	Jesmin Akther M.Sc. Eng. (ICE)2018 NSTU	

1.

CLASSIFICATION OF EARLY AND LATE BLIGHT DISEASE OF POTATO USING CONVOLUTIONAL NEURAL NETWORK

Jesmin Akther

MASTER'S OF SCIENCE ON ENGINEERING

NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

CLASSIFICATION OF EARLY AND LATE BLIGHT DISEASE OF POTATO USING CONVOLUTIONAL NEURAL NETWORK

Jesmin Akther

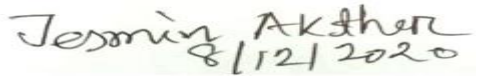
Thesis submitted in fulfillment of the requirements for the award of the degree of masters of science of Engineering in in Information and Communication Engineering.

Department of Information and Communication Engineering.

NOAKHALI SCIENCE AND TECHNOLOGY UNIVERSITY

STUDENT DECLARATION

I hereby declare that the work in this thesis is my own except for quotations and summaries which have been duly acknowledged. The thesis has not been accepted for any degree and is not concurrently submitted for award of other degree.



Signature

Name: Jesmin Akther.

ID Number: BKH1711MSc114F

Date: 8 December 2020

SUPERVISOR'S DECLARATION

We, hereby declare that we have checked this thesis and in our opinion, this thesis is adequate in terms of scope and quality for the award of the degree of Master of engineering of science in Information and Communication Engineering.

Signature

Name of Supervisor: Dr. Md. Ashikur Rahman Khan

Position: Professor & Chairman
Dept. of ICE.

Date: 8 December 2020

Signature

Name of Co-supervisor: Sultana Jahan Soheli

Position: Assistant Professor

Dept. of ICE.

Date: 8 December 2020

ACHNOWLEDGEMENT

I am grateful and indebted to my supervisor Professor Dr. Md. Ashikur Rahman Khan for his invaluable guidance, continuous encouragement and constant support in making this research possible. I appreciate his consistent support from the first day I applied to graduate program to these concluding moments. In true sense, he is a friend, philosopher and guide. Even volumes of words will fall short in thanking him for the kind of support and guidance given by him.
I also would like to express very special thanks to my co-supervisor Assistant Professor Sultana Jahan Soheli for her suggestions and co-operation throughout the study. I also sincerely thanks for the time spent proofreading and correcting my many mistakes.

My sincere thanks go to all my members and staff of the Information and Communication Engineering Department, NSTU, who helped me in many ways and made my stay at NSTU pleasant and unforgettable. Many special thanks go to member programming research group for their excellent cooperation, inspirations and supports during this study.

I acknowledge my sincere indebtedness and gratitude to my parents for their love, dream and sacrifice throughout my life. Special thanks should be given to my mother who helps me a lot data research and study. I would like to acknowledge her suggestions which was crucial for the successful completion of this study.

ABSTRACT

Crop diseases are a major threat to cultivate, and need to supervise growth and detrimental diseases in time. Potato early and late blight disease symptoms and detection both are vague to distinguish and isolate. The rising combination of smartphone penetration and recent advances in deep learning has paved the way for smart device assisted disease prognosis. Relying on pure naked-eye observation to detect and classify diseases can be very cumbersome. Absolute detection process proves to be effective and convenient for researchers. The technique of training deep learning models on increasingly vast and globally available image datasets presents a clear path toward smartphone aid crop disease experiment around the world. The color and analyze layer features are used to best match to recognize and classify different agriculture produce into early blight and late blight affected disease. Both features prove to be very effective in disease detection. This paper deployed a Sequential convolutional neural network (CNN) model to detect and identify diseases in real time survey potato leaves labeling early and late blight. In addition to normalization, divide and extract the images to prepare data prior CNN. This work adopts slight variation during CNN model finalization with the help of Tensorboard analysis. This analyzed effectively optimized model validation accuracy each layer by layer hierarchically. Best layers ensure to create the final model which assures 94% accuracy for this dataset and short timing classification. The experimental results indicate that the approach significantly can be modified model accuracy in automatic detection of both affected bight.

TABLE OF CONTENTS

CHAPTER 2 LITERATURE REVIEW

CHAPTER 4 RESULT AND DISCUSSION

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

xx	feature
yy	label
ACC	Accuracy
VAL_ACCURACY	Validation Accuracy
VAL_LOSS	Validation_LOSS

LIST OF ABBREVIATIONS

JPEG	Joint Picture Expert Group
CNN	Convolutional Neural Networks
SVM	Support Vector Machine
MLP	Multilayer perceptron
ConvPool	Convolution and Pooling
ReLU	Rectified linear unit
GPU	Graphics Processing Unit
Colab	Google Colaboratory
NN	Neural Network
DCNN	Deep Convolutional Neural Networks
API	Application Programing Interface

CHAPTER 1

INTRODUCTION

1.1 Background

This chapter discuss about the importance of potato cultivation, their disease symptoms and related work that already applied for disease detection. Bangladesh's potato export world's fourth largest food crop after maize, wheat, and rice. Potato export reached a record high of over 100,000 tonnes in 2013-14, in April 2019 the amount reach 27,811.6 tonnes found by Molla [1]. The Russian Federal Service for Veterinary and Phytosanitary Surveillance said the potatoes from Bangladesh were infected by brown rot disease and potato tuber moth, as there is no accredited or standard lab in Bangladesh to test the quality of potatoes found by Zahangir [2]. Including brown rot disease and potato tuber moth diseases, there are so many disease of potatoes such as potato late blight and potato early blight are detrimental both of crops growth and glut production. Disease symptoms occur at any age of plants including yellowing leaves, stunting and wilting plants [3]. Potato early blight and late blight are difficult to detect and prevent on naked eye for farmers, in consequences, glut growth of crops lack of proper steps. Late blight can be controlled applying costly fungicides. Moreover, the degree of control heavily depends on the timing of the fungicide application in relation to local weather conditions, crop development and disease pressure. In Bangladesh 1922 late blight was first reported, screening for the blight disease resistance was carried out under both natural and artificial conditions [4]. This disease appears to be severe to succulent plants by wilting of leaves and collapse of stems. Annual potato yield losses due to late blight have been estimated at 25-57%. So, the efficiency of late blight control can be improved considerably by informing farmers in time about predicted

infection periods of the potato crop and the effectiveness of past spray applications illustrated by Kessel [5]. Trials at Michigan State University (MSU) in 2013 indicated that Ridomil applied curatively to blighted foliage did not successfully prevent further disease development Control of foliar late blight with fungicides in 2013 at the Clarksville Research Station [6].First symptoms of early blight appear as small, circular or irregular, dark-brown to black spots on the older leaves. The affected potato and potato crops in early blight and late blight diseases is shown in, Figure 1.1.

(a) Early blight effected potatoes

(b) Effected potato crops

(c) Late blight effected potatoes

(d) Effected potato crops

Figure 1.1. Effected potatoes and potato crops.

These spots enlarge up to 3/8 inch in diameter and gradually may become angular-shaped (Figure 1.1 b). Late blight damages leaves, stems and tubers shows on figure 1.1(c). Affected leaves eventually rot and dry out. When drying out, leaves turn brown or black in color. When active infections, spots appear on the underside of leaves blanketed in what looks like flour. Affected stems begin to blacken from their tips, and eventually dry out. Severe infections cause all foliage to rot, dry out and fall to the ground, stems to dry out and plants to die. Affected tubers display dry brown-colored spots on their skins and flesh. This disease acts very quickly. If it is not controlled, infected plants will die within two or three days [7].

1.2 Statement of Problem

Plant diseases Identification is the key to preventing the losses in the yield and quantity of the agricultural product. The revisions of the plant diseases means the studies of visually observable patterns seen on the plant. Health monitoring and disease detection on plant is very critical for sustainable agriculture. This is very difficult to monitor the plant diseases manually. The diagnostician must have very good observation skills, and needs to be a good detective. Without proper identification of the disease and the disease-causing agent, disease control measures can be a waste of time and money and can lead to further plant losses. Identification of affected plants is one of the first steps in diagnosing a plant disease. Proper disease diagnosis is therefore essential. Often, plant pathologists have to rely on symptoms for the identification of a disease problem. It requires tremendous amount of work, expertize in the plant diseases. Manually disease detection complex in nature and is a time consuming process. Also requires the excessive processing time. Recurrently ranchers are maladroit to virtual pattern detection that leads inaccurate imagination. Due to misconstrue intensity of the disease, pesticide's under-dosage or over-dosage retract healthy crops. Therefore need a soothe manual disease prediction through boon of methodology of deep learning along minimal error.

1.3 Objectives

The objective for potato disease are pointed below:

a) To find a solution to the problem of classify between potato early blight and late blight disease detection using Convolutional Neural Network (CNN).
b) CNN model can be deployed into smart device that can pave the way identify disease by smartphone or fitted various smart devices.
c) Earlier potato blight disease detection via CNN model classification.
d) To the simplest approach while making use of minimal computing resources to achieve results comparable to state of the art techniques.
e) By monitoring service for disease control and cure in early stage.
f) The timely recognition of potato plant infection by disease or other could be helpful and valuable source of information for performing the exact pest management with cure and infection, manage measures to avoid the growth and the spread of potato harmful diseases.

1.4 Scope of the Research

Advance technology could be ensure disease strains as early and as quickly as possible to protect crop yields and reduce dependency on pesticides. A vital step in inscribe potato crop disease is the implementation of adequate surveillance strategies so that rapid, in field diagnosis can be made. The diagnostic system will deliver in-field results in minutes while current lab tests can take up to one week. Diseased plants lead to declassification or even rejection of the seed lots resulting in a financial loss shown by authors Gerrit Polder et al [8]. Farmers put in a lot of effort to detect diseased plants and remove diseased plants from the field. Nevertheless, dependent on the cultivar, diseased plants can be missed during visual observations in particular in an early stage of cultivation. Therefore, there is a need for fast and objective disease detection. Early detection of diseased plants with modern vision techniques can significantly reduce costs.

1.5 Overview of the Thesis

Chapter 1 gives the introduction covering brief account of plant disease, Problem and difficulties and contribution of information technology in agriculture. In chapter 2 practical and theoretical concepts of disease detection and classification method are discussed. The limitation and difficulties of classification and detection method are explained. Brief analysis of two potato disease problems are identified. Chapter 3 deals with the experiment layout of potato disease detection. The main motive of this chapter is to find out the infected lesion in potato leaves. Infected lesion is found with the help of image pre-processing background separation and color filtering operation. Then prepare data to train dataset and analyze and optimize dataset for classify dataset. Chapter 4 the classification of disease is discussed and developed model. The train model discriminate analysis based classification method is used for identification of disease and the result are quite accurate. Last chapter 5 conclusion and future of this research are enumerated in detail. In the next chapter related word with various way to crop disease detection, boon of deep learning role in advance era are reviewed. Mobile assisted different algorithm are discuss for reviewing related work. Limitation and research are discuss in literature review chapter.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

Agricultural crop have become a beneficial resource of power energy and also mystery to resolve the difficulty of global warming. There are various disease that influence agriculture plants cause to disturbing cost-effective, environmental losses. There are numerous ways to identify plant disease. Generally various kind of laboratory analysis, usually by way of powerful and accurate microscope, is compulsory. With increasing penetration of internet and smart devices this is possible using remote sensing technique remote disease detection through image process capture by multi-hyperspectral digital image captures. Digital image processing helps to solve many problems or reduce by use of pattern recognition and statistical analysis tools. In chapter two various technique are discussed for plant disease detention and classification. The chapter is divided that is potato blight disease, disease sensing, Disease detection method, disease classification.

2.2 CNN Analysis

A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers typically consist of a series of convolutional layers that convolve with a multiplication or other dot product. The activation function is commonly a rectified linear unit (RELU) layer, and is subsequently followed by additional convolutions such as pooling layers, fully connected layers and normalization layers, referred to as hidden layers because their inputs and outputs are masked by the activation function and final convolution. Mathematically, convolution is technically a sliding dot product or cross-correlation.

2.3 Convolutional layer

The convolutional layer is the core building block of a CNN. The layer's parameters consist of a set of learnable filters or kernels, which have a small receptive field, but extend through the full input volume. The image data is represented into each color intensity in a color channel at a given point, the most common one being RGB, which means Red, Blue and Green. The information contained into an image is the intensity of each channel color into the width and height of the image During the forward pass, each filter is convolved across the width and height of the input volume, computing the dot product between the entries of the filter and the input and producing a 2-dimensional activation map of that filter up pose that we have some N×N square neuron layer which is followed by our convolutional layer. If we use an m × m filter ω, our convolutional layer output of size (N−m+1) × (N−m+1). In order to compute the pre-nonlinearity input to some unit xℓij in our layer, we need to sum up the contributions (weighted by the filter components) from the previous layer cells generally, the convolutional layer output can be represented by Eq. (1),

$$x_{ij}^{l} = \sum_{a=0}^{m-1} \cdot \sum_{b=0}^{m-1} w_{ab} Y_{(i+a)(j+b)}^{l-1} \qquad (1)$$

This is just a convolution.

2.3.1 Depth

In Deep Neural Networks the depth refers to how deep the network is but in this context, the depth is used for visual recognition and it translates to the dimension of an image. For an image size 32x32x3, In this case the size of this input is 32x32x3 stand as width, height, depth. The neural network should be able to learn based on this parameters as depth translates to the different channels of the training images. For example, if the first convolutional layer takes the raw image as input, then different neurons along the depth dimension may activate in the presence of various oriented edges, or blobs of color.

2.3.2 Stride

Stride controls how depth columns around the spatial dimensions (width and height) are allocated. When the stride is 1 then we move the filters one pixel at a time. This leads to heavily overlapping receptive fields between the columns, and also to large output volumes. In practice, stride lengths of stride>3 are rare. The spatial size of the output volume can be computed as a function of the input volume size W, the kernel field size of the convolutional layer neurons K, the stride with which they are applied S, and the amount of zero padding P used on the border. The formula for calculating how many neurons "fit" in a given volume is given by Eq. (2)

$$1 + \frac{W - k + 2p}{S} \tag{2}$$

2.4 Pooling layer

The pooling layer operates independently on every depth slice of the input and resizes it spatially. The most common form is a pooling layer with filters of size 2×2 applied with a stride (S) of 2 down samples at every depth slice in the input by 2 along both width(x) and height(Y), discarding 75% of the activations. Pooling layer output can be represented by Eq. (3)

$$f_{x,y}(S) = \max_{a,b=0}^{1} S2x\,a_{0} + a, 2Y + b \tag{3}$$

2.5 ReLU activition

ReLU layer applies the non-saturating activation function f(x) =max (0, x). It effectively removes negative values from an activation map by setting them to zero. Other function are saturating hyperbolic tangent f(x)=tanh(x), f(x)=|tanh(x)| and the sigmoid function sigma (x)=(1+e^-x)^-1. ReLU is often preferred to other functions because it trains the neural network several times faster without a significant penalty to generalization accuracy.

2.6 Fully connected layer

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is in principle the same as the traditional multi-layer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

2.7 Receptive field

In a convolutional layer, neurons receive input from only a restricted subarea of the previous layer. Typically the subarea is of a square shape size 5 by 5. The input area of a neuron is receptive field. So, in a fully connected layer, the receptive field is the entire previous layer. In a convolutional layer, the receptive area is smaller than the entire previous layer.

2.8 Weights

The function that is applied to the input values is determined by a vector of weights and a bias typically real numbers. Learning, in a neural network, progresses by making iterative adjustments to these biases and weights.

2.9 Hyper parameters Selection

We cannot look at a dataset and know the best algorithm to use, let alone the best data transforms to use to prepare the data or the best configuration for a given model. We must use controlled experiments to discover what works best for a given dataset. CNNs use more hyper parameters than a standard multilayer perceptron (MLP). While the normal rules for learning rates and regularization constants still apply, the following should be kept in mind when adjusting.

2.9.1 Number of filters

Since feature map size decreases with depth, layers near the input layer will tend to have fewer filters while higher layers can have more. To equalize computation at each layer, the product of feature values with pixel position is kept roughly constant across layers. Preserving more information about the input would require keeping the total number of activations non-decreasing from one layer to the next.

2.9.2 Feature maps

The number of feature maps directly controls the capacity and depends on the number of available examples and task complexity.

2.9.3 Filter shape

Common filter shapes found in the literature vary greatly, and are usually chosen based on the dataset. This is a challenge to find the right level of granularity so as to create abstractions at the proper scale, given a particular dataset, and without overfitting.

2.10 Regularization methods

Regularization is a process of introducing additional information to solve an ill-posed problem or to prevent overfitting. To use a larger model that may be required to use regularization during training that keeps the weights of the model small. These techniques not only reduce overfitting, but they can also lead to faster optimization of the model and better overall performance. CNNs use various types of regularization.

2.10.1 Dropped out

At each training stage, individual nodes are either "dropped out" of the net with probability 1-p or kept with probability p, so that a reduced network is left; incoming and

outgoing edges to a dropped-out node are also removed. Only the reduced network is trained on the data in that stage. The removed nodes are then reinserted into the network with their original weights. The biggest contribution of the dropout method is although it effectively generates {\displaystyle 2^{n}}2^{n} neural nets, and as such allows for model combination, at test time only a single network needs to be tested.

2.10.2 Drop-Connect

Drop Connect is a generalization of Dropout, for regularizing large fully-connected layers within neural networks. When training with Dropout, that time a randomly selected subset of activations are set to zero within each layer. Drop Connect instead of sets a randomly selected subset of weights within the network to zero. Each unit hence receives input from a random subset of units in the previous layer. Drop Connect is the generalization of dropout in which each connection, rather than each output unit, can be dropped with probability 1-p. Each unit thus receives input from a random subset of units in the previous layer.

2.10.3 Stochastic pooling

A major drawback to Dropout is that it does not have the same benefits for convolutional layers.. A novel type of regularization for convolutional layers that where the neurons are not fully connected enables the training of larger models without over-fitting, and produces superior performance on recognition tasks. The main idea is to make the pooling that occurs in each convolutional layer a stochastic process. Conventional forms of pooling such as average and max are deterministic, the latter selecting the largest activation in each pooling region. In stochastic pooling, the conventional deterministic pooling operations are replaced with a stochastic procedure. This approach is free of hyper parameters and can be combined with other regularization approaches, such as dropout and data augmentation. Stochastic Pooling for Regularization of Deep Convolutional Neural Networks, for regularizing large convolutional neural networks. Neural network models are prone to over-fitting due to their high capacity. A range of regularization techniques are used to prevent this, such as weight decay, weight tying and the augmentation of the training set.

2.10.4 Number of parameters

For convolutional networks, the amount of parameters meaning modification of weights and bias that make up the cost function assure validation accuracy. Additionally the filter size also affects the number of parameters. This is a simple way to prevent overfitting is to limit the number of parameters. Limiting the number of parameters restricts the predictive power of the network directly. Reducing the complexity of the function that it can perform on the data, and thus limits the amount of overfitting.

2.10.5 Weight decay

A simple form of added regularizer is weight decay, which simply adds an additional error, proportional to the sum of weights or squared magnitude of the weight vector, to the error at each node.

2.11 Performance Modelling

Performance Modelling Increasing or decreasing contains theoritical dimentional shaping with CNN. Increasing or decreasing the number of new convolutional layers, Increasing or decreasing the number of nodes in each new convolutional layer. This predict the inference time and training time of the CNN, which enables the system to optimize the performance in time. Tuning hyper-parameters such as activation functions and learning rates. Experimenting with the image, unfreezing the fully-connected layer and adjusting or training that as well. Unfreezing one or more preexisting convolutional layers and (re-)training those as well. The best performing code implemented using ImageDataGenerator, flow from data frame, various data augmentation techniques, L2 regularization, batch normalization, one hot encoding on a subset of data considering the class weights in Keras. Keras library easy to work with python where background Tensorflow.

2.11.1 Hyper-parameter tuning

Hyper-parameter tuning refers to different types of function activation and layers. Namely function are ReLU, sigmoid,Tanh Softmax Dense layer. Apart them epochs, learning rate increase model performance with overfitting and underfitting.

- **Regularization**

Penalizing complexity of the model, penalizing big weights. Regularization encourage weights to be small but doesn't force them to exactly 0.

- **Learning Rate (LR) optimization**

Starting with a base LR and subsequently decreasing it for next epoch.

- **Batch Size**

Usually try with max batch size your GPU can handle, depends on the memory of the GPU.

- **Increase model capacity**

Increasing model depth that is more number of layers and width (number of filters in each convolution layer.

2.11.2 Data redesigning

- **Progressive resizing:**

  From 128 x 128 x 3 to 256 x 256 x 3 or to even higher size.

- **Random image rotations:**

  Change orientation of the image.

- **Random image shifts:**

  Useful when object is not at the center of the image.

- **Vertical and horizontal shift:**

  Randomly flip the image vertically or horizontally. Algorithm should identify a glass whether it's face up or face down.

2.11.3 Model optimization

- Fine tuning the model with subset data:
  Dropping few data samples for some of the overly sampled data classes.
- Class weights:
  Used to train highly imbalanced (biased) database, class weights will give equal importance to all the classes during training.
- Fine tuning the model with train data:
  Use the model to predict on training data, retrain the model for the wrongly predicted images.

2.12 Improving Performance of CNN

CNN among algorithms of deep learning has been one of the most influential innovations in the field of deep learning. To improving performance certain ideas to lift the performance of CNN such as Tune Parameters, Image Data Augmentation, Deeper Network Topology And Handel Overfitting and Underfitting problem.

2.12.1 Tune Parameters

In section 2.11.1 Tune Parameters are discussed. Number of epochs definitely affect the performance. But need to do certain experimentation for deciding epochs, learning rate. Decision could make after certain epochs if there is not any reduction in training loss and improvement in training accuracy. Also dropout layer in the CNN model can be used and proper optimizer during compilation of model. Various optimizer e.g. Adam, SGD, rmsprop these things affect the performance of CNN.

2.12.2 Image Data Augmentation

CNN requires the ability to learn features automatically from the data, which is generally only possible when lots of training data is available. Image augmentation parameters that are generally used to increase the data sample count are zoom, shear, rotation, preprocessing function and so on. Definitely it will increase the accuracy of system.

2.12.3 Deeper Network Topology

Deeper networks capture the natural "hierarchy" that is present everywhere in nature. See a convent for example, it captures low level features in first layer, a little better but still low level features in the next layer and at higher layers object parts and simple structures are captured. The advantage of multiple layers is that they can learn features at various levels of abstraction. The wider network will take longer time to train. Deep networks are very computationally expensive to train. Hence, make them wide and deep enough that they work well.

2.12.4 Handel Overfitting and Underfitting problem

- Overfitting refers to a model that models the training data too well. The Example of underfiiting is your model is giving 50% accuracy on train data and 80% accuracy on test data. The meaning of it, In the overfitting your model gives very nice accuracy on trained data but very less accuracy on test data. Overfitting model is having good memorization ability but less generalization ability. This also refers model doesn't generalize well from our training data to unseen data.

- Underfitting refers to a model which works unwell on train as well as test data. Its very dangerous. The Example of overfitting is your model is giving 99% accuracy on train data and 60% accuracy on test data. Model is not fitted well on training data itself. There

are certain solutions to avoid overfitting Train with more data that is by augmenting achieve this. Then early stopping means stopping the training process before the learner passes that point and Cross validation refers Partition the original training data set into k equal subsets. Each subset is called a fold.

2.13 Literature Review

Computer vision, deep learning, and object recognition in particular, has made tremendous advances in the past few years. Deep neural networks have recently been successfully applied in many diverse domains as examples of end to end learning. Neural networks provide a mapping between an inputs such as an image of a diseased plant to an output such as a crop disease pair. Computer vision diverse technique and penetration of internet merge the way to smartphone or smart devise aid disease image diagnosis. In addition to image data security issue can be raised as data provided by users Microscope and DNA sequencing-based methods are effective to identify different kinds of diseases. Even though many of the farmers around the world do not have access to these diagnostics tools, the vast majority of them possesses a cell phone. In fact, the Ericsson Company forecasts that mobile subscriptions will reach 9.3 billion in 2019 and 5.6 billion of these smartphone subscriptions shown by Interim report. Various type of efforts have been developed to prevent crop loss due to diseases. Traditional approaches of widespread application of pesticides have in the past decade increasingly been supplemented by integrated pest management (IPM) approaches according to Ehler [9]. Before decades of time disease identification has been supported by agricultural extension organizations or institutions, like as local plant clinics. In more recent times, leveraging the increasing Internet penetration worldwide, such efforts have additionally been supported by providing information for online disease diagnosis. Even more recently, tools based on mobile phones have proliferated, taking advantage of the historically unparalleled rapid uptake of mobile phone technology in widespread of the world .Various types of images are using mobile phone, which are state by Geneva, International Telecommunication Union held on ICT Facts and Figures the World in 2015, can be utilized, and even nuances detection, classification, differences and brainy jobs [10]. Food security threatened by a number of factors

including climate change [11], the decline in pollinators [12], plant diseases [13] and others. Identification of rice diseases using deep convolutional neural networks and pooling, resulted accuracy was 95.48% by the authors Yang Lu et al [14] . The experiment evaluated with three different type of pooling methods namely max pooling, mean pooling, and stochastic pooling. Softmax regression learning algorithm can solve multi-classification problem they finally conclude that the Stochastic pooling method produces more accuracy 95.48% than others method. Limitation of these process does not answered how many layers and how many neurons are optimal at last. Process need high quality rice disease samples, faster and optimizing algorithm. Authors Jihen Amara et al [15] A Deep Learning-based Approach for Banana Leaf Diseases Banana sigatoka and Banana speckle Classification. Deep learning models were trained under certain challenging conditions, for example comprise of illumination, complex background, different images resolution, size and orientation effectively demonstrate the accuracy of this approach and the very less computational efforts required. Limitation of this paper does not confirm exact accuracy percentile and timing effort. Hughes et al [16] demonstrate the technical feasibility using a deep learning approach utilizing 54,306 images of 14 crop species with 26 diseases or healthy made openly available through the project PlantVillage, the limitation here is complex in feature. Computer vision, and object recognition in particular, has made tremendous advances in the past few years. The PASCAL VOC Challenge e. g PASCAL VOC project provides standardized image data sets for object class recognition. Also provides a common set of tools for accessing the data sets and annotations. Enables evaluation and comparison of different methods. Ran challenges evaluating performance on object class recognition from 2005-2012, now finished shown by Everingham et al [17] and more. And more recently the Large Scale Visual Recognition Challenge ILSVRC) [18] Russakovsky et al based on the ImageNet dataset have been widely used as benchmarks for numerous visualization-related problems in computer vision, including object classification. A method that identifies whether a rice leaf is healthy or infected accuracy of 99.83% is shown by Usama Mokhtar et al [19]. In their process, the input image was first pre-processed by removing the background and, the noise present was eliminated by erosion technique. Next Gray Level Co-occurrence Matrix (GLCM) was given texture feature extraction from the enhanced image. Support Vector Machine (SVM) classifier was trained using different kernel functions,

and the performance has been evaluated using N-fold cross-validation technique. Their average accuracy 99.83%, it is not high enough to predict or differentiate between healthy also validation accuracy is not simplest as like CNN. Additionally, the type of disease was not identified. Authors Patil and Khirade et al [20] used various segmentation, namely feature extraction and classification techniques that identify, classify and detect the type of diseases using the diseased image. The leaf images pre-process either smoothing or enhancing applying histogram equalization. To confirm the affected area, various segmentation techniques like K-Means clustering have been . The infected region were then extracted from the segmented region and calculated using GLCM. After feature extraction, the diseases can be detected with the help of Artificial Neural Networks (ANN) or Back Propagation Neural Networks. Drawback of segmenting the image using K-Means clustering is that the process was semi-automated as the user has to explicitly select the cluster which contains the infection. Another disease detection approach Malus Domestic by image process by Bashir and Sharma [21]. Texture segmentation used algorithm names co-occurrence matrix and color segmentation used algorithm affect the quality of a system which understands image. K-means clustering has been used to partition the leaf image into four clusters in which one or more clusters contain the disease in case when the leaf shows the symptoms that has been infected by more than one disease. Limitations of these technique is K-means is semi-automated technique causes time consuming. Multi-Task Learning for Food Identification and Analysis with Deep Convolutional Neural Networks (DCNN) by Zhang et al. [22]. To achieve model, various segmentation techniques are provided, SVM trained with DCNN by feature vector, achieves good classification results. After DCNN fully-connected layers are calculated by an activation function fact. Most of the activation functions are Rectified Linear Units (ReLU).Primary hindrances of the work is SVM is not fully automated process so this time consuming process. Phone utilize tool that helps in diagnosing crop diseases based on capturing and analyzing automatically a picture of a plant leaf is a promising solution by author Oerke [23]. CNNs are known for their robustness toward low variation in inputs, they require low preprocessing for their execution. They are also able to extract appropriate features while simultaneously performing discrimination correspondingly Deng, Arel et al [24] [25]. More specifically, authors Jihen Amara et al [26] in their paper implement to make use of the LeNet, LeCun et al [27] architecture for the convolution neural network.Apart these several LeNet efficient architecture utilized number of layer Yann LeCun

et al [28] and Rizwan [29] or not Prajwala TM et al [30] or based on fixed layer Yann LeCun et al [31]. In gradient based learning problem, the more relevant measure is the error rate of the system. The error rate is estimated by measuring accuracy. The set of samples, disjoint from training set called test set, about these theatrical and practical job accordingly has shown by Sompolinsky and Seung, Vladimir Vapnik et al, Corinna Cortes et al, [32] [33] [34]. The difference between the expected error rate on the test E_test and the error rate on train set E_train decrease with number of train sample approximately as E_test – E_train =k(h/O)pow(a). P number of train sample, h measure capacity either complexity of machine between .5 and .1 and k is constant. The expected error rate shown by Vapnik and W. H. Press et al [35] [36] . If capacity increase, lowest E_test most algorithm try to minimize E_train where the variance is known as structural risk minimization. The cutting-edge generation of CNNs) has accomplished impressive results in the field of image classification more than others technique. The use of CNN model is innovative way of training fully automated process so this is better process, the methodology used enable a quick and easy system implementation in practice. The developed model is able to recognize two different types of potato blight diseases, with the ability to distinguish plant leaves from their surroundings.

2.14 Conclusion

This chapter revised practical and theoretical concepts of disease detection and classification different method. In addition to the limitations and difficulties of classification and detection method are explained in the field of image classification. The CNN model can be deploy into smart devices, disease shadowing smart devices applications can assistance from robust computer vision models that detect or classify objects in images. Trained using deep learning, where CNNs are a class of deep learning models that are currently state of the art in many computer vision tasks, including object classification and detection. Part of this success lies in the ability of a CNN to perform feature extraction along with lebel.Apart these brief analysis of two potato disease problems are recognized. The acknowledgment lagging met when a trained plant pathologist or a farmer can diagnose disease.

CHAPTER 3

METHODOLOGY

3.1 Introduction

In deep learning, CNN is a class of deep neural networks, most commonly applied to analyzing visual imagery also known as space invariant or shift invariant artificial neural networks (SIANN).CNNs are regularized versions of multilayer perceptron's usually mean fully connected networks(FCN). In FCN each neuron in one layer is connected to all neurons in the next layer which may makes them prone to overfitting data. In this chapter apply the methodology of convolutional neural network with need prerequisite to detect potato blight disease detection. In the previous chapter shows and discuss different approach of disease detection. Among them the easiest way disease detect could be applied with appropriate method.

3.2 Research Design

In this section, the basic steps for plant disease detection and Classification using image processing are shown. The thesis design is experimental observation analyzing light changing in the classification model. Basic steps for disease detection of the thesis is draw Figure 3.1.

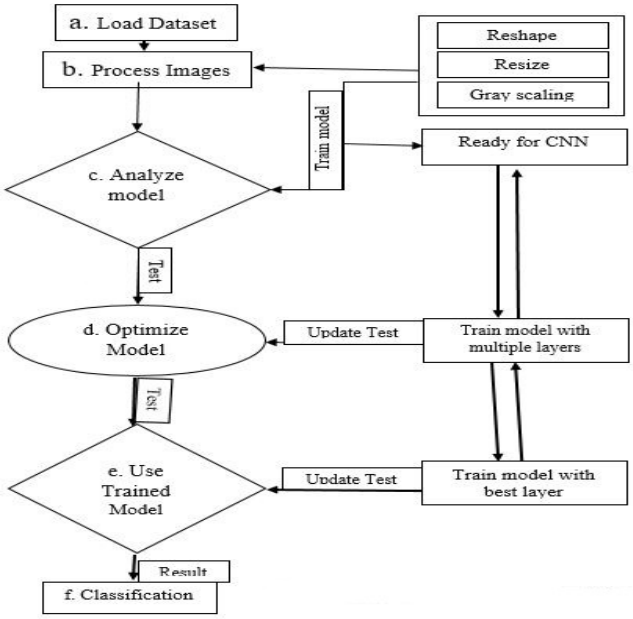


Figure 3.1: Basic steps for disease detection.

a). **Load images:** Dataset are loaded into jupyter notebook, importing uses libraries by python computer language.

b).**Process Images:** Then dataset are resize, greyscaling to run speed up. Process with segmentation, edge detection and softing.

c).**Analyze model:** To apply CNN any architecture need to process data in step a,b and off course ensure balance datasets. Then the model created are save in directory and analyze with the tensorboard for better analyzing.and save in directory.

d).**Optimizing Model:** Save model are optimized also with tensorboard and better decision making aproach are here by train model.

e). **Use Train Model:** For this need testing data by get googling or capture image. The data is train and test with save model.

f).C**lassification:** After train and test the data with created model show the disease classification.

3.3 Sampling

Symptoms of late blight in the field are small, light to dark green, circular to irregular shaped water soaked spots (Figure 1.1). These lesions usually appear first on the lower leaves. Lesions often begin to develop near the leaf tips or edges, where dew is retained the longest.During cool, moist weather, these lesions expand rapidly into large, dark brown or black lesions, often appearing greasy, effected late blight symptoms over stem and leaf shows on Figure 2. During cool, moist weather, these lesions expand rapidly into large, dark brown or black lesions, often appearing greasy.

(a) Late blight effected stem (b) Late blight effected leaves

Figure. 3.2. Effected late blight symptoms on stem and leaf

A long period of growing.one or two spot severely damaged. Underside of leaves blanketed like flour. Consequently quickly get spoiled both leaves and stems, Figure 2(b) and 2(a). Leaf lesions also frequently are surrounded by a yellow chlorotic halo.The lesions are not limited by leaf veins, and as new infections occur and existing infections coalesce, entire leaves can become blighted and killed within just a few days. The lesions also may be present on petioles and stems of the plant. For sample potato leaves are used. Effected potato crops in early blight diseases is shown in, Figure 3.

(a) Early blight effected stem

(b) Early blight effected leaves

Figure. 3.3 Effected early blight symptom on stem and leaf

Symptoms of early blight appear as small, circular or irregular, dark-brown to black spots on leaves Figure 3.3(a). Innumerable spot spread across the leaves cause to dry out Figure 3.3(b).

Symptoms of early blight appear as small, circular or irregular, dark-brown to black spots on the older leaves. These spots enlarge up to 3/8 inch in diameter and gradually may become angular-shaped. Disease image leaves are collected from the district of Noakhali Sonapur -314, Sonaimuri- 3827 belongs two villages Papua and Shimulia and Dhaka Banasree in potato yard and garden. Approximately three thousand image are collected 2,100 images are collected, for train process use 1960 data, rest of data are separated for test purpose. Affected potato of early blight and late blight potatoes are twenty.

(a) Late blight

(b) Early blight

Figure 3.4: Affected potatoes early and late blight.

Potato blight disease crops are collected from the district of Noakhali Sonapur -3814, and Sonaimuri- 3827 survey three different potato garden. Acquisition are manually place by single potato and capture with smart devices. Device having default measurements and resolutions are and have change device to device. For capturing digital devices are Iphone6, Alcatel, Xiomi both Android and ios platform. Effected potato crops collection are fifty in number. To feed the world agricultural crops must be cooprated through production and should be protected from from disease for giving assurance of high rate of production.

3.4 Instrument

a) Digital camera or Smart Phone: This is need for diseased image capture, HD resolution provide images details clearly, will use for train and test.
b) Anaconda software: Images are prepared as input, required input loaded from jupyter notebook web application and google colaboratory.
c) Bulkresizephotos, RIOT, MATLAB software or site: To Image resize and further effect python coding and these software needed.
d) WPS server: Data uploaded into an online data storage and make directory which ensure security and prepare each disease images as a directory or class.

e) Python language and libraries: Train images as data required programming and development has done with API of python programming language, Tensorflow in backend and package along high level API of Keras.

f) GPU, Tensor board, Computer and smart phone: Experiment and Accuracy has tested with Dell Inspiron 3567 7th Gen Core i3 Laptop with Genuine Win 10 and keras API in data storage and cloud storage with default drive.

g) Online site and smart devices: Classification result and detect suggestion has modified with online site https://playground.tensorflow.org and Jupyter Notebook, google colab. These process ensure disease classification using conducted model.

3.5 Steps for Method

The scheme includes these important steps namely Data Acquisition, Data pre-processing and Classification etc. In data acquisition contain steps are Raw Leaves collection, process image, and store image for next step. The methodology full process here,

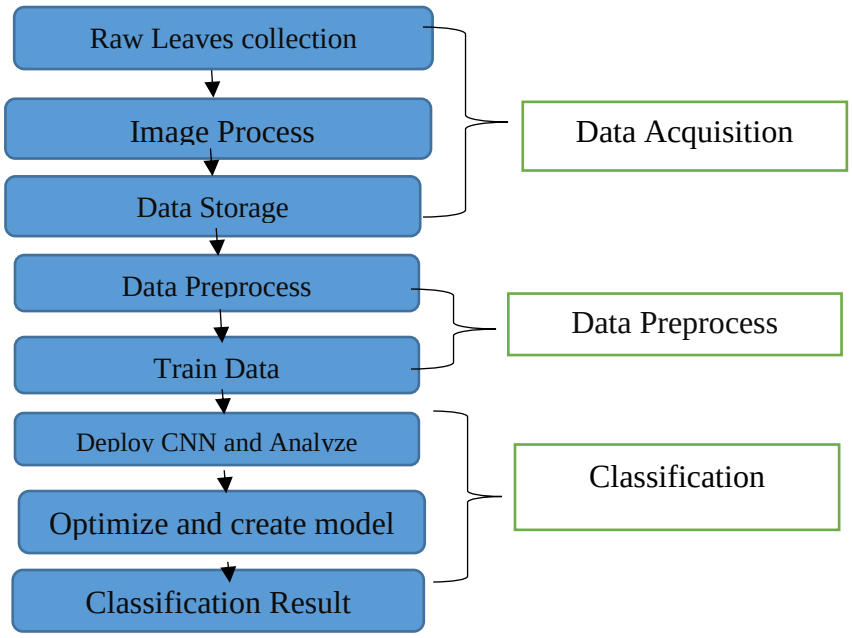


Figure 3.5: Proposed Methodology

3.5.1 Raw Leaves collection

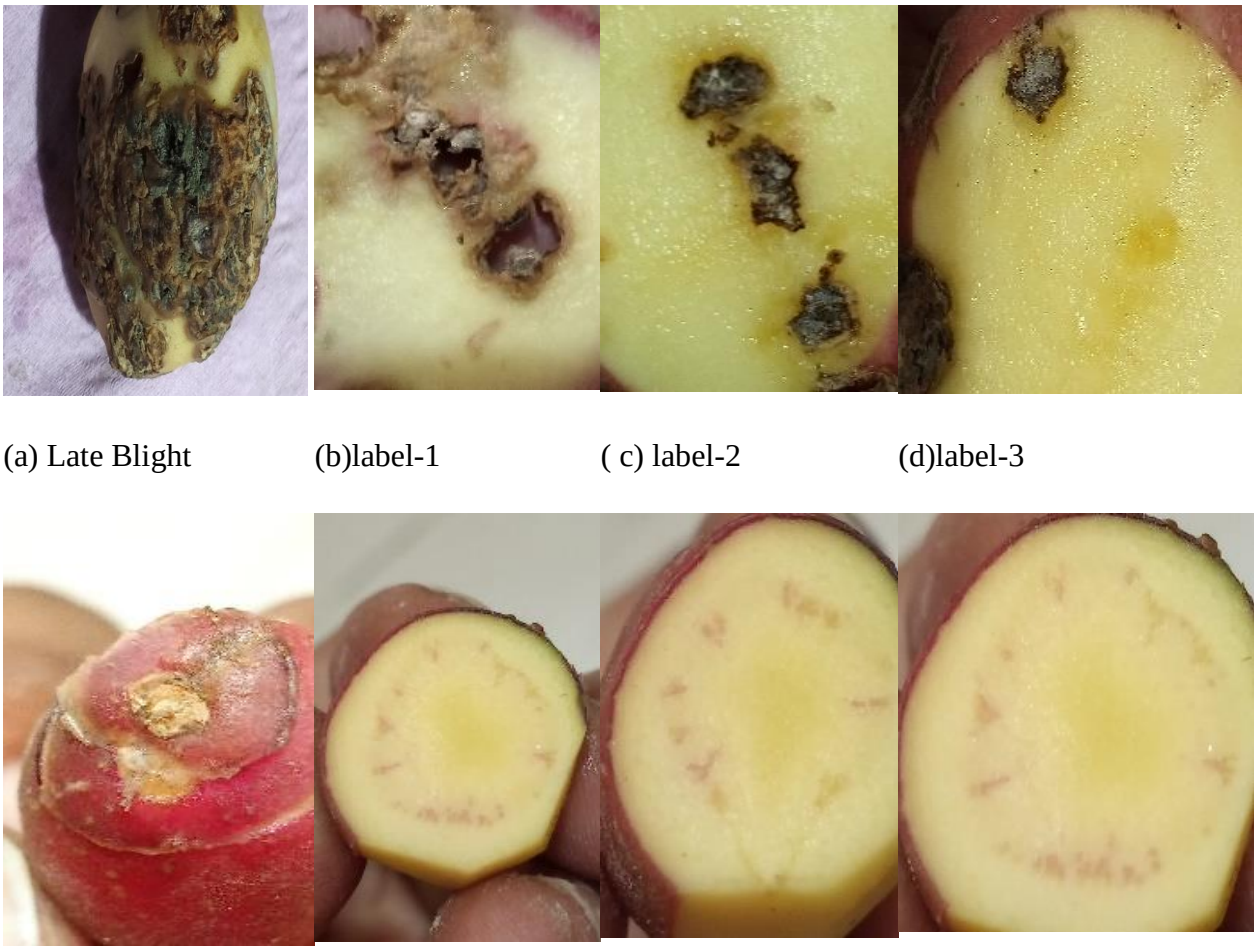
Collected or survey leaves are collected from different yard garden etc. Figure 3.6 shows the image of agriculture field survey with different locations where raw leaves are collected.

Figure 3.6: Field survey of potato leaves.

Potato blight disease leaves or raw leaves are collected from the district of Noakhali Sonapur - 3814, Sonaimuri- 3827 belongs two villages Papua and Shimulia containing three different potato garden. Also Dhaka Banasree in potato yard and garden. Acquisition are manually place by single leaves and capture with smart devices. Device having default dimensions and resolutions are and have variance device to device. Devices are Iphone6, Alcatel, Xiomi Android ios, and digital devices. Around 2,100 leaves are store from collected images. 980+980 image are prepared for process, 1960 rest of data are separated for test purpose. All the collected data firstly need to process for being same feature.

3.5. 2 Image Process

As previous section disease images are separately organized and exquisite for further process due to same capacity, feature, channel, labeling corresponding devices etc. Figure 8 shows potato early and late blight affected potato image and deeper severity label corresponding disease.



(a) Late Blight (b)label-1 (c) label-2 (d)label-3

(e) Early Blight (f) label-1 (g) label-2 (h) label-3

Figure 3.7 Affected Late Blight and Early Blight potato with deeper severity.

Late blight disease potato rotten prior dry out Figure 8(b) deeper slice, more deeper cut in Figure 8(c), mostly deeper cut on Figure 8(d) where early blight potato spot circular get larger by splicing Figure 8 (f), deeper slice, more deeper cut Figure 8(g), mostly deeper cut on Figure8(h).Late blight and early blight severity visually leaves and affected on potatoes by slicing shows on Figure 3.7, also indicate second row early blight symptoms and affected level in potatoes, early blight problem occur as circular lesion and quickly become severe if any action not taken. Early blight circular spotted in potato with potato leaves lesion growth, these symptoms shows early growth time of potato plant. Late blight disease occur late age of plant and dry out affected leave spot besides potato get rotten. Also affected leaves stem and below surface of leaves black spot with whites appeared. A deep convolutional neural network is trained to identify 14 crop species and 26 diseases with accuracy 99.35% [37]. In this work binary classification used to predict potato disease into two category. Disease leaves images have been consists of approximate around three thousand images belonging to two different class's early blight and late blight in storage. The dataset includes images of all major kinds of leaf diseases that could affect the potato crop. By default each of the images belongs to the RGB color space and are JPG format configuration of each capture digital device. CNN does work with same capacity, size, feature, color and capacity. With the help of online web application bulk resize and windows default editor edited images to same capacity, size 148 x 148 and make sure are same image format. All the images needed to be noise free and for data preprocess for saving in the cloud storage as sample.

3.5. 3 Data Storage

In the cloud, sample are stored separately in two class for two purpose. One reason for train data with one class and other reason test model with the other class sample. Data are stored in cloud storage for the further processing and cloud GPU support. Require as convolutional neural network maintain same capacity, size, shape so data stored classes must be same in capacity that is for train data early blight 980 image and late blight 980. Rest of sample are belongs to test class. To make them same as primarily dimension prior implemented data. All the images are color of three channel.

3.5.4 Data preprocess

The work dataset consisted of digital images with minimal dimension such dimension stands for height, width and depth correspondingly. The modern digital image and signal process are discussed by author Lim [38] .preprocess with coding has plotted with the first image in dataset and check its size using the shape function by import matplotlib library. The images in the dataset had resized to 148 × 148 resolution in Image Process section, convert them into image array. Read Image as one channel format. Further speed up the training process to make model training computationally manageable. Resize array matrix steps.

- Resize Image Array (Computation)

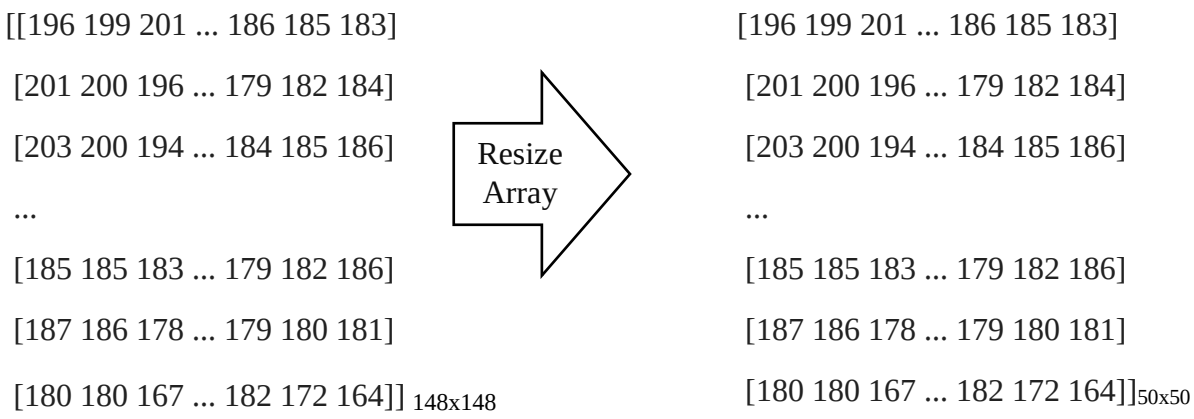


Figure 3.8**:** Resize Image Array

3.5. 5 Train Data

In this steps data are manually classes, so now in implemented with python, need to train data. To create training data and all that, but, first, set aside some images for

final testing which already done in section Data Storage, Just manually create a directory called Testing and then create 2 directories inside of there, one for Early Blight and one for Late Blight. Make sure all the images are not copy of other class. Testing class images are use t for final tests. Now, want to begin building training data. Give index to run data as directory list of categorical or indexing. For each class contains nineteen hundred eighty images. For the first train of dataset took high speed internet connection provided GPU support.

Figure 3.9: train data time speed

Following Figure 3.9 shows that almost 1960 samples are trained between balanced data as CNN prerequisite. In the case of this dataset, the dataset started off as being balanced, mean there are the same number of samples for each class such as same number of early blight and late blight. If not balance, the model initially learn that the best thing to do is predict only one class, and get stuck there. In this case, this data is already balanced, so that's easy enough. Also, if the dataset that is too large to fit into ram, need to batch-load in data. There are many ways to do this, some outside of Tensorflow and some built in. Certainly data for now is balanced, in this section mainly trying to cover how data should look, be shaped, and fed into the models. Need shuffle the data because data is just all early blight, then all late blight. This will usually wind up causing trouble too, as, initially, the classifier learn to just predict early blight always. So need to import random library to shuffle or mix data nicely [39]. By iterating over a few of the initial samples and printing out the class can confirm mix data. Build model and save this data, so that don't need to keep calculating it every time. Got dataset, ready to cover convolutional neural networks and implement one with prepared data for classification.

3.5. 6 Deploy CNN and Analyze model

The Convolutional Neural Network gained huge popularity through its use with image data, and is currently the state of the art for detecting what an image is [40] [41] and classification of large classes [42]. The basic CNN structure [43] is as follows: Convolution, Pooling, Convolution, Pooling, Fully Connected Layer, Output Convolution is taking the original data, and creating feature maps from it. Pooling is down-sampling, most often in the form of "max-pooling," where select a region, and then take the maximum value in that region, and that becomes the new value for the entire region. Fully Connected Layers are typical neural networks, where all nodes are "fully connected." The convolutional layers are not fully connected like a traditional neural network. Each convolution and pooling step is a hidden layer. After this, we have a fully connected layer, followed by the output layer. The fully connected layer is typical neural network (multilayer perceptron type of layer, and same with the output layer.

- Model Prerequisite

Prior convolutional neural network there are some prerequisite. First convert this dataset to training data. As a few issues right out of the gate. The largest issue is not all of these images are the same size. While eventually have variable sized layers in neural networks, this is not the most basic thing to achieve. Want to reshape things, so every image has the same dimensions. Next, may or may not want to keep color. After training data need to randomize for clarification. As provided indexing this shown as 0 or 1 as the validation result. For feature say xx and label say yy of each image need to reshape data.

Reshape Image array:

Data = image array * (-1, 50, 50, 1)

Or

Data = image array * (-1, 70, 70, 1)

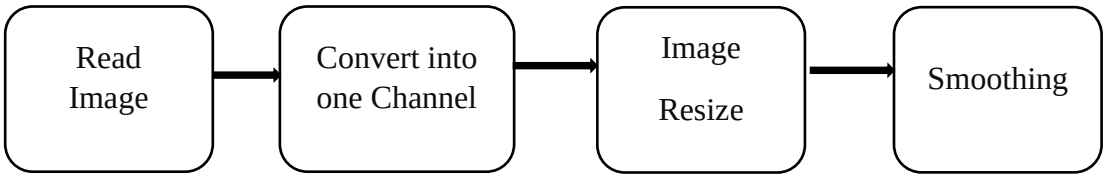


Figure 3.11: Image pre-process Implement-View.

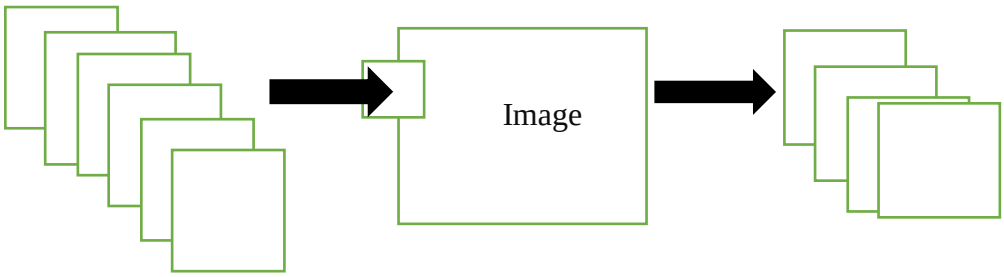
Need to reshape dataset inputs to the shape that model expects when train the model. The last number is 1, which signifies that the images are greyscale. The process of input variables tends to speed up the training process through improvement of the numerical condition of the optimization problem. Several default values involved in initialization and termination are appropriate. Image segmentation [44] providing a label to every pixel in an image so pixels with the same label share certain features. In consequence's region of interest (ROIs) find out the doubtful mass candidates from the ROI by using segmentation [45].But approach get feature and label by reshaping. Rather their feature and label are normalized with per image each pixel divided 255 as the one channel image pixel value range 0 to 255. The result of each pixel will get 0 or 1 to speed the computation more efficiently [24] [25] both of these feature extraction need more computation but current makes more simply.

- Model Analyze

Computers see images using pixels. Pixels in images are usually related. Normalization applied with image array for single value zero or one. Sequential model are utilized here with Adam optimizer along CNN [46] [47]. Convolutional two layer are use, each layer Conv filter 256, MaxPooling use 2x2, advance activation ReLU are used. Convolutions use this to help identify images. A convolution multiplies a matrix of pixels with a kernel that is filter matrix and sums up the multiplication values shows on Figure 3.12. The convolution slides over to the next pixel and repeats the same process until all the image

pixels have been covered. Assume each square is a pixel. That window's features are just a single pixel-sized feature in a new feature map, but will have multiple layers of feature maps in reality. Next slide that window over and continue the process. There were some overlap, can determine how much wanted, just do not want to be skipping any pixels, of course .Now continue this process until have covered the entire image, and then will have a feature map. Typically the feature map is just more pixel values, just a very simplified one.

- Conv2D Filters:



K kernel applied Each Kernel conv with image Feature Map

With input image

Figure 3.11: Conv2D parameter filters

Conv2D determines the number of kernels to convolve with the leaves image and produces a 2D activation map or feature map each time when scan process are 2x2. Scan over the input image 2x2 and iterate it left to right till scan whole pixel of the image. Each scan of the process full out the max pixel value, and do all the other scan time on the image. After that all the max pooling value create feature map corresponding image. Each feature map again convolve with the next image of the dataset and again max pooling to produce better feature map from first layer. This process is consequence process tile multilayer perceptron and get classified. Apart max pooling there are another types of

pooling layer average pooling normal pooling but the best one for convolutional network is ax pooling.

- MaxPooling:

Kernel size 2 by 2 and image size 3 by 3 scan result shows on Figure 3.12. Following figure data pixels are scan with kernel size and select highest pixel by given kernel. Kernel size can be variance either 1x1, 2x2, 3x3, 4x4, 5x5. Here image pixel 3x3, kernel size 2, 2 produce MaxPooling result 2, 2.

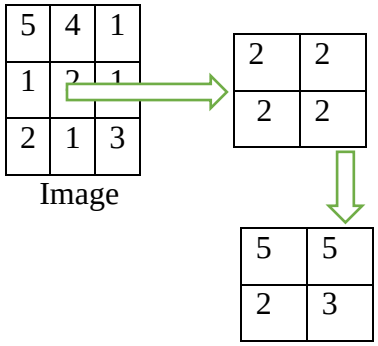


Figure 3.12: MaxPooling

I. ReLU Activations can be used through an Activation layer. This returns neuron fire or not any feature. Activation is the activation function for the layer. The activation function for first 2 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks.

II. Dropout: due to disabling neurons, some of information about each sample is get loss. And the subsequent layers attempt to generate the data result basing on incomplete representations.

The exponential linear activation:

X if x > 0 and alpha * (exp (x)-1) if x < 0.

This is a layer of CNN then another one used for this CNN deployment. After that 3D vector need to Flatten to convert 1D vector.

III. **Dense Layer:** Dense means input parameters, with neurons in the FIRST hidden layer.

IV. **Compilation** the model compiling the model takes three parameters: optimizer, loss and metrics [46].

Prior train the model, learning process need to configure via the compile method [48]. It receives three arguments:

An Optimizer. The optimizer controls the learning rate. We have used Adam as optimizer in this work. Adam is generally a good optimizer to use for many cases. The Adam optimizer adjusts the learning rate throughout training.

A Loss function. This is the objective by which model will try to minimize. It can be the string identifier of an existing loss function such as categorical crossentropy or other. This is the most common choice for classification. A lower score indicates that the model is performing better. The learning rate determines how fast the optimal weights for the model are calculated.

A list of metrics. For any classification problem need to set this to metrics= ['accuracy']. To make things even easier to interpret, use the 'accuracy' metric to see the accuracy score on the validation set when train the model.

3.5.7 Optimize and Build Model

According to authors Jihen Amara et al [15] a deep learning-based approach on CNN under certain challenging conditions, Author Hughes et al [16] demonstrate the technical feasibility using a deep learning approach, Authors Yang Lu et al [14] trained a model for rice disease identification, they are not analyze model with Tensorboard callback and compare model accuracy layer by layer. Tensorboard is a handy application

that enable us to view aspects of model, or models, in browser. Here, to optimize model check with conv layer or nodes sizes 32, 64, 128 etc. with filter window 3x3. So which conv layer accurate check with 64 here.

- Optimize model

In this work to finalize the model we optimized model by utilize Tensorboard into notebook and run with three different nodes 32, 64,128 sizes, this running process need background GPU support in colab, 3 different conv layer namely 1,2,3 and 3 different dense layer 0,1,2. Layers size are sequential with power of 2. Loop in 27 node-conv-dense layer build for train and validate model. During keras implement model compile method pass Adam optimizer and binary cross entropy. According to Kingma et al. [49], the method is computationally efficient, has little memory requirement, invariant to diagonal rescaling of gradients, and is well suited for problems that are large in terms of data or parameters. Update with CNN model with smaller convolutional filter, 32, 64 then 128 sizes correspondingly. Activation layer of ReLU and MaxPooling with window size 2 by 2 shows on Figure 3.13.

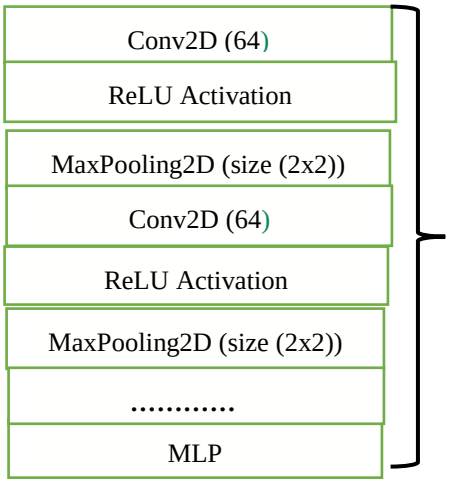


Figure 3.13: A small design of CNN.

- Build Model

After compile and show result on Tensorboard trigger 10 best layers depending on model accuracy. Among them run three each layer to find better accuracy shows on Table 3.1.

```
1-conv-64-nodes-0-dense-1583741819;
2-conv-256-nodes-0-dense-1583741819;
3-conv-64-nodes-0-dense-1583741819;
```

conv	nodes	dense
1	64	0
2	256	0
3	64	0

Table 3.1: Three best layers table.

In this step model with the best layer 1-conv-64-nodes-0-dense is gives accuracy 98% and validation accuracy 94%. Now this time model is created for classification.

3.5.8 Classification

CNN can be used for the creation of a computational model that works on the unstructured image inputs and converts them to corresponding classification output labels. For the purpose of potato leaf disease detection, we have experimented with several standard deep learning architectures like AlexNet [50], GoogleNet method [48] and the best results could be seen with the use of a variation of the LeNet architecture [51]. LeNet is a simple CNN model that consists of convolutional, activation, pooling and fully connected layers. The architecture used for the classification of the potato leaf diseases is a variation of the LeNet model [52]. Here CNN use sequential model Adam optimizer with default params and values. With the increase in depth, the complexity of the extracted

features increases. The size of the filter is fixed to 5 × 5 and MaxPooling is 2 × 2 and two hidden layer is applied analyze with params are shows on Table 3.2.

Name	Keyword Params
Adam Optimizer	Optimization algorithm, Learning Rate, Momentum, Weight Decay, beta_1, beta_2, amsgrad.
Model Fit	Batch Size, Epoch iteration
CNN layer	Conv layer, Activation function, Pooling
Compile	binary_crossentropy, Adam, metrics.
Epochs	50

.

Table 3.2: Model Parameter table

Adam is a replacement optimization algorithm training deep learning models and combines the best properties of the AdaGrad and SProp algorithms. This optimizer can handle sparse gradients on noisy problems and relatively easy to configure. Here the default configuration parameters do well on most problems including this work. CNN block used conv2D filter 64 then 256, kernel size 3by 3 and input shape same as input image. For model fit function parameter are fed namely Optimization algorithm, Learning Rate, Momentum, Weight Decay, beta_1, beta_2, amsgrad. Batch size 32 and epoch iterated for fifty times. Batch size of 20 has been used and the model has been trained for 50 epochs. The initial learning rate has been set to 0.01 and it is reduced by a factor of 0.3 on plateau where the loss stops decreasing. Compile function that is compiling the model takes three parameters: optimizer, loss and metrics. Here used params are binary_crossentropy as the two classification in the output, Adam, metrics. Adam optimizer adjusts the learning rate throughout training,

3.6 Conclusion

This chapter brush up practical models of disease detection and classification. In addition to the troubles of classification and detection are elucidated. Brief analysis of two potato disease problems are recognized. Part of this implementation lies in the ability of a CNN to perform feature extraction, tuning parameter optimize the model.The feature maps are also same size in order to domain the size of the image after the use of the convolution operation. The max pooling layer 2×2, perform the feature maps, speeding up the training process, and making the model less variant. Advance ReLU activation layer is used in each of the blocks for the introduction of non-linearity to minor changes in input. With the help of compile, model fit and other usage method hyper parameter are altered, epochs also altered like ten epochs, twenty epochs, fifty epochs eighty epochs hundred epochs etc. Apart these run result fifty epochs with current system shows the best result. For overfitting and Underfitting problem special focus pay heed on the parameter selection timing and data process step.

CHAPTER 4

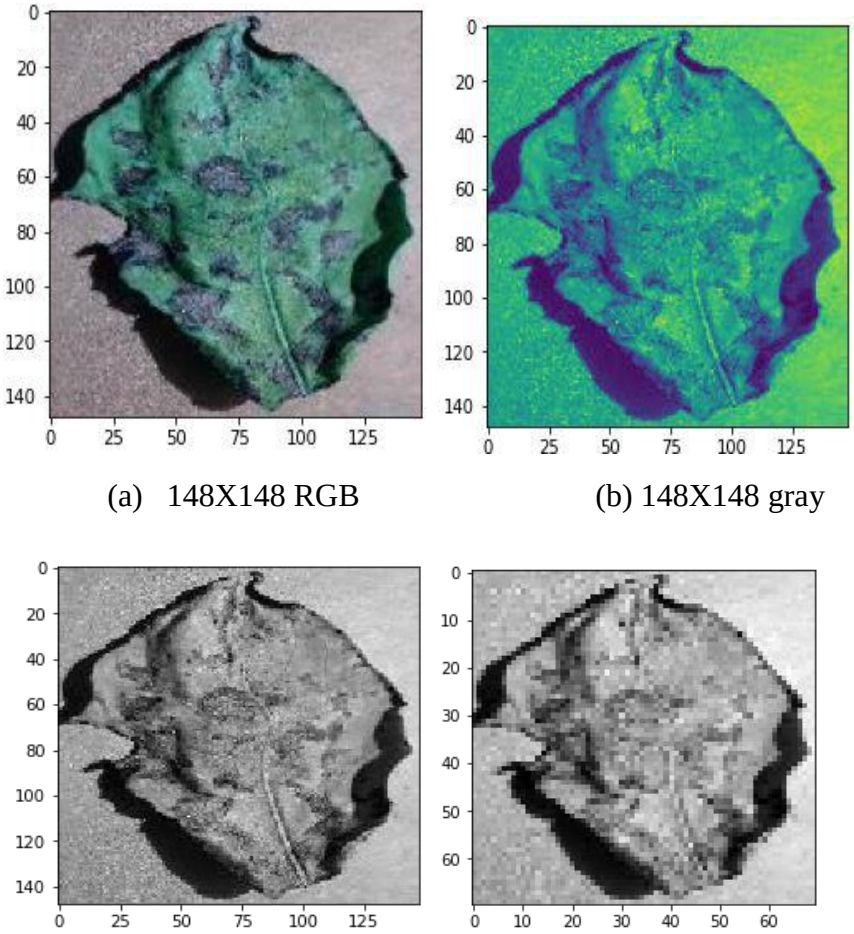
RESULT AND DISCUSSION

4.1 Introduction

This chapter shows result which are applied in the methodology chapter all the steps result and details about the output. The work main steps are pre-process, train model, analyze model, optimize model and then classify with distinct ample. In the last of this chapter could be compare with the methodology according result.

4.2 Pre-process Output

For resize check with the output either with 50x50 or 60x60 or 70x70.Prepared sample check an image with shape outputted as (148, 148, 3) where height 148, weight 148 and color channel is 3 because it's RGB. RGB image take high speed running process but there is large data set which are not be able to run at easy moment and simple hardware. The larger dataset make sure more accuracy of convolutional neural network but here the dataset in rgb run crash on system. So implement image into rgb to gray scale as gray scale image are fast to run in the system. When converted into one channel that time also need to change with various shape to detect run time and speed, definitely don't want the images that big, but also various images are different shapes. Check with 100 x 100 would better but 70 x 70 is much better for the dataset. Single channel image refers to gray scale image in this implementation with the help python shown the two version of gray scaling. One for the converted version rgb to gray scaling and the other output show as second tie gray. Resize Image Array in graphical show on Figure 4.1:



(a) 148X148 RGB (b) 148X148 gray

(c) 70X70 gray (d) 50X50 gray

Figure 4.1: Image channel modification.

Figure 4.1 shows top left 4.1(a) is 148x148 RGB that is 3 channel, (b) is 148x148 with color map gray, (c) is 70x70 with color map gray and read grayscale on one channel, (d) 50x50 with color map gray and read grayscale in one channel.

4.3 Train Data Output

In this steps data with early blight images class train speed 1.86IT/S the other class late blight image train speed 2.66IT/S. GPU alter memory to accelerate the creation of images in a frame buffer intended for output to a display device here display device laptop.

Data speed during train data with the help of GPU support on coleb notebook shows on Figure 4.2.

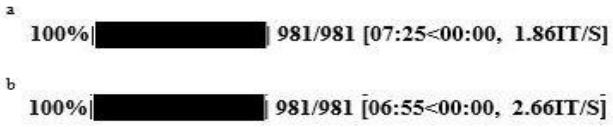


Figure 4.2: Train data with speed.

Without GPU compile these image is so time consuming and lengthy process. Modern GPUs are very efficient at manipulating computer graphics and image processing. In a personal computer, a GPU can be present on a video card or embedded on the motherboard.

4.4 First Deploy CNN

MaxPooling and ReLU activation are mutually represent a hidden layer. Uses CNN compile with two of such hidden layer. After just three epochs, 78% validation accuracy. If keep going, probably do even better, but should probably discuss how to improve validation accuracy. To help with this, use Tensorboard, which comes with Tensorflow and it helps to visualize models as they are trained. CNN model is mention on Figure 4.3.

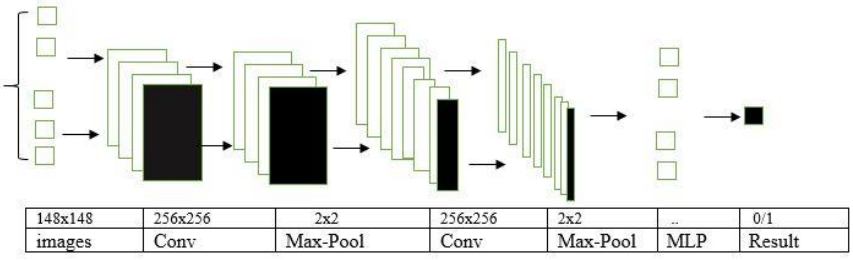


Figure 4.3: Deployed CNN overview

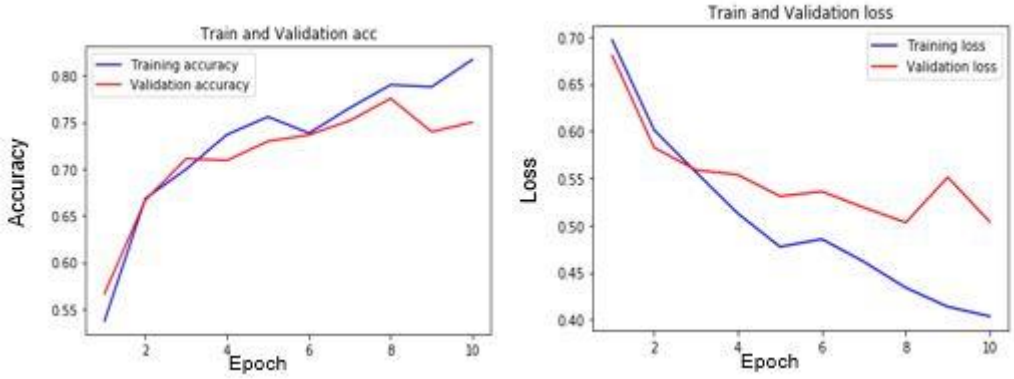
Conv2D parameter is the numbers of filters that convolutional layers will learn from. It is an integer value and also determines the number of output filters in the convolution. Figure 4.3 learning a total of 256 filters and then use 2 by 2 Max Pooling to reduce the spatial dimensions of the output volume with two hidden layer. ReLU activation function with dense layer prior use flatten function to convert one dimension. Softmax dimension into one dimension format. For three epochs first Running CNN model shows on Figure 4.4.

```
Model: "sequential_37"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_74 (Conv2D)           (None, 48, 48, 128)       1280
_________________________________________________________________
activation_111 (Activation)  (None, 48, 48, 128)       0
_________________________________________________________________
max_pooling2d_74 (MaxPooling (None, 24, 24, 128)       0
_________________________________________________________________
conv2d_75 (Conv2D)           (None, 22, 22, 256)       295168
_________________________________________________________________
activation_112 (Activation)  (None, 22, 22, 256)       0
_________________________________________________________________
max_pooling2d_75 (MaxPooling (None, 11, 11, 256)       0
_________________________________________________________________
flatten_37 (Flatten)         (None, 30976)             0
_________________________________________________________________
dense_74 (Dense)             (None, 64)                1982528
_________________________________________________________________
dense_75 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_113 (Activation)  (None, 1)                 0
=================================================================
Total params: 2,279,041
Trainable params: 2,279,041
Non-trainable params: 0
_________________________________________________________________
Train on 1372 samples, validate on 588 samples
Epoch 1/3
1372/1372 - 1s - loss: 0.7080 - accuracy: 0.5751 - val_loss: 0.6352 - val_accuracy: 0.6207
Epoch 2/3
1372/1372 - 1s - loss: 0.5180 - accuracy: 0.7194 - val_loss: 0.5569 - val_accuracy: 0.7109
Epoch 3/3
1372/1372 - 1s - loss: 0.4683 - accuracy: 0.7522 - val_loss: 0.4869 - val_accuracy: 0.7857
```

Figure 4.4: First CNN Compilation.

From the Figure 4.4 run output accuracy is 78% accuracy as this found rerun the CNN deployment and Train on 1372 samples, validate on 588 samples. Separate the data 30% for validation and 70% for accuracy testing. So need further analyze. So the model does not create here to create the final model. Training and testing purpose splitting result are plot on graph. Training testing are separated classes and their result plotted on graph and analyze on Tensorboard provided background GPU support. Each plotted graph contains

features are accuracy, loss, validation accuracy and validation loss. Figure 4.4 provide the both train and test loss and accuracy accordingly epochs.

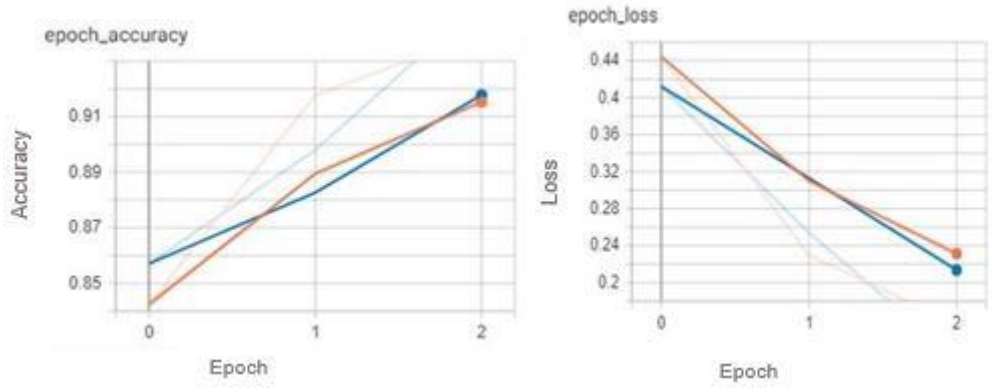


(a) 70% Accuracy and Validation of 30% (b) 70% Loss accuracy and Validation 30%.

Figure 4.5: Data Accuracy and Loss graph

Figure 4.5 (a) horizonat axis for epoch and vertical for accuracy in percent unit. Color as blue for training accuracy where data is 70% and red for validation accuracy where data amount is 30%. graph 4.5 (a) shows validation accuracy get almost accurate with training accuracy that means training accuracy above 80% after three epochs. Graph 4.5 (b) horizonat axis for epoch and vertical for Loss in percent unit. Training loss blue and red for validation loss which shows 75%. The more minimize loss the data get validate.Training loss is 40% and validation loss nearly 50% after three epochs. To increase validation accuracy, Tensorboard can help with this, which comes with Tensorflow and visualize models as they trained. From figure 4.3 first model run output is 78% accuracy, CNN deployment and Train on 1372 samples, validate on 588 samples. Separated the data 30% for validation and 70% for accuracy testing. So need further analyze with Tensorboard. Tensorboard can be used directly within notebook experiences such as Colab and Jupyter. This can be helpful for sharing results, integrating Tensorboard into existing workflows, and using Tensorboard without installing anything locally.

In this research load first deployed CNN model in Tensorboard with a few command. Both of train and test loss and accuracy accordingly epochs are shows on Figure 4.6.



(a) Accurcy Graph (b)Loss graph

Figure 4.6: Epoch Accuracy and Loss with the Tensor Board.

Dashboards use for graphs feature figure 4.6(a) show horizonat axis for epoch and vertical for accuracy in percent unit. The graph shows blue progress line for model training accuracy 93% and red for validation accuracy 92%. 4.6(b) show horizonat axis for epoch and vertical for loss in percent unit. Graph shows blue progress line for model training loss 0.2 and red for validation loss slight above .24. The same Tensorboard backend is reused by issuing command, it monitor model in progress. If a different logs directory was chosen, a new instance of Tensorboard would be opened. A logs class is create in the cloud storage to load and run the model with Tensorboard. Ports are managed automatically. Start training a new model and watch Tensorboard update automatically every 30 seconds or refresh it with the button in the Tensorboard.

4.5 Analyze and Optimize model

Tensorboard is a handy application that allows to view aspects of the model. For certain challenging conditions Jihen Amara et al [15] worked a deep learning-based approach on CNN, Hughes et al [16] demonstrate the technical feasibility using a deep learning approach, yang Lu et al [14] trained a model for rice disease identification, where in this work, the model accuracy hierarchy analyzed with Tensorboard callback. The way

that use Tensorboard with Keras is via a Keras callback. There are actually quite a few Keras callbacks. Model Checkpoint is another useful one. For this work focused on the Tensorboard callback. Callbacks is a list and definitely check the others and could pass other callbacks into this list as well and enable visualizations for Tensorboard. Inherits from Callback check out Keras Callbacks [53].

4.5.1 Analyze with different size model

```
Model: "sequential_46"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_92 (Conv2D)           (None, 48, 48, 64)        640
_________________________________________________________________
activation_146 (Activation)  (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d_92 (MaxPooling (None, 24, 24, 64)        0
_________________________________________________________________
conv2d_93 (Conv2D)           (None, 22, 22, 64)        36928
_________________________________________________________________
activation_147 (Activation)  (None, 22, 22, 64)        0
_________________________________________________________________
max_pooling2d_93 (MaxPooling (None, 11, 11, 64)        0
_________________________________________________________________
flatten_46 (Flatten)         (None, 7744)              0
_________________________________________________________________
dense_92 (Dense)             (None, 64)                495680
_________________________________________________________________
activation_148 (Activation)  (None, 64)                0
_________________________________________________________________
dense_93 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_149 (Activation)  (None, 1)                 0
=================================================================
```

Figure 4.7: A small model implementation summery.

For the model type that used is Sequential. This is the easiest way to build a model in Keras. Sequential allows to build a model layer by layer dot add '.add ()' function to add layers to the small model. First 2 layers are Conv2D layers. These are convolution layers that will deal with input images, which are seen as two dimensional matrices. 64 in the first layer and 64 in the second layer are the number of nodes in each layer. This number can be adjusted to be higher or lower, depending on the size of the dataset. In this case, 64 work well. Kernel size is the size of the filter matrix for our convolution. So a kernel size of 3 means a 3x3 filter matrix. Activation is the activation function for the layer. The activation function being used for first 2 layers, is the ReLU, or Rectified Linear

Activation. Since this activation function has been proven to work well in neural networks. In between the Conv2D layers and the dense layer, there is a ‘Flatten’ layer. Flatten serves as a connection between the convolution and dense layers. Dense is the layer type we will use in for our output layer. Dense is a standard layer type that is used in many cases for neural networks. Figure 4.8 shows a small model with val_accuracy 75%.

```
Total params: 533,313
Trainable params: 533,313
Non-trainable params: 0
_________________________________________________________________
Train on 1372 samples, validate on 588 samples
Epoch 1/10
1372/1372 - 1s - loss: 0.6964 - accuracy: 0.5372 - val_loss: 0.6800 - val_accuracy: 0.5663
Epoch 2/10
1372/1372 - 0s - loss: 0.6011 - accuracy: 0.6684 - val_loss: 0.5824 - val_accuracy: 0.6667
Epoch 3/10
1372/1372 - 0s - loss: 0.5563 - accuracy: 0.6997 - val_loss: 0.5587 - val_accuracy: 0.7109
Epoch 4/10
1372/1372 - 0s - loss: 0.5125 - accuracy: 0.7369 - val_loss: 0.5536 - val_accuracy: 0.7092
Epoch 5/10
1372/1372 - 0s - loss: 0.4773 - accuracy: 0.7558 - val_loss: 0.5309 - val_accuracy: 0.7296
Epoch 6/10
1372/1372 - 0s - loss: 0.4855 - accuracy: 0.7383 - val_loss: 0.5359 - val_accuracy: 0.7364
Epoch 7/10
1372/1372 - 0s - loss: 0.4615 - accuracy: 0.7653 - val_loss: 0.5189 - val_accuracy: 0.7517
Epoch 8/10
1372/1372 - 0s - loss: 0.4339 - accuracy: 0.7901 - val_loss: 0.5031 - val_accuracy: 0.7755
Epoch 9/10
1372/1372 - 0s - loss: 0.4138 - accuracy: 0.7879 - val_loss: 0.5510 - val_accuracy: 0.7398
Epoch 10/10
1372/1372 - 0s - loss: 0.4037 - accuracy: 0.8171 - val_loss: 0.5039 - val_accuracy: 0.7500
```

Figure 4.8: Small model with val_accuracy

Figure 4.8 shows data in Validation Split as 70% and 30%. The small implementation result shows that loss 40% which less than test loss 50%, train accuracy is 81% which is greater than validation accuracy 75%. Total parameter is 533,313, Trainable params 5333,313 and non-triable params 0. As validation accuracy is 75% that means should increase conv layer from 64.

4.5.2 Full model analyze

Exemplify might build a workflow to optimize model's architecture. To begin, A few things could do to the model figure 4.5. A full model update shows on figure 4.9.

Layer (type)	Output Shape	Param #
conv2d_173 (Conv2D)	(None, 48, 48, 128)	1280
activation_300 (Activation)	(None, 48, 48, 128)	0
max_pooling2d_173 (MaxPoolin	(None, 24, 24, 128)	0
conv2d_174 (Conv2D)	(None, 22, 22, 256)	295168
activation_301 (Activation)	(None, 22, 22, 256)	0
max_pooling2d_174 (MaxPoolin	(None, 11, 11, 256)	0
flatten_87 (Flatten)	(None, 30976)	0
dense_165 (Dense)	(None, 64)	1982528
activation_302 (Activation)	(None, 64)	0
dense_166 (Dense)	(None, 1)	65
activation_303 (Activation)	(None, 1)	0

Figure 4.9: Full update model implementation summery.

The most basic things to modify are layers and nodes per layer, such as 0, 1, or 2 dense layers. Even parameters will take some significant time like varying layer sizes, activation functions, learning rates, dropouts, and much more. In previous compilation 64 node-per-layer is working for 75% validation accuracy. So more optimize compile 128 and a 256, along with 64, list is [32, 64, 128] is try 64. 128 shows better results, so then tried [64, 128, 256] and so on. Change in certain parameters, may need to revisit older ones, so ensure that is add or just increase dropout, likely have a larger overall model in terms of layers or nodes per layer than before. A larger model overall might want a larger starting

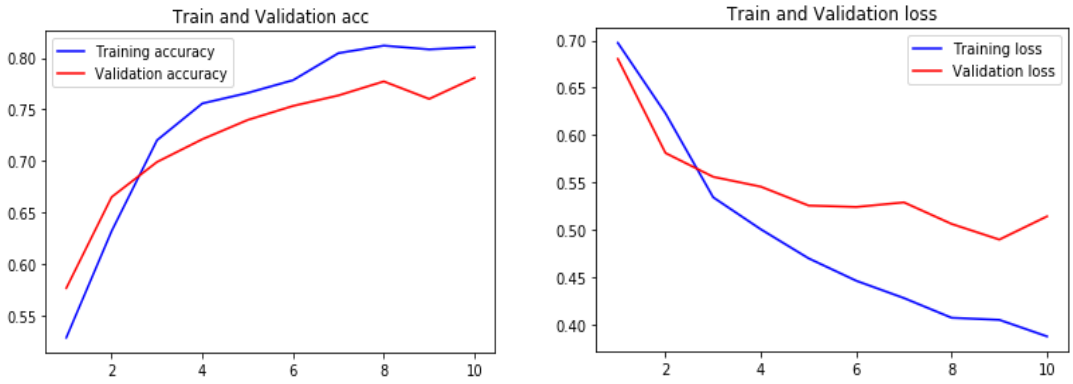
learning rate or a slower rate of decay for the learning rate. Small and few of run increasing or decreasing conv layer get this output. Compilation result of the full update model compilation of 70% and 30% loss, accuracy, val_loss, and val_accuracy shows on Figure 4.10:

```
Total params: 2,279,041
Trainable params: 2,279,041
Non-trainable params: 0
_________________________________________________________________
Train on 1372 samples, validate on 588 samples
Epoch 1/10
1372/1372 - 1s - loss: 0.6974 - accuracy: 0.5284 - val_loss: 0.6804 - val_accuracy: 0.5765
Epoch 2/10
1372/1372 - 1s - loss: 0.6228 - accuracy: 0.6319 - val_loss: 0.5809 - val_accuracy: 0.6650
Epoch 3/10
1372/1372 - 1s - loss: 0.5341 - accuracy: 0.7201 - val_loss: 0.5558 - val_accuracy: 0.6990
Epoch 4/10
1372/1372 - 1s - loss: 0.5003 - accuracy: 0.7558 - val_loss: 0.5454 - val_accuracy: 0.7211
Epoch 5/10
1372/1372 - 1s - loss: 0.4697 - accuracy: 0.7660 - val_loss: 0.5254 - val_accuracy: 0.7398
Epoch 6/10
1372/1372 - 1s - loss: 0.4460 - accuracy: 0.7784 - val_loss: 0.5239 - val_accuracy: 0.7534
Epoch 7/10
1372/1372 - 1s - loss: 0.4277 - accuracy: 0.8047 - val_loss: 0.5288 - val_accuracy: 0.7636
Epoch 8/10
1372/1372 - 1s - loss: 0.4068 - accuracy: 0.8120 - val_loss: 0.5060 - val_accuracy: 0.7772
Epoch 9/10
1372/1372 - 1s - loss: 0.4047 - accuracy: 0.8083 - val_loss: 0.4895 - val_accuracy: 0.7602
Epoch 10/10
1372/1372 - 1s - loss: 0.3872 - accuracy: 0.8105 - val_loss: 0.5140 - val_accuracy: 0.7806
```

Figure 4.10: Full update model compilation

Following figure implementation shows in the last that loss 38% which less than test loss 50% , train accuracy is 81% which is greater than validation accuracy 78%.This accuracy result is more validate than small model. Total parameter is 2,279,041, Trainable params 2,279,041 and non-triable params 0.

Figure 4.9 provide the both train and test loss and accuracy accordingly epochs.



(a) Train and validation accuracy (b) Train and validation loss

Figure: 4.11. Full model train and validation accuracy Plot.

Figure 4.11 (a) plot as blue for training accuracy where data is 70% and red for validation accuracy where data amount is 30%. In addition validation accuracy gets almost accurate with training accuracy that means training accuracy above 80% after iterating three epochs Figure 4.11(b) shows that blue color for training loss and red for validation loss which shows near80%. As well as graphs, training loss is 40% and validation loss nearly 50% after three epochs.

4.6 Experiment Result

Selected data predicted with the save model. Work as a function every for selected data similar to predicted model optimization size capacity dimension to predict result as early blight disease or late blight disease. Experiment result shows fluctuation of the model summary with model compilation parameter such as loss, accuracy, validation accuracy and Validation_LOSS. All the compilation and optimization done with the help of keras API background upgrade Tensorflow in python. Tensorboard analyzer contains more than one parameter. Such image histogram, graph, epoch accuracy epoch loss etc.

Toggle of Tensorboard function can toggle to visualize each selected layer performances. For CNN usually besides optimize and analyze sometime. Epoch number increase and compilation number increase can give best performance of the model.

3-conv-64-nodes-1-dense-1581577627

Train on 980 samples, validate on 980 samples

Epoch 1/20
980/980 -3s – loss: 0.7016 – accuracy:0.5204 - val_loss:0.6713 - val_accuracy:0.5898
Epoch 2/20
980/980 -1s – loss: 0.4640 – accuracy: 0.7776 - val_loss: 0. 2191- val_accuracy: 0.9102
Epoch 3/20
980/980 -1s – loss: 0.1988 – accuracy: 0.9184- val_loss: 0.1848- val_accuracy: 0.9316
Epoch 4/20
980/980 -1s – loss: 0.1696– accuracy: 0.9327- val_loss: 0.1466 - val_accuracy: 0.9510
Epoch 5/20
980/980 -1s – loss: 0.1414 – accuracy: 0. 9439- val_loss: 0.1399 - val_accuracy: 0.9531
Epoch 6/20
980/980 -1s – loss: 0. 1123– accuracy: 0.9561- val_loss: 0.1540- val_accuracy: 0.9378
Epoch 7/20
980/980 -1s – loss: 0.1582 – accuracy: 0.9337 - val_loss: 0. 1464 - val_accuracy: 0.9531
Epoch 8/20
980/980 -1s – loss: 0.1509– accuracy: 0.9398- val_loss: 0.1676 - val_accuracy: 0.9367
Epoch 9/20
980/980 -1s – loss: 0.0907 – accuracy: 0.9653- val_loss: 0.1045 - val_accuracy: 0.9582
Epoch 10/20
980/980 -1s – loss: 0.0809 – accuracy: 0.9714- val_loss: 0.1243 - val_accuracy: 0.9561
Epoch 11/20

980/980 -1s – loss: 0.0727 – accuracy: 0.9684- val_loss: 0.1173 - val_accuracy: 0.9592

Epoch 12/20

980/980 -1s – loss: 0.0612 – accuracy: 0. 9745- val_loss: 0. 1500- val_accuracy: 0.9367

Epoch 13/20

980/980 -1s – loss: 0.0454 – accuracy: 0.9816- val_loss: 0.1683- val_accuracy: 0.9439

Epoch 14/20

980/980 -1s – loss: 0.0633 – accuracy: 0.9786- val_loss: 0.1573 - val_accuracy: 0.9418

Epoch 15/20

980/980 -1s – loss: 0.0809 – accuracy: 0.9714- val_loss: 0.1243 - val_accuracy: 0.9561

Epoch 16/20

980/980 -1s – loss: 0.0727 – accuracy: 0.9684- val_loss: 0.1173 - val_accuracy: 0.9592

Epoch 17/20

980/980 -1s – loss: 0.0672 – accuracy: 0. 9745- val_loss: 0. 1420 - val_accuracy: 0.9469

Epoch 18/20

980/980 -1s – loss: 0.0662 – accuracy: 0.9816- val_loss: 0.1683 - val_accuracy: 0.9439

Epoch 19/20

980/980 -1s – loss: 0.0391 – accuracy: 0. 9878- val_loss: 0. 1405 - val_accuracy: 0.9561

Epoch 20/20

980/980 -1s – loss: 0.0173 – accuracy: 0.9949- val_loss: 0.1539 - val_accuracy: 0.9469

Table 4.1: Final accuracy compilation.

For step wise accuracy, loss, validation accuracy and validation loss optimization; parameter use Adam, loss function and epochs at most 20. Figure 3.4 shows the function loss and accuracy according to their data process. Result of model prediction tested with the 30% data from the storage as the model is used for classifier for five best layers shows the approximately better result along with other layers. Plot graph and model analyze optimize result give this accuracy for prediction. Tensorboard use with the keras API with tensorflow using python language. All the plot value are used for the graphical estimation.

The more loss minimize the better accuracy obtained. Five best layer result such as accuracy, loss, and validation accuracy and validation loss shows on Table 4.2.

loop	loss	accuracy	Val_loss	Val_accuracy
15/20	0.0454	0.9816	0.1683	0.9439
16/20	0.0633	0.9786	0.1573	0.9418
17/20	0.0672	0.9745	0.1420	0.9418
18/20	0.0622	0.9724	0.1476	0.9490
19/20	0.0391	0.9878	0.1405	0.9561
20/20	0.0173	0.9949	0.1539	0.9469

Table 4.2: Five best layer result.

The above table shows all the parameter running result of our model. Last fifteen to twenty compilation shows the previous figure 3.4 best five result.

4.7 Model Test Overview

Another model function are used for preprocess due to validation test. For early testing purpose several data are deploy with the preserve model by implementing in 4.6 section. Model feature label are would need to same as previously import method such test image on the directory, data directory resize reshape converted one channel etc. The full work has tested with a single method with these process. The save model path has used for predictions with the tested images.

The result predicted with model either 0 or 1 value predict early and late potato blight. An overview of the full testing process in Figure 4.11.

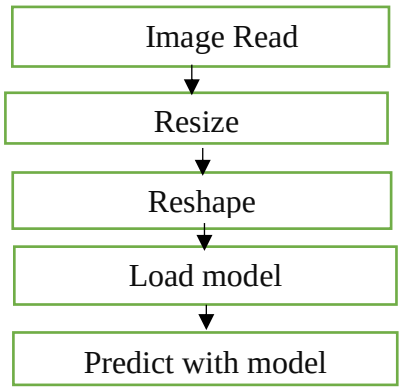


Figure 4.12: Testing Process.

4.8 Data Preparation

Data are collected for prepare as the previous approach like image data prepare same size dimension as any class or can be single data retrieval. For this step real collected data are converted as 148X148 dimension with bulkresize Photo's online application same process used here also. For the first image collection from local area data which are separately stored on storage model. Make sure Tested data are not used before.

4.9 Step-by-Step Method

Separately stored data are manually stored before on the cloud storage drive. Image are read here randomly maintain same size 70x70 and reshapinig one-dimensional vector.

Above all for model finalize which we implemented also deploy in Figure 4.13.

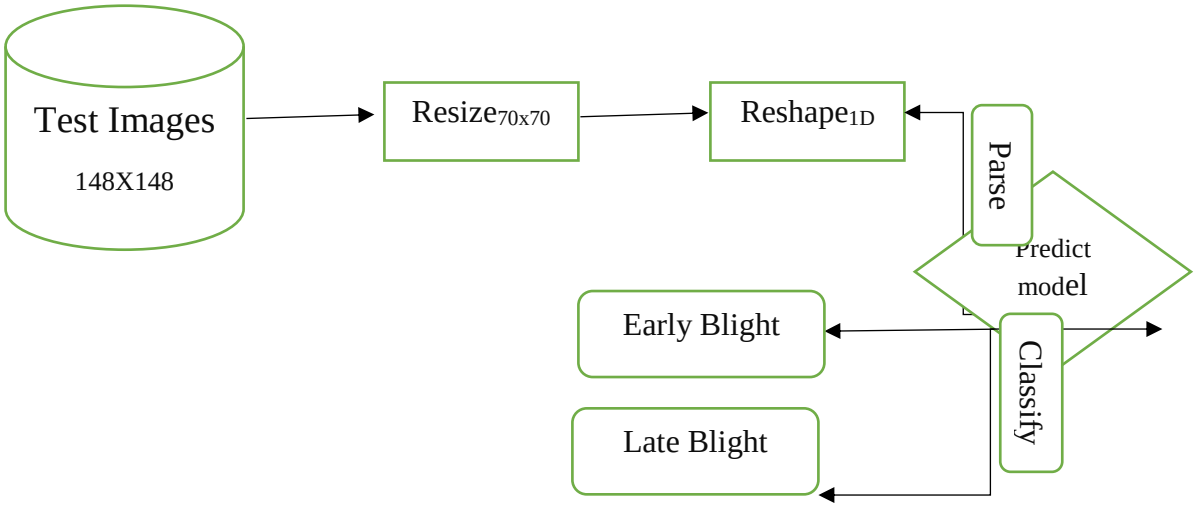


Figure 4.13: Model Predict process.

The figure 4.12 shows the summary of the predict process. Test images with 148x148 is resize with python opencv library. Reshape use for one vector which applied with our model previously. Parse the test image to test with the predicted model which classify as early blight and late blight.

4.10 Conclusion

This chapter revise practical models of disease detection with classification real image. In addition to the output of every part such as image preprocess resize, reshape training result. Update and modify in accordance loss and accuracy obtained. Optimize with best layers conv dense layer correspondingly. Full update and final model adjustment with the unseen real data. In the next chapter discuss future work and other modification model with varieties circumstance

CHAPTER 5

CONCLUSION

Conclusion chapter discuss very traditional and manual approach to compare the implemented method and result. The rising combination of smartphone penetration and recent advances deep learning has paved the way for smart device assisted disease prognosis. Thus, the performance of deep learning models on progressively vast in use and globally available and image datasets presents a clear path toward smartphone aid crop disease detection. CNN model can be deployed into android studio, therefore can be mobile aid prognosis this work. This trained CNN model assures 94% accuracy by modifying optimized model layer by layer. As food one of the basic human need so the deep learning based mobile assisted technology in this era provide a great a role. Traditional approach SIFT [54] , HoG [55], SURF [56] etc,In addition, traditional approaches to disease classification via machine learning typically focus on a small number of classes usually within a single crop. Examples include a feature extraction and classification pipeline using thermal and stereo images in order to classify tomato powdery mildew against healthy tomato leaves according to Raza et al [57] , the detection of powdery mildew in uncontrolled environments using RGB images [58] the use of RGBD images for detection of apple scab [59] the use of fluorescence imaging spectroscopy for detection of citrus huanglongbing [60] the detection of citrus huanglongbing using near infrared spectral patterns [61] and many others. A very recent review on the use of machine learning on plant phenotyping [62] extensively discusses the work in this domain. While neural networks have been used before in plant disease identification [63], the approach required representing the images using a carefully selected list of texture features before the neural network could classify them. The developed methodology has been carried out on a stored server or kernel. Images belonging to three different classes of potato leaf diseases. Keras, a neural network API written in Python, has been used for the

model implementation along tensor flow library. Image classes are mainly two parts namely train dataset and test dataset. All the experiments were performed on Intel Core i7-5567 CPU and processor clock speed up to 2.3GHz.The crop disease prevails food insecurity to eat, export import problem etc. But computer vision and penetration of smartphones have paved the way to disease detection. Thus, the above mentioned model can be made use of as a decision tool to help and support farmers in detecting the diseases that can be found in the potato leaf methodology accuracy 94% optimization can make an accurate detection of the leaf diseases abridges diagnosis effort. By the effective of this methodology mobile based further detection bring to image based disease detection is handier to use.

References

[1] M.Molla (2019) Potato glut badly hurts growers, thedailystar.net, pp. 10-15, https://www.thedailystar.net/backpage/potato-production-in-bangladesh-glut-badly-hurts-growers-1748572. Accessed 25 May 2019.

[2] Z. Jahangir (2014), Export product of mega group: fresh potato, Available at http://www.megagroupbd.com/potato.html. Accessed 26 march 2014.

[3] Pam Mercure (2012), Early Blight and Late Blight of Potato. Dissertation University of Connecticut.

[4] M. Hossain, T.K. Dey, M. Iqbal Hossain, S.N. Begum, M.S. Kadian (2009), Research experience on potato late blight disease management in Bangladesh, Acta Hortic. 834, 175-186 DOI: 10.17660/ActaHortic.2009.834.19.

[5] G. Kessel (2019), Geodata to control potato late blight in Bangladesh (GEOPOTATO). Available at http://www.fao.org/e-agriculture/news/geodata-control-potato-late-blight-bangladesh-geopotato.

[6] H. M. Rasel, M. R. Hasan, B. Ahmed and M. S. U. Miah MSU 2013 Investigation of soil and water salinity, its effect on crop production and adaptation strategy.

[7] Mitchell J. Bauske, Andrew P. Robinson,Neil C. Gudmestad(2018), Early Blight in Potato, North Dakota State University (NDSU) Extensions, pp 1892. https://www.ag.ndsu.edu/publications/crops/early-blight-in-potato/pp1892.pdf

[8] G. Polder, P. M. Blok, Hendrik A. C. de Villiers, Jan M. van der Wolf, and Jan Kamp(2019), Potato Virus Y Detection in Seed Potatoes Using Deep Learning on Hyper spectral Images. Front Plant Sci. 2019; vol: 10, p: 209. Doi: 10.3389/fpls.2019.00209.

[9] EHLER, Lester E (2006), integrated pest management (IPM): Definition, historical development and implementation, and the other IPM. Wiley, Chichester United Kingdom. Vol 62, No 9, pp 787-789.

[10] ITU International Telecommunication Union (2015), ICT Facts and Figures the World in 2015, Place des Nations CH-1211 Geneva Switzerland 2015. https://www.itu.int/en/ITUD/Statistics/Documents/publications/misr2015/MISR2015-ES-E.pdf.

[11] Tai, A.P.K., M. Val Martin and C.L. Heald (2014), Threat to future global food security from climate change and ozone air pollution, 2014. Doi: 10.1038/nclimate2317.

[12] J. Msuya (2019), Report of the Plenary of the Intergovernmental Science-PolicyPlatform 2019. Germany, Accessed on 19 April 2019.

[13] Richard N Strange, Peter R Scott (2005) Plant disease: a threat to global food security.pp 83-116.DOI: 10.1146/annurev.phyto.43.113004.133839.

[14] Yang Lu, Shujuan Yi a,Nianyin Zeng,Yurong Liu,Yong Zhang (2017) Identification of rice diseases using deep convolutional neural networks, ELSEVIER, pp. 20-40, 2017.

[15] Jihen Amara, Bassem Bouaziz (2017), A Deep Learning-based Approach for Banana Leaf Diseases, In: BTWorkshops, pp.7989.

[16] David P. Hughes & Marcel Salathe (2015), an open access repository of images on plant health to enable the development of mobile disease diagnostics, Cornell University, Computer and Science 2015.

[17] Luc Van Gool, Christopher K. I. Williams · John Winn, Andrew ZissermanJohnWinn ·Andrew Zisserman (2010), the pascal visual object classes (voc) challenge. Int. J. Comput. Vis. 88, pp 303–338.

[18] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, Li Fei-Fei (2015), ImageNet large scale visual recognition challenge. Int. J. Comput. Vis. 115, 2015. Doi: https://arxiv.org/abs/1409.0575.

[19] U. Mokhtar, N. El-Bendary, Aboul Ella Hassenian, E. Emary, M. Mahmoud, Hesham A. Hefny, M. Tolba (2015), SVM-Based Detection of Rice Leaves Diseases, Springer, 2015. DOI: 10.1007/978-3-319-11310-4_55.

[20] Sachin D. Khirade, A. B. Patil (2015), Plant Disease Detection, In: 2015 International Conference, pp. 768-770, on February 2015.

[21] Sabah Bashir, Navdeep Sharma (2012), Remote Area Plant Disease Detection Using Image Processing, IOSR Journal of Electronics and Communication Engineering, vol. 2, no. 6, p 31-33, DOI:10.1109/ICCUBEA.2015.153

[22] Xi-Jin Zhang, Yi-Fan Lu, Song-Hai Zhang(2016), Multi-Task Learning for Food Identification and Analysis with Deep, journal of computer science and technology, 31(3): pp 489–500, May 2016.

[22] Ericsso (2015), Information and Communication Technology (ICT) to service providers, Interim Update: Ericsson Mobility Report, 2015.

[23] E.C. Oerke, Crop losses to pests, The Journal of Agricultural Science, vol. 144(01), p. 31–43. DOI: https://doi.org/10.1017/S0021859605005708

[24] Li Deng (2014) a tutorial survey of architectures, algorithms, and applications for deep learning, APSIPA Transactions on Signal and Information Processing. DOI: https://doi.org/10.1017/atsip.2013.9

[25] I. Arel, Derek C. Rose, T. Karnowski (2016), Deep Machine Learning - A New IEEE Comp. Int. Mag. 11 April 2016. DOI:10.1109/MCI.2010.938364.

[26] Jihen Amara, Bassem Bouaziz, Alsayed Algergawy, A Deep Learning-based Approach for Banana Leaf Diseases, B. Mitschang et al. (Hrsg.): BTW 2017 – Workshop band, 2017.

[27] Y. LeCun Boser, B.; Denker, J. S. Henderson, D.; Howard, R. E.; Hubbard, W.Jackel, L. D. (1989), Backpropagation applied to handwritten zip code recognition, Neural computation, AT&T Bell Laboratories, Holmdel, NJ 07733 USA; p. 541–551, 1989.

[28] Yann LeCun, Leon Bottou, Yosuha Bengio and Patrick Heffner (1990), a neural network architecture for handwritten and machine-printed character recognition called LeNet-5.

[29] M. Rizwan (2020), LeNet-5 – A Classic CNN Architecture available at https://engmrk.com/lenet-5-a-classic-cnn-architecture/, 3 March 2020.

[30] T. M. Prajwala, Alla Pranathi, Kandiraju Sai Ashritha, Nagaratna B. Chittaragi, Shashidhar G. Koolagudi (2018) Tomato Leaf Disease Detection using Convolutional," Proceedings of 2018 Eleventh International Conference on Contemporary Computing (IC3), 2-4 August 2018.

[31] Yann LeCun, L eon Bottou, Yoshua Bengio, and Patrick Heffner (1998),Gradient-Based Learning Applied to Documennt Recognition, IEEE, NOVEMBER 1998.

[32] H. S. Seung and H. Sompolinsky (1992), Statistical mechanics of learning from examples, volume 45, number 8, 15 APRIL 1992.

[33] Vladimir Vapnik, Esther Levin and Yann Le Cun (1994), Measuring the VC-dimension of a Learning Machine. Doi: 10.1.1.32.8584.

[34] Corinna Cortes, Lawrence D. Jackel, Sara A. Solla, Vladimir Vapnik, John S. Denker (1993), Learning Curves: Asymptotic Values and Rate of Convergence. NIPS.

[35] Vapnik, Vladimir (1998), the Nature of Statistical Learning Theory. Springer.DOI 10.1007/978-1-4757-2440-0 AT&T Bell Laboratories, Holmdel, March 1995.

[36] William H.Press, Saul A. Teukolsky, William T.Vetterling, Brlan P. (1986) Flannery Numerical Recipes: The art of scientific computing Cambridge university press, Cambridge, 1986.

[37] Sharada P. Mohanty, David P. Hughes and Marcel Salathé (2016), Using Deep Learning for Image-Based Plant Disease Detection, Frontiers in Plant Science, 22 September 2016. https://doi.org/10.3389/fpls.2016.01419.

[38] Jae. S. Lim (1990), Two-dimensional signal and image processing, Englewood Cliffs, N.J: Prentice Hall, 1990. p. 710.

[39] Harrison Kinsley (2018), Deep Learning basics with Python, Tensorflow and Keras,accessed27August2018.Available:https://pythonprogramming.net/loading-custom-data-deep-learning-python-tensorflow-keras/.

[40] G. Cheng, P. Zhou and J. Han(2016), Learning Rotation-Invariant Convolutional Neural Networks for Object Detection in VHR Optical Remote Sensing Images, vol. 54, no. 12, pp. 7405 - 7415, 2016.

[41] G. Cheng, J. Han, P. Zhou and D. Xu (2019), Learning Rotation-Invariant and Fisher Discriminative Convolutional Neural Networks for Object Detection, vol. 28, no. 1, January 2019.

[42] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich(2015) Going Deeper with Convolutions, https://arxiv.org/abs/1409.4842.

[43] Jiuxiang Gua, Zhenhua Wang, Jason Kuenb, Lianyang Ma, Amir Shahroudy (2017), Recent advances in convolutional neural networks, Elsevier.

[44] Yinhui Deng, Yuanyuan Wang, Senior Member, IEEE, and Ping Chen (2008) Automated detection of Polycystic Ovary Syndrome from ultrasound images, Conf. Proc.IEEE, pp. 4772–4775.

[45] Palak Mehrotra, Jyotirmoy Chatterjee, Chandan Chakraborty, Biswanath Ghoshdastidar, Sudarshan Ghoshda (2011) Automated Screening of PCOS using Machine learning technique palak., IEEE India Conference, 2011.

[46] E. Allibhai (2018), Keras tutorial – build a convolutional neural network. Towards data Science. Accessed on 17 October 2018.

[47] Ji Young Lee, Franck Dernoncourt (2016), Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks, in conference paper at NAACL, in 2016. p. 515–520 DOI:10.18653/v1/N16-1062.

[48] Chollet, François (2020), Getting started with the Keras Sequential model. Accessed on 12 April 2020.United States online open source project by keras SIG. DOI:https://keras.io/guides/sequential model/

[47] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton (2012), Imagenet classification with deep convolutional neural networks, In: Advances in neural information processing systems, pp. 1097–1105.

[48] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich (2015) Going Deeper with Convolutions, DOI:https://arxiv.org/abs/1409.4842.

[49] Diederik P. Kingma, Jimmy Ba (2014), Adam: A Method for Stochastic Optimization, a conference paper at ICLR 2015.

[50] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton (2012), ImageNet classification with deep convolutional neural networks, p. 1097–1105.

[51] Yann Lecun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L.D. Jackel (1989) Backpropagation applied to handwritten.,IEEE, p. 541–551.

[52] T. M. Prajwala, Alla Pranathi, Kandiraju Sai Ashritha, Nagaratna B. Chittaragi, Shashidhar G. Koolagudi (2018), Tomato Leaf Disease Detection using Convolutional Neural Networks (CNN), Proceedings of 2018 Eleventh International Conference on Contemporary Computing(IC3), in IEEE, 2018. DOI:10.1109/IC3.2018.8530532.

[53] Harrison Kinsley (2018), Deep Learning basics with Python, Tensorflow and Keras, Analyzing Models with Tensorboard, accessed 2018.

[54] David G. LOWE (2004), Distinctive Image Features from Scale-Invariant Key points, International Journal of Computer Vision), vol. 60, no. 2,pp , 91–110.

[55] N. Dalal and B. Triggs (2005), Histograms of Oriented Gradients for Human Detection, IEEE. DOI:10.1109/CVPR.2005.177

[56] Herbert Bay, Andreas Ess, Tinne Tuytelaars, Luc Van Gool (2008), Speeded-Up Robust Features (SURF), ELSEVIER, p. 346–359.

[57] Shan-e-Ahmed Raza; Prince, Gillian; P. Clarkson, John; M. Rajpoot, Nasir (2015), Automatic Detection of Diseased Tomato Plants Using Thermal and Stereo Visible Light Images, 2015. https://doi.org/10.1371/journal.pone.0123262

[58] Deny Lizbeth Hernández-Rabadán, Fernando Ramos-Quintana and Julian Guerrero Juk (2014), Integrating SOMs and a Bayesian Classifier for Segmenting, Hindawi Publishing Corporation, Oct 6, 2014.

[59] Chéné Y, Rousseau D, Lucidarme P, Bertheloot J, Caffier V, Morel P, Belin É, Chapeau-Blondeau F(2012), On the use of depth camera for 3d phenotyping of entire, Computers and Electronics in agriculture., 01 Mar 2012 DOI: 10.1016/j.compag.2011.12.007.

[60] Caio B. Wetterich, Ratnesh Kumar, Sindhuja Sankaran, José Belasque Junior,Reza Ehsani,and Luis G. Marcassa, "A comparative study on application of computer vision and fluorescence imaging spectroscopy for detection of Huanglongbing citrus disease in the USA and Brazil.," Journal of Spectroscopy, 2013. Volume 2013,https://doi.org/10.1155/2013/841738

[61] Francisco Garcia-Ruiz , Sindhuja Sankaran, Joe Mari Maja, Won Suk Lee c, Jesper Rasmussen,Reza Ehsani(2013), "Comparison of two aerial imaging platforms for identification of," Computer and Electronics in Agriculture, pp. 106-115, 2013. http://dx.doi.org/10.1016/j.compag.2012.12.002

[62] Arti Singh, Baskar Ganapathy Subramanian, Asheesh Kumar Singh, Soumik Sarkar (2015), Machine Learning for High-Throughput Stress Phenotyping in Plants, 2015. 21(2), P110-124 DOI:https://doi.org/10.1016/j.tplants.2015.10.015

[63] Kuo-Yi Huang (2007) "Application of artificial neural network for detecting Phalaenopsis seedling diseases using color and texture features ELSEVIER - Computer and Electronics in Agriculture,Volume57,Issue1,May2007,pp 3-11. doi.org/10.1016/j.compag.2007.01.015

Appendix A.

Collected data: Early blight data

And late blight data:

Appendix B.

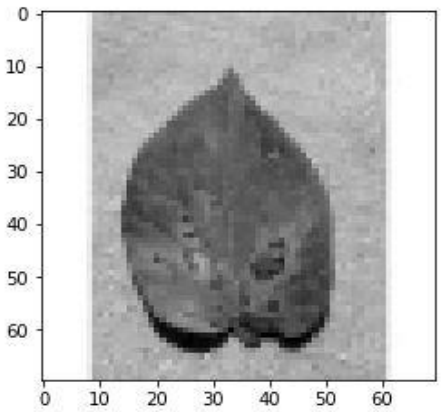
Code snippet: Path for Class marge and run as single directory.

```
DATADIR = "/content/drive/My Drive/Colab Notebooks/Thesis"
CATEGORIES = ["Potato_Early_blight", "Potato_Late_blight"]
i = 0
for category in CATEGORIES:
    path = os.path.join(DATADIR,category)
    for img in os.listdir(path):
        print(img)
        i=i+1
        if i==5:
           break
        img_array = cv2.imread( os.path.join(path,img),cv2.IMREAD_GRAYSCALE)
        plt.imshow(img_array,cmap="gray")
        plt.show()
        #break
    break
```

Reshape Data Snippet:

```
IMG_SIZE = 70

new_array = cv2.resize(img_array, (IMG_SIZE, IMG_SIZE))
plt.imshow(new_array, cmap='gray')
plt.show()
```



Two Model summery snippet prior best Model:
1) Small model with 64 conv layer

```
1-conv-64-nodes-1-dense-1583608673
Model: "sequential_121"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_233 (Conv2D)          (None, 48, 48, 64)        640
_________________________________________________________________
activation_424 (Activation)  (None, 48, 48, 64)        0
_________________________________________________________________
max_pooling2d_233 (MaxPoolin (None, 9, 9, 64)          0
_________________________________________________________________
flatten_119 (Flatten)        (None, 5184)              0
_________________________________________________________________
dense_229 (Dense)            (None, 64)                331840
_________________________________________________________________
activation_425 (Activation)  (None, 64)                0
_________________________________________________________________
dense_230 (Dense)            (None, 1)                 65
_________________________________________________________________
activation_426 (Activation)  (None, 1)                 0
=================================================================
Total params: 332,545
Trainable params: 332,545
Non-trainable params: 0
_________________________________________________________________
Train on 1372 samples, validate on 588 samples
```

2) Best model with 256 Conv layer:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_14 (Conv2D)           (None, 68, 68, 256)       2560
_________________________________________________________________
activation_23 (Activation)   (None, 68, 68, 256)       0
_________________________________________________________________
max_pooling2d_14 (MaxPooling (None, 34, 34, 256)       0
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 32, 32, 256)       590080
_________________________________________________________________
activation_24 (Activation)   (None, 32, 32, 256)       0
_________________________________________________________________
max_pooling2d_15 (MaxPooling (None, 16, 16, 256)       0
_________________________________________________________________
flatten_7 (Flatten)          (None, 65536)             0
_________________________________________________________________
dense_14 (Dense)             (None, 64)                4194368
_________________________________________________________________
activation_25 (Activation)   (None, 64)                0
_________________________________________________________________
dense_15 (Dense)             (None, 1)                 65
_________________________________________________________________
activation_26 (Activation)   (None, 1)                 0
=================================================================
```