

Plant Disease Detection Using Convolutional Neural Network

by

Mohammad Shifat Hossain

15101044

Fatin Ishraq Noor

15301086

Mir Ayman Ali

15101104

Rafiul Alam

15101130

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
April 2020


© 2020. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:



Mohammad Shifat Hossain
15101044



Fatin Ishraq Noor
15301086



Mir Ayman Ali
15101104



Rafiul Alam
15101130

Approval

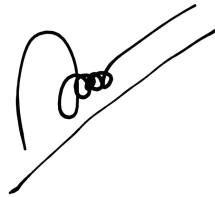
The thesis/project titled "Plant Disease Detection Using Convolutional Neural Network" submitted by

1. Mohammad Shifat Hossain (15101044)
2. Fatin Ishraq Noor (15301086)
3. Mir Ayman Ali (15101104)
4. Rafiul Alam (15101130)

Of Spring, 2020 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on April 7, 2020.

Examining Committee:

Supervisor:
(Member)



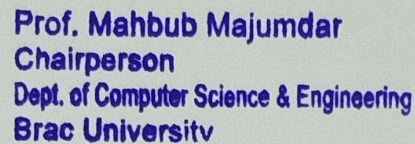
Dr. Muhammad Iqbal Hossain
Assistant professor
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)



Md. Golam Rabiul Alam, PhD
Associate Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)



Mahbubul Alam Majumdar
Professor and Chairperson
Department of Computer Science and Engineering
Brac University

Abstract

Rice is a staple crop of Bangladesh and many metric tons of it are being destroyed every year due to diseases. If the diseases can be efficiently and accurately classified and recognized at early stage, the farmers can get the required help resulting in better rice crop yields. Thus, in an attempt to better increase the rice crop, yield our proposal is to make a website prototype system by using different machine learning algorithms to analyze and recognize different rice crop diseases. By utilizing CNN and its variations for the detection of rice plant diseases, we aim to guide individuals and assist farmers in identifying the infected plants early. By doing so, automated systems can be made to find out the infected crops and suggest diagnosis based on the problems. The photographs of rice plant leaves are taken for brown spot, Hispa and leaf blast diseases. We have used Convolution Neural Network (CNN) which comprises of different layers which are used for prediction. In addition, we have implemented other 4 CNN structures such as GoogleNet, ResNet-152 and VGG19 which is 19-layer deep structure. On the other hand, the features from the infected area are extracted using Histogram Oriented Gradient (HOG) features and for distinguishing between their category these features were given to the Support Vector Machine (SVM). To sum up, by experimentation we will be able to conclude which structure or algorithm has the most success rate. As a result, by this approach the information will be provide at the initial stage so that one can take necessary steps at the beginning to prevent the rice plant diseases and minimize the loss of production.

Keywords: Machine Learning; Rice Plant Disease; Detection; Prediction; ResNet-152; Convolutional Neural Network

Acknowledgement

First, We thank the Almighty for whom we could complete our thesis without any hindrance.

Our Advisor Dr. Muhammad Iqbal Hossain sir was kind enough to take us as his thesis students and guide us in every step of the way to reach at this position. He apprised us beforehand what issues we would face and urged to go on the right path. And obviously our Computer Science and Engineering Department (Brac University) for providing us the necessary equipment without which we couldn't complete our thesis.

We would like to mention our unsung heroes, our parents who supported us till date and kept us in their prayers.

Table of Contents

Declaration	i
Approval	ii
Abstract	iii
Acknowledgment	iv
Table of Contents	v
List of Figures	vii
List of Tables	viii
Nomenclature	ix
1 Introduction	1
1.1 Motivation	1
1.2 Problem Statement	1
1.3 Objective and Contribution	3
1.4 Thesis Structure	3
2 Background	4
2.1 Literature Review	4
2.2 Algorithms	6
2.2.1 SVM	6
2.2.2 CNN	7
2.2.3 Transfer Learning	7
2.2.4 GoogLeNet	8
2.2.5 VGG19	9
2.2.6 ResNet-152	9
3 Proposed prediction model	11
3.1 Dataset Description	11
3.1.1 Dataset	11
3.1.2 Data preprocessing	11
3.2 Proposed Approach	12
3.2.1 Image Collection	12
3.2.2 Pre-processing Images	12
3.2.3 Training Model	13

3.2.4	Detection of affected plants	14
4	Experimentation	16
4.1	Support Vector Machine (SVM)	16
4.2	Convolutional Neural Network(CNN)	17
4.3	GoogLeNet	18
4.4	Visual Geometry Group (VGG19)	19
4.5	Residual Network (ResNet-152)	20
5	Discussion	21
5.1	Qualitative Analysis	21
5.2	Comparative Analysis	21
6	Conclusion and Future Work	23
	Bibliography	26

List of Figures

1.1	Rice Blast	2
1.2	Leaf Brown Spot	2
1.3	Rice Hispa	3
1.4	Marks on the leaves	3
2.1	Architecture of SVM	6
2.2	Architecture of CNN	7
2.3	Block Diagram of transfer-learning	8
2.4	Traditional Learning vs Transfer Learning	8
2.5	Inception with dimensional reduction	8
2.6	GoogLeNet architecture	9
2.7	Architecture of VGG19	9
2.8	Comparison of architectures between VGG19, ResNet-152 and Feed-forward Neural Network	10
3.1	Proposed Approach	12
3.2	Augmentation Configurations	13
3.3	Image Normalization	13
3.4	Training Model CNN	14
3.5	Identify the diseased rice plants	15
4.1	RGB to grayscale to HOG feature extraction	16
4.2	Accuracy and Confusion Matrix SVM	16
4.3	using image augmentation CNN	17
4.4	CNN configuration	17
4.5	Training and validation accuracy; and loss graph CNN	17
4.6	Dataset images after image augmentation GoogLeNet	18
4.7	GoogLeNet Configuration	18
4.8	test accuracy of GoogLeNet	18
4.9	VGG19 Configuration	19
4.10	VGG19 Hyperparameters	19
4.11	Training and Validation accuracy; and loss VGG19	19
4.12	Single layer of ResNet152	20
4.13	Model generating phase for ResNet152	20
4.14	ResNet-152 Test Result	20
5.1	Model Accuracy Comparison	22

List of Tables

3.1	Dataset	11
3.2	Data distribution	11

Nomenclature

The next list describes several symbols & abbreviation that will be later used within the body of the document

API Application Program Interface

BF Brute-Force

CIELAB CIE L*a*b* is color space specified by the International Commission on Illumination

CNN Convolutional Neural Network

HIS Hue Intensity Saturation

HOG histogram of oriented gradients

KNN k-nearest neighbors

NN Neural Network

ResNet Residual Network

RGB Red, Green, Blue

SIFT Scale-invariant feature transform

SOM Self-organizing map

SURF Speeded-Up Robust Features

SVM Support Vector Machine

VGG Visual Geometry Group

YCbCr is Digital MPEG compression

Chapter 1

Introduction

1.1 Motivation

Bangladesh is an agriculture dependent country. Rice is the primary food of the people, accounting for approximately 93 percent of the entire meals produced, about 70 percent of common strength intake and 35 percentage household expenditure. It has 85.77 lacs of hectares of fertile land according to the Bangladesh Bureau of Statistics. 13992874 metric tons of Aman rice was produced in the fiscal year 2017-2018. The major step for reducing the amount of loss in the production and amount of agriculture produced is to identify the diseases of rice plants. By using automatic technique to detect the disease is useful as it reduces a lengthy time-consuming works such as monitoring of crop fields, as it can detect the symptoms of disease at a very early stage i.e. when observed on plant leaves. Plant disease detection is necessary because it can increase yield of harvest if proper and timely precautions are taken to treat the disease. The motivation behind our research arises from the need to lower rice harvest wastage due to disease which will help to lower deficit of food.

1.2 Problem Statement

Due to rice plant disease the living cost of a country is increased and lowers the of quality and quantity of agriculture crops. Due to these two aspects occurring at the same time can lead to affect the production of crops in the nation. These challenges have created a great interest among the researchers to discover involuntary methods that can help to identify the diseases with great correctness related to rice plants and help the stakeholders detect the diseases early to reduce economic loss. Hence, the specific fertilizer can be chosen by the farmers. In addition, there are many models to detect various types of plant diseases solving image classification problems and machine learning techniques, such as [23],[25], [10],[2]. As we know, in Asian countries rice is the vital crop [21] including [24]; thus, in this paper, we have implemented CNN and its variants along with SVM to identify and detect rice plant diseases in both early and current state of the plants. First thing to remember only because rice plant diseases around 10-15% of rice production is hampered in Asia [13]. According to studies it is can be deducing the major reason for rice plant disease are fungus and bacteria.



Figure 1.1: Rice Blast



Figure 1.2: Leaf Brown Spot

In these days, deep learning techniques are used by many researchers due to its extraordinary execution to classify images. The ability to execute feature engineering by its own is the greatest benefits of deep learning technique and not at all like the classic algorithms used to solve this problem. The most relevant deep learning technique for image classification is CNN. [21],[8]. The relationship between the layers and set of spatial data of the picture is given by CNN. As a result, it is dependable for the classification of images [3]. In addition, number of experimentation on rice plant diseases with various CNN architectures is much lower comparing with other experimentation with CNN in other fields. In [21], examined the capacity of CNN to for classification of rice plant diseases.

However, experimentation with CNN is very challenging as CNN requires huge dataset for training data for efficient training and gathering as well to classify the images of diseased rice plants is a very challenging task. We have collected data set of 3 of the major rice plant diseases and implemented various CNN architectures to distinguish which has the most success rate. Whereas, the limitations with CNN we have encountered can be beaten by the implantation of transfer learning. To begin with, the existing trained networks are used by transfer learning on huge data set. Therefore, for the current small data set it is useful to update weights [5]. According to our research, this paper describes different models to identify rice plant diseases using SVM and implementation of various CNN architectures.

To begin with, we will be focusing on the 3 of the major rice plant diseases which are Leaf Blast, Brown Spot, Hispa, Firstly, Rice blast as shown in Figure 1 is a fungal disease and the most common disease which has the higher percentage to occur than any of the other rice plant diseases. It causes serious issue, particularly within the zones where the safe assortments are not accessible and easily gets infected due to bad environment. The dreadful the blast becomes the more it affects the plant. A total of 779 images were gathered of rice plants which were affected by rice blast disease.

Secondly, another important rice disease also caused by fungus named Leaf Brown Spot. We were able to collect 523 images of plants having brown spot. The land which has a lacking in silicon is more likely to cause Brown Spot and gets deadliest. Brown spot, shown in figure 2 also has the major role to decrease the overall production. Its major area to attack is leaf and leaf sheath. The affected areas might be observed having measuring 0.5 – 2.0mm in size in an oval structure. This infection might keep on affecting the plant during its entire lifetime.

Lastly, the leaves of rice plants affected by the Rice Hispa pest as shown in Figure



Figure 1.3: Rice Hispa



Figure 1.4: Marks on the leaves

3(a) and the pattern is shown in Figure 3(b), can be recognize by the long line marks that appeared on the surface of the leaf. There were around 565 images we have collected as it is a very common disease and the steps to prevent it must be done immediately. The adult's pests and larvae of the rice Hispa are responsible for the major damage and they usually cause harm to the upper side of the leaves. The traditional way to detect only this is to holding the damaged leaf in bright light or by touching the fingers and rubbing against them. Therefore, with simple pictures of the leaves it might be detected and classify the disease in an efficient and time-consuming manner and this is our major goal.

1.3 Objective and Contribution

Our objectives include Detection of rice plant diseases from images captured via digital camera which were uploaded to a website. Work on detection of rice plant disease are done via SIFT or SURF to extract the features and then other machine algorithms are used to classify the diseases such as KNN, SVM, Naïve-Bayes etc. Features extraction by Transfer Learning has not yet been done thus our contribution is we will use Transfer Learning to extract features from the images. Hence, features will then be fed to another neural network which will then classify the diseases.

1.4 Thesis Structure

Chapter 1: Introduction where motivation, problem statement, objectives and contributions were discussed.

Chapter 2: In literature review, we discussed the previous work and related works. We described about various works on detection of rice plant diseases via various machine learning algorithms. We also discussed about Neural Network and how it works describing some of the algorithm that are related to our project.

Chapter 3: We described about our proposed model and provided a workflow diagram. There was a description data-set about how we pre-processed data and also performed feature selection. There was a brief description about our model and also about its detail.

Chapter 4: We discussed about the experiments we performed and analyzed the results.

Chapter 5: It includes conclusion and the plan about our future works

Chapter 2

Background

2.1 Literature Review

In [24], researchers used a method called SIFT to extract key features and calculate descriptors. Then for image classification, BoW (Bag of Words) is used to convert extracted features identified by SIFT to words. After classification, they used BF(Brute-Force) matcher to match the descriptors of a feature of the first set with the second sets containing extra features. Lastly, SVM (Support Vector Machine) was used to train and detect the disease-ridden rice leaves from a dataset.

In [1], for segmentation of images and to identify the diseased part of the leaves, researchers used HIS (Hue Intensity Saturation) model. According to this paper, with the help gray value of the pixels of spot images are fed as input and then for classification Self-organizing map (SOM) neural network has been implemented.

In [19], image is pre-processed and clipping is performed to get the infected regions. Smoothing filter function is used over the images and the contrast is enhanced. Then the segmentation is done using Genetic Algorithm. For doing clustering to a set of unlabeled points in N-dimension into K clusters. The segmented data is represented as HIS color space.

In [17], there are two phases in detection of the diseases. In the first phase they are using a feature extraction technique called SIFT to detect the disease, then they have filtered the image at various scales. The second phase is disease recognition. Haar-like features and AdaBoost classifier are used to identify affected regions of the paddy plant. SIFT, k-NN and SVM are used to recognize the various categories of diseases like brown spot, leaf blast and bacterial blight.

In [25], researchers used image processing, segmentation technique and classification algorithm SVM. They used Otsu's method to calculate all possible threshold values, measure the spread for pixels to distinguish between foreground and background and find the minimum spread. Using genetic method they tried to find the best coefficients for RGB planes. They tried to extract feature from texture of the object using Blob Analysis where the spread is minimum. Using Quantitative color measurement they found the lesion in the leaf. Using supervised machine learning (SVM) they plotted the features in n-dimensional space and found the hyper plane

to differentiate the two classes and thus finding the disease.

In [14][18], transfer learning is executed which is a method where the at the beginning weights of CNN occurs from pre-trained network. Furthermore it is concluded that on the small dataset transfer learning's optimization is greater than training from the starting position.

Ghaiwat in [22] for classification of plant disease demonstrate a survey on various classification methods. In this paper it is discussed about the simplest of all algorithm, k-nearest-neighbour for prediction of class and which is the perfect method for given test example. Whereas, SVM is also implemented and discussed about one of the drawbacks of it. If training data is not linearly separable then to find the optimal parameters will be very challenging in SVM.

Moreover author in paper [11] explained about the major four methods for development of the processing scheme. Firstly, a colour transformation structure is created for the input RGB image. As HSI is used for color descriptor that is why RGB is used to generate color. It is also needed to transfer or to convert image of RGB. Secondly, green pixels are masked and removed by threshold value. Thirdly, the image is segmented from the useful segments that were extracted in previous step. Lastly the main step concludes and make sure that the segmentation is done.

Mrunalini in [4] also discussed about the technique to identify plant diseases. According to the paper by machine learning based recognition system will boost Indian Economy by saving money and time and also by giving much less effort. To begin with for feature set extraction is the color co-occurrence method. Here, neural networks are used for detection of diseases in leaves. This approach is very efficient and without giving much efforts in computation the accurate detection of leaf and roots diseases can be found out.

According to [9] to identify plant disease histogram matching is executed. The method deals with the edge detection and also colour for histogram matching technique as in plants the diseases occurs mainly on the leaves. For the training process layer separation is implemented which have the training of the given samples. Hence, by separating the layers of RGB image into blue, red, green layers. Furthermore, the edge of the layered images are done using edge detection technique. For development of the color co-occurrence texture analysis method Spatial Gray-level Dependence matrices are used.

In addition, triangle threshold methods and basic threshold methods are well explained in [6]. For detecting the infected area and also to segment the leaf area those two methods are used. In infected region the triangle threshold and basic threshold techniques are used. In addition, the leaf area is segmented. Lastly, by finding the quotient of leaf area and infected area the diseases are categorized. Also the leaf area calculation is executed by using threshold segmentation. To conclude the given method is time consuming and very fast according to the paper for determining the level of severity of the diseases. Authors in [7] introduce how disease spot segmentation can be detect in plant leaf using an algorithm of image processing techniques [7]. In this paper, by comparing the effect of HSI, CIELAB, and YCbCr color space the

disease is detected. Median filter is used for soothing the images. Lastly, calculation of threshold is done by applying Otsu method on color component to find the disease spot. Moreover, CIELAB color model is used to remove the noises which were due to camera flash and vein to minimize the limitations to obtain an efficient result.

Moreover, diverse methods of image processing techniques to detect plant disease is presented in the paper [12]. Existing methods studies are designed to increase subjectivity output and reduction resulting from naked eye observation through which plant disease identification and detection is performed.

By considering the literature review, it can be deduce that to increase the efficiency, researchers have explored and implemented various methods for rice plant diseases detection build on conventional machine learning such as k-nearest neighbor, and support vector machine (SVM) etc. Gradually, deep CNN is becoming efficient day by day with higher success rate and more efficient result in classifying amongst the classes. Though training period needs a lot of time but on the other hand, the trained models can distinguish and classify images very easily and quickly. For pattern detection with massive amount of data CNN is the perfect model nowadays.

2.2 Algorithms

2.2.1 SVM

The Support Vector Machine is a technique that uses machine learning in a supervised way. Which is applied when we have to classify between two or more groups. The algorithm works by plotting all the data as points in an n-dimensional region where n is the quantity of relevant features and the location of every feature corresponds to a particular coordinate of that region. The next task of SVM is to find the optimal hyper-plane that separates the two classes. To achieve maximum accuracy, the hyper-plane is selected in such a way that most of the data points fall on either sides of the plane since SVM is a binary classifier. The number of features available from the training set can also give a rise to equal number of hyper-planes. For such cases, the optimal hyper-plane is one when the margin between the data points is maximum. This margin is calculated by doubling the difference between the hyper-plane and the closest data. The ideal hyper-plane is chosen from the margins having the highest value and no data points within its region. The higher valued margin is selected to maintain a higher precision, since a lower margin can lead to higher faulty classification.

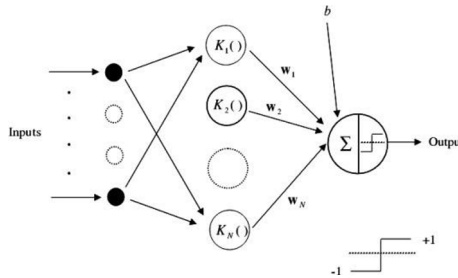


Figure 2.1: Architecture of SVM

2.2.2 CNN

CNN (Convolutional Neural Network) is a deep learning algorithm which takes an image as an input and extracts all possible features from the images making the need for hand-engineered feature extraction obsolete. Compared to other algorithms, the pre-processing complexity of CNN is much less. In the earlier algorithms, filters were manually provided, but with server iterations of training, CNN has the capacity to learn those filters by themselves. The CNN was designed according to the circuit of neurons in the human brain. The role of CNN is to compress the images in way that reduces processing complexities so that no features are lost and maximum accuracy is achieved. This characteristic is essential when we want to create a model does not only adept at extracting features but can also adapt to large datasets. The pooling layer has the task of decreasing the spatial area of the convolved features. This serves the purpose of decreasing the effort to minimize the dimension of the matrix still maintaining matrix integrity. Max pooling and average pooling are the two types of pooling available. Max pooling gives the maximum value that the kernel encloses on the part of the image. On the other hand, average pooling gives the average of every value that the kernel encloses. High-level features are learned from the fully connected layer. Then the images are converted to a column to its reduced form. These variables define the prominent features using provided numbers of iterations using softmax.

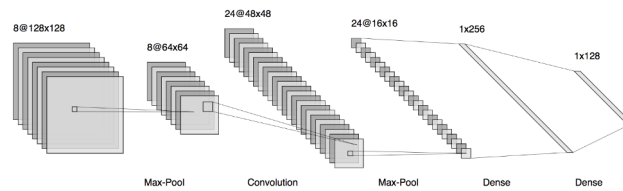


Figure 2.2: Architecture of CNN

2.2.3 Transfer Learning

A unique event can be perceived in numerous deep neural networks: in the starting layers of the network, a deep learning model tries to learn a low level of features, like detecting edges, colours, variations of intensities, etc. These features do not seem to fall under a specific dataset or an errand because of no matter what type of images we are processing either for detecting a lion or cars. In both cases, we must detect these low-level features. All these features occur regardless of the exact cost function or image dataset. Thus, learning these features in one task of detecting lion can be used in other tasks like detecting humans. This is what transfer learning is. Nowadays, it is very hard to see people training a complete CNN from the ground up, and it is common to use any pretrained model trained on a plethora of images in a similar task, e.g. models trained on ImageNet (1000 categories of around 1 million images), and use features from them to solve a new task.

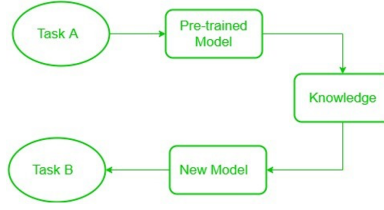


Figure 2.3: Block Diagram of transfer-learning

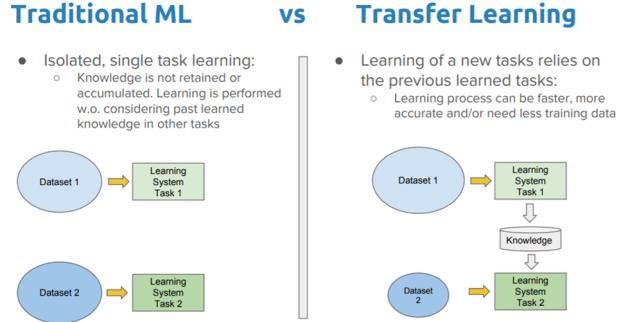


Figure 2.4: Traditional Learning vs Transfer Learning

2.2.4 GoogLeNet

This architecture uses masks of three different sizes for the same input image and merges the identified classes to get a vigorous output. It is a neural network of 22 layers which helps to decrease the amount of params from sixty million in case of AlexNet to four million. To reduce the feature space a 1×1 conv2d is introduced [16]. This architecture naturally selects the appropriate features by finding the best weights when training of the network is ongoing.

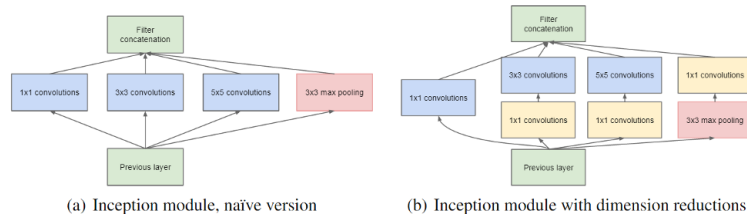


Figure 2.5: Inception with dimensional reduction

This Fig. illustrates the multiple convolution with 1×1 filter, 3×3 filter, 5×5 filter, and max-pooling layer.

There are multiple Inception modules combined to form a deeper network by which high accuracy can be obtained. Transfer learning provides new training for GoogLeNet and AlexNet Network to recognize new faults in the fabric. The image analysis pre-trained network was trained using numerous images to categorize different fabric fault including Knots, Foreign fiber, Oil stain, and Needle line among others [15]. Images are put to the nas input, and there is labeling of the objects in the images making up the output on the basis of probabilities of every of the respective object trajectory.

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 2.6: GoogLeNet architecture

2.2.5 VGG19

VGG is a convolutional neural network that takes an input image (224x224 RGB) of dimension. For pre-processing, the mean RGB value is deducted from every pixel that is calculated over the whole training set. The image is passed through an array of N convolutional layers which uses 3x3 masks. To preserve spatial resolution a spatial padding of 3x3 pixel windows with stride 1 was used. Five max pooling layers are used to perform spatial pooling. Using a 2x2 pixel window of stride 2, max pooling was performed. To make the model classify better and to speed-up computation, ReLu is required to perform non-linearity function. Next, there are 3 FC layers, among which the first 2 have a size of 4096 channels each step, then the 3rd having thousand channels to perform all the way to detection. The last layer does the soft max. In VGG19 there are a total of 16 conv2d layers and 3 FC layers.

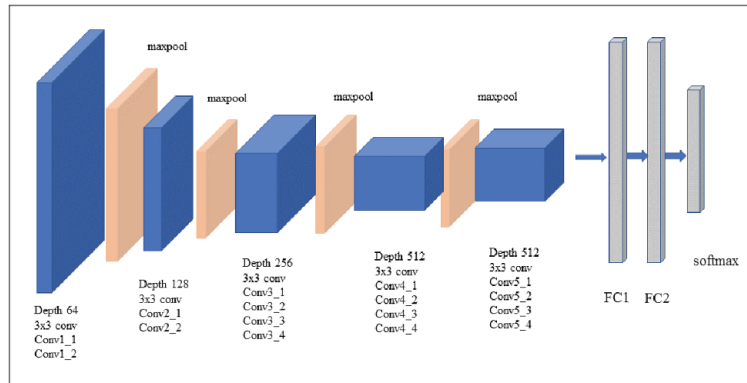


Figure 2.7: Architecture of VGG19

2.2.6 ResNet-152

ResNet learns from residual functions by skipping over some of the layers. This enables the residual networks to be optimized easily and gain increased accuracy with depth. ResNet-152 has a depth of 152 layers. The convolutional layers contain

3x3 filters but, for an equivalent output feature map size, these layers contain same number of filters; the feature map when halved the amount is made twice, it helps to preserve the reduce the time complexity. For down sampling conv2d layer of 7x7 and stride 2 is use. finishes with a global average pooling layer and a FC layer of a thousand progressive nodes and a softmax function to finish it all. Based on the network, shortcut connections are inserted that turns the network to counterpart residual version. With each increase in dimension the shortcut still performs character mapping, with additional 0 entries cushioned for expanding dimensions which produces no additional parameters.

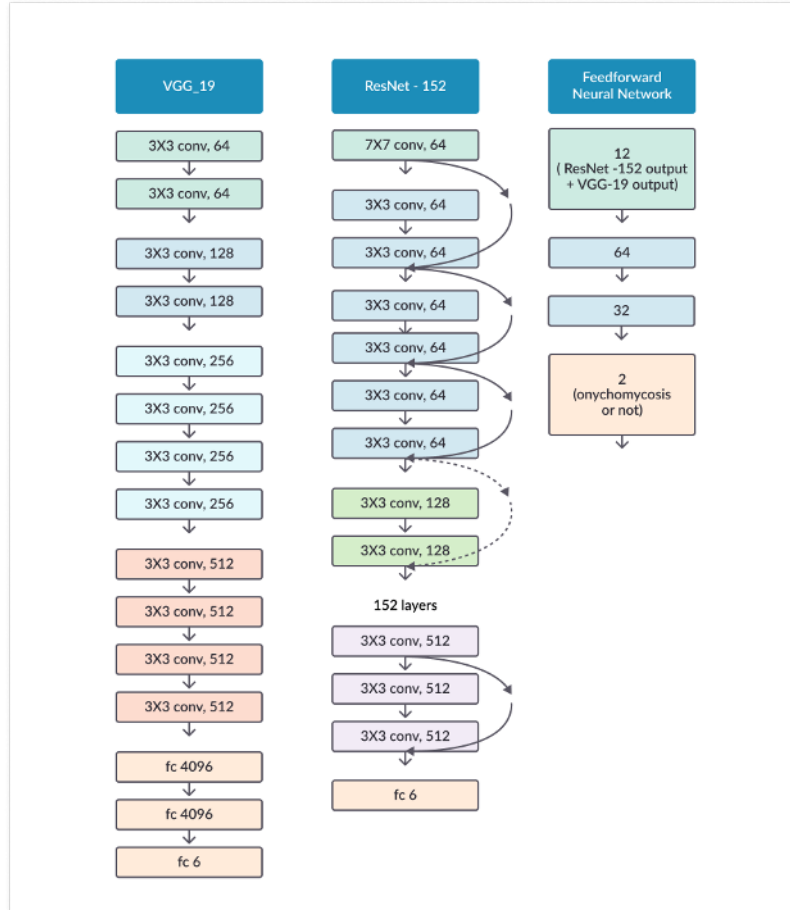


Figure 2.8: Comparison of architectures between VGG19, ResNet-152 and Feedforward Neural Network

Chapter 3

Proposed prediction model

3.1 Dataset Description

3.1.1 Dataset

In this project we have used a dataset is collected from and online dataset repository called Kaggle. The images were already nicely labelled with related images placed stacked together. It contained 3355 high quality labelled images with clear background as shown in Table 3.1. It comprises of 3 classes of images for rice plant leaves infected with Brown Spot, Hispa and Leaf Blast. Another class contained images of healthy leaves. The images were then split into 3 sets (A) training, (B) testing and (C) validation as show in the figure.

Class & No. of images	Training	Testing	Validation
Brown Spot	523	418	105
Healthy	1488	1190	298
Hispa	565	452	113
Leaf Blast	779	623	156

Table 3.1: Dataset

3.1.2 Data preprocessing

The samples in the dataset were already properly labelled and the background noise was pretty low. We split the data into 3 parts for (a) Training (b) Testing (c) Validation as shown in the Table 3.2.

	(&)	Amount
Training	80%	2683
Testing	10%	2683
Validation	10%	336

Table 3.2: Data distribution

3.2 Proposed Approach

This section contains the proposed methodology for the detection rice plant disease mainly Brown Spot, Leaf Blast and Hispa. Figure 3.3 shows the steps how the rice disease should be identified from the data set. The labels denote each high-level step that is performed in our proposed work without any specialized method i.e. various algorithms were used for learning the data (CNN, SVM etc). The detailed process will be discussed in the next part of this section.

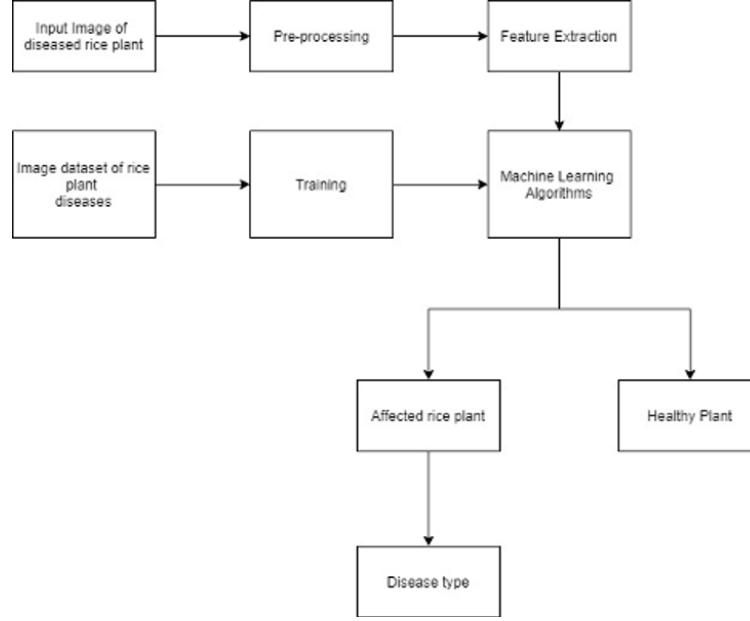


Figure 3.1: Proposed Approach

The main objective of the proposed methodology is to recognize the performance methods in identifying Hispa, Brown Spot and Leaf Blast diseases of rice plants apart from the healthy ones. For experiment, we are using the dataset of various samples of rice plants from each category of diseases. Then the images are passed through various steps: (A) Image Collection (B) Pre-processing Images, (C) Training Model (D) Detection of Affected Plants.

3.2.1 Image Collection

The images collected were already labelled in different folder. 10% of the images were separated from the data set for validation. It was ensured whether the labelled images were not false positives provided the given class. 4 folders representing 4 classes were used for the training phase.

3.2.2 Pre-processing Images

Different methods in our work required different image pre-processing steps. We have experimented the values and looked for the best fit that supported better accuracy for test results. Various methods have been employed to achieve better diagnosis of rice plant diseases as discussed.

Image Augmentation

Images used for training can come up of different size, orientation, rotation, shearing and levels of zoom etc. We do not want these factors to hamper the accuracy of our learning model. To solve this problem, we have employed image augmentation technique to increase the training set by generating multiple variants of images from a single one. The parameters we have used is show in Figure 3.2.

```
rescale=1./255,  
rotation_range=20,  
zoom_range=0.15,  
shear_range=0.15,  
width_shift_range=0.2,  
height_shift_range=0.2,  
horizontal_flip=True,  
vertical_flip=True,  
fill_mode="nearest",
```

Figure 3.2: Augmentation Configurations

Image Normalization

For better recognition of images by the learning model we had to employ another technique to increase the contrast of the images to create distinction between the rice plants and the background. It helps to create consistency between image samples. The normalization is done for $N \times N$ pixel image with the mean values of pixels $M1 \dots Mn$ provided the standard deviation $S1 \dots Sn$ such that:

$$Input[channel] = Input[channel] / Mean[channel] / StdDeviation[channel]$$

The following values for Mean and Standard Deviation were used for image normalization in our project as shown if Figure 3.3.

```
transforms.Compose([transforms.Resize(225),  
                    transforms.CenterCrop(224),  
                    transforms.ToTensor(),  
                    transforms.Normalize([0.485, 0.456, 0.406],  
                                         [0.229, 0.224, 0.225]))]
```

Figure 3.3: Image Normalization

Image segmentation: After preprocessing the images are converted to HSI model so that we can extract the hue, saturation and intensity of each pixel. Boundaries are used for 8-connectivity method.

3.2.3 Training Model

In image classification problems the traditional approaches have two steps in this phase (A) Image Segmentation (B) Feature Extraction. As for most part of the experimentation Neural Networks are used including several variants of CNN for the image classification problem, models are trained in this phase via NNs. These models are trained through series of convolutional layers, fully connected layers and finally the output layer. The specification of each network is different from each

other.

As for general CNN, it has 3 stages- (A) The convolutional layer scans the images pixels by pixels and creates a features map using filters. (B) Pooling happens after the feature scans, each layers of pooling reduces the dimensions of the layers keeping the crucial features and finally, (C) Fully-connected layers are formed when there is a probable class detection. There are usually stacks of different quantity of convolutional and pooling layers in different orders before reaching the FC layers. The process is shown in Figure 3.4.

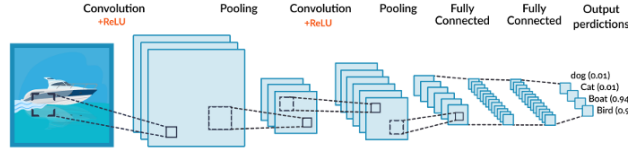


Figure 3.4: Training Model CNN

As a traditional machine learning method, SVM is used for image segmentation and feature classification. HOG (Histogram of Oriented Gradients) is used for feature detection as it describes the features as vectors provided the dimension of the image. HOS highlights the useful information and reduces the focus on the less informative ones. Then, the HOG described image can be fed to SVM to detect the model. This traditional approach is still very powerful in detecting features from images.

3.2.4 Detection of affected plants

Convolution method of detection use of filter has been used for a long time. Classic filters as Sobel (3x3) is used to detect edges effectively from sample images. As shown in figure 3.7 the convolution between the sample image and the filter G_x , G_y are used to detect horizontal and vertical edges respectively.

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \quad \text{and} \quad G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

But this filter works well with images with sharp edges but, not the sample image with fading edges (gradient edge). In this scenario Scharr filter works very well when convolution is applied with a sample image.

The numbers of these matrices are predefined and can be tested with various numbers. But, the goal of the CNN network is to find the best fit of numbers to these filters. The series of convolutional layers learns the numbers that suits the best for the training set. These layers are then pooled with various parameters until it reaches the FC layer. These parameters can be stride, padding etc. Thus, if these layers are known we can used prebuild layers of various variation of CNN. Out of the numerous variants this project uses prebuilt ResNet-152, VGG19, GoogLeNet CNNA variants and one custom CNN network to identify the diseased rice plants from sample images as shown in Figure 3.5.

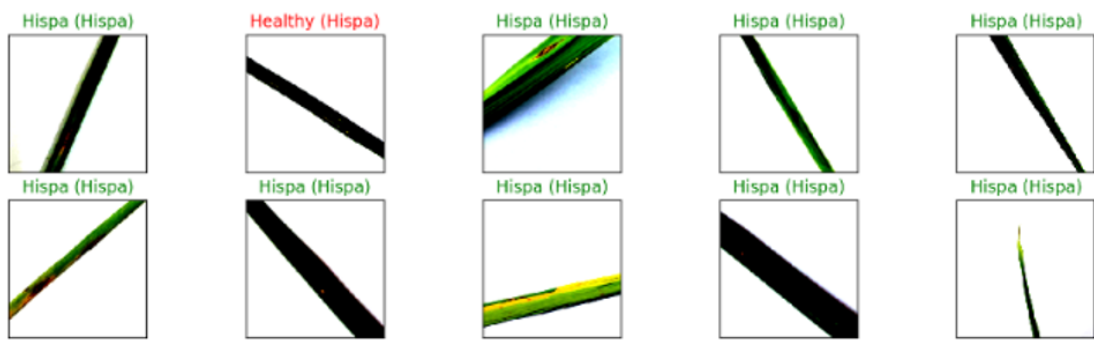


Figure 3.5: Identify the diseased rice plants

Chapter 4

Experimentation

4.1 Support Vector Machine (SVM)

Our initial approach to analyze the dataset and making a model was to use Support Vector Machine (SVM) as a classification algorithm. We converted the dataset images into 256x256 image size and extracted gray from RGB colors and extracted the histogram of oriented gradients (HOG) features out of it to run the SVM.

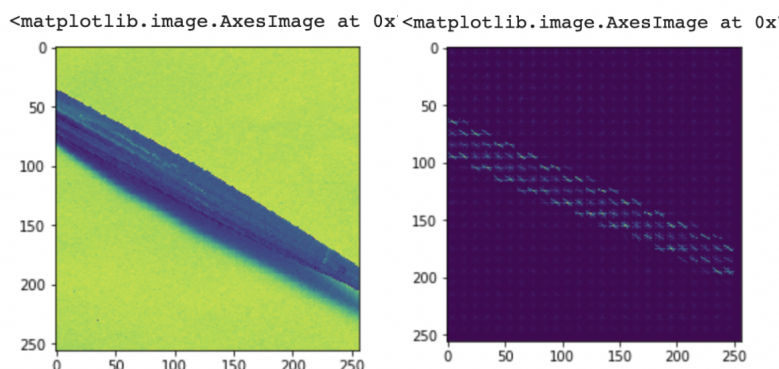


Figure 4.1: RGB to grayscale to HOG feature extraction

It extracted 69569280 vectors from matrix (3355, 20736). We used 80% of the datasets as a training dataset and ran the algorithm.

As you can see it gave us an accuracy of 76%. It has a Macro Avg of 38% and

Accuracy: 0.7630402384500745

	precision	recall	f1-score	support
0.0	0.76	1.00	0.87	512
1.0	0.00	0.00	0.00	159
accuracy			0.76	671
macro avg	0.38	0.50	0.43	671
weighted avg	0.58	0.76	0.66	671

Figure 4.2: Accuracy and Confusion Matrix SVM

Weighted Avg of 58%. Which is very low when we are classifying on 4 classes of Rice plant.

4.2 Convolutional Neural Network(CNN)

After getting poor result in SVM we wanted to use Neural network as a classifier, Convolutional Neural Network to be precise. We used PyTorch libraries, using same datasets. From this point we wanted to use image augmentation to better data preprocessing stage. We changed the rotation of the images, wide shifted the rage, zoomed the images and for some we flipped the images horizontally. Used this configuration to run on our model.

```
aug = ImageDataGenerator(  
    rotation_range=30, width_shift_range=0.15,  
    height_shift_range=0.15, shear_range=0.15,  
    zoom_range=0.2, horizontal_flip=True,  
    fill_mode="nearest")
```

Figure 4.3: using image augmentation CNN

```
model.add(Conv2D(32, (3, 3), padding="same", input_shape=input_shape))  
model.add(Activation("relu"))  
model.add(BatchNormalization(axis=chanDim))  
model.add(MaxPooling2D(pool_size=(3, 3)))  
model.add(Dropout(0.25))  
model.add(Conv2D(64, (3, 3), padding="same"))  
model.add(Activation("relu"))  
model.add(BatchNormalization(axis=chanDim))  
model.add(Conv2D(64, (3, 3), padding="same"))  
model.add(Activation("relu"))  
model.add(BatchNormalization(axis=chanDim))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
model.add(Dropout(0.25))  
model.add(Conv2D(128, (3, 3), padding="same"))  
model.add(Activation("relu"))  
model.add(BatchNormalization(axis=chanDim))  
model.add(Conv2D(128, (3, 3), padding="same"))  
model.add(Activation("relu"))  
model.add(BatchNormalization(axis=chanDim))  
model.add(MaxPooling2D(pool_size=(2, 2)))  
model.add(Dropout(0.25))  
model.add(Flatten())  
model.add(Dense(1024))  
model.add(Activation("relu"))  
model.add(BatchNormalization())  
model.add(Dropout(0.5))  
model.add(Dense(n_classes))  
model.add(Activation("softmax"))
```

Figure 4.4: CNN configuration

We used Rectified Linier Unit (ReLU) as a configuration for CNN and MaxPooling of pool size (2,2). Used Sequential model and 30 epochs to run the algorithm which we found suitable for running CNN.

Here we can find the accuracy is almost 75%. And we can see a validation accuracy

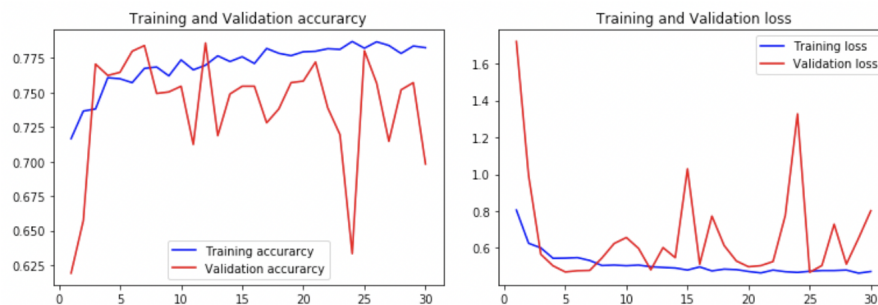


Figure 4.5: Training and validation accuracy; and loss graph CNN

peek at 25th epoch which is too perfect so ran for 30 epochs. But still 75% is lower than SVM.

4.3 GoogLeNet

After some research we found a better version of CNN developed by Google named GoogLeNet which is inspired by LeNet. Going for the same image augmentation operation for our datasets and here are the clipping of some images from datasets after augmentation.



Figure 4.6: Dataset images after image augmentation GoogLeNet

```
GoogLeNet{
  (conv1): BasicConv2d(
    (conv): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (maxpool1): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
  (conv2): BasicConv2d(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (conv3): BasicConv2d(
    (conv): Conv2d(64, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (maxpool2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
  (inception3a): Inception3a(
    (branch1): BasicConv2d(
      (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch2): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(192, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicConv2d(
        (conv): Conv2d(96, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (branch3): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(192, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(16, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicConv2d(
        (conv): Conv2d(16, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (branch4): Sequential(
      (0): MaxPool2d(kernel_size=3, stride=1, padding=1, dilation=1, ceil_mode=True)
      (1): BasicConv2d(
        (conv): Conv2d(192, 32, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(32, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
}
```

Figure 4.7: GoogLeNet Configuration

At this point we used high end GPUs from Google CoLab to fasten our training process. Running the algorithm using 50 epochs we got the algorithm to hit 91% accuracy.

Test Loss: 0.209819

Test Accuracy: 91% (307/336)

Figure 4.8: test accuracy of GoogLeNet

We hit the accuracy of 91% which is very high as we were continuously getting 75% accuracy and found our research to be accurate.

4.4 Visual Geometry Group (VGG19)

Through our research we found VGG19 to be one of the best algorithms for object recognition. So, wanted to try running the algorithm to find if we could get better results. To run this algorithm, we used Keras as a Library to get the implementation of the algorithm. Going through the same augmentation process we used this configuration for the VGG with 19 Layers.

```

VGG(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (1): ReLU(inplace=True)
    (2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU(inplace=True)
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (5): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU(inplace=True)
    (7): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU(inplace=True)
    (9): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (10): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (13): ReLU(inplace=True)
    (14): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (15): ReLU(inplace=True)
    (16): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (17): ReLU(inplace=True)
    (18): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (19): Conv2d(256, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (20): ReLU(inplace=True)
    (21): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (22): ReLU(inplace=True)
    (23): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (24): ReLU(inplace=True)
    (25): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (26): ReLU(inplace=True)
    (27): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    (28): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (29): ReLU(inplace=True)
    (30): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (31): ReLU(inplace=True)
    (32): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (33): ReLU(inplace=True)
    (34): Conv2d(512, 512, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (35): ReLU(inplace=True)
    (36): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(7, 7))
  (classifier): Sequential(
    (0): Linear(in_features=25088, out_features=4096, bias=True)
    (1): ReLU(inplace=True)
    (2): Dropout(p=0.5, inplace=False)
    (3): Linear(in_features=4096, out_features=4096, bias=True)
    (4): ReLU(inplace=True)
    (5): Dropout(p=0.5, inplace=False)
    (6): Linear(in_features=4096, out_features=1000, bias=True)
  )
)

```

Figure 4.9: VGG19 Configuration

```

EPOCHS = 20
INIT_LR = 1e-3
BS = 32

```

Figure 4.10: VGG19 Hyperparameters

For this approach we found 20 Epochs to be suit spot but found very fluctuating result. Here is the graph for VGG19. Which gave us an accuracy rate of 75% which seems odd for this algorithm.

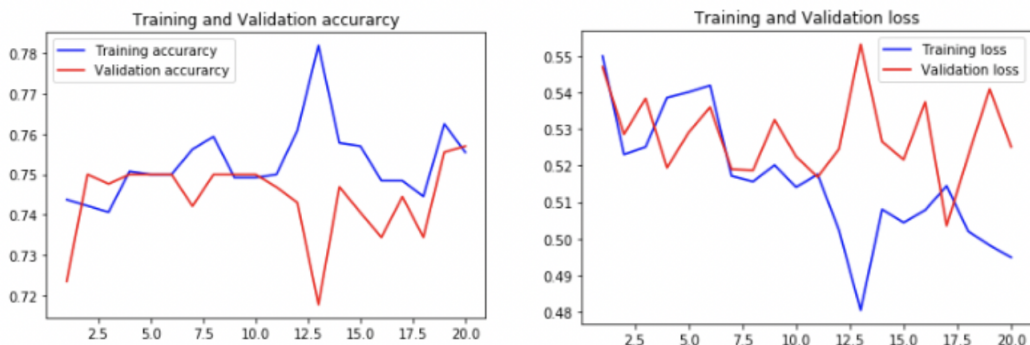


Figure 4.11: Training and Validation accuracy; and loss VGG19

4.5 Residual Network (ResNet-152)

After getting disappointing result from VGG19 and to get better result than GoogLeNet we wanted to use a different approach of an algorithm using deep Residual Network (ResNet). To be precise more modern version of ResNet which acts and performs as if it has 152 Convolutional Layers using layer skipping algorithm yet has less complexity than VGG19 which has 19 layers. Using PyTorch again as a library to get the implementation of ResNet152. Going through the same image augmentation we preprocessed our data and divided our data into training (80%), testing (10%) and validation (10%).

```
ResNet(
  (conv1): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
  (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (relu): ReLU(inplace=True)
  (maxpool): MaxPool2d(kernel_size=3, stride=2, padding=1, dilation=1, ceil_mode=False)
  (layer1): Sequential(
    (0): Bottleneck(
      (conv1): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
      (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (relu): ReLU(inplace=True)
      (downsample): Sequential(
        (0): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (1): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
  )
  (1): Bottleneck(
    (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
  (2): Bottleneck(
    (conv1): Conv2d(256, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn1): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (conv3): Conv2d(64, 256, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn3): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
    (relu): ReLU(inplace=True)
  )
)
```

Figure 4.12: Single layer of ResNet152

```
Validation loss decreased (0.33331 --> 0.23744). Saving model ...
Epoch: 36      Training Loss: 0.31290 Validation Loss: 0.34869
Epoch: 37      Training Loss: 0.28707 Validation Loss: 0.30826
Epoch: 38      Training Loss: 0.27310 Validation Loss: 0.45088
Epoch: 39      Training Loss: 0.26845 Validation Loss: 0.63444
Epoch: 40      Training Loss: 0.26908 Validation Loss: 0.39845
Epoch: 41      Training Loss: 0.29634 Validation Loss: 0.25491
Epoch: 42      Training Loss: 0.28249 Validation Loss: 0.38953
Epoch: 43      Training Loss: 0.25741 Validation Loss: 0.35588
Epoch: 44      Training Loss: 0.24248 Validation Loss: 0.38537
Epoch: 45      Training Loss: 0.26930 Validation Loss: 0.36976
Epoch: 46      Training Loss: 0.25265 Validation Loss: 0.26039
Epoch: 47      Training Loss: 0.23506 Validation Loss: 0.29833
Epoch: 48      Training Loss: 0.22954 Validation Loss: 0.20307
Validation loss decreased (0.23744 --> 0.20307). Saving model ...
Epoch: 49      Training Loss: 0.26843 Validation Loss: 0.35564
Epoch: 50      Training Loss: 0.23962 Validation Loss: 0.08683
Validation loss decreased (0.20307 --> 0.08683). Saving model ...
```

Figure 4.13: Model generating phase for ResNet152

We used 50 epochs and ran the algorithm on our dataset and hit the accuracy of 99% which is very high. Which is better than a human making difference between rice disease. At this point we got the algorithm we needed. Though it took ages to build the model it ran the checking in seconds.

```
Test Loss: 0.009012

Test Accuracy: 99% (335/336)
```

Figure 4.14: ResNet-152 Test Result

Chapter 5

Discussion

5.1 Qualitative Analysis

The number of classes used for training the models was only 4. Due to this, it not surprising that some learning models performed very well. The images were very clear and could be easily segmented using the HOG and deep learning models implemented. Because of this, SVM has matched the accuracy rate of that of the CNN models. As the number of classes increase, the CNN should outperform SVM in all aspects.

5.2 Comparative Analysis

A major issue we faced in this project is overfitting. Overfitting happens if models perform very well at classifying data from the training sample but performs poorly on testing sample. Thus, resulting high variance and low bias. One of the reasons that caused this problem is that the images used for training are fine tuned with less background noise. 80% of the training images were used for training, 10% for testing 10% for validation. All of pictures belonged to the same dataset with finely tuned images. This led to very high accuracy results with ResNet-152, GoogleNet, VGG and CNN. To overcome this problem, we have used separate images for validation and optimize image augmentation parameters. Even then, GoogleNet and ResNet-152 perform exceptionally well in predicting the outcome.

We calculated the accuracy of the data based on the following equation:

$$Accuracy = (TrueNegatives + TruePositives) / TotalImages \quad (5.1)$$

ResNet-152 and GoogLeNet are very deep convolutional neural networks. They both have top 5 records of a very low error rate with 3.57% and 6.67% respectively. The architecture of GoogLeNet consist of 22 layers of convolutional networks and with upto 4 million parameters. ResNet-152 has 152 convolutional layers with residual connections. They both tend to have human level performance in predicting outcome when trained correctly.

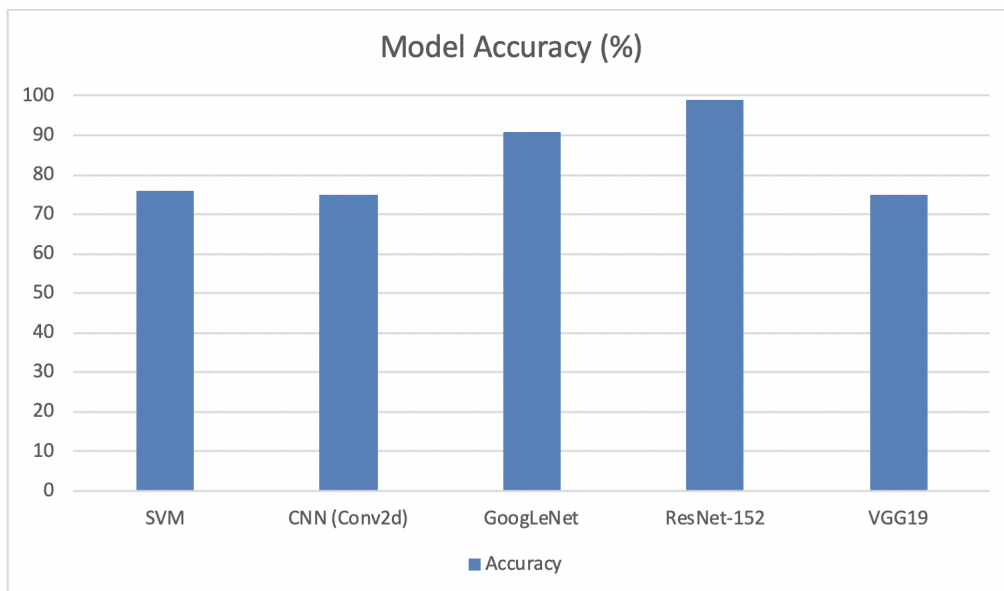


Figure 5.1: Model Accuracy Comparison

Chapter 6

Conclusion and Future Work

We as a nation depend on healthy rice crops as it is our staple food. We, now, do not face the lack of rice crops in our country. However, rice Blast Disease alone is the reason of the most damage, around 21.19%, in rice crops in Bangladesh [20]. Which is higher than the damage inflicted by rain in the monsoons, around 11.98%. Where Bangladesh have around 20 rice diseases, and each having their own prevention and remedies, classifying between them for a common farmer is very difficult and getting help to understand the issue and solve them is very costly for them. This is the only reason why we wanted to make a classifier to better detect rice disease and give the opportunity to the farmers of our country to better produce rice crops and reduce their damage. With 99% accuracy we are ensuring more than human level detection of rice disease in seconds.

Though we are showing 99% accuracy rate, but we had a very short dataset that had just 3.5 thousand images including healthy rice plants which is very short. If we had a large number of datasets, we could have built a better model and better detect the disease.

We are currently detecting only 3 diseases out of 20 rice diseases running in our country. Because of one, we had the dataset for only those diseases and two, shortage of time to collect datasets in large quantity for those varieties of diseases.

In theory if and when we have dataset of rest of the diseases, we can run the dataset through the algorithm we found perfect for detecting rice disease. However, it is not true for now. We need to collect more dataset of rice diseases to run through our algorithm and make our model ready for detecting those diseases.

We wanted to make a simple app that can be run on only camera phone with internet which can be used by the common farmers of our country to take a picture of their suspicious rice plant and show them their necessary process that they need to take to prevent or resolve the disease. Which is not done because of the shortage of time. In theory we can make a python REST API around the model to make the backend of the app. However, it is not done yet. And we couldn't build the app yet. These are the work left for us or anyone taking interest in our approach to do.

Bibliography

- [1] J. S. Santanu Phadikar, “Rice disease identification using pattern recognition techniques”, *2008 11th International Conference on Computer and Information Technology*, pp. 420–423, 2008. DOI: 10.1109/ICCITECHN.2008.4803079. [Online]. Available: <https://doi.org/10.1109/ICCITECHN.2008.4803079>.
- [2] L. Liu and G. Zhou, “Identification method of rice leaf blast using multilayer perception neural network”, vol. 25, pp. 213–217, Oct. 2009. DOI: 10.3969/j.issn.1002-6819.2009.z2.041.
- [3] I. Arel, D. Rose, and T. Karnowski, “Deep machine learning - a new frontier in artificial intelligence research [research frontier].”, *IEEE Comp. Int. Mag.*, vol. 5, pp. 13–18, Jan. 2010.
- [4] M. R. Badnakhe and P. Deshmukh, “An application of k-means clustering and artificial intelligence in pattern recognition for crop diseases”, 2011.
- [5] Y. Bengio, G. Guyon, V. Dror, G. Lemaire, D. Taylor, and D. Silver, “Deep learning of representations for unsupervised and transfer learning”, vol. 7, Jan. 2011.
- [6] S. B. Patil and S. K. Bodhe, “Leaf disease severity measurement using image processing”, *International Journal of Engineering and Technology*, vol. 3, no. 5, pp. 297–301, 2011.
- [7] P. Chaudhary, A. K. Chaudhari, A. Cheeran, and S. Godara, “Color transform based approach for disease spot detection on plant leaf”, *International journal of computer science and telecommunications*, vol. 3, no. 6, pp. 65–70, 2012.
- [8] A. Krizhevsky, I. Sutskever, and G. Hinton, “Imagenet classification with deep convolutional neural networks”, *Neural Information Processing Systems*, vol. 25, Jan. 2012. DOI: 10.1145/3065386.
- [9] S. Naikwadi and N. Amoda, “Advances in image processing for detection of plant diseases”, *International Journal of Application or Innovation in Engineering & Management*, vol. 2, no. 11, 2013.
- [10] S. Phadikar, J. Sil, and A. Das, “Rice diseases classification using feature selection and rule generation techniques”, *Computers and Electronics in Agriculture*, vol. 90, pp. 76–85, Jan. 2013. DOI: 10.1016/j.compag.2012.11.001. [Online]. Available: https://www.researchgate.net/publication/257018762_Rice_diseases_classification_using_feature_selection_and_rule_generation_techniques.
- [11] M. P. Prof. Sanjay B. Dhaygude, “Agricultural plant leaf disease detection using image processing”, *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering*, vol. 2, Jan. 2013. eprint: https://www.ijareeie.com/upload/january/5_Agricultural%20plant.pdf.

- [12] A. N. Rathod, B. Tanawal, and V. Shah, “Image processing techniques for detection of leaf disease”, *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 3, no. 11, 2013.
- [13] L. P. Gianessi, “Importance of pesticides for growing rice in south and south east asia”, 2014.
- [14] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, “Learning and transferring mid-level image representations using convolutional neural networks”, *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Jun. 2014. DOI: 10.1109/CVPR.2014.222.
- [15] S. Bell, C. Zitnick, K. Bala, and R. Girshick, “Inside-outside net: Detecting objects in context with skip pooling and recurrent neural networks”, Dec. 2015.
- [16] Z. Zhu, X. Wang, S. Bai, C. Yao, and X. Bai, “Deep learning representation using autoencoder for 3d shape retrieval”, Jan. 2015. DOI: 10.1016/j.neucom.2015.08.127.
- [17] S. P. Katta Jagan Mohan Mahesh Balasubramanian, “Detection and recognition of diseases from paddy plant leaf images”, *International Journal of Computer Applications*, vol. 144, pp. 34–41, 2016. DOI: 10.5120/ijca2016910505. [Online]. Available: <https://doi.org/10.5120/ijca2016910505>.
- [18] H.-c. Shin, H. Roth, M. Gao, L. Lu, Z. Xu, I. Nogues, J. Yao, D. Mollura, and R. Summers, “Deep convolutional neural networks for computer-aided detection: Cnn architectures, dataset characteristics and transfer learning”, *IEEE Transactions on Medical Imaging*, vol. 35, Feb. 2016. DOI: 10.1109/TMI.2016.2528162.
- [19] A. M. Vijai Singh, “Rice disease identification using pattern recognition techniques”, *Information Processing in Agriculture*, vol. 4, no. 1, pp. 41–49, 2016. DOI: 10.1016/j.inpa.2016.10.005. [Online]. Available: <https://doi.org/10.1016/j.inpa.2016.10.005>.
- [20] M. Hossain, M. Ali, and D. Hossain, “Occurrence of blast disease in rice in bangladesh”, *American Journal of Agricultural Science*, vol. 4, pp. 74–80, Aug. 2017.
- [21] Y. Lu, S. Yi, N. Zeng, Y. Liu, and Y. Zhang, “Identification of rice diseases using deep convolutional neural networks”, *Neurocomputing*, vol. 267, Jul. 2017. DOI: 10.1016/j.neucom.2017.06.023. [Online]. Available: https://www.researchgate.net/publication/318082906_Identification_of_Rice_Diseases_using_Deep_Convolutional_Neural_Networks.
- [22] M. Singh, S. Chetia, and M. Singh, “Detection and classification of plant leaf diseases in image processing using matlab”, *International Journal of Life sciences Research*, vol. 5, pp. 120–124, Dec. 2017. [Online]. Available: https://www.researchgate.net/publication/321587357_Detection_and_Classification_of_Plant_Leaf_Diseases_in_Image_Processing_using_MATLAB.

- [23] P. U., “Smart paddy crop disease identification and management using deep convolution neural network and svm classifier”, *International Journal of Pure and Applied Mathematics*, vol. 118, no. 15, pp. 255–264, Feb. 2018. [Online]. Available: https://www.researchgate.net/publication/323392707_Smart_Paddy_Crop_Disease_Identification_and_Management_Using_Deep_Convolution_Neural_Network_and_SVM_Classifier.
- [24] M. R. KOMAL BASHIR and M. BARI, “Detection and classification of rice diseases: An automated approach using textural features”, *Mehran University Research Journal of Engineering Technology*, vol. 38, no. 1, 2019. DOI: 10.22581/muet1982.1901.20. eprint: <http://oaji.net/articles/2019/2712-1546616651.pdf>.
- [25] N. Mangla, P. Raj, and S. Hegde, “Paddy leaf disease detection using image processing and machine learning”, vol. 7, pp. 2321–5526, Feb. 2019. DOI: 10.17148/IJIREEEICE.2019.7220. eprint: <https://ijireeice.com/wp-content/uploads/2019/03/IJIREEEICE.2019.7220.pdf>. [Online]. Available: <https://doi.org/10.1016/j.inpa.2019.09.002>.