and Engineering.

Supervisor/Advisor	Signature	Date
Chairperson	Signature	Date
Internal Examiner	Signature	Date
External Examiner	Signature	Date

Acknowledgment

First, I would like to thank my merciful God for the unlimited strength he gave me in this very challenging and complex situations throughout the study, by keeping me with his everlasting and unstoppable love to keep me calm and focused on my work to bring this success. Next, I would like to thank my adviser Dr. Rajeesh Sharma for helping me to bring new ideas and approaches during my difficulties in the study.

I want to express my deepest gratitude to Dr. Mesfin Abebe, for giving me the opportunity to work on this area and being there for me any time I need help from him. He has been giving me his time to help me develop as a professional and researcher, taught me problem-solving skills. I am extremely thankful for his kind assistance that he has been showing me like a best friend, that I will not forget the love and respect he has shown me whenever I needed help from him, He will stay as my example for kindness and optimism in my life and it is a big pleasure to work with him.

My Heartful thanks also go to all my family for being with me and giving me all the support I needed from you and I know it would have been very difficult without you. My mom Lili, My Woman Mira, my brothers Mehari and Chala, my Daughter Hillsun, you are all the reason I was so strong and dedicated to my study and I want to say you all stay in my heart and life forever.

Zelalem Birhan, I will never forget the assistance you were giving me and it was a lot for me, my brother, thank you for the bold and sacrifice help you gave me and may God bless you and give you all the success you want in your whole life.

Finally, I want to pass all my thanks and respect to all my classmates, you were loving and respectful from the start and it was amazing to have this chapter of my life with all of you, especially my SIG members Anteneh and Tadele, I want to say thank you for your support in completion of my thesis work.

Table of Contents

List of Figures

List of Tables

List of Equations

List of Abbreviations

Adam	Adaptive learning rate optimization algorithm
CPU	Central processing unit
CNTK	Cognitive tool kit
Conv	Convolution
CL	Convolution layer
CNN	Convolutional neural networks
ConvNets	Convolutional neural networks
FN	False negatives
FP	False positives
FCL	Fully connected layer
GLM	Generalized linear model
GHz	Giga hertz
GB	Gigabytes
GPU	Graphical processing unit
HD	High definition
K	Kernel size
NumOf	Number of
Num of epochs	Number of epochs
PL	Pooling layer
PPV	Positive predictive value
P	Precision
RAM	Random access memory
R	Recall
RELU	Rectified linear unit
RGB	Red green blue
SNNP	Southern nations and nationalities people
S	Stride

SIG	System Interest Group
train_accuracy	Training accuracy
train_loss	Training loss
TN	True negatives
TP	True positives
val_accuracy	Validation accuracy
val_loss	Validation loss
VGG16	Visual geometry group

Abstract

Crop reduction is the major problem the world is facing currently in ensuring food security and creating a sustainable agricultural system because of different crop diseases. Tracking the health status of crops is a critical job we have to do to prevent the spread of different crop diseases and it should be in a technological manner rather than by the labor force. This study presents developing a model that detects three types of wheat rust disease detection using a Deep Learning approach, especially the Convolutional Neural Network (CNN) by using the image of the wheat crop. These wheat diseases are wheat leaf rust, stem rust, and yellow rust which also differ in the way they affect the crop and in their level of damage. Color code segmentation is proposed in this study to extract only the needed information from the wheat image to identify the healthy crop from the infected one. After conducting more than 200 hundred experiments using different impacting factors such as Learning rate, Dropout, and train test split ratio the study has finally achieved a 99.76% accuracy to detect the wheat rust from healthy crops.

Keywords: *Deep Learning, Convolutional Neural Networks, Color Code Segmentation, Classification, Detection, MosNet, Wheat Rust, Model.*

CHAPTER ONE

1. Introduction

1.1 Background of the Study

Nowadays, technologies deliver to humankind the ability to produce enough food for billions of people over the world. Even though the world is producing a huge amount of crops to ensure food security of the people, there are a lot of factors that are threats in the process of ensuring food security. The threats that basically occur on crops can be with climate changes, pollinators, and plant diseases. Plant diseases are not only threats to global food security, but they also have devastating consequences on smallholding families in Ethiopia, who are responsible for supporting many, in one family. In Ethiopia, this crop is the second[1] most used crop to ensure food security and covers an estimated 17%[2][3] of Ethiopia's total agricultural land use. It is not only a critical crop to smallholder farmers' livelihood but also a means of ensuring food security for millions of people in Ethiopia.

The outbreak of this disease is threatening the crop production gain of the country in wheat-producing regions of Ethiopia, following a long period of El-Niño caused drought across the country. The wheat rust was first[4] spotted in Southern Nations(SNNP) regions and Oromia. The crop-damaging fungus has since spread to different regional states, especially to the northern part of the country, especially to Amhara and Tigray regions. These diseases basically occur by funguses, well known to bring stunting in plants and cause pre-harvest losses between 50 to 100 percent. Wheat rusts are known to cause stunting in plants and pre-harvest losses of between 50 and, in severe cases, 100%[2][5][6]**,** the diseases have been detected in nearly 2,200 communities and on close to 300, 000 hectares of land across the country.

As of 24 October 2016, an estimated 92% of wheat farms in the country are identified as wheat rust affected have been sprayed with fungicides by regional authorities, as a measurement of proactive response by the Government. However, in some cases, it persists to survive from the fungicide if it is not inspected and action is taken in time**.** One of the issues was the fungicide spray equipment and protective gear are in limited supply in parts of the country, not only affecting the timely control of the outbreak but also causing serious

problems on the health and safety of farmers and experts who spray the fungicide. In addition, many farms required spraying more than once. This shows the importance of developing effective methods to help these farmers by detecting and recognizing the rust leaves, in turn, this can be implemented on drones or robots to detect and spray the antifungal chemical without any involvement of the farmers. Therefore I propose a deep learning methodology to detect if a wheat crop is infected by one of the three rust types of wheat or it is healthy. This study delivers a CNN model that can be used on large governmental and private agricultural fields for early detection and measurement, but not for statistical data on which kind of diseases happened on the field, which is already suggested on future works.

Deep Learning can be summarized as a sub-field of machine learning [7] that study statistical models called deep neural networks. The latter are able to learn complex and hierarchical representations from raw data, unlike the artisanal models which consist of an essential features engineering step. This scientific field has been known by a variety of names and has had a long history of research, performing an alternative wave of excitement and periods of oblivion. Early work on deep learning, or rather Cybernetics, as it was called at the time, was done in the 1940s and 1960s, and describes biologically inspired models such as Perceptron, Adaline, or Multi-layer Perceptron. Then, a second wave called Connectionism came in the years 1960-1980 with the invention of "backpropagation". This algorithm persists today and is currently the algorithm of choice for optimizing deep neural networks. The ambition for creating a feasible neural network for detection and recognition of images that have automatic feature extraction and translation invariance promotes a new category of neural networks called Convolution Neural Network (ConvNet). These networks have special types of processing layers in the extractor module that learns to extract discriminated features from the given input image.

The extracted features are then forwarded to the classification module that resembles the Multi-Layer Feed Forward Neural Network in terms of architecture. The advantage of having a Convolutional Networks (ConvNets) is that it requires a very small trainable number of parameters that support the sharing of weight and partially connected layers. With a reduced number of trainable parameters, a ConvNets design also makes the model invariant to translation, making it a state of the art for image detection and recognizing tasks.

1.2 Motivation

More than 85% of the Ethiopian population's livelihood relies on agriculture and our farmers should be profitable from what they have done and should achieve to the best of their hardworking. Additionally wheat is a kind of versatile crop that is used to ensure food security of the society, not only that, it is also the subject of millions of dollars import from foreign countries, as an example Ethiopian government approved[8] 4 billion birr **(137,931,034.48 USD)** to import wheat for the internally displaced and drought-affected people in the country, which is a fire on gas to the shortage of dollar currency the country is facing currently. Since the Ethiopian government is planning to halt the wheat import after a couple of years to expand large irrigation farms across the country, this study will be opening a door for other forthcoming studies to advance on crop image-based disease detection technologies. Previously done researches also have some drawbacks that should be addressed like being constrained on single leaves, using images with homogenous backgrounds, manual cropping, and Segmentation of images, using laboratory images, which decreases dynamicity and efficiency of the models.

That's why I wanted to make my own contribution by designing a model that can detect these diseases effectively and efficiently and help to take immediate measurements before the disease brings total damage to the crop, specifically on wheat rust diseases.

1.3 Statement of the Problem

Ethiopia's economy is dependent on agriculture, which accounts for 40 percent of the GDP, 80 percent of exports, and an estimated 75 percent of the country's workforce. But, the outbreak of different diseases on different crops is the most difficult challenge the country is facing, which brings different socio-economic problems such as food insecurity, market inflation hard currency shortage because of the necessity of imports to cover it in import from foreign countries. Wheat is the second most important crop for food security [2] in Ethiopia and covers an estimated 17% of Ethiopia's total farmland use, according to the country's statistical agency. The crop is not only a means of financial income to smallholder families but it is critical for the food and nutrition security of tens of millions of Ethiopians. Ethiopia is expecting a production of 4.6 million tons in 2019/20, which is an increment of 0.1 ton only, which is still not good enough to cover wheat consumption of the country[9]. Even though

yellow rust the most commonly damaging and widespread in many areas, stem[10] and leaf rust have also been detected across the country many times.

The biggest challenge in Ethiopia to deal with this problem is that it could not be addressed easily with the help of technology at an early stage so that it cannot spread through all the fields; because every premature detection is done by deploying a large number of experts in the field and inspect it manually.

From study wise, there are a lot of studies worked on different kinds of plant disease detections. But, with all their performances, at the same time, they do have their own drawbacks as no research has a perfect ending. This study tried to figure out some of the drawbacks that occurred in previous researches. Here are some basic problems tried to be figured out in this study:

- Using laboratory images and when the model is tested in the real field or cultivation area the accuracy of the models degrades
- Being constrained to single leaf images that are with a homogenous background.
- Using manually cropped images specifically focusing only on the disease area which decreases the efficiency when it is taken to a real-world test. These drawbacks are discussed in Chapter two (Related works) in detail with their references.
- Being dependent on previously trained models which brings future insecurity.

1.4 Research Questions

RQ1: How can the model detect wheat rust disease with high speed and accuracy?

RQ2: How to improve the detection with images captured at any state?

RQ3: How the model can work on images at any state, without any single leaf constraint and heterogeneous background?

1.5 Objectives

1.5.1 General Objective

The general objective of this study is to produce an efficient, effective and more accurate wheat rust detection model using Convolutional Neural Network (CNN) which can easily detect whether a wheat crop is infected with rust diseases or not.

1.5.2 Specific Objectives

- Preparing the necessary data set by using different mechanisms such as image preprocessing and Augmentation.
- Analyze the Healthy wheat and Rust Infected wheat dataset and evaluate the effect of different pre-processing techniques.
- Designing the Standard and HUF depth separable convolutional neural network model.
- Evaluate the effect of learning parameters including learning rate, batch size, number of epochs as well as train test split ratio and dropout.
- Give a comparative evaluation of the designed model and discuss its potential applicability.

1.6 Significance of the Study

The study provides different benefits for different stakeholders and other researchers. After the evaluations of the results, it can be applied in different areas:

- The study opens a door for different researchers in Ethiopia, who have the interest to work on Agro Technologies.
- Agricultural Research Institutes can be beneficiary from this study.
- The model also can be applied to other crop disease detection which can be detected based on color segmentation.
- Private industries working in agriculture can also get benefit from the study.

1.7 Scope of the study

The study focuses on designing and implementing a model that can easily detect if a wheat crop is infected by one of the three rust types or if it is healthy, but the model doesn't identify which type of the disease it is, because of a dataset limitation. The model is only applicable to wheat crops.

1. 8 Limitations of the Study

The highest limitation in this study is all about the dataset and better computational power. Even if this study has achieved a model with high efficiency, it could have progressed beyond the scope which is identifying a type of disease that happened, but with a small amount of data, it is impossible to achieve a CNN model with high efficiency.

1.9 Organization of the Study

This study is organized as follows:

Chapter Two discusses different literature reviews which gives a brief explanation of wheat and wheat rust diseases, plus on the general knowledge of Machine learning, Deep learning and Convolutional neural networks with previous related works done by other researchers.

Chapter Three explains some important research methodologies and techniques used in the research progress, to design and implement the model, data preprocessing and segmentation technique used in this study is discussed.

Chapter Four discusses the proposed model implementation which explains the proposed standard CNN model which is implemented from scratch and also the design of the model applying it with HUF Depth Separable convolution explained in section *[general description of MosNet model* is elaborated in section [*4.1.1 MosNet Model*] which is explained in the above schematic diagram, below, there is a description of each bloack in the architecture.

- The three above layers(Conv Layer1, Conv Layer3, Conv Layer3) are input layers expecting images from their previous layers, as Example Conv Layer 1 takes input image as (None, 32,148,148): None indicates the number of images the layer can take, but in case of this None indicates the layer can take as many as possible images as it can , so it is indicated as None by default. 32: indicates feature map or the number of hidden layers the layer has, 148: is the size of the input image, but originally our image is resized into 150 by 150 dimension, and it becomes 148 because we are using 2X2 pooling and it minimizes 2 from the original image. The size of input image will decrease by half in eachas the number of convolution layers increase, Example: 148 is decrease into 74, which will be 76-umber of pooling which is 2, 76-2=74., the same procedure takes place in each layer.
- Each input layer has Convolution layer, ReLu activation layer and pooling Layer which are explained in sections [*2.8 Convolutional Neural Networks*] and [*2.9 Activation*].
- Sigmoid activation function is an output activation function which is used to predict probability like predictions like (0 and 1➔(“Healthy” and “Infected ” in case of this study))

- In the output layer our output features are flatten from grid format to flatten information to feed it for the output layer so that the layer can learn each features from the previous layer in a sequential manner. Nextly, there are two dense layers, one for the number of hidden layers in the output layer and one for the number of classes we have for prediction. Example the first dense layer is Dense(64) means we have 64 feature maps or hidden layers and the next dense layer Dense(2) means there 2 classes for prediction whicha are ‘’Healthy’’ and “Infected” classes.

4.1.2 HUF Depth Separable Convolutional Architecture*].*

Chapter Five discusses intermediate implementations which are found before we get our final result, these implementations are like augmenting data, resizing and the color code segmentation sample progressive results are shown to give clear understanding for the reader of the paper.

Chapter Six discusses the results found in my implementation and gives reasonable discussion depending on the results found and which features have a better effect on the efficiency of the model found. The resulting model is evaluated based on different comparable parameters that have impacts on the performance of the model

Chapter Seven gives a reasonable conclusion depending on the results found in Chapter six and also gives some constructing recommendations on what the government should do to take this study to real-world implementation and how to enhance it to take the country’s agricultural development by using it on other crop species, through conducting more researches on other crops. Finally, some future works are discussed especially, on how any governmental and private institutes should change the culture of handling data, so that that would be helpful for researches and reduces the challenges they face to collect data.

CHAPTER TWO

2. Literature Review

2.1 Wheat and Wheat Rust Diseases

2.1.1 Wheat in Ethiopia

Wheat is one of the world's most commonly consumed cereal crops. It comes from a type of grass (Triticum) that is grown in a lot of varieties worldwide. Basically, Wheat is the second most prominent crop for food security and subject to millions of tones[11] of import with millions of hard cash in dollars in Ethiopia. Despite the importance of the crop in Ethiopia it is the most commonly rust infected crop, basically with three most common rusts: Leaf rust, Yellow (stripe) rust, and Stem rust.

Table 2. 1: Wheat import data of Ethiopia since 2010 [12]

Year	*Import amount in (MT)*	*Growth Rate from the previous year*
2010	700	-34.58 %
2011	1300	85.71 %
2012	1100	-15.38 %
2013	1000	9.09%
2014	1075	7.50%
2015	2600	141.86%
2016	1100	-57.69%
2017	1500	36.36%
2018	1300	13.33%
2019	1600	23.08%

2.1.2 Wheat Rusts

Wheat rusts are fungal diseases that can cause devastating losses[4][13] to yields of the crop. In Ethiopia, where wheat is a key part of the diet and of the economy, a poorly performing crop can have a huge impact. Yet fungicides are only half of the answer. Spray a field at the wrong time or in the wrong area, and not only will the crop still be at risk, but the cost of spray

will also have been wasted. Wheat rust is the most common fungal disease that is challenging Ethiopia's agricultural economy currently. There are three most common wheat rusts, which are very severe and have the capability of damaging 50 to 100% of the crop unless it is controlled as soon as it occurs on the crop. These rusts are Leaf rust, Yellow rust, and Stem rust. From these rusts, Stem and Yellow rusts represent the greatest disease threat to Ethiopian wheat farmers; with both diseases capable of causing huge losses. Even though yellow and leaf rusts are both types of leaf rusts, they are treated as different kinds of rust races, but most of the time they are identified in laboratories, because of the similarities they make on the structure and color of the fungus on the crop.

2.1.2.1 Leaf Rust

Leaf rust, also known as brown rust, is caused by the fungus[14][13] (Puccinia triticina). This rust disease occurs wherever wheat, barley, and other cereal crops are grown. Leaf rust attacks foliage only. Symptoms are dusty, reddish-orange to reddish-brown fruiting bodies that appear on the leaf surface.

2.1.2.2 Yellow Rust

Yellow rust (Puccinia striiformis)[13], also known as wheat stripe rust, is one of the three wheat rust diseases principally found in wheat grown in cooler environments. Such locations are generally associated with northern latitudes or cooler seasons. The disease usually occurs early in the growing season, when temperature ranges between 2 and 15 °C (36 and 59 °F); but it may occur to a maximum of 23 °C (73 °F). Even though Leaf and Yellow rusts are types categorized under leaf rust family, they are in different races, but they are sometimes difficult to recognize them easily by normal visual inspection and they need to be tested in Laboratory**.**

2.1.2.3 Stem Rust

The stem, rust is caused by the fungus like other rust types, (Puccinia graminis**)**[13] and is a significant disease that affects wheat crops. Crop species that are affected by the disease include bread wheat, durum wheat, barley, and triticale. These diseases have affected cereal farming throughout history.

Figure 2. 1: Sample Healthy Wheat Image

Figure 2. 3: Sample Stem Rust Imag

Figure 2. 2: Sample Leaf Rust Image

Figure 2. 4: Sample Yellow Rust Image

2.2 How to Fight Wheat Rust?

In Ethiopia, Detecting and protecting plant diseases are all performed by waging human labor and experts on the agricultural fields, which is almost very costly and time-wasting. Beside these expenses sometimes in large agricultural fields, we may lose from 50 – 100% of the crop, before we detect and take immediate actions against the rust diseases, since it is done traditionally by human labor, as the expense of the human labor to cover all the fields in short period of time to spray fungicides to the infected crops. Fortunately, in the case of wheat rust diseases, we use one type of fungicide to kill all three types of rust, which is called with its scientific name **"Vivax"** chemical as the information from Research institutes shows and explained in **[***Appendix A***]**, even though it is still done by using human experts on the field.

2.3 Image in Computer Vision

Image is a binary representation of visual information that can be represented into the two-dimensional axis which has height, and width, and it can also be represented as a collection of pixels that form visual information, such as drawings, pictures, logos, and graphs. In the case of this study, there are two types of image sets.

2.3.1 Training Data(Image)

Training data(image set) is a data that is used to train or fit a model, and it is not recommended to use training data, because it is more likely biased since the model already has seen the data during training the model.

2.3.2 Test Data (Image)

It is a set of unbiased or unseen data used to test our model and used to provide an unbiased evaluation of our final model. Test data should be data that is different from the data that is used for training the model. Using test data helps our model to be more robust and effective because the model evaluates the result of unseen data.

2.4 Image Processing

Digital image processing is a branch of computer vision in which images are converted into an array of integers or pixels[16][17] or processing a digital image by using a digital computer so that we can get an enhanced image which we can extract useful information from it.

2.5 Image Detection

Detection is the process of detecting or noticing something in the real world. So Detection in this study is used to refer to detecting if a wheat crop is infected with wheat rusts or healthy.

2.6 Data, Data Mining, and Big Data

Enethough it is not the part of this study before we dive into Deep Learning, we need to have some understanding of data and some of its related issues because deep learning is all about a sub-field of machine learning which can only be achieved with a large amount of data set. Data is any kind of raw data that can be processed into meaningful information, and this raw data can be a text, image, voice and any unstructured data that can be processed to meaningful information.

2.6.1 Big Data

Big data deals with a huge amount of data that can be structured, semi-structured, and which has 5 fundamental V's. The 5 V's[18] are Volume, Variety, Velocity, Veracity, and Value. Most of the time more than 1TB of data is known to be big data.

2.6.2 Data Mining

Data mining refers to extracting knowledge from a huge[19] amount of data through the process of discovering various types of patterns that are derived from the data and which are accurate, new and very useful. Here are some comparisons between Big data and Data mining.

Table 2. 2: Comparison table for Big Data and Data Mining

Big Data	*Data Mining*
Focuses on lots of relationships between data.	Focuses on lots of details of data.
Is a huge picture f data	Is a close up view of data
Explains why of the data	Explains what is about the data
Deals with a huge amount of data sets	Can be used for both small and large data sets
Is a concept than a precise term	Is a way for analyzing data

2.6.3 Data Mining Vs Machine Learning

Data mining is the process of extracting knowledge from large amount of data, whereas machine learning is a way to invent new algorithm the knowledge[19][20] and machine

learning uses data mining techniques and different learning algorithms to build a model which predicts future from the extracted knowledge, which is like teaching a machine to predict an output based on some knowledge learned from data. Here are some basic Comparisons of Data mining and machine learning.

Table 2. 3: Comparison table for Data mining and Machine learning

Tips for Comparison	Data Mining	Machine Learning
Meaning	Extracts knowledge from data	Find a new algorithm from processed data and previous experience
Responsibility	Used to get rules from existing data	Teaches machines to understand from given rules.
Nature	Needs human interference to extract rules	Once designed, it is self-implementing
Abstraction	Abstracts from a data warehouse	Reads machine
Origin	Unstructured data from Traditional databases	Existing data and algorithms

2.7 Why Deep Learning over Machine Learning?

2.7.1 Machine Learning

Machine learning is a branch of Artificial intelligence which is a way of making machines intelligent to learn from a bunch of learning algorithms and apply what they have learned to make brilliant decisions like human being, but as long as they are still machine and in some situations they need human intervention, because they work only for what they are designed for no more no less. That's why we need deep learning, which gives us a bit better performance.

2.7.2 Deep Learning

Practically, it is[21][22] a subset of machine learning with more power and flexibility than traditional machine learning approaches. The biggest advantage of **Deep Learning** algorithms is that they try to learn high-level features from data in an incremental manner. This prevents the need for manual feature extractions which is done by human intervention in traditional

machine learning. Deep learning emulates the human brain[20], and it works with unstructured data, especially for applications like computer vision(image processing) and speech recognition.

2.7.3 When to Use Deep Learning?

Deep learning performs more and more when we have a large data size and good computational performances. It outperforms other approaches when it comes to really complex problems like image classifications, natural language processing, and speech recognition. Basically, we need to have two crucial things in Deep Learning. The first one is large data size to train our model and the second one is a good computational device because deep learning learns up to millions of parameters in a model and our device should be capable of overcoming those bunches of parameters easily.

2.8 Convolutional Neural Networks

CNN is the class of deep neural networks (deep learning), most commonly used for image processing tasks. It can obtain effective representations of the original image, which makes it possible to recognize visual patterns directly from raw pixels with little-to-none preprocessing. ConvNets are constructed with processing layers namely, Convolutional, pooling and fully connected or Dense. In CNN, each layer takes information from the previous layer[23] and processes it to the succeeding layer, especially CNN works best for image data[24], basically large image data set. The Design and layout of CNN the layers are decided by the model designer. The design depends on the type and complexity of the problem to be solved as well as the expected output of the network. The output of a ConvNets can be either a posterior probability of a sample belonging to various classes, or discriminant features that can be used with any other classifier.

Figure 2. 5, shows the general architecture CNN where a digital image is used as an input in the first layer of CNN which is the Convolution layer and processed by the next layer which is the pooling layer before it is finally processed by the Dense Layer or Fully connected Layer. The Convolution and pooling layer performs the feature extraction module whereas the last fully Connected layers perform the Classification part. Basically, there is no limited boundary between these modules and Fully connected Layers may form a part of the classification module, however, the classification module can consist of one or more fully connected layers.

This fully connected layer is a layer that is found as a last element of the architecture which is also called the output layer calculates the final probability of the given input data or image given to one of the classes. The removal of the output layer along with other fully connected layers means transforming the CNN into only feature extractor without a classifier.

The operation of all three types of processing layers, namely convolutional, pooling and fully connected is described in the following sections.

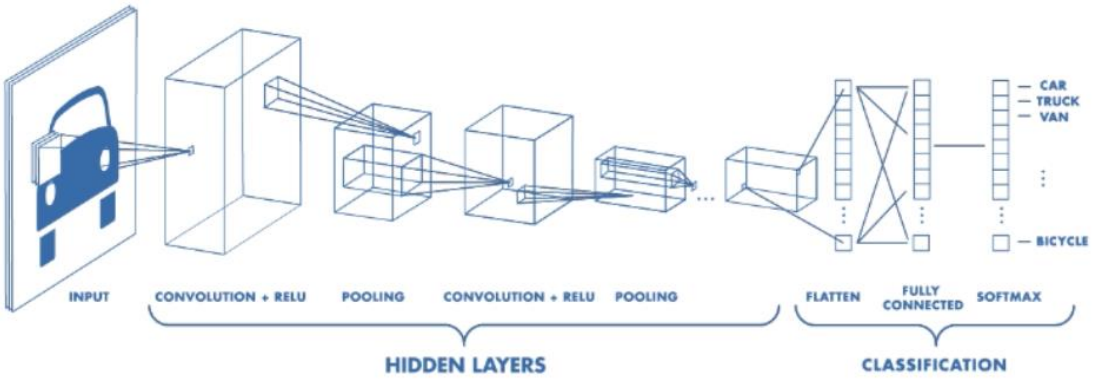


Figure 2. 5: General Architecture of CNN[25]

2.8.1 CNN Layers

2.8.1.1 Convolution Layer (CL)

In computer vision, images are read as pixels and are represented as a matrix of (NxNx3) — (height by width by depth respectively). Colored Images make use of three channels (RGB), so that is why we have a depth of 3 [26][27]**.** Convolution filter in basic CNNs is a Generalized Linear Model (GLM) for the underlying local image patch. It works well for abstraction when instances of latent concepts are linearly separable. Here we introduce some works which aim to enhance its representation ability. The main processing component of this layer is a filter or mask which is a matrix of weights. This mask is applied to a specified region of the feature map. The Convolutional Layer makes use of a set of learnable filters. A filter is used to detect the presence of specific features or patterns present in the original image (input). It is usually expressed as a matrix (MxMx3), with a smaller dimension but the same depth as the input file.

The convolution operation is mathematically defined as:

$$S[i, j] = \Sigma_m \Sigma_n [m, n] . K[u-m, j-n]$$

Equation 2. 1: Convolution Operation

where I is the input image with dimensions m and n, K is the kernel, and S is the resulting output at location i and j. Applying the convolution operation provides several benefits that are based on three important ideas: sparse interactions (or sparse weights), parameter sharing, and equivalent representations. The term sparse interactions means that fewer connections between input and output values exist. Therefore, fewer parameters are needed which reduces the memory requirements and improves the statistical efficiency. This is achieved by applying smaller kernel sizes than the input image because fewer operations are required to produce the output. In contrast, traditional neural network layers, which do not employ convolutions, use matrix multiplication to describe the relationships between each input value and each output value. This means that all outputs are connected to all inputs.

Convolutions are very efficient operations to describe transformations of small receptive fields across input images. Parameter sharing refers to using the same values for more than one function in a network. In traditional neural networks, each element of the weight matrix is used exactly once for computing the layer output. Each value of the kernel is used at every position of the input. The parameter sharing used by the convolution operation means that rather than learning a separate set of parameters for every location, only one set is learned. This does not affect the runtime but further reduces the storage requirements of the model to k parameters. When convolution is applied to images, convolutions create 2D feature maps that show where certain features appear in the input. Parameter sharing causes the layer to be equivariant to translation, i.e. if the input changes, the output changes in the same way. If the object in the input is moved, its representation will move the same amount in the output.

2.8.1.2 Pooling Layer (PL)

The pooling layer is an important component of CNN which lowers the computational burden, by reducing the number of connections between convolutional layers[26][28], which is basically also used controlling overfitting by progressively reducing the spatial size of the network**.** Unlike the convolution layer, the PL does not alter the depth of the network, the

depth dimension remains constant. Max pooling is the most commonly used pooling layer in the Convolutional neural network.

2.8.1.2.1 Max-Pooling

As the name states; it takes out only the maximum from a pool. This is actually done with the use of filters sliding through the input; and at every stride, the maximum parameter is taken out and the rest is dropped.

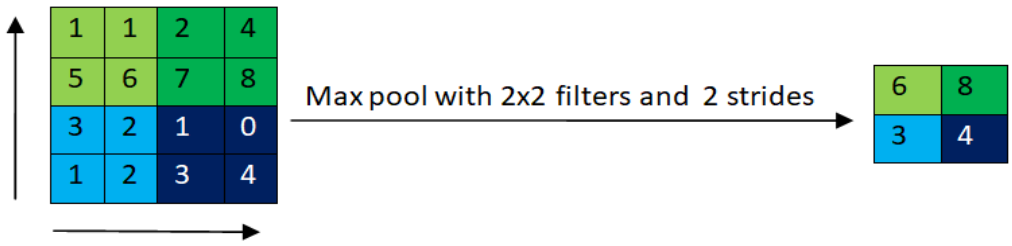


Figure 2. 6: Max Pooling

The most important parameters to play:

- Input: H1 x W1 x DepthIn x N
- Stride: Scalar that controls the number of pixels that the windows slide.
- K: Kernel size

The output will be:

$$W2 = (W1 - K)/S + 1$$
$$H2 = (H1 - K)/S + 1$$
$$DepthOut = DepthIn$$

Equation 2. 2: Output Image Equation

2.8.1.3 Fully Connected Layer (FCL)

The last layers in the network are fully connected, meaning that neurons of preceding layers are connected to every neuron in subsequent layers. This mimics high-level reasoning where all possible pathways from the input to output are considered. Linear or Fully Connected Mathematically, we can think of a linear layer as a function that applies a linear transformation on a vectored input of dimension I and output a vector of dimension *O*. Usually the layer has a bias parameter. In this layer, the neurons have a complete connection to all the activations

[26][29] from the previous layers. Their activations can hence be computed with a matrix multiplication followed by a bias offset. This is the last phase for a CNN network.

The Convolutional Neural Network is actually made up of hidden layers and fully-connected layer(s).

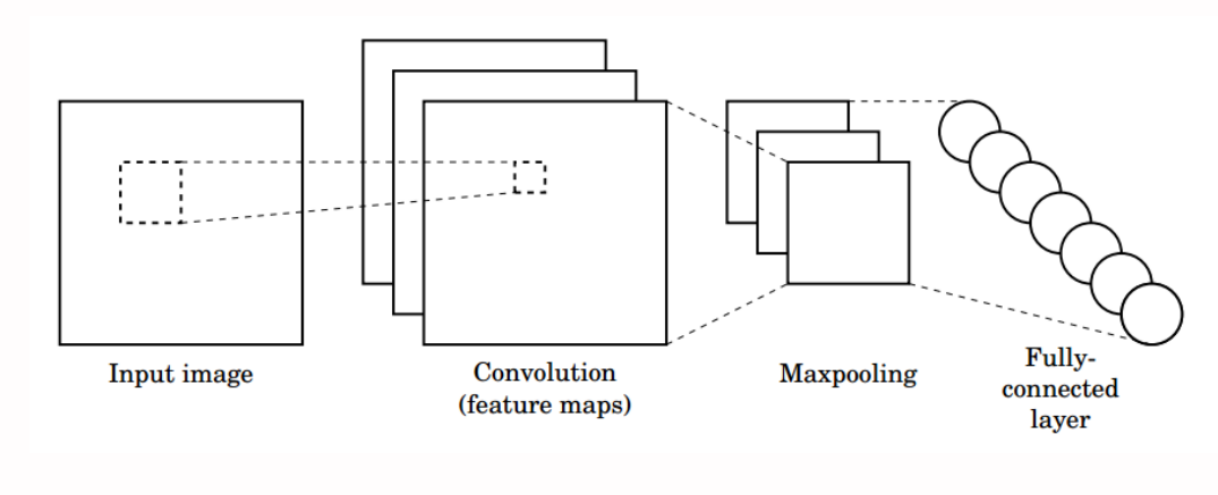


Figure 2. 7: Fully Connected Layer of CNN [30]

2.9 Activation

The activation layer controls how the signal flows from one layer to the next, emulating how neurons are fired in our brain. Output signals which are strongly associated with past references would activate more neurons, enabling signals to be propagated more efficiently for identification. ConvNet is compatible with a wide variety of complex activation functions to model signal propagation, the most common[31] function is the Rectified Linear Unit (ReLU), which is favored for its faster training speed. The capacity of the neural networks to approximate any functions, especially non-convex, is directly the result of the non-linear activation functions. Every kind of activation function takes a vector and performs a certain fixed point-wise operation on it. Some of the activation functions explained in the following subsection.

2.9.1 Rectified Linear Unit (ReLu)

The ReLU is the most used activation function in the world right now. Since it is used in almost all the convolutional neural networks or deep learning[26][31]. It does not suffer from

the vanishing or exploding gradient. Another advantage is that it involves cheap operations compared to the expensive exponentials.

The ReLU has the following mathematical form:

$$f(x) = \max(0, x)$$

Equation 2. 3: ReLu Function

2.9.2 Sigmoid and Softmax

There are two most common activation functions that are used in deep learning approaches, which are softmax and Sigmoid. A **Sigmoid**[7][31] **function** is a mathematical function having a characteristic "S"-shaped curve or **sigmoid curve**. Often, we use a sigmoid function when our problem is solved with binary classification, which means our result lies between zero and one(**0 & 1**) [32].

It takes a real value and squashes it between 0 and 1. However, when the neuron's activation saturates at either a tail of 0 or 1, the gradient at these regions is almost zero. Thus, the backpropagation algorithm fails at modifying its parameters and the parameters of the preceding neural layers. The sigmoid activation function is the most suitable activation function for binary classifications. Softmax is different because they are most effective in multi class classifications.

2.10 Overfitting and Underfitting

Overfitting and underfitting are the two most critical problems in neural network models and these problems happen because of different reasons

2.10.1 Overfitting

Overfitting happens when a model learns effectively[33] on the training data but performs poorly on validation(hold out data).

2.10.2 Underfitting

Underfitting happens when the model fails to effectively[33] learn the problem and performs poorly on training data and also does not perform well on validation data. Therefore if we see a model that performs badly both on training and validation data, it should be known that the model suffers from underfitting and need to try to change our approach or by updating our parameters.

2.11 Dropout

'Dropout' is a technique that helps[34] to avoid overfitting by setting the outputs of hidden neurons to zero with a pre-set probability. The neurons which are set to zero do not contribute to the forward pass or back-propagation, i.e. they are 'dropped out'. Due to these 'dropped-out' neurons, the network samples a different architecture every time. This forces neurons to learn more robust features as they cannot rely on the presence of other neurons. A dropout ratio of 0.5 doubles the number of iterations required to converge but reduces overfitting.

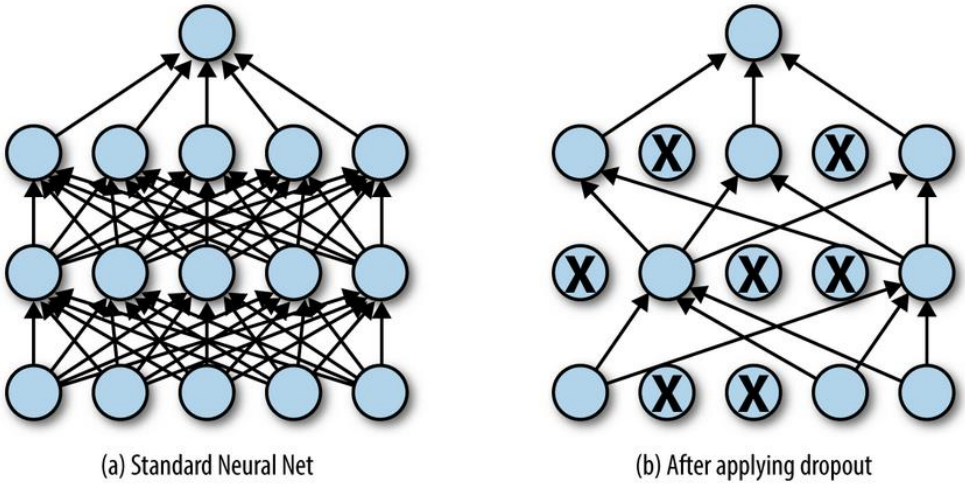


Figure 2. 8:Dropping neurons randomly from the network[35]

2.12 Batch Normalization

This layer quickly became very popular mostly because it helps to converge faster[7]. It adds a normalization step (shifting inputs to zero-mean and unit variance) to make the inputs of each trainable layer comparable across features. By doing this it ensures a high learning rate while keeping the network learning. In addition to that, it also allows activations functions such as Sigmoid to not get stuck in the saturation mode (e.g. gradient equal to 0).

2.13 Tools and Techniques.

2.13.1 Language

2.13.1.1 Python

Python is an interpreted high-level, general-purpose programming language. It is an easy and powerful[36] programming language that is dynamic and easy to catch up with. As it's a lot of built-in libraries for Artificial intelligence and Machine Learning, well documented, simple enough to catch up with, python is preferred in this study. Some of the libraries are TensorFlow, Keras, and sci-kit–learn. In this study, to produce the result, python 3.7.3 is used and runs on Windows 10, 64- bit operating system, with 4 GB RAM and 3.20 GHz of processor.

2.13.2 Built-in Libraries

2.13.2.1 Tensor flow

Tensor Flow is a Python library for fast numerical computing created and released by Google. It is a foundation library that can be used to create Deep Learning models and it's an open-source library for fast numerical computing, which is an end to end[37] open-source machine learning(ML) framework. Tensor Flow was designed for use both in research and development and in production systems. It can run on single CPU systems, GPUs as well as mobile devices and large scale distributed systems on hundreds of machines. So we can say TensorFlow is a brilliant tool, with lots of power and flexibility. However, for quick prototyping work, it can be a bit verbose.

2.12.2.2 Keras

Keras is a high-level neural network, API[38], written in Python and capable of running on top of TensorFlow, CNTK or Theano. It was developed with a focus on enabling fast experimentation. It is a model-level library providing high-level building blocks for developing deep learning architectures. It does not allow low-level operations like tensor manipulation and differentiation, Instead of that, it uses a specialized, well-optimized tensor library, serves as a backend engine for Keras. Keras allows for easy and fast prototyping through user-friendliness, modularity, and extensibility.

2.14 Related Works

Previously, several machine learning applications are used including traditional machine learning and deep learning approaches in agricultural problem solving, using image processing techniques. Even though machine learning applications advance in such fields, efficiency from different angles still remains a question.

Amanda Ramcharan et al.[39] used 15000 manually cropped RGB images into a single leafs to detect only the infected area of the crop. These images used to classify three types of cassava leaf diseases, by applying a different set of a train, validate and test split ranges, in which 10 percent is used for validating the model, and other are used for train and test of 10/80, 20/70/, 40/50 and 50/40 percents respectively. They also used Google InceptionV3 and achieved 98% of accuracy, but at the same time, this study cannot achieve good performance when we have random images, which are captured under random conditions, which will not allow the model to be applied in real-world conditions.

David Hughes et al.[40], used GoogleNet and AlexNet models to train 54,306 images from the plantVillage website, in which GoogleNet performs better and consistently with a training accuracy of 99.35%. But in this study, the accuracy degrades to 31.4% when it is tested using images taken under conditions different from images used for training the model. In this study, we have used three train test split distribution of 75/25, 60/40, and 70/30 in percent with three types of image types which are RGB color images, grayscale images, and segmented images.

Konstantinos Ferentinos et al. [41] used an automated pattern recognition using CNN is used in, to detect three types of plants and their diseases, based on simple leaves of the plants, using the 5 basic CNN models, from pre-trained models. The study uses 70,300 for training and other 17,458 images for testing, with a standard size of 256x256 pixel size. Models fine tuned[42][40] and *Alvarez Gila et al.* [43] in these studies are AlexNet, AlexNetOWTBN, GoogleNet, OverFeat, and VGG, with the highest accuracy on the training set with 100% and 99.48 percent on the testing set in VGG model.

2.15 Summary

Table 2. 4: Related work Summary Relevant to this Study

Ref No.	Technique	Accuracy	Gaps
[40]	Applied fine-tuning using pre-trained deep learning models.	99.35%	The study constrained to single leaf images with homogenous background
[41]	Applied different pre-trained networks to train it on laboratory images	99.48%	Accuracy degrades when the model tested on images from real cultivation field
[43]	Fine-tuned Resnet50 model to	87%	Images are segmented manually by expert technicians.
[39]	Transfer Learning using InceptionV3	96%	Images are manually cropped to a single leaflets

As shown in *Table 2. 4*, there are some gaps that should be addressed to achieve a better performing model. As the table shows the previous studies lack heterogeneity in background and image nature, real-world applicability and constrained to single leaf images. Therefore this study solves the gaps shown in the table including adding some values on the efficiency of the model, by achieving high accuracy.

CHAPTER THREE

3. Research Methodology

3.1 General Approaches

Here are the general procedures I have followed to develop my own MosNet model.

- The has been necessary data collected to train the model from different agricultural research institutes (Kolomsa (Arsi), Bishoftu and Ambo).
- The necessary tools used to develop the models are downloaded from their sites and set up on the computer to be trained on.
- Data Augmentation is used to generate a large amount of training data from a small amount of data since it is a must condition to have a lot of data in Deep Learning.
- Images used to train the models are preprocessed using different preprocessing techniques like resizing the images to a uniform standard so that the model can get normalized and uniform data to train and validate the data set.
- After preprocessing, the images are segmented by using the color code the diseases posses, which is proposed in this study.
- The MosNet is designed and implemented using the Segmented images.

3.2 Training from Scratch

3.2.1 Initialization

Choosing the proper initialization for each layer is the key to achieving high performance. Initializing the weights in an appropriate way can be significantly influential to how easily the network learns from the training (as it effectively preselects the initial position of the model parameters with respect to the loss function to optimize in designing the model to experiment with the network parameters are initialized with Xavier sometimes called Glorot initialization. The key idea behind this initialization scheme is to make it easier for a signal to pass through the layer during forward as well as backward propagation, for a linear activation this also works nicely for sigmoid activations because of the interval where they are unsaturated is roughly linear as well. It draws weights from a probability distribution (uniform or normal) with variance.

3.3 Data Collection

In deep learning, data is the most crucial thing that we need to take care of. Deep Learning algorithms are different from other traditional machine learning algorithms in the size of the data they use, that they need a huge amount of data for training. In this study we have used three classifications of data sets for implementing our model which are training set , validation set and test set. The data set collection process takes place through travelling to three agricultural research institutes in Ethiopia namely, Kolomsa Agricultural Research Institute, Bishoftu Agricultural Research Institute and Ambo Agricultural Research Institute. The Image data collected totally are 192 original images which are augmented to increase the size which is discussed in the next section.

3.4 Data Augmentation for Increasing Dataset Size

Overfitting happens when we have too few samples of training data, and Data Augmentation is the best way to fight this overfitting [7] when we work on image data. It is a method of boosting the size of the training set so that the model cannot memorize all of it. This can take several forms depending on the dataset. For instance, if the objects are supposed to be invariant to rotation such as galaxies or planktons, it is well suited to apply different kinds of rotations to the original images. In this study there were totally 192 original images and these images are augmented to 2113 images using 10 features which are discussed in *[4.3.1 Increasing Data Set size with Data Augmentation]*.

3.5 Pre–Processing to resize Images

Images needs to be resized before they are fed to the model, so that the model can get a uniform image size format and will have the a uniform response time during the real worls implementations. There is no a standard set to resize our images but we can resize it based on the computational resources we have, because as the size of image increased, it needs more computational resources to be processed. In this study we have resized out images into 150X150 height and width dimension.

3.6 Optimization Algorithms

The loss function of a convolutional neural network is highly non-convex. Hopefully, the latter is also fully derivable so that gradient-based optimization algorithms can be applied. However, convolutional neural networks are usually made of millions of parameters. Thus, only the first-order derivatives are used in practice most widely used first-order optimization algorithm is Gradient descent. These algorithms tell us whether the function is decreasing or increasing at a particular point basically it gives a tangential line to a point on its error surface. The second derivative which is also called Hessian is a Matrix of second-order partial Derivatives. Since the second derivates are costly in term of memory and computational effort are not used much.

3.6.1 Gradient Descent

Gradient descent[44] is a way to minimize an objective function (θ) parameterized by a model's parameters θ E $R2$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla\theta$ (θ) with respect to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum. In other words, we follow the direction of the slope of the surface created by the objective function downhill until we reach a valley.

3.6.2 Rmsprop Algorithm

Rmsprop is an unpublished[45][46] optimizer but, a kind of optimizer that became the most popular and effective. Rmsprop is similar to gradient descent with momentum. It increases our learning rates by restricting the vertical oscillations, and converging the horizontal oscillations faster so that our model can learn fast and effectively.

3.7 Performance Measurement Methods

3.7.1 Loss Function (objective function)

The loss function is the crucial element[47] in artificial neural networks, which is used to measure the inconsistency between predicted value (y^) and the actual label (y). It is a non-negative value, where the robustness of the model increases along with the decrease of the value of loss function. The loss function is the hardcore of empirical risk function as well as a significant component of structural risk function.

3.7.2 Cross-Entropy

Cross-entropy loss, or log loss, measure the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverge from the actual label. The Mathematical computation behind[48] cross-entropy is tough, but fortunately, it is easy to understand and implement it.

3.8 Regularization Approaches

Deep and large enough neural networks can memorize any data. During training, their accuracy on the train set typically converges towards perfection while it degrades on the test set. This phenomenon is called overfitting. Thus different regularization approaches used to handle this overfitting problem.

3.8.1 Dropout

Dropout is one of the most effective and most commonly used regularization techniques for neural networks, developed by Geoff Hinton and his students at the University of Toronto. Applied to a layer, consists of randomly dropping out or setting to 0, a number of output features of the layer during training. In MosNet model Dropout is used to minimize the overfitting that may happen during the model training.

3.8.2 Early Stopping

It consists of stopping the training before the model begins to overfit the training set. In practice, it is used a lot during the training of neural networks, and it is a means of halting model training before it starts to overfit. Early stopping is another regularization technique we have used in this study to terminate our model training before it starts to overfit after the model learned enough features from the images.

3.9 Evaluation Metrics to Evaluate Accuracy of the Model

After implementing or training the model, the next step will be measuring the success rate and efficiency of the model by using different metrics. Here are the basic performance metrics I have used for the “Wheat Rust Disease Detection” model.

3.9.1 Precision (P)

Precision is a performance metrics which answers the question of:
What proportion of positive identifications was actually correct? It is also called the Positive Predictive Value (PPV). Precision can be thought of as a measure of classifiers' exactness. A low precision can also indicate a large number of False Positives. Before we discuss the formulas of Precision and Recall, let's have some views about their measurements[49].

In this study we have 2 classes:

Healthy crop: Positive class

Not Healthy(Infected): Negative class

True Positive: It is a result where the model predicts the positive class correctly.

True Negative: It is a result where the model predicts the negative class correctly.

False Positive: It is a result where the model predicts the positive class incorrectly.

False Negative: It is a result where the model predicts the negative class incorrectly.

$$P = TP / TP + FP$$

Equation 3. 1: Precision Formula [50]

3.9.2 Recall (R)

Recall is the number of True Positives divided by the number of True Positives and the number of False Negatives. Put another way it is the number of positive predictions divided by the number of positive class values in the test data. It is also called Sensitivity or the True Positive Rate. Recall can be thought of as a measure of classifiers' completeness. A low recall indicates many False Negatives. Recall tries to answer the question of: What proportion of actual positives was identified correctly?

$$R = TP / TP + FN$$

Equation 3. 2: Recall Formula [50]

3.9.3 F1 Score

F1 score combines precision and recall relative to a specific positive class. The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst at 0.n

F1 Score = 2 *(precision*recall) / (Precision+ recall)

Equation 3. 3: F1 score Formula [50]

CHAPTER FOUR

4. Proposed model

4.1 Convolutional Neural Network Architectures

A lot of convolutional architectures have been developed from the 1990s. In the following, this study has proposed MosNet, convolutional neural network architecture, which is implemented from scratch.

4.1.1 MosNet Model

MosNet is a convolutional neural network architecture which I named it after my own name, and implemented from scratch with totally 2113 data of the infected and healthy image classes, with three different training and test split set which are categorized as 1233 for training , 411 for validating the model and the rest for explicit singlet test imagining as a real world image test if needed. These train test split distributions are 50% - 50%, 75% - 25% and 70% - 30% for train and test respectively. Sigmoid activation function for the output layer is used with the loss function of the binary cross-entropy and the efficient Adam gradient descent algorithm is used to learn the weights using the most common and efficient learning rates of 0.001 and 0.0001.

- ✓ The first layer is a convolution layer called a Conv2D. The layer has 32 feature maps, which with a size of 3×3 and a rectifier activation function. This is the input layer, expecting images shape with (pixels, width, and height) which I fed it with 150X150 image sizes.
- ✓ The next layer contains a pooling layer that takes the max called MaxPooling2D. It is configured with a pool size of 2×2.
- ✓ The second layer contains a convolutional layer with 32 feature maps with a size of 3x3 and a rectifier activation function called ReLU and a pooling layer with a size of 2x2.
- ✓ The third layer contains a convolutional layer with 64 feature maps with the size of 3x3 and rectifier activation function followed by the Max pooling layer with a pool size of 2x2.

- ✓ The next layer converts the 2D matrix data to a vector called flatten it allows the output to be processed by standard fully connected layers flowed by a rectifier activation function.
- ✓ The next layer is a regularization layer using drop out called Drop out. It is configured to randomly exclude 30% of neurons in the layer in order to reduce overfitting.
- ✓ The next layer is a fully-connected layer with 64 neurons with a rectifier activation function.
- ✓ The output layer has 2 neurons for the 2 classes and a sigmoid activation function to output probability-like predictions for each class. The model is trained using binary cross-entropy as loss function and the ADAM gradient descent algorithm with different learning rates.

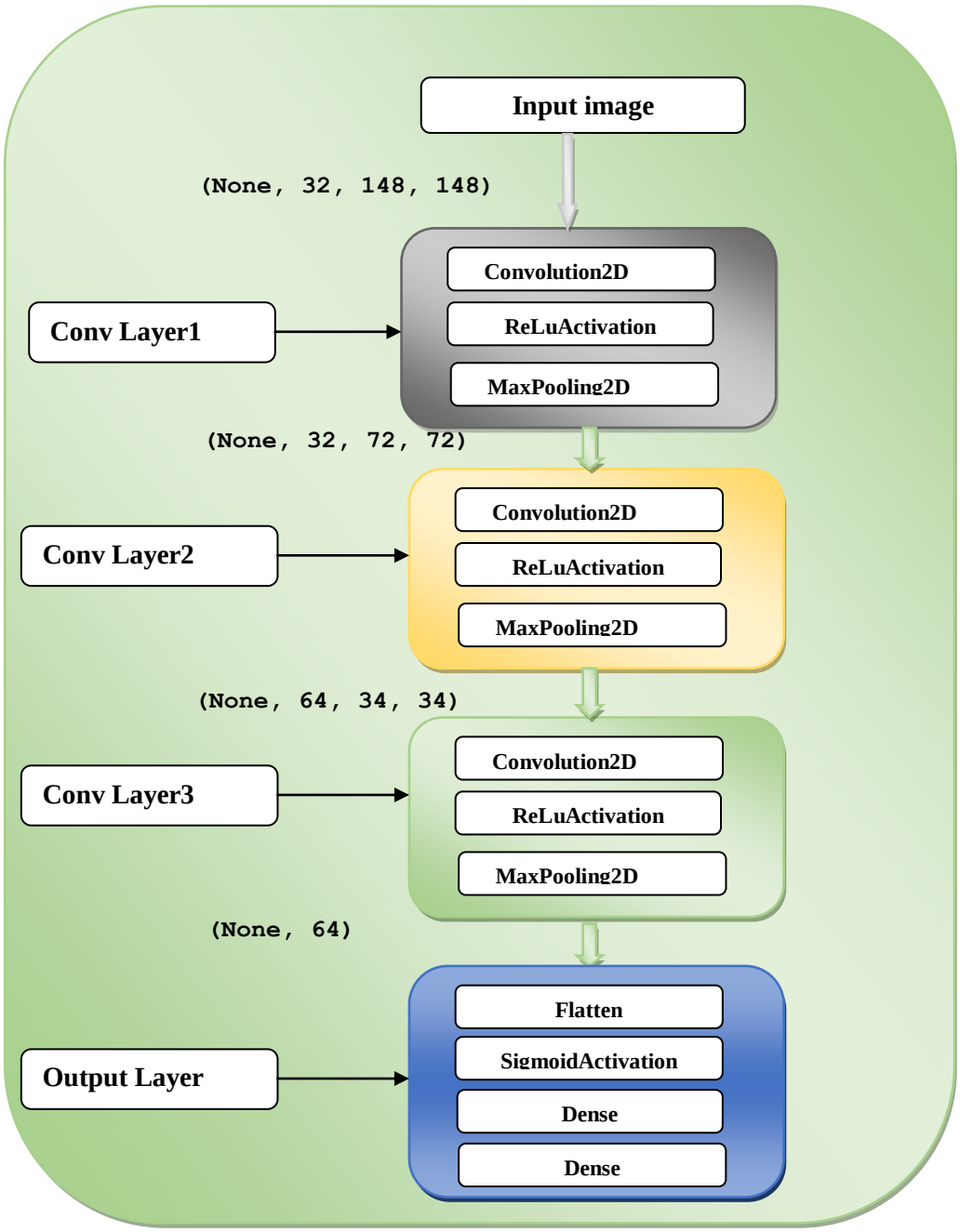


Figure 4. 1: Schematic Diagram of MosNet Model

4.2 Explanation for the Schematic Diagram

The general description of MosNet model is elaborated in section [*4.1.1 MosNet Model*] which is explained in the above schematic diagram, below, there is a description of each bloack in the architecture.

- The three above layers(Conv Layer1, Conv Layer3, Conv Layer3) are input layers expecting images from their previous layers, as Example Conv Layer 1 takes input image as (None, 32,148,148): None indicates the number of images the layer can take, but in case of this None indicates the layer can take as many as possible images as it can , so it is indicated as None by default. 32: indicates feature map or the number of hidden layers the layer has, 148: is the size of the input image, but originally our image is resized into 150 by 150 dimension, and it becomes 148 because we are using 2X2 pooling and it minimizes 2 from the original image. The size of input image will decrease by half in eachas the number of convolution layers increase, Example: 148 is decrease into 74, which will be 76-umber of pooling which is 2, 76-2=74., the same procedure takes place in each layer.
- Each input layer has Convolution layer, ReLu activation layer and pooling Layer which are explained in sections [*2.8 Convolutional Neural Networks*] and [*2.9 Activation*].
- Sigmoid activation function is an output activation function which is used to predict probability like predictions like (0 and 1➔(“Healthy” and “Infected ” in case of this study))
- In the output layer our output features are flatten from grid format to flatten information to feed it for the output layer so that the layer can learn each features from the previous layer in a sequential manner. Nextly, there are two dense layers, one for the number of hidden layers in the output layer and one for the number of classes we have for prediction. Example the first dense layer is Dense(64) means we have 64 feature maps or hidden layers and the next dense layer Dense(2) means there 2 classes for prediction whicha are ‘’Healthy’’ and “Infected” classes.

4.1.2 HUF Depth Separable Convolutional Architecture

Depth-wise separable convolutions reduce the number of parameters and calculations used in convolutional operations while increasing the efficiency of representation. They have been shown to be successful in image classification models, both in obtaining better models than previously possible for a given parameter and in significantly reducing the number of parameters needed to perform on a given level. Unlike standard CNN's in which convolution is applied to all M channels, Depthwise separable applied to a single channel at a time and it reduces the number of calculations in the model.

- The first layer, which is called SeparableConv2D with 32 feature maps with 3x3 kernel size and ReLu activation layer. This layer expects image with shape of (density, height, width). This layer performs a spatial convolution on each channel before proceeding to point-wise convolution (1x1) convolution.
- The next layer has a separable convolution with 32 feature maps, 3x3 kernel size, and ReLu activation function
- Next, there is a regularization layer, which Dropout used to randomly exclude different neurons in the layer to reduce overfitting happens in the layer.
- The next layer has a pooling layer, which is a max-pooling known as MaxPooling2D with a pool size of 2x2.
- The next layer again has a separable convolution layer with 32 feature maps, 3x3 kernel size, and Relu activation function.
- The next layer again has a separable convolution layer with 32 feature maps, 3x3 kernel size, and Relu activation function.
- Pooling Layer with a size of 2x2.
- The next layer again has a separable convolution layer with 64 feature maps, 3x3 kernel size, and ReLu activation function.
- Dropout regularization techniques to reduce overfitting.
- The next layer again has a separable convolution layer with 64 feature maps, 3x3 kernel size, and ReLu activation function.

- Global Average Pooling Layer It treats the convolutional layers as feature extractors and fed to a fully connected layer called a dense layer with 32 neurons and rectifier activation function.
- The output layer has 6 neurons for the 2 classes and a sigmoid activation function to output probability-like predictions for each class. The model is trained using categorical cross-entropy as loss function and the ADAM gradient descent algorithm with different learning rates and batch sizes.

4.2 Difference between Standard and Depthwise Convolutions

4.2.1 Standard Convolution

Suppose there is an input data for the convolution process with the size of h * w * M, where h and w are the size of the image with height and width and M is the number of channels in the image (3 in RGB image and 1 in Grayscale image). Again, there are N filters (kernels) of size X * X * M. So for the standard convolution the output image size will be Y * Y * N.

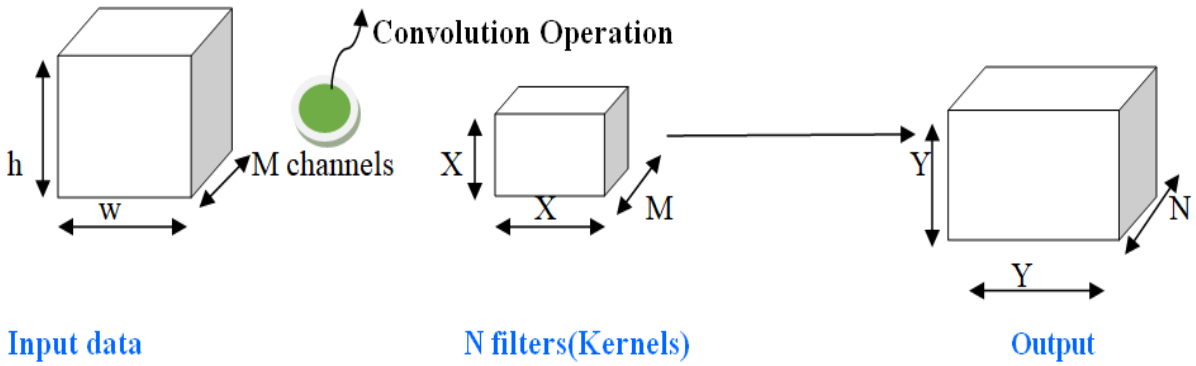


Figure 4. 2: Standard Convolution Operation

$$numOf * X * M$$

Since there are N filters and each filter slides vertically and ***multiplications in one Conv = X*** horizontally X times,

$$total\ NumOf\ multiplications = N * Y * Y$$

Therefore, in Conv operation:

$$\textit{total NumOf multiplications} = M * X^2 * Y^2 * M$$

4.2.2 Depth-wise Separable Convolutions

Depthwise separable convolutions are broken into two operations.

A. Depth-wise convolution

B. Point-wise convolution

A. Depth-wise convolution

In a depth-wise operation, convolution is applied to a single channel at a time, which is different from the standard convolution, which is done for all M channels. Therefore, in depth-wise convolution, the filters or kernels will be $X * X * 1$. For M channels in the image data, then M filters are required. The output will be the size of Y * Y * M.

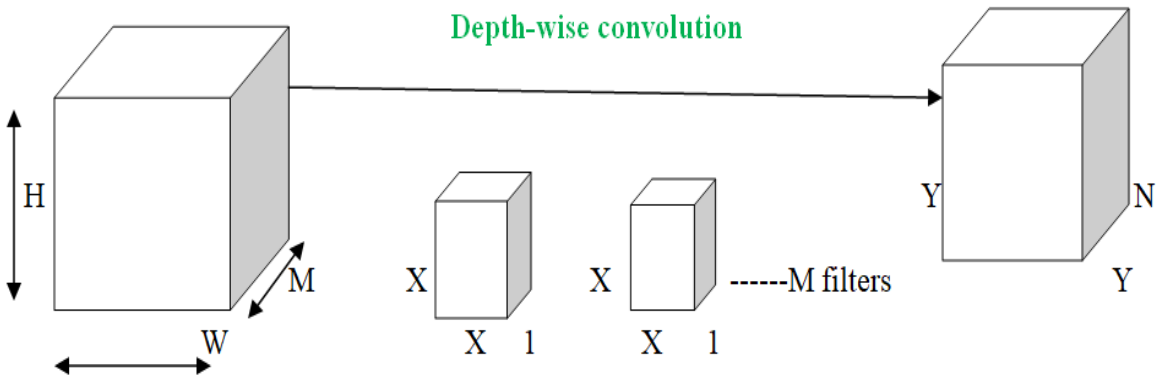


Figure 4. 3: Depth-Wise Convolution

Cost of Operation:

Single Conv operation requires $X * X$ multiplications.

The total number of multiplications is equal to $M \times X \times X \times Y \times Y$

$$\textit{total numOf multiplications} = M \times X^2 \times Y^2$$

B. Point-wise convolution

In point-wise operation, a 1x1 convolution operation is applied on the M channels. Therefore the filter size for the operation will be 1 x 1 x M. Suppose we use N filters, the output size will be, Y * Y * N.

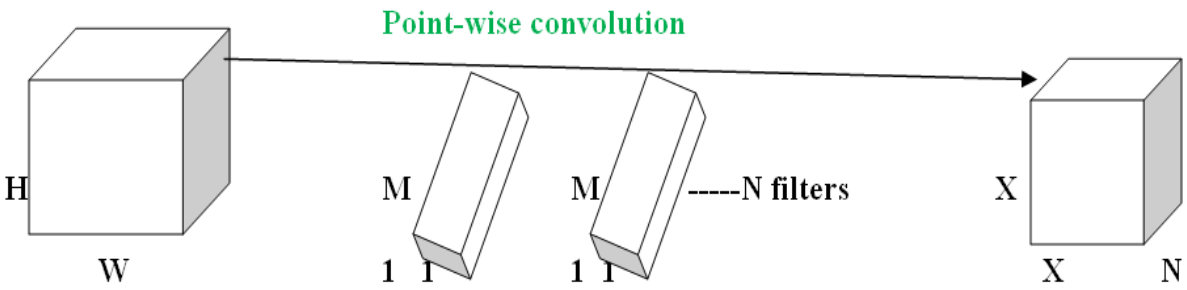


Figure 4. 4: Point-Wise Convolution

Cost of Operation:

A single convolution operation requires 1 x M multiplications.

The total number of multiplications is equal to M x X x X x (no. of filters)

total no of multiplications = M x X^2 x N.

Therefore , for overall operation:

Total multiplications = Depth wise conv. multiplications + Point wise conv. multiplications.

Total multiplications = M x X^2 x Y^2 x + M x X^2 x N = M x Y^2 x (X^2 + N)

Therefore Depth-wise separable convolution operation:

Total numOf multiplications = M x Y^2 x (X^2 + N)

4.3 Dataset Preparation

To Train and Test MosNet model, 2113 RGB images are used, that are augmented from a small set of original data. After augmenting the original data, the new dataset will be preprocessed to set on new standardized and uniform data, with the likes of setting the data to the same size.

4.3.1 Increasing Data Set size with Data Augmentation

Data Augmentation is a mechanism by which we can generate large data set sizes from small size data set. Because of many reasons, we may not get access to enough data that we needed in our study. Especially, Deep Learning is a kind of approach which performs well when we have only a large data set. Therefore, in order to get a well-performing model, we need to augment the little amount of data we collected originally. We can perform this process by adding some effects such as increasing some contrasts, flipping, rotating in different angles, shifting vertically and horizontally are some of the techniques. This Augmentation process helps us to create many different data examples from a single image data so that we can generate more images from a single image. As an example, 10 features are used, which means creating 10 different images from a single image, as described below.

- Rotation
- Width shift
- Height shift
- Rescaling
- Shear
- Zoom
- Horizontal flip
- Fill mode
- Data format
- Brightness

4.3.1.1 Pseudo Code for Data Augmentation

- Read the images from the local disk.
- Change the image into an array (because images are information that can be described as 2D arrays).
- Set a value for each feature factors explained above. (Example: width_shift_range = 15, brightness_range = [0.5, 0.15]).
- Save data.

4.4 Pre-Processing

Because the Training images are collected from different agricultural research Institutes with different formats, resolutions, and sizes; I needed to normalize my data to the same format and resolution so that my model can easily learn and catch up with those data set. Since I am using RGB images with a limited computing power of my computer, then the images are resized into 150X150 RGB image dimensions.

Input: Distorted data

Output: Resized data (150X150 dimensions)

4.4.1 Pseudo Code for resizing the images

- Read the images from the training directory

```
imgPath = trainingdirectory ()
img = readimg(FromTrainingdirectory)
trainingImage = resize (img)(150x150)
```

4.5 Color Code Segmentation

It is very difficult to easily understand plant images for classification purpose. Because they are not as easy as identifying the images of cats from dogs, or lions, which can easily be detected by the size and existence of patterns from the image. Therefore this study proposes a color code segmentation, to classify infected images from the healthy one, by identifying the yellow-orange and red-orange colors from the RGB images which will give a pattern results from the infected images and empty space from the green or healthy image.

The three disease types have a bit color difference when they infect the crop and I needed to segment them with their unique color code they belong in RGB image.

Therefore:

- Yellow Rust has color in between B>0 and B<45
- Stem rust has color in between R>130 and R <200
- Leaf rust has color in between G>50 and G<150

The Segmentation process is performed by finding out a specific color code for each rust color type from the RGB color image. After segmenting these images using this color segmentation, we get a segmented image which is different from the healthy wheat crop, which is a black solid image without any pattern, unlike those infected images. To get a unique RGB value of every color we can use online RGB color code pickers.

4.5.1 Pseudo Code for Segmentation

- Read the preprocessed data from the training directory

  if (blueColor > 0 and <100) and (redColor < 130 and > 245) and (greenColor >50 and <245)

  return [j, i, 0] =r

  return [j, i, 1] =g

  return [j, i, 2] =b

4.6 Classification

The Segmented images give two results which are a pattern from infected images and empty space (solid-color image) from healthy images and the model can easily learn from those patterns.

4.7 Performance Evaluation

The loss function is the most important element in CNN, which helps us to evaluate our model's performance. It is a value that varies between 0 and 1, which measures the inconsistency between the predicted and actual values. As the loss tends to 1 it implies the model is not doing well and as it tends to 0, it is showing that the model is predicting our values well.

4.8 Experiment Parameters

MosNet has a totally conducted more than 200 experiments which are made of different features that have an impact on the performance of the model.

1. Architecture

- MosNet model

2. Data Set Type

- Grayscale Image
- Color(RGB segmented)

3. Train – Test Split

- Train - 50% : Test - 50%
- Train - 75% : Test - 25%
- Train - 70% : Test - 30%

4. Number of Epochs

- 100, 200, 300

5. Learning Rate

- Rmsprop
- Adam(0.001)
- Adam(0.00001)

6. Dropout Rate

- 30%, 50%

7. Training Mechanism

- Training from Scratch
- Fine Tuning

To calculate the number of experiments done for the model we can multiply the number of chosen parameters in each feature factor.

Therefore it is going to be:

- ✓ Data set type = 2(Gray scale, Segmented RGB Image)
- ✓ Train – Test split ratio = 3(50% -50%, 70% - 30%, 75% - 25%)
- ✓ Number of Epochs = 3(100, 200,300)
- ✓ Learning rates = 3(Rmsprop, Adam(0.001), Adam(0.00001)
- ✓ Dropout rate = 2(50%, 30%)

The above factors give us total experiment combination of:

- ✓ 2x3x3x3x2 = 108 total Experiments.

But, it will be wasting time to discuss all these 108 experiments in this study and experiments with only the top 10 results are discussed under section *[4.3.1 Increasing Data Set size with Data Augmentation]*.

CHAPTER FIVE

5. Implementation

5.1 Data Set Preparation

The first thing we need to be curious about before we start training a model is all about to sort out our data set and the computational power we have in our hand, otherwise, it will be difficult to achieve the kind of efficient model we want. Even if we cannot afford the best computational power we want, at least we need to have a sufficient dataset to train our model. Therefore before starting to train the model, it is needed to well prepare the data. Originally the collected images from the fields are not enough to use for deep learning. So the first thing done to increase the data set size is using a Data Augmentation. There are totally around 213 original data and the data is augmented into around 2113 images, by applying 10 feature factors which are explained in section [*4.3.1 Increasing Data Set size with Data Augmentation*], which the 10 feature factors are applied to every single image in the original image, so that have generated more than 2000 images from those original images and save it in the same format, in my case which I used a PNG image format, eventhough the image format does not matter, but JPEG, JPG and PNG are the most commonly used image formats. Here is an example in which we can create 20 images from 2 original images.

Note: PNG is an image extension format where as RGB is a color ordering channel. As an example, an image with any color channel whether it is a Grayscale or RGB it might have different format extentions like PNG, JPEG, JPG and etc. like we can have an extension for a text document for example: a file extension for a text document can be .doc, .docx,.pdf and others, PNG is the same way for images, it is a file extension format. RGB is a color channel order which is described as **R for Red**, **G for Green** and **B for Blue.** Every color in the world is the combinational value of these three color combinations.

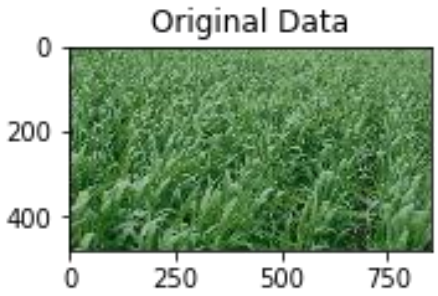


Figure 5. 1: Sample Original Image Before Augmentation

The next thing is to increase the data set size (amount) by using the 10 feature factors, as described in the above section.

Here are the augmented images of the above image.

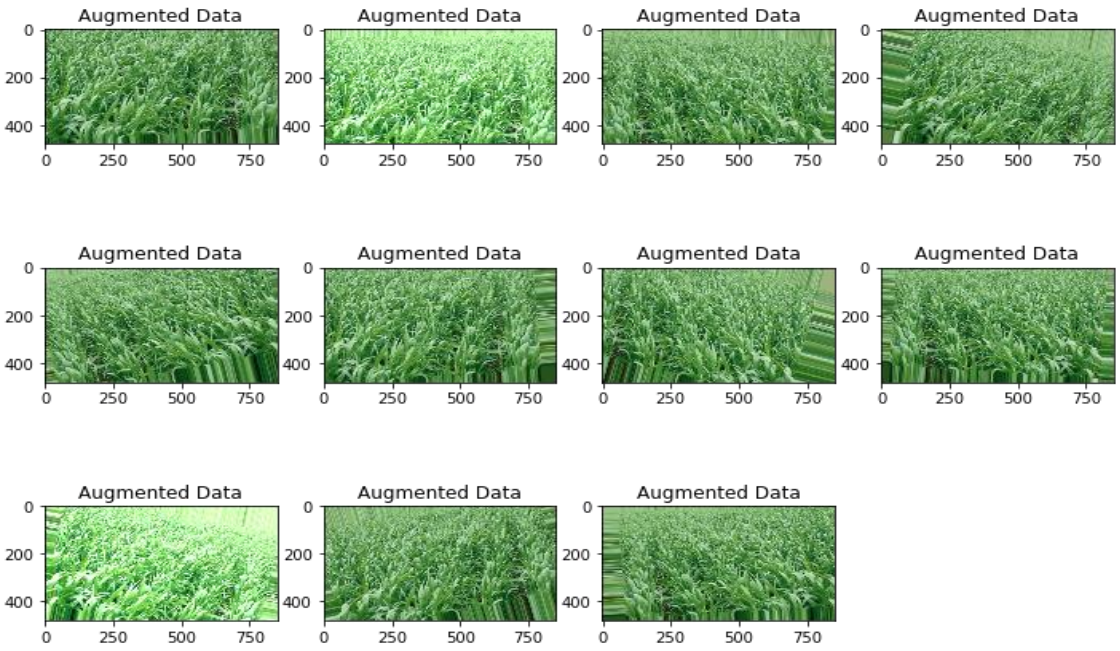


Figure 5. 2: Augmented Images Using 1 Original Image

5.1.1 Sample code for Data Augmentation

```
datagen = ImageDataGenerator(rotation_range =15,
                      width_shift_range = 0.1,
                      height_shift_range = 0.1,
                      rescale=1./255,
                      shear_range=0.1,
                      zoom_range=0.1,
                      horizontal_flip = True,
                      fill_mode = 'nearest',
                      data_format='channels_last',
                      brightness_range=[0.5, 1.5])

Aug_dir = "C:/Users/Dude/Documents/NewImages"
```

Figure 5. 3: Sample Code for Data Augmentation

The above images are images that are generated from one original image on [Fig 5.1:] by using augmentation technique so that I have generated 21 different images, which means 20 images using 10 feature factors for a single image[1x10+1 original image], which results in 21 total images. Therefore from 213 images, 2113 training images are generated for the designed model.

5.2 Image Pre- Processing

As we can see in *Figure 5. 1*, the size of image is 480x852 height and width respectively, but if we see all images in the training data, they have a sparse nature with different dimension size, thus we need to normalize to the same image dimension, which is making the training data standardized to the same image dimension size. Therefore I have used a dimension of 150X150 for all images in the training set.

Here are some samples of distorted data that needs to be pre-processed or resized to the same image dimension, which is 150X150 image size.

Here are sample images that are not pre-processed or distorted images.

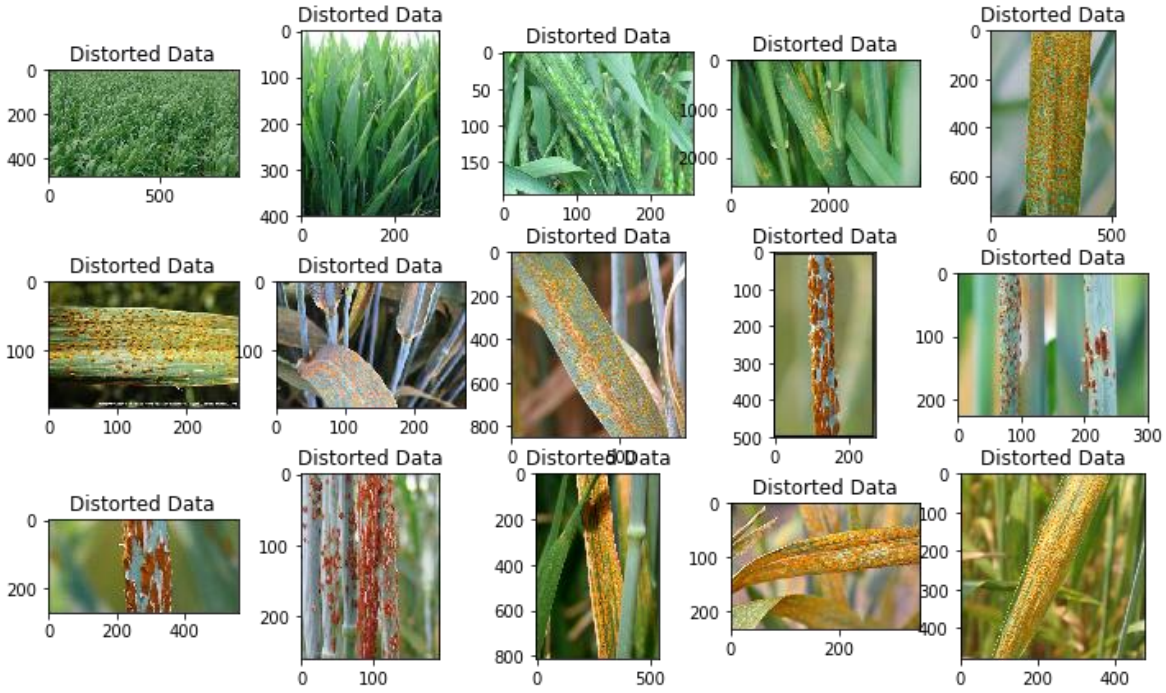


Figure 5. 4: Sample Distorted Images

Here are sample pre-processed or resized images to 150X150.

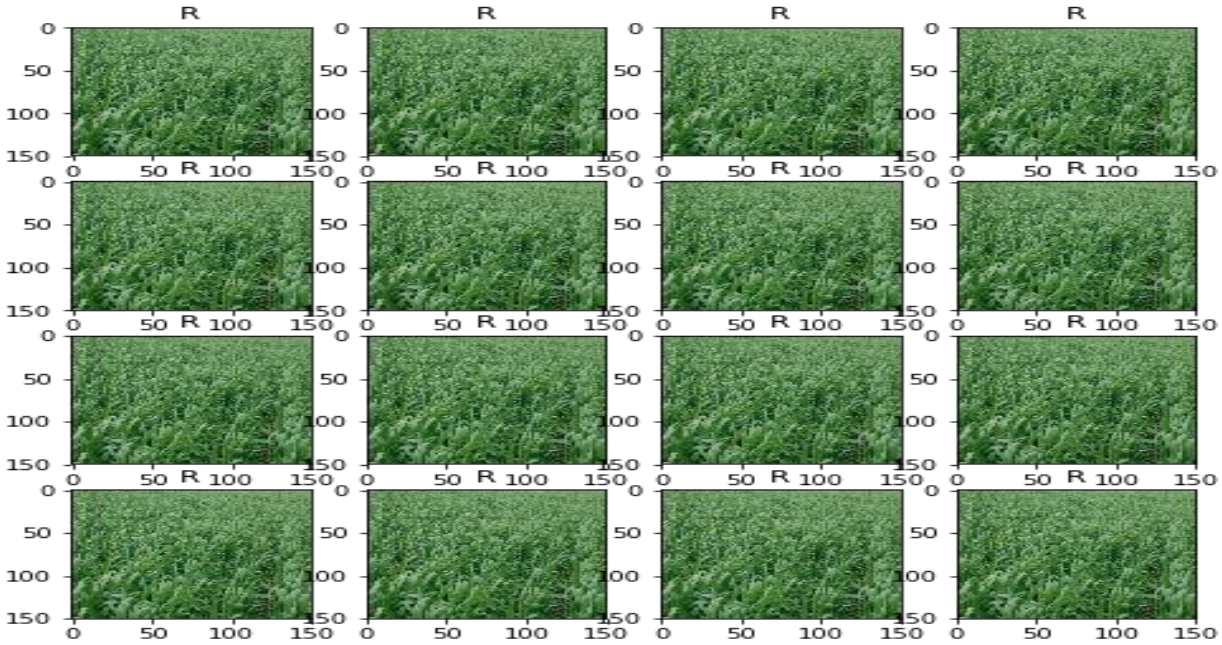


Figure 5. 5: Sample Preprocessed Images

5.2.1 Sample code for resizing the images into 150X150

```
for i in range(1, columns*rows +1):
    fig.add_subplot(rows, columns, i)
    img = cv2.resize(img,(150,150))
    im_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    plt.title("R")
    plt.imshow(im_rgb)
```

Figure 5. 6: Sample Code for Image Resizing

5.3 Image Segmentation

After sorting out our images by performing the preprocessing step through resizing, the next step is to segment the image based on the color code from the infected crop images as I explained it in section[4.4].

As explained in Chapter 1, we have identified problems such as single leaf constraincy, manual cropping of infected crop images to identify healthy crops from the infected crops. But this study solves this issues through color code Segmentation which uniquely identifies every

color over the world by their unique RGB value. This technique enables to clearly classify the infected crops from healthy crops based on the color of the rusts on the wheat color.

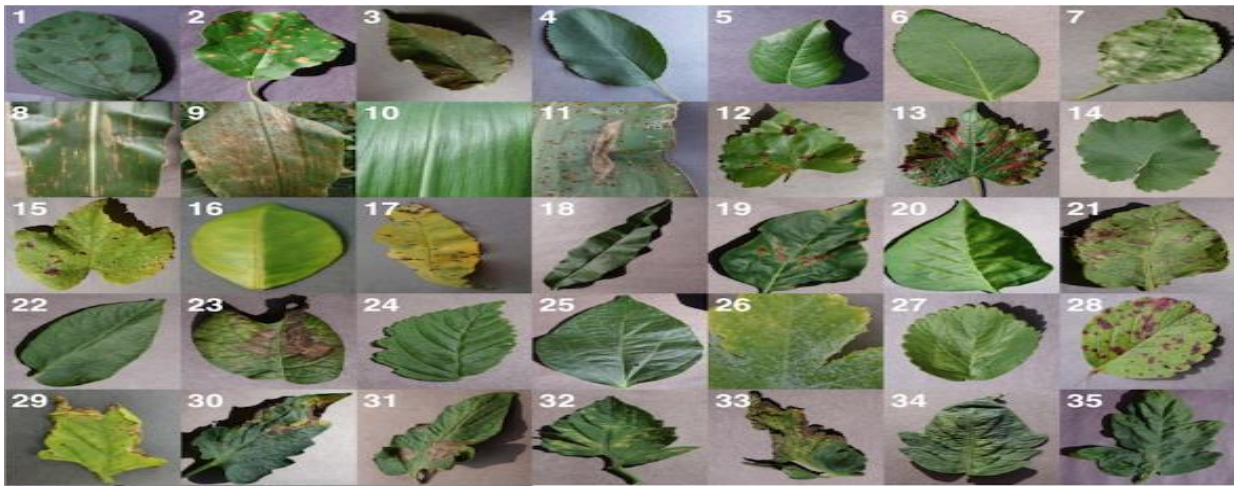


Figure 5. 7: Example of Single Leaf Constraincy

The above image shows images that are taken in single leaves only and homogenous background, so this technique does not work for images with different kinds of backgrounds and images taken in any state of shape and from different angles. Therefore, the proposed model solves these problems by taking images in any condition with heterogenous backgrounds.

Figure 5. 8: Image Comparison of Original with Color Segmented

As you can see above, there are two kinds of images, the above four images, which are the normal RGB images (before segmentation), and the four images with their respective below are segmented images for their respective original images on the above. Therefore, by segmenting all images in the training set to feed to the model, it can easily classify them by identifying the patterns made in the segmentation part, In this case the healthy images hold solid black without any pattern and the infected images form a specific pattern which is totally different from the healthy images and by comparing images with patterns and without patterns it perfectly classifies them in to healthy or infected crops.

5.3.1 Sample Source code for the color Segmentation

```
for i in range(img.shape[1]):
    for j in range(img.shape[0]):
        b = img[j, i, 0]
        g = img[j, i, 1]
        r = img[j, i, 2]

        if((b < 45 and b > 0) and (r > 130 and r < 245) and (g > 50 and g < 245)):
            out[j,i, 0] = 255
            out[j,i, 1] = 255
            out[j,i, 2] = 255
```

Figure 5. 9: Sample Source Code for Color Segmentation

CHAPTER SIX

6. Results and Discussion

After performing all the evaluation on the Grayscale and color segmented dataset on the MosNet model, experiments with top results are discussed below. The **MosNet** model is evaluated using different parameters that can affect the efficiency of the model. These parameters are a number of epochs; train test split ratio, learning rate and dropout ratio which are explained briefly in Chapter 4. Different parameter ratios are used for the evaluation purpose and the most successful ratios are discussed in this chapter.

Note: When we read our confusion matrix, it classified into two labels in the table, which are the horizontal line is labeled as True Value whereas the vertical line is labeled as the predicted value.

6.1 Table Summary for Evaluation of MosNet Model

Table 6. 1: Summary Table for MosNet model

No	*Dataset Type*	*Epochs*	*Learning Rate*	*Test Ratio*	*Drop out*	*Training time*	*Accuracy*	*Error*
1	*Grayscale*	*100*	*Adam(0.001)*	*25%*	*50%*	*60.49*	*85.89%*	*14.11%*
		100	*Adam(0.00001)*	*25%*	*50%*	*60.95*	*81.63%*	*18.37%*
		200	*Adam(0.001)*	*25%*	*30%*	*120.77*	*80.78%*	*19.22%*
		200	*Adam(0.001)*	*25%*	*50%*	*120.43*	*86.62%*	*13.38%*
		300	*Adam(0.001)*	*25%*	*50%*	*179.25*	*79.08%*	*20.92%*
2	*RGB*	*200*	*Adam(0.001)*	*25%*	*50%*	*110.34*	*98.78%*	*1.22%*
		200	*Adam(0.001)*	*20%*	*50%*	*115.65*	*98.48%*	*1.52%*
		300	*Adam(0.001)*	*25%*	*30%*	*170.90*	*99.51%*	*0.49%*
		300	*Adam(0.001)*	*25%*	*50%*	*196.24*	*99.27%*	*0.73%*
		300	*Adam(0.0001)*	*25%*	*50%*	*176.19*	*97.57%*	*2.43%*
		300	*Adam(0.00001)*	*25%*	*50%*	*169.17*	*96.84%*	*3.16%*
3	*RGB Segmented*	*300*	*Adam(0.001)*	*25%*	*30%*	*165.61*	*99.76%*	*0.24%*
		300	*Adam(0.001)*	*25%*	*50%*	*168.14*	*98.05%*	*1.95%*

The above table summarizes top results found for the evaluation of MosNet model on three types of dataset types which are used for the purpose of comparison. These data set types are the grayscale image type, which is types of images with only one (1) color channel, the RGB image types, which are images containing more than 16million colors in one RGB image and the RGB segmented images which are images proposed in this study.

6.2 Evaluating MosNet model on Grayscale Images.

Table 6. 2: Result Summary of MosNet model on Grayscale images

No.	*Epochs*	*Learning Rate*	*Test ratio*	*Dropout*	*Training time(minutes)*	*Accuracy*	*Error*
1	***100***	***Adam(0.001)***	***25%***	50%	***60.49***	***85.89%***	***14.11%***
2	***100***	***Adam(0.00001)***	***25%***	50%	***60.95***	***81.63%***	***18.37%***
3	***200***	***Adam(0.001)***	***25%***	30%	***120.77***	***80.78%***	***19.22%***
4	***200***	***Adam(0.001)***	***25%***	50%	***179.25***	***86.62%***	***13.38%***
5	***300***	***Adam(0.001)***	***25%***	50%	***120.43***	***79.08%***	***20.92%***

1. Model Evaluation with parameters, Epoch of 100, learning rate Adam (0.001), Test data ratio of 25%, and Dropout 50%, which results with an accuracy of 85.89%, from *Table 6. 2*, row 1.

	precision	recall	f1-score	support
Healthy	0.85	0.85	0.85	192
Infected	0.87	0.86	0.87	219
accuracy			0.86	411
macro avg	0.86	0.86	0.86	411
weighted avg	0.86	0.86	0.86	411

Figure 6. 1: Classification Report of Validation Data for Table 6. 2, row 1

	precision	recall	f1-score	support
Healthy	1.00	1.00	1.00	630
Infected	1.00	1.00	1.00	603
accuracy			1.00	1233
macro avg	1.00	1.00	1.00	1233
weighted avg	1.00	1.00	1.00	1233

Figure 6. 2: Classification Report of Training Data for Table 6. 2, row 1

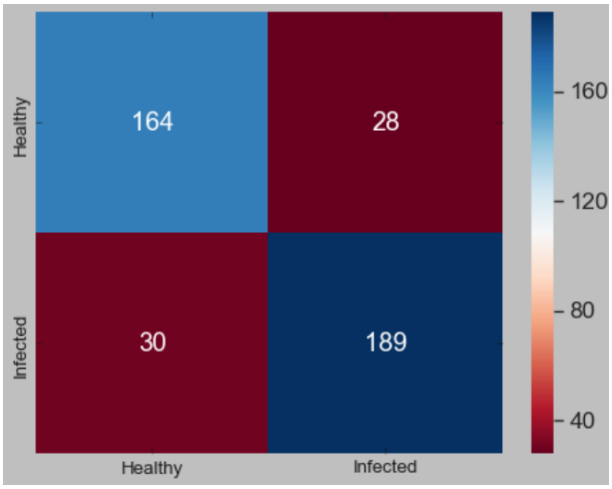


Figure 6. 3: Confusion Matrix of Validation Data for Table 6. 2, row 1

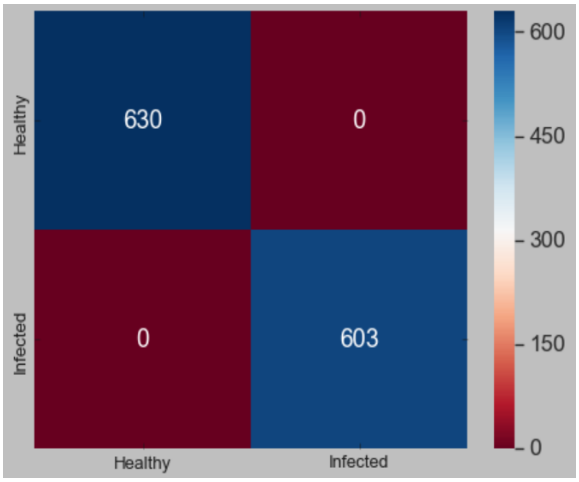


Figure 6. 5: Confusion Matrix of Training Data for Table 6. 2, row1

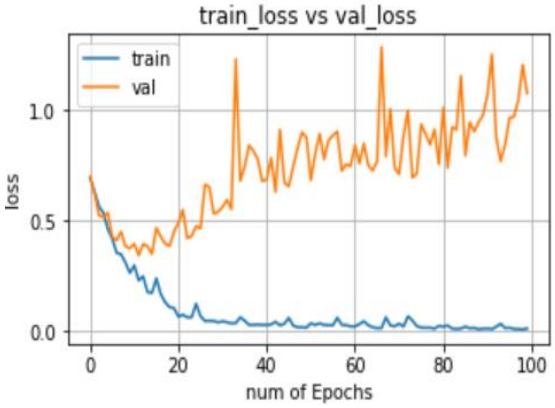


Figure 6. 4: Loss Graph of Train vs Val Data for Table 6. 2, row 1

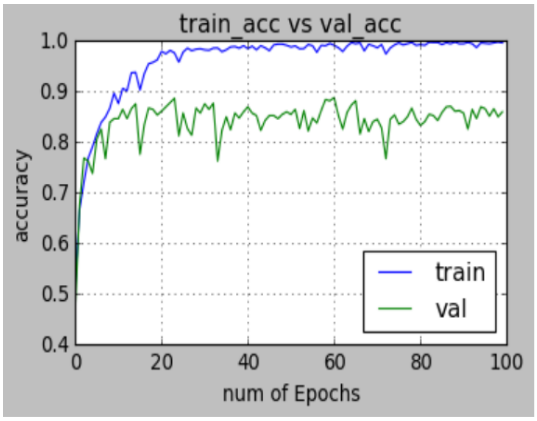


Figure 6. 6: Accuracy Graph of Train vs Val Data for Table 6. 2, row 1

2. Model Evaluation with parameters with Epoch of 100, learning rate Adam (0.00001), Test data ratio of 25%, and Dropout 50%, which results with an accuracy of 81.63% from *Table 6. 2*,row2

	precision	recall	f1-score	support
Healthy	0.87	0.75	0.81	206
Infected	0.78	0.89	0.83	205
accuracy			0.82	411
macro avg	0.83	0.82	0.82	411
weighted avg	0.83	0.82	0.82	411

Figure 6. 7: Classification Report for Validation Data for Table 6. 2, row 2

	precision	recall	f1-score	support
Healthy	0.93	0.79	0.86	616
Infected	0.82	0.94	0.88	617
accuracy			0.87	1233
macro avg	0.88	0.87	0.87	1233
weighted avg	0.88	0.87	0.87	1233

Figure 6. 8: Classification Report of Training Data for Table 6. 2, row 2

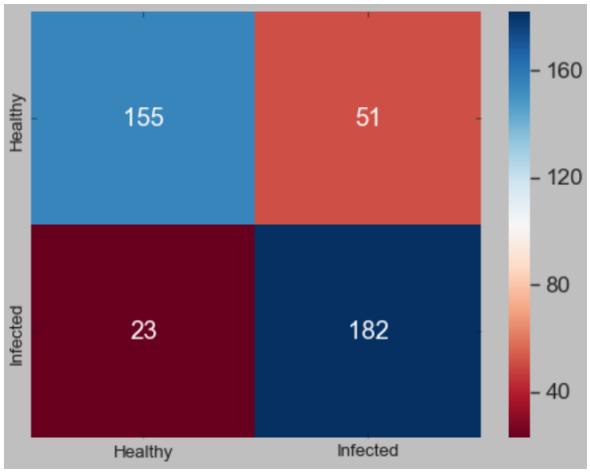


Figure 6. 9: Confusion Matrix of Validation Data for Table 6. 2, row 2

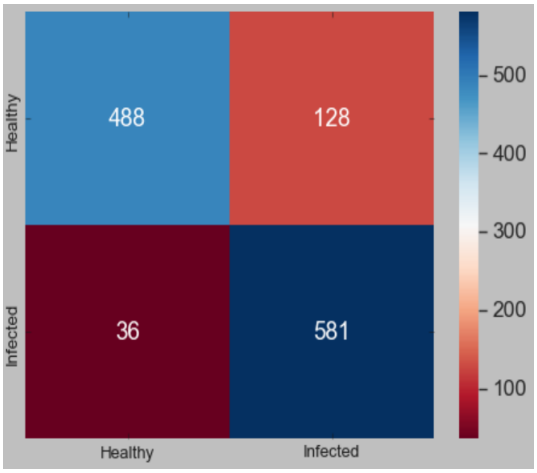


Figure 6. 10: Confusion Matrix of Training Data for Table 6. 2, row 2

6.3 Discussion for MosNet Model on Grayscale Images

Grayscale images are images with only one color channel, which cannot hold enough information to be processed or extracted from the image data. Evaluating the same model on

the same dataset in Table 6.2, by changing their parameters that can affect the efficiency of the model, the effect of the learning rate is discussed for result1 and 2 in the table. The only parameter changed from the two results is their Learning rate, in which 0.001 is used for the first result and 0.00001 is used for the second result, which results with an accuracy of 85.89% and 81.63% respectively. This shows that the learning rate in result1 which is equal to 0.001 has decreased to 0.00001 in result 2.

This sounds, as the learning rate decreases, the accuracy of the model decreases proportionally, which sounds like, decreasing the learning rate means, decreasing in the speed of the learning ability of the model. As long as the model is using the same number of epochs, Dropout rate, and test ratio, the result shows the model is taking too much time to learn features from the data. As we can understand from the table, the model starts to degrade on the 300^{th} epoch and this shows for Grayscale images it has nothing more to extract and learn because there is only one channel this makes it lose more features from the data and that will degrade the efficiency of the model after the 200^{th} epoch. This result will force us to use RGB images which have three channels and can contain more than 16 million colors. This helps the model to extract and learn from the color nature of images and helps to prevent losing information from the images, which can be extracted from the color of the images since in the case of this study wheat rusts are identified by their colors from the healthy wheat images.

To see some numerical results from the two results, result1 performs better than result 2, with better accuracy of 85.89%, but if we see the confusion matrix of this result in *Figure 6. 9*, there are still 84 misclassified images from 411 validation data, which is not good enough to take this model to take it to real-world try and risk our field for rust spread out before taking immediate action.

6.4 Evaluating MosNet Model on RGB Images

After evaluating MosNet model on grayscale images, images with only one channel, which results in a model with a lot of misclassified data on both training and validation data, it is a must option to find a new solution to get a better model that can perform in a better way.

Table 6. 3: Result Summary of MosNet model on RGB images

No.	Epochs	Learning Rate	Test ratio	Dropout	Training time(minutes)	Accuracy	Error
1	200	Adam(0.001)	25%	50%	110.34	98.78%	1.22%
2	200	Adam(0.001)	20%	50%	115.65	98.48%	1.52%
3	300	Adam(0.001)	25%	30%	170.90	99.51%	0.49%
4	300	Adam(0.001)	25%	50%	196.24	99.27%	0.73%
5	300	Adam(0.0001)	25%	50%	176.19	97.57%	2.43%
6	300	Adam(0.00001)	25%	50%	165.61	96.84%	3.16%

The above table summarizes the top 6 results found when evaluating the MosNet model on RGB images using different learning parameters and their effects on the model. As we can learn from the table each and every parameter change has its own effect on the result the model is delivering with the training time needed and accuracy of the model. Let's discuss some results that are affected by the change made on different parameters.

6.4.1 Result comparison of MosNet Model on RGB Images Based on Learning Rate

From the

*Table 6. **3***, we can take results on rows 4, 5 and 6 which have the same parameters except they differ in their learning rates, in which result on row 4 has LR of 0.001, result on row 5 with LR of 0.0001 and result on row 6 with LR of 0.0001. By comparing the three results with respect to their learning rate we can conclude the effect of the learning rate on the model.

***Table 6. 3*, Row 4:** Result on row 4 is evaluated using 300 numbers of Epochs with test ratio of 25% and dropout ratio of 50%, and LR of 0.001, which results in an accuracy of 99.27% taking a training time of 196.24 minutes.

	precision	recall	f1-score	support
Healthy	1.00	0.99	0.99	219
Infected	0.99	0.99	0.99	192
accuracy			0.99	411
macro avg	0.99	0.99	0.99	411
weighted avg	0.99	0.99	0.99	411

Figure 6. 11: Classification Report of Validation Data for Table 6. 3, row 4

	precision	recall	f1-score	support
Healthy	1.00	1.00	1.00	603
Infected	1.00	1.00	1.00	630
accuracy			1.00	1233
macro avg	1.00	1.00	1.00	1233
weighted avg	1.00	1.00	1.00	1233

Figure 6. 12: Classification Report of Training Data for Table 6. 3, row 4

```
[[217   2]
 [  1 191]]
```

Figure 6. 13: Confusion Matrix of Validation Data for

Table 6. 3, row 4

```
[[603   0]
 [  0 630]]
```

Figure 6. 14: Confusion Matrix of Training Data for Table 6. 3, row 4

As we can see from the classification report of the above figures the model achieves a 100% training accuracy and, but it has a few misclassifying data which is explained below in the confusion matrix.

The above confusion matrix shows that the model works perfectly when tested on the images it already knows or trained on. But it has some misclassifying images on validation data, which are 3 misclassifying images from the total 411 validation data. The model misclassified 2 healthy wheat images as infected and 1 infected wheat image as healthy as stated on the confusion matrix for validation data.

Figure 6. 15: Loss Graph of Train vs Val Data for Table 6. 3, row 4

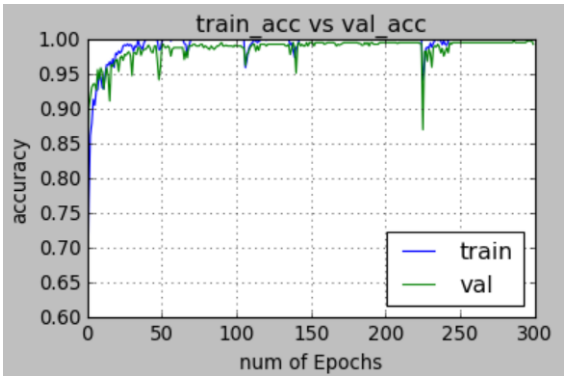


Figure 6. 16: Accuracy Graph of Train vs Val Data for Table 6. 3, row 4

The Loss graph shows that the model loss which starts from around 0.6(60%) has gradually decreased and tends to almost zero and this is what we are looking for and at the same time if we see the accuracy starts from 0.70(70%) and gradually increased to 0.99(99%) which is almost tending to 1(100%). In both graphs, both the training and validation data have no significant distance between them, which is a healthy move for the model.

***Table 6. 3*, Row 5:** Result on row 5 is evaluated using 300 numbers of Epochs with test ratio of 25% and dropout ratio of 50%, and LR of 0.0001, which results in an accuracy of 97.57% taking a training time of 176.19 minutes.

	precision	recall	f1-score	support
Healthy	0.97	0.98	0.98	200
Infected	0.98	0.97	0.98	211
accuracy			0.98	411
macro avg	0.98	0.98	0.98	411
weighted avg	0.98	0.98	0.98	411

A) For validation data

	precision	recall	f1-score	support
Healthy	1.00	1.00	1.00	603
Infected	1.00	1.00	1.00	630
accuracy			1.00	1233
macro avg	1.00	1.00	1.00	1233
weighted avg	1.00	1.00	1.00	1233

B) For training data

Figure 6. 17: Classification report of result in Table 6. 3, row 5

The above classification report shows that, by decreasing the learning rate from 0.001 to 0.0001 from the result on row 4, there is a little change in the accuracy of the model which decreases from 99.27% to 97.57%. But it still achieves a training accuracy of 100%, which cannot be an issue of discussion as long as the model knows the training data when we fit the model. What matters is achieving high accuracy on the validation data. It's obvious that as the accuracy of the model decreases, misclassifying data increases inversely, so in order to minimize the number of misclassifying data we need to increase the accuracy of the model. In the Figure below, it is shown that as the accuracy of the result on row 5 decreased, we can figure it out as the numbers of misclassifying images are increased from the previous result.

```
[[196   4]
 [  6 205]]
```

A) For validation data

```
[[622   0]
 [  0 611]]
```

B) For training data

Figure 6. 18: Confusion Matrix of result in Table 6. 3, row 5

On the above figure, there are a totally 10 misclassifying images that increased from the previous result in ***Table 6. 3***, row 4, the one with only 3 misclassifying images. On the result of row 5, there are 4 healthy wheat images which are classified as infected wheat images and 6 infected wheat images which are classified as healthy. But here there is no misclassifying image for the training data, which means it has achieved training accuracy of 100%.

***Table 6. 3*, Row 6:** Result on row 6 is evaluated using 300 numbers of Epochs with test ratio of 25% and dropout ratio of 50%, and LR of 0.00001, which results in an accuracy of 96.84% taking a training time of 165.61 minutes.

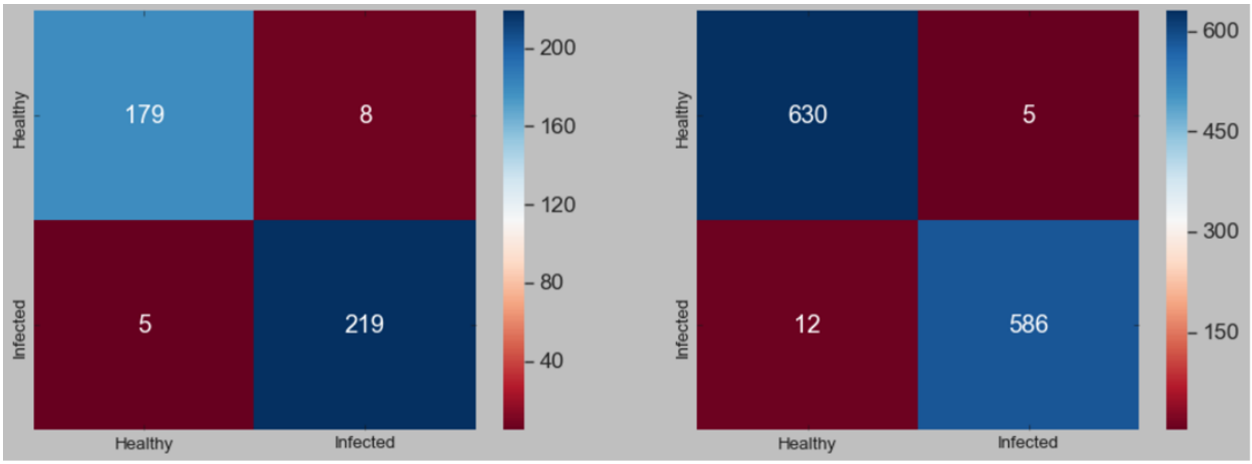
	precision	recall	f1-score	support
Healthy	0.97	0.96	0.96	187
Infected	0.96	0.98	0.97	224
accuracy			0.97	411
macro avg	0.97	0.97	0.97	411
weighted avg	0.97	0.97	0.97	411

A) For validation data

	precision	recall	f1-score	support
Healthy	0.98	0.99	0.99	635
Infected	0.99	0.98	0.99	598
accuracy			0.99	1233
macro avg	0.99	0.99	0.99	1233
weighted avg	0.99	0.99	0.99	1233

B) For training data

Figure 6. 19: Classification report of result in Table 6. 3, row 6



A) For validation data B) For training data

Figure 6. 20: Confusion matrix of result in Table 6. 3, row 6

6.4.2 Conclusion on Results in

Table 6. 3, **Row 4, 5 and 6**

As the results of the three evaluations discussed above in detail, we can easily conclude the effect of the learning rate on the model based on the results the model brought. Starting from the result on row 4, the model is trained using LR 0f 0.001 and achieved an accuracy of 99.27% whereas with result on row 5, it is evaluated using LR of 0.0001 which is lesser learning rate from the previous model and achieved an accuracy of 97.57%, which has also decreased accuracy from the previous result, finally we continue decreasing the LR, like the result on row 6 which decreases the LR to 0.00001, it still decreased the accuracy to 96.23%. This clearly shows that decreasing the LR when still other parameters are the same, also decreases the efficiency of the model. Therefore MosNet model the LR value of 0.001 has achieved an efficient model. Bear in mind that we have the same parameters for all 3 models and the only difference is their learning rate value:

LR=0.001➔Accuracy of **99.27% --------Best model**

LR=0.0001➔Accuracy of **97.57%**

LR=0.00001➔Accuracy of **96.23%**

Table 6. 4: Comparison Table for Results With Different Dropout Rates

No.	Epochs	Learning Rate	Test ratio	Dropout	Training time(minutes)	Accuracy	Error
1	300	Adam(0.001)	25%	30%	170.90	99.51%	0.49%
2	300	Adam(0.001)	25%	50%	196.24	99.27%	0.73%

6.4.3 Result comparison based on the Dropout ratio.

As explained in section [2.11 Dropout], it helps us to overcome overfitting by setting outputs on previous neurons, so that the model couldn't generalize features which drives it to overfit easily. Dropout drops some neurons from the previously hidden layers, but that doesn't mean we have to dropout all the neurons from the previous layer which affects the model not to learn some features from its preceding layers. In MosNet model 30%(0.3) dropout rate has performed best from all, but it depends on the amount of data set we have and this dropout rate might not work on other data sets, so we have to bear in mind that the dropout rate performed best on MosNet model doesn't necessarily mean it will perform the same on another data set types and models.

	precision	recall	f1-score	support
Healthy	1.00	0.99	0.99	219
Infected	0.99	0.99	0.99	192
accuracy			0.99	411
macro avg	0.99	0.99	0.99	411
weighted avg	0.99	0.99	0.99	411

A) Dropout = 0.5

	precision	recall	f1-score	support
Healthy	0.99	0.99	0.99	199
Infected	1.00	1.00	1.00	212
accuracy			1.00	411
macro avg	1.00	1.00	1.00	411
weighted avg	1.00	1.00	1.00	411

B) Dropout = 0.3

Figure 6. 21: Classification report comparison based on dropout rate.

On *Figure 6. 21*, we can see that the model with dropout rate 0.3(30%) has an f1-score of almost 1.00 which means, it has a better performance than the model with a dropout rate of 0.5(50%) both models performing an accuracy of 100% when tested on training data.

Figure 6. 22: Confusion matrix comparison based on Dropout rate.

As we can see in *Figure 6. 22*, it is clear that the dropout ratio of 0.3(30%) worked well, with only two misclassifying images, but if we see the confusion matrix with a dropout rate of 0.5(50%), there are three misclassifying images.

6.4.4 Result Comparison Based on Train Test Split

There is no common standard for splitting training and test ratio because it depends on the type of data used, the ability of the model to filter information quickly and the amount of data used to train our model. But, there are some ratios that are most commonly effective in different researches and models and besides that, the logic behind this concept is every model needs to learn with more and more data to reach its highest accuracy. For example, a model that is trained with 60% of total available data and a model that is trained with 20% data does not perform equally, because the one which is trained with more data can learn more features and its accuracy will be more than the one which is trained with fewer data. In the case of this study, the ratio that performs best is 75% train and 25% test split ratio.

6.5 Discussion of MosNet Model Evaluated on RGB Images

After training the MosNet model on grayscale images, the model couldn't achieve a satisfying result. So it is needed to try it on RGB images so that the model can extract more features from colored images. As discussed in the above section, we have achieved the highest accuracy level on MosNet model after trying different methods and data set types on RGB images, achieving 99.51% of accuracy. After doing a lot of experiments for the MosNet model by applying different parameters such as different learning rates and dropout rates, which are the two most affecting parameters when creating different models. Therefore with a learning rate value of 0.001 and a dropout rate value of 0.3(30%).

6.6 Evaluating MosNet Model on Segmented Images

After evaluating MosNet model on RGB images we have achieved the maximum accuracy of 99.51%, but even if the model has achieved high accuracy, it may decrease as the number of validation data increases in the real world, so as much as possible it should be to the model's maximum capacity to detect an image with a better accuracy than the previous one. Segmentation helps to get more important and the only needed features from the image, and that will increase the accuracy of the model because the model can get more refined information from the images. Therefore as explained in section [4.5 Color Code Segmentation], the wheat dataset is color segmented to increase the accuracy of MosNet model.

Table 6. 5: MosNet Best Result Found on Segmented Images

No.	Epochs	Learning Rate	Test ratio	Dropout	Training time(minutes)	Accuracy	Error
1	300	Adam(0.001)	25%	30%	166.40	99.76%	0.24%

As shown in *Table 6. 5*, the accuracy of the model has increased from the previous model, which was 99.51% to 99.76%, this is a remarkable change when it's taken to the real world, because, there will be a huge number of data in the real world and as long as the accuracy is increased, the model will not be vulnerable for more losses.

	precision	recall	f1-score	support
Healthy	1.00	1.00	1.00	610
Infected	1.00	1.00	1.00	623
accuracy			1.00	1233
macro avg	1.00	1.00	1.00	1233
weighted avg	1.00	1.00	1.00	1233

Figure 6. 23: Classification report of result for Table 6. 5

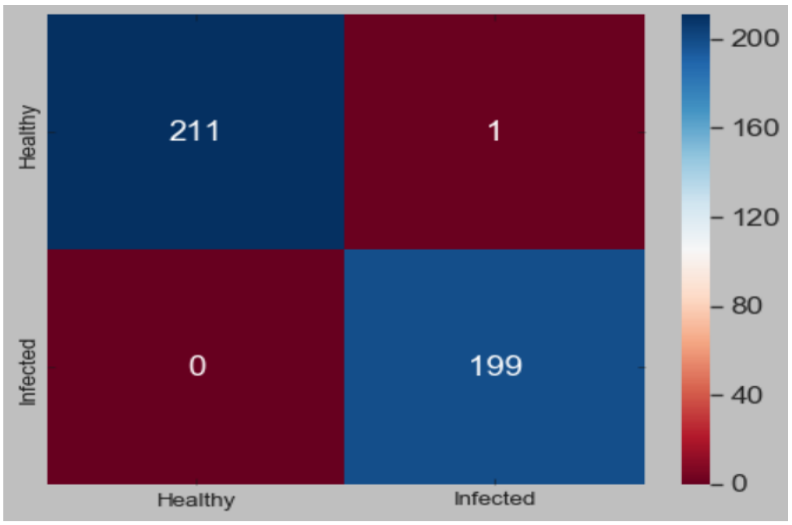
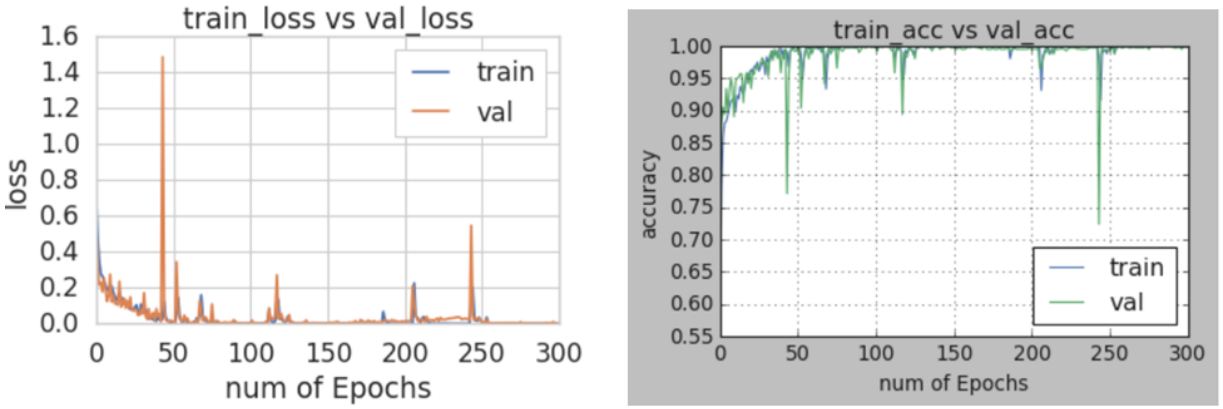


Figure 6. 24: Confusion matrix of result for Table 6. 5



A) Loss *B)Accuracy*

Figure 6. 25: Loss and Accuracy graph of the best model with segmented images

6.6.1 Discussion of MosNet Model Evaluated on Segmented Images

As shown in *Figure 6. 24*, there is only one misclassifying image in the validating data, that is one healthy image is misclassified as infected image and we can see that on the classification report in *Figure 6. 23*, that the infected label scored a precision value of 99% while all others are 100%. Having only 1 misclassifying image from a total of 411 validating data is a nice achievement which is improved decreased by 1 misclassifying image from the previous model, the one without segmented images.

In *Figure 6. 25*, we can see that the Loss and accuracy graph moves in inversely proportional if we see the Loss graph, both the training and validation losses are diverging to zero starting from loss percentage of around 1.5(150%) loss. On the last epoch, the loss is almost zero and at the same time, there is no significant space gap between the training and validating losses(the blue and orange colored lines respectively) that both have the closest values, which shows the model is neither overfitting nor underfitting. This shows the model is learning perfectly. For the accuracy part, the plots move in the opposite way to the Loss graph, as the loss graph starts from a bigger number and degrades to zero, which means the model loss is decreasing and the probability of delivering false information is less. The opposite way works for the accuracy graph, as the graph starts from a number close to zero and increases to 1 which means accuracy is converging to 100%. This implies the probability of the model to deliver accurate or righteous information is high. Additionally, there is still no way that the model overfits or underfits, as it is shown on the graph, that both the training and validation line on the plot(the blue and green lines respectively) have closer results, which shows the model is predicting almost 100% for validation data as trained on for training data.

6.7 Evaluating HUF Depth Approach for MosNet on Segmented Images

Table 6. 6: Summary table for HUF depth approach on MosNet model

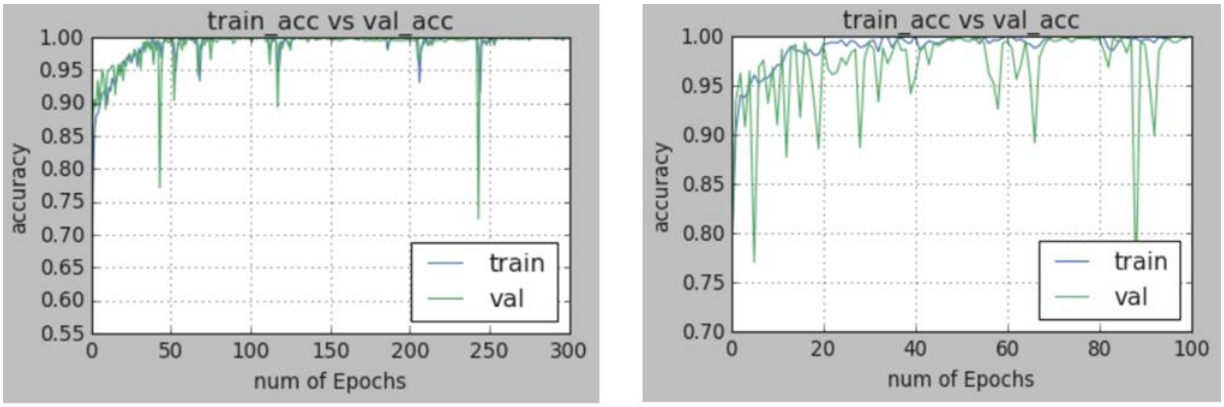
No.	Epochs	Learning Rate	Test ratio	Dropout	Training time(minutes)	Accuracy	Error
1	100	Adam(0.001)	25%	30%	69.89	99.76%	0.24%
2	100	Adam(0.00001)	25%	30%	63.70	92.21%	7.79%
3	100	Adam(0.001)	25%	50%	61.90	98.05%	1.95%
4	100	Adam(0.001)	20%	50%	64.41	97.57%	2.43%

6.7.1 Discussion of MosNet Model Evaluated Using HUF Depth Approach

As explained in section [4.2.2 Depth-wise Separable Convolutions], Depthwise Separable convolutions minimize the number of calculations and operations in the convolution. Decreasing the number of operations decreases the number of parameters computed in the convolution, which proportionally decreases the computing power. There are a total of 1.2 million parameters and when using HUF depth separable convolution, it is decreased to 14

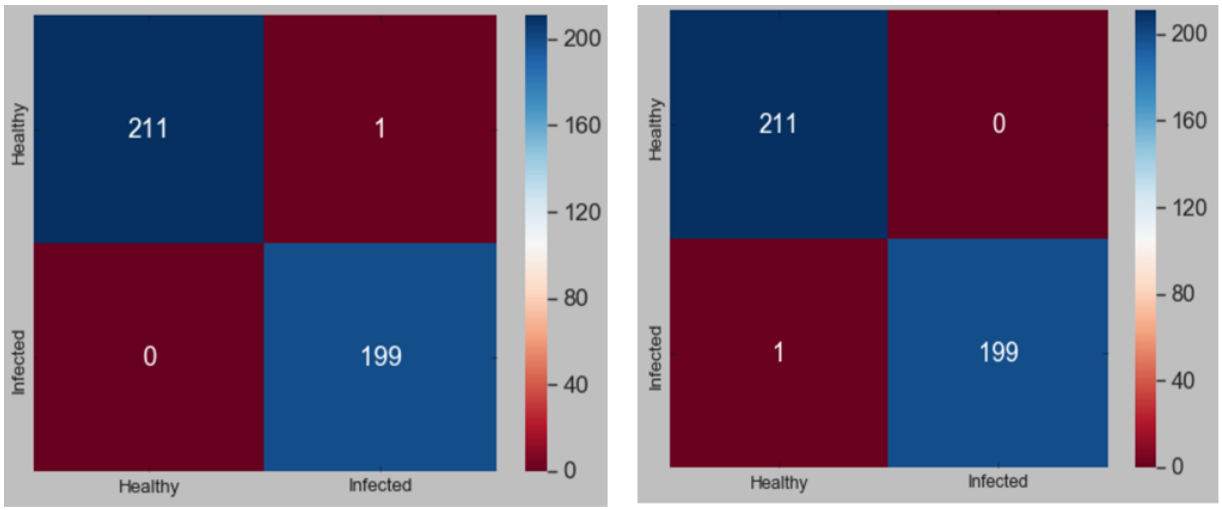
thousand parameters, which creates a huge change in the number of epochs to be applied to achieve the same accuracy. When the number of epochs decreased from 300 to 100 to achieve the same accuracy means, the model will be easy to be applicable on devices with a limited number of computational resources and this eases the model to be deployed on simple mobile devices with minimum computational powers.

As shown in the first row of *Table 6. 6*, the model has achieved an accuracy of 99.76% with only 100 epochs and training time of 69.39 minutes only, which is much improved in different aspects from the model which achieved the same accuracy in *Table 6. 5*, but taking too much time which is 166.40 minutes and 300 epochs. Taking both models to real-world scenarios has a great impact on issues like computational resources and computing time. The model that uses HUF depth separable performs best with a minimum computing time and resource.



A)MosNet without HUF depth *A)MosNet with HUF depth*

Figure 6. 26: Accuracy graph of MosNet model with and without HUF depth approach



A)MosNet without HUF depth *A)MosNet with HUF depth*

Figure 6. 27: Confusion matrix of MosNet model with and without HUF depth approach

As shown in *Figure 6. 26*, there are two accuracy graphs(***A and B***), the first one is the model which didn't use the HUF depth approach and the second one is the model which used HUd depth approach, but the difference is that the first model takes 300 epochs to achieve 99.76% of accuracy and the second one which achieves the same accuracy with only 100 epochs .

In *Figure 6. 27*, the confusion matrix shows that there is only one misclassifying image in each model, the one difference is the models misclassifies on different classes, The first model misclassifying the healthy image as infected and the second one misclassifies infected image as healthy.

CHAPTER SEVEN

7. Conclusion, Recommendation and Future Works

7.1 Conclusion

The world should be saved from suffering from disasters that happened in agriculture because it is the reason to live for every humankind in the world but, being threatened by crop diseases which are the main reason for crop reduction, especially in developing countries like Ethiopia. Digital technologies that can help for early detection of these crop diseases should be delivered from computer scientists and these systems should be quite efficient as much as possible and crop disease detection using images of the infected crops are the most commonly used techniques used in machine learning so far.

This study discussed different CNN models by applying different important factors that can affect the model designed in the study. Three data set types are used in the study to conduct the experiments for the MosNet model, and these data set types are: Grayscale image data set, a data set which only contains one channel images, RGB data set, a data set with 3 channel images and RGB color segmented data set, a data set which is RGB and segmented with the diseases color code. MosNet model has achieved an accuracy of 86.62% with 200 epochs. 0.001 learning rate and a 50% dropout rate. This result is improved when the model is trained on the RGB image data set, which climbed to an accuracy of 99.51%. Finally, after segmenting the images using the color of the infected images, the model extracted better information than the previous model and achieved an accuracy of 99.76% with 300 training epochs, the learning rate of 0.001 and the dropout rate of 30%.

HUF depth separable convolution approach is used to minimize the number of calculations and parameters, which helps the model whether to improve the model accuracy or achieve the same accuracy within small computational resources. Therefore MosNet achieved using the HUF Depth approach an accuracy of 99.76% within only 100 training epochs, which minimizes the training time, and this helps to use devices with small computational resources and can do the detection in a better timely manner than the previous model.

Finally, the most important factors that have impacts on the efficiency and accuracy of the model are a number of training **epochs**, **learning rate**, and **dropout ratio**. The number of epochs is the number of iterations the model learns the whole data set. It is not fixed that in how many epochs a model should be trained, but it obvious that the model learns more as it is trained more and more having a limit to stop somewhere and depends on the model architecture, type and size of the dataset. This study has achieved the highest accuracy with 300 epochs for the MosNet model, but it is decreased 100 epochs when it is used by the HUF depth separable convolution approach because the approach minimizes the number of calculations in the convolution operation. The other factor is the learning rate and MosNet achieved the highest accuracy of 99.76% on both the standard convolution and HUF depth convolution with the learning rate of 0.001 ratios. The last factor that impacts the efficiency of the model is the dropout rate which helps to drop some features from the neurons before the model overfits and the 30% dropout rate is the best performing ratio in this study.

7.2 Recommendation

To be more precise on the result of this study, it would be better to apply it to other types of crop diseases that can be identified by their colors from the healthy crops in a large and complex real-world environment.

The very critical issue I recommend to move this study one step forward especially in Ethiopia is that there should be governmental or private institutes that can provide researches with enough data that can enable them to conduct a genuine and real-world application that can help the agricultural sector move forward to a modern agricultural era for Ethiopia.

7.3 Future Works

This study delivered a CNN model that can effectively monitor wheat crop health which is quite helpful for early protection of the wheat farm before the disease spreads out and make total damage to the crop. This is a nice thing for early prevention of total crop loss, but it is not enough for statistical data, which means the detection should be with identifying which type of wheat rust disease occurred and in how much extent it occurred on the farm. When we say ‘’type of wheat rust diseases” are explained in section [2.1.2 Wheat Rusts] briefly.

To do this we need to have enough and quality image data from different areas, taken at different times, because, images taken during raining time and during sunny time have also their own impact on the learning efficiency of a model when we have so many classes. Therefore having data with a variety and large set will help us to take this study a long way to forward to help the agricultural system of the country.

Collecting enough and well-refined data on different humidities, on other crops also will help this study to progress to work on other different kinds of crops and their disease types to apply CNN models to the real world in short period of time in Ethiopia since lack of enough data limits the study not to progress more than the result found currently.

References

[1] A. Seyoum Taffesse, P. Dorosh, and S. A. Gemessa, “Crop production in Ethiopia: Regional patterns and trends,” *Food Agric. Ethiop. Prog. Policy Challenges*, vol. 9780812208, pp. 53–83, 2013.

[2] FAO, “Ethiopia battles wheat rust disease outbreak in critical wheat-growing regions,” 2016. [Online]. Available: http://www.fao.org/emergencies/fao-in-action/stories/stories-detail/en/c/451063/. [Accessed: 20-Mar-2019].

[3] A. Fallis, “Wheat”. [Online]. Available: http://www.ecx.com.et/Pages/Wheat.aspx?AspxAutoDetectCookieSupport=1. [Accessed: 12-Feb-2019].

[4] ReliefWeb, “Ethiopia battles wheat rust disease outbreak in critical wheat-growing regions,” 2016. [Online]. Available: https://reliefweb.int/report/ethiopia/ethiopia-battles-wheat-rust-disease-outbreak-critical-wheat-growing-regions. [Accessed: 10-Mar-2019].

[5] Teklay Abebe Teferi, "Wheat Leaf Rust (Pucinnia triticina) Epidemics and Host Plant Response in South Tigray, Ethiopia" , Interantional Journal of Plant Pathology , Malaysia, 2015-21-2015.

[6] M. Sticklen, “Advances in Crop Science and Technology Transgenic , Cisgenic , Intragenic and Subgenic Crops,” vol. 3, no. 2, pp. 2–3, 2015.

[7] P. A. Miceli, W. D. Blair, and M. M. Brown, *Isolating Random and Bias Covariances in Tracks*. 2018.

[8] B. HaileMariam, “Gov’t to Spend 4b Br Buying Wheat.” [Online]. Available: https://addisfortune.news/govt-to-spend-4b-br-buying-wheat/. [Accessed: 09-May-2019].

[9] “Ethiopia’s Wheat Production on the Rise.” [Online]. Available: https://www.2merkato.com/news/alerts/5672-ethiopia-s-wheat-production-on-the-rise. [Accessed: 19-May-2019].

[10] E. Hailu, “Distribution of Stem Rust (Puccinia graminis f. sp. tritici) Races in Ethiopia,” *Plant*, vol. 3, no. 2, p. 15, 2015.

[11] USDA, “Ethiopia Wheat Production.” [Online]. Available: https://ipad.fas.usda.gov/rssiws/al/crop_production_maps/eafrica/ET_wheat.gif. [Accessed: 19-Jan-2019].

[12] IndexMundi, “Ethiopia Wheat Imports by Year.” [Online]. Available: https://www.indexmundi.com/agriculture/?country=et&commodity=wheat&graph=imports. [Accessed: 19-May-2019].

[13] GreenLife, “Wheat Rust (1).” [Online]. Available: https://www.greenlife.co.ke/wheat-rust/. [Accessed: 25-Mar-2019].

[14] S. N. Wegulo, E. P. Pathologist, and E. Byamukama, “Rust Diseases of Wheat,” 2000. [Online]. Available: https://ohioline.osu.edu/factsheet/plpath-cer-12.

[15] X. Xie, S. Roither, D. Kartashov, L. Zhang, A. Baltuška, and M. Kitzler, “Channel-resolved subcycle interferences of electron wave packets emitted from H2 in two-color laser fields,” *High Power Laser Sci. Eng.*, vol. 4, no. 1, pp. 1–13, 2016.

[16] P. Walstra, J. T. M. Wouters, and T. J. Geurts, *Second Edition Second Edition*, no. June. 2005.

[17] “BI vs. Big Data vs. Data Mining,” 2015. [Online]. Available: https://selecthub.com/business-intelligence/bi-vs-big-data-vs-data-mining/. [Accessed: 19-Apr-2019].

[18] “Data Mining vs. Statistics vs. Machine Learning,” *DeZyre*, 2017. [Online]. Available: https://www.educba.com/data-mining-vs-machine-learning/. [Accessed: 12-Mar-2019].

[19] J. Hurwitz, D. Kirsch, and J. Wiley, *Machine Learning Machine Learning For Dummies*, no. june. 2018.

[20] Sambit Mahapatra, “Why Deep Learning over Traditional Machine Learning?,” *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/why-deep-learning-is-needed-over-traditional-machine-learning-1b6a99177063. [Accessed: 15-Feb-2019].

[21] A. K. Singh, B. Ganapathysubramanian, S. Sarkar, and A. Singh, “Deep Learning for Plant Stress Phenotyping: Trends and Future Perspectives,” *Trends Plant Sci.*, vol. 23, no. 10, pp. 883–898, 2018.

[22] H. Wehle, “Machine Learning – Artificial Intelligence- COGNITIVE,” July, 2017.

[23] S. Albawi, T. A. Mohammed, and S. Al-Zawi, “Understanding of a convolutional neural network,” *Proc. 2017 Int. Conf. Eng. Technol. ICET 2017*, vol. 2018–Janua, no. April 2018, pp. 1–6, 2018.

[24] M. Mishra, “Convolutional Neural Networks,” *Medical Imaging and Information Sciences*, 2017. [Online]. Available: https://www.datascience.com/blog/convolutional-neural-network. [Accessed: 12-Apr-2019].

[25] U. Udofia, “Basic Overview of Convolutional Neural Network (CNN),” *Medium*, 2018. [Online]. Available: https://medium.com/@udemeudofia01/basic-overview-of-convolutional-neural-network-cnn-4fcc7dbb4f17. [Accessed: 20-May-2019].

[26] R. Prabhu, “Understanding of Convolutional Neural Network (CNN) Deep Learning,” *Medium.Com*, 2018. [Online]. Available: https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148. [Accessed: 19-Feb-2019].

[27] J. Brownlee, “A Gentle Introduction to Pooling Layers for Convolutional Neural Networks,” *Deep Learning for Computer Vision*, 2019. [Online]. Available: https://machinelearningmastery.com/pooling-layers-for-convolutional-neural-networks/. [Accessed: 14-Feb-2019].

[28] V. Zhou, “An Introduction to Convolutional Neural Networks,” 2019. [Online]. Available: https://victorzhou.com/blog/intro-to-cnns-part-1/. [Accessed: 14-May-2019].

[29] R. Venkatesan, B. Li, R. Venkatesan, and B. Li, “Convolutional Neural Networks,” *Convolutional Neural Networks in Visual Computing*, 2018. [Online]. Available: https://divishd.wordpress.com/2017/08/24/convolutional-neural-networks-cnn-applications-in-nlp-for-sentence-classification/. [Accessed: 12-Apr-2019].

[30] SAGAR SHARMA, “Activation Functions in Neural Networks,” *Towardsdatascience.Com*, 2017. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed: 25-Apr-2019].

[31] N. C. Steele, C. R. Reeves, and E. I. Gaura, “Activation Functions,” *Artificial Neural Nets and Genetic Algorithms*, 2001. [Online]. Available: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6. [Accessed: 16-Mar-2019].

[32] J. Brownlee, “How to Avoid Overfitting in Deep Learning Neural Networks,” 2018. [Online]. Available: https://machinelearningmastery.com/introduction-to-regularization-to-reduce-overfitting-and-improve-generalization-error/. [Accessed: 12-Jun-2019].

[33] N. Schlüter, “How to prevent Overfitting in your Deep Learning Models.” [Online]. Available: https://towardsdatascience.com/dont-overfit-how-to-prevent-overfitting-in-your-deep-learning-models-63274e552323. [Accessed: 12-May-2019].

[34] R. Zadeh and B. Ramsundar, “Tensorflow for Deep Learning,” 2018. [Online]. Available: https://www.oreilly.com/library/view/tensorflow-for-deep/9781491980446/ch04.html. [Accessed: 11-Apr-2019].

[35] G. Van Rossum and F. L. Drake, “Python Tutorial,” *History*, 2010. [Online]. Available: http://docs.python.org/tutorial/. [Accessed: 03-Dec-2018].

[36] Tensorflow, “Tensorflow.” [Online]. Available: https://www.tensorflow.org/tutorials/. [Accessed: 29-Dec-2018].

[37] Keras, “Keras.” [Online]. Available: https://keras.io/. [Accessed: 12-Dec-2018].

[38] A. Ramcharan, K. Baranowski, P. McCloskey, B. Ahmed, J. Legg, and D. P. Hughes, “Deep Learning for Image-Based Cassava Disease Detection,” *Front. Plant Sci.*, vol. 8, no. 2002, pp. 1–10, 2017.

[39] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using Deep Learning for Image-Based Plant Disease Detection,” *Front. Plant Sci.*, vol. 7, 2016.

[40] K. P. Ferentinos, “Deep learning models for plant disease detection and diagnosis,” *Comput. Electron. Agric.*, vol. 145, no. January, pp. 311–318, 2018.

[41] E. C. Too, L. Yujian, S. Njuki, and L. Yingchun, “A comparative study of fine-tuning deep learning models for plant disease identification,” *Comput. Electron. Agric.*, vol. 161, no. October 2017, pp. 272–279, 2019.

[42] A. Picon, A. Alvarez-Gila, M. Seitz, A. Ortiz-Barredo, J. Echazarra, and A. Johannes, "Deep convolutional neural networks for mobile capture device-based crop disease classification in the wild," *Comput. Electron. Agric.*, vol. 161, no. October 2017, pp. 280–290, 2019.

[43] S. Here, "Gentle Introduction to the Adam Optimization Algorithm for Deep Learning," *Neural Networks*, 2017. [Online]. Available: https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/. [Accessed: 10-Dec-2018].

[44] Vitaly Bushaev, "Understanding RMSprop — faster neural network learning," *Towards Data Science*, 2018. [Online]. Available: https://towardsdatascience.com/understanding-rmsprop-faster-neural-network-learning-62e116fcf29a. [Accessed: 01-May-2019].

[45] Gandhi rohith, "A Look at Gradient Descent and RMSprop Optimizers – Towards Data Science," 2018. [Online]. Available: https://towardsdatascience.com/a-look-at-gradient-descent-and-rmsprop-optimizers-f77d483ef08b. [Accessed: 21-Apr-2019].

[46] I. Changhau, "Loss Functions in Neural Networks," 2017. [Online]. Available: https://isaacchanghau.github.io/post/loss_functions/. [Accessed: 18-Apr-2019].

[47] B. J. Mccaffrey, "Neural Network Cross Entropy Error," 2014. [Online]. Available: https://visualstudiomagazine.com/articles/2014/04/01/neural-network-cross-entropy-error.aspx. [Accessed: 16-May-2019].

[48] Google, "Classification: True vs. False and Positive vs. Negative," *Machine Learning Crash Course Google Developers*, 2019. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/true-false-positive-negative. [Accessed: 12-May-2019].

[49] S. Saxena, "In Reply: BEHAVIOUR THERAPY," *The British Journal of Psychiatry*, 1966. [Online]. Available: https://towardsdatascience.com/precision-vs-recall-386cf9f89488. [Accessed: 23-Apr-2019].

Appendix A-Interview Questions

A.1 Interview questions and answers found from Research Institutes.

1. Interviewer: How is the process of inspecting or detecting wheat rusts takes place?

Expert: By waging human labor or experts on the farm.

2. Interviewer: which rust type is most severe?

Expert: Yellow rust and Stem rusts.

3. Interviewer: How are the diseases identified one from the other?

Expert: Yellow and Leaf rusts occur on the leaf of the wheat crop, and sometimes they can be identified by their color and sometimes in laboratory tests. Stem rust occurs on the stem of the wheat and has more similarity in color with leaf rust.

4. How many experts are used to inspect the occurrence of these diseases in large agricultural fields?

Expert: It depends on the expert resource and size of the farm, but not enough yet to inspect huge farm fields.

5. Are these rust diseases transmitting diseases from one wheat crop to the other?

Expert: Yes.

6. How they transmit from one to the other?

Expert: By wind and by any physical contact.

7. How to protect the rest of other crops before spreading?

Expert: By spraying fungicides as quick as possible.

8. Interviewer: Are the fungicides unique for each rust type?

Expert: No.

9. Interviewer: What is the name of the fungicide used to control the rusts?

Expert: Triazoles family fungicide(specifically called ''Vivax'').

Appendix B - Python Source Code

B.1 Important Python Libraries Used

```
from keras.models import Sequential
from keras.layers import Dense,Activation
from keras.layers import Dropout
from keras.layers import Flatten
from keras import optimizers
import time
from keras.layers.convolutional import Conv2D
from keras.layers.convolutional import MaxPooling2D
from keras.utils.vis_utils import model_to_dot
from keras.preprocessing.image import ImageDataGenerator
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,confusion_matrix
import seaborn as sns
from matplotlib import pyplot
import os,cv2
from random import shuffle
from tqdm import tqdm
import tensorflow as tf
from keras.utils import np_utils
import matplotlib.image as mpimg
import matplotlib.pyplot as plt
import numpy as np
import pandas as pa
from sklearn.preprocessing import LabelEncoder
from keras import backend as K
```

B.2 Python Code for Reading and Segmenting Image Data

```
def create_train_data():
    training_data=[]
    for img in tqdm(os.listdir(Train_dir)):
        label=label_img(img)
        path=os.path.join(Train_dir,img)
        #img=cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img=cv2.imread(path)
        #img=cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img=cv2.resize(img,(Img_size,Img_size))
        out = np.zeros ((img.shape[0], img.shape[1], img.shape[2]), np.float32)

        for i in range(img.shape[1]):
            for j in range(img.shape[0]):
                b = img[j, i, 0]
                g = img[j, i, 1]
                r = img[j, i, 2]

                if((b < 100 and b > 0) and (r > 130 and r < 245) and (g > 50 and g < 245)):
                    out[j,i, 0] = b
                    out[j,i, 1] = g
                    out[j,i, 2] = r
```

B.3 MosNet Model (The Standard CNN Model Part)

```
def mos_model():
    # create model
        model = Sequential()
        model.add(Conv2D(32, (3, 3), input_shape=(3, Img_size,Img_size), padding='valid'))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(32, (3, 3), padding='valid'))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))

        model.add(Conv2D(64, (3, 3), padding='valid'))
        model.add(Activation('relu'))
        model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
        model.add(Flatten())
        model.add(Dense(64))
        model.add(Activation('relu'))
        model.add(Dropout(0.3))
        model.add(Dense(2))
        model.add(Activation('sigmoid'))
        adam=optimizers.Adam(lr=0.001)

        model.compile(loss='binary_crossentropy',
                      optimizer=adam,
                      metrics=['accuracy'])
        return model
```

B.4 Python code for MosNet Model With HUF Depth Approach

```
def huf_depth_model():
    model = Sequential()
    model.add(layers.SeparableConv2D(32, 3,input_shape=(3, 75, 75),kernel_initializer='he_uniform',activation='relu'
    model.add(BatchNormalization())
    model.add(layers.SeparableConv2D(32, 3, activation='relu',kernel_initializer='he_uniform'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))
    model.add(layers.MaxPooling2D(2))
    model.add(layers.SeparableConv2D(32, 3, activation='relu',kernel_initializer='glorot_uniform'))
    model.add(BatchNormalization())
    model.add(layers.SeparableConv2D(32, 3, activation='relu',kernel_initializer='glorot_uniform'))
    model.add(BatchNormalization())
    model.add(layers.MaxPooling2D(2))
    model.add(layers.SeparableConv2D(64, 3, activation='relu',kernel_initializer='glorot_uniform'))
    model.add(BatchNormalization())
    model.add(Dropout(0.3))
    model.add(layers.SeparableConv2D(64,3, activation='relu',kernel_initializer='glorot_uniform'))
    model.add(BatchNormalization())
    model.add(layers.GlobalAveragePooling2D())
    model.add(layers.Dense(32, activation='relu',kernel_initializer='glorot_uniform'))
    model.add(BatchNormalization())
    model.add(layers.Dense(num_classes, activation='sigmoid'))
    adam=optimizers.Adam(lr=0.001)
    model.compile(loss = 'binary_crossentropy',
                  optimizer=adam,
                  metrics= ['accuracy'])
    return model
model = huf_depth_model()
model.summary()
```

B.5 Python Code for Fitting or Running the Model

```
callbacks = [EarlyStopping(monitor='val_loss', patience=15),
             ModelCheckpoint(filepath='MosNet_RGB.h5', monitor='val_loss', save_best_only=True)]
model.fit(X_train, # Features
                    y_train, # Target vector
                   epochs=nb_epoch, # Number of epochs
                   callbacks=callbacks, # Early stopping
                   verbose=1, # Print description after each epoch
                   batch_size=batch_size, # Number of observations per batch
                   validation_data=(x_test, y_test)) # Data for evaluation
```

Appendix C- Model Summaries

C.1 MosNet Model Summary of Standard CNN

```
(1233, 3, 150, 150)
(411, 3, 150, 150)
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 32, 148, 148)      896
_________________________________________________________________
activation_11 (Activation)   (None, 32, 148, 148)      0
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 32, 74, 74)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 32, 72, 72)        9248
_________________________________________________________________
activation_12 (Activation)   (None, 32, 72, 72)        0
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 32, 36, 36)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 64, 34, 34)        18496
_________________________________________________________________
activation_13 (Activation)   (None, 64, 34, 34)        0
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 64, 17, 17)        0
_________________________________________________________________
flatten_3 (Flatten)          (None, 18496)             0
_________________________________________________________________
dense_5 (Dense)              (None, 64)                1183808
_________________________________________________________________
activation_14 (Activation)   (None, 64)                0
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_6 (Dense)              (None, 2)                 130
_________________________________________________________________
activation_15 (Activation)   (None, 2)                 0
=================================================================
Total params: 1,212,578
Trainable params: 1,212,578
Non-trainable params: 0
_________________________________________________________________
```

C.2 MosNet Model Summary With HUF Depth Approach

```
(1233, 3, 75, 75)
(411, 3, 75, 75)
```

Layer (type)	Output Shape	Param #
separable_conv2d_65 (Separab	(None, 32, 73, 73)	155
batch_normalization_76 (Batc	(None, 32, 73, 73)	292
separable_conv2d_66 (Separab	(None, 32, 71, 71)	1344
batch_normalization_77 (Batc	(None, 32, 71, 71)	284
dropout_23 (Dropout)	(None, 32, 71, 71)	0
max_pooling2d_22 (MaxPooling	(None, 32, 35, 35)	0
separable_conv2d_67 (Separab	(None, 32, 33, 33)	1344
batch_normalization_78 (Batc	(None, 32, 33, 33)	132
separable_conv2d_68 (Separab	(None, 32, 31, 31)	1344
batch_normalization_79 (Batc	(None, 32, 31, 31)	124
max_pooling2d_23 (MaxPooling	(None, 32, 15, 15)	0
separable_conv2d_69 (Separab	(None, 64, 13, 13)	2400
batch_normalization_80 (Batc	(None, 64, 13, 13)	52
dropout_24 (Dropout)	(None, 64, 13, 13)	0
separable_conv2d_70 (Separab	(None, 64, 11, 11)	4736
batch_normalization_81 (Batc	(None, 64, 11, 11)	44
global_average_pooling2d_12	(None, 64)	0
dense_23 (Dense)	(None, 32)	2080
batch_normalization_82 (Batc	(None, 32)	128
dense_24 (Dense)	(None, 2)	66

```
Total params: 14,525
Trainable params: 13,997
Non-trainable params: 528
```