Deep Learning Based Algorithm to Predict Plant Diseases: A Case Study with Rice Plant Disease Prediction

A thesis
Submitted in partial fulfillment of the requirements for the Degree of
Bachelor of Science in Computer Science and Engineering

Submitted by

Asifur Rahim	**180104109**
Rumana Akter	**180104106**
Ashif Reza	**180104107**
Tauhidur Rahman	**180104148**

Supervised by

Prof. Dr. Mohammad Shafiul Alam

Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dhaka, Bangladesh

December 2022

CANDIDATES' DECLARATION

We, hereby, declare that the thesis presented in this report is the outcome of the investigation performed by us under the supervision of Prof. Dr. Mohammad Shafiul Alam, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, Dhaka, Bangladesh. The work was spread over two final year courses, CSE4100: Project and Thesis I and CSE4250: Project and Thesis II, in accordance with the course curriculum of the Department for the Bachelor of Science in Computer Science and Engineering program.

It is also declared that neither this thesis nor any part thereof has been submitted anywhere else for the award of any degree, diploma or other qualifications.

Asifur Rahim
180104109

Rumana Akter
180104106

Ashif Reza
180104107

Tauhidur Rahman
180104148

CERTIFICATION

This thesis titled, **“Deep Learning Based Algorithm to Predict Plant Diseases: A Case Study with Rice Plant Disease Prediction”**, submitted by the group as mentioned below has been accepted as satisfactory in partial fulfillment of the requirements for the degree B.Sc. in Computer Science and Engineering in December 2022.

Group Members:

Asifur Rahim	**180104109**
Rumana Akter	**180104106**
Ashif Reza	**180104107**
Tauhidur Rahman	**180104148**

Prof. Dr. Mohammad Shafiul Alam
Professor & Supervisor
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

Dr. Md. Shahriar Mahbub
Professor & Head
Department of Computer Science and Engineering
Ahsanullah University of Science and Technology

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our honourable thesis supervisor Dr Md. Shafiul Alam, Professor, Department of Computer Science and Engineering, Ahsanullah University of Science and Technology, for his patient guidance, enthusiastic encouragement and constructive critiques of this thesis work. We would also like to thank all our respected teachers, lab assistants, and friends for their moral support and helpful discussions. Finally, we would like to thank our parents for their constant support and encouragement throughout our study.

Dhaka
December 2022

Asifur Rahim

Rumana Akter

Ashif Reza

Tauhidur Rahman

ABSTRACT

One of the most popular staple meals consumed worldwide is Rice. Bangladesh, one of the top ten producers and consumers of rice in the world, is largely dependent on rice for both its economy and its dietary needs. To ensure the healthy and proper growth of the Rice plants it is essential to detect any disease in time and prior to applying the required treatment to the affected plants. The purpose of this research is to develop a simple method for identifying diseases that harm rice plants. This research is conducted on a dataset with 5932 images of four different kind of Rice plant diseases. This dataset is a unique dataset which is not been previously used in any other research. Moreover, most of the previous existing works related to Rice plant diseases in Bangladesh have been done using smaller datasets and the performance is not up to the mark. A unique model using CNN is proposed which provides satisfactory results and outperforms most of the previous works. Deep learning and Machine learning models are used for comparison purposes. Transfer Learning techniques are used to get higher prediction accuracy. Pre-trained models (vgg-16, Inception-V3, vgg-19, ResNet-50) is used for feature extraction by modifying the input and output layers based on the disease classes of the dataset and a Convolutional Neural Network (CNN) is used for disease classification. By flipping, zooming, and resizing the dataset, augmentation is done which is providing better training and test accuracy. An accuracy of as high as **97%** is achieved by the custom model. Also, an even higher accuracy of **98%** test accuracy is achieved by VGG-19 model which has given best performance among all the trained models. From traditional Machines learning models KNN, SVM, AdaBoost, Decision Tree and Random Forest are used. Here for feature extraction, the Gabor filter followed by the Sobel filter is used for better texture analysis and edge detection. Random forest gives the highest accuracy among all machine learning classifiers which is of **96.6%**. This model outperforms most of the previous existing works.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

Rice being the most consumed food product in Bangladesh, needs to be maintained and mass-produced for the massive population. Rice plant diseases are a common problem that needs to be combated to make sure Rice production does not get hindered at any point in the process. This paper deals with the identification and classification of Rice plant diseases based on image datasets. Four of the most common and major Rice plant diseases have been considered for this paper [1]. They are Bacterial Blight, Blast, Brown Spot, and Rice Tungro. The dataset consists of 5932 images of these 4 diseases [2].

1.2 Rice Plant Diseases

1.2.1 Bacterial Blight

Xanthomonas Oryzae [32] pv. Oryzae is the culprit behind bacterial blight. Seedlings wilt as a result, and leaves turn yellow and dry out. The disease is most prone to spread to areas with weeds and harmful plant residue. It can appear in both tropical and temperate settings, especially in lowland areas that receive irrigation and rainfall. The disease generally prefers conditions with relative humidity above 70% [32] and temperatures between 25 and 34 °C [32]. Strong winds and persistent downpours frequently cause it, which makes it easy for the disease-causing bacteria to spread through oozing droplets on lesions of afflicted plants. Under conditions of strong nitrogen fertilization, bacterial blight in sensitive rice types can be severe. One of the most dangerous illnesses that can affect rice is bacterial blight. The yield loss increases with the onset of the disease. When vulnerable types are grown in conditions conducive to the disease, yield loss from bacterial blight can reach 70%.

Bacterial blight does not affect yield when plants are infected at the booting stage, but it does cause poor grain quality and a high percentage of damaged kernels. Yellowing or withering of the leaves or seedlings (also called Kresek [32]) is a disease symptom. Infected seedling leaves curl up and turn a grayish-green color. The illness spreads, causing the leaves to wilt and turn yellow to straw-colored, which causes the entire seedling to dry out and perish. Kresek damage to seedlings can occasionally be mistaken for early rice stem borer damage.

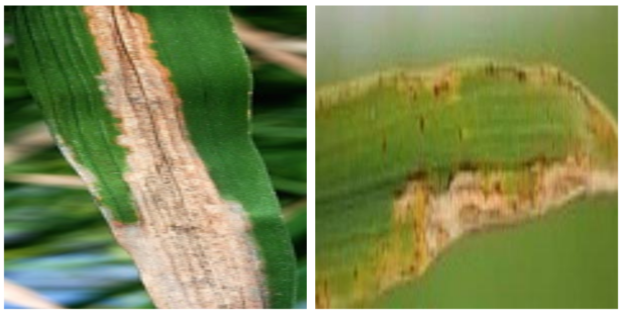


Figure 1.1: Bacterial Blight [2]

1.2.2 Blast

The fungus Magnaporthe Oryzae [33] is responsible for the blast. A rice plant's leaf, collar, node, neck, sections of the panicle, and occasionally the leaf sheath are all susceptible to it. Anywhere there are blast spores, the blast can happen. Low soil moisture, frequent and protracted rain showers and chilly daytime temperatures are all factors that contribute to its occurrence. Large day-night temperature fluctuations that result in dew forming on leaves and generally lower temperatures in upland rice enhance the growth of the disease. Rice can develop blast at every stage of growth. Up until the tillering stage, a leaf blast infection can harm seedlings or plants. A severe leaf blast infection limits leaf area for grain fill at later growth stages, lowering grain output. In cases of severe infection, leaf blasts can destroy rice plants at the seedling stage and reduce output. Brown spot lesions tend to be more rounded, brown in color, and have a yellow halo surrounding the lesion, whereas leaf blast lesions are typically elongated and pointed at each end.

1.2.3 Brown Spot

The Coleoptile, leaves, Leaf Sheath, panicle branches, glumes, and spikelets are all affected by the fungus known as Brown Spot. The multiple large blotches on the leaves that can kill the entire leaf are the most noticeable harm. Unfilled grains or speckled or discolored seeds develop when the seed becomes infected. High relative humidity (86 to 100%) [34]

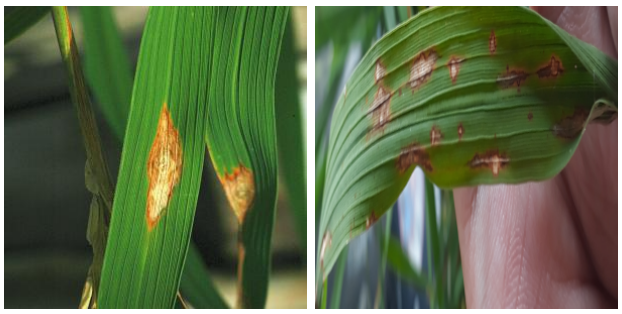


Figure 1.2: Blast [2]

and temperatures between 16 and 36 °C are conducive to the disease's development. It frequently occurs in soil that is neither wet nor nourished, nor in soil that builds up harmful compounds. The leaves need to be damp for 8 to 24 hours before an infection can start. Brown spots can appear at any stage of the crop's development, although the infection is most dangerous during maximum tillering and the crop's ripening stages. Small, round, yellow-brown, or brown lesions that may encircle the coleoptile and deform primary and secondary leaves are present on infected seedlings. On the leaves, lesions can be seen beginning at the tillering stage. They start out small, round, dark brown to purple-brown in color. Fully formed lesions have an oval or circular shape, a light brown to gray center, and a reddish brown edge brought on by the poison the fungi generate. Losses due to brown spots affect both quantity and quality. In South and Southeast Asia, the disease reduces lowland rice yields by an average of 5%. A severely infected field may see a yield loss of up to 45% [34]. Heavy infection in the seeds causes seedling blight, which kills 10–58% of the seedlings [34]. Additionally, it lowers the kernel weight and has an impact on the quality and quantity of grains produced per panicle. According to historical accounts, the Brown spot was the main cause of the Great Bengal Famine in 1943 [34].

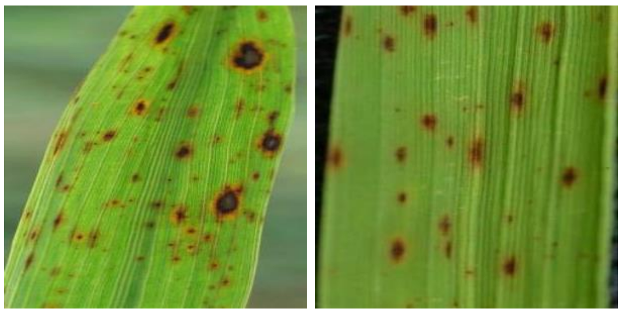


Figure 1.3: Brown Spot [2]

1.2.4 Rice Tungro

Rice Two viruses that combine to cause the Tungro [35] are spread by leafhoppers. It results in infertile or partially filled grains, slowed development, reduced tiller numbers, and discolored leaves. As well as some wild rice relatives and other grassy weeds typically seen in rice paddies, Tungro also infects cultivated rice. Leafhoppers that eat Tungro-infected plants spread the viruses that cause Tungro disease from one plant to another. The green Leafhopper is the most effective vector. By feeding on any portion of the infected plant, even for a brief period of time, Leafhoppers can pick up the viruses. The viruses can then be immediately spread to other plants within 5–7 days. Unless the leafhopper eats another contaminated plant and re-acquires the viruses, the viruses do not stay in its body. Every stage of the rice plant's growth is susceptible to Tungro infection. When it is in the vegetative stage, it is most frequently observed. Tillering stage [35] is when plants are most sensitive. In South and Southeast Asia, Tungro is one of the most harmful and devastating diseases of rice. In extreme circumstances, Tungro sensitive types that are infected at an early growth stage could have a yield loss of up to 100% [35]. When Tungro first appears in the field, it quickly spreads to new rice plants. Young rice plants are the preferred food of Leafhopper vectors [35]. Additionally, younger diseased plants are a more effective source of Tungro virus acquisition.

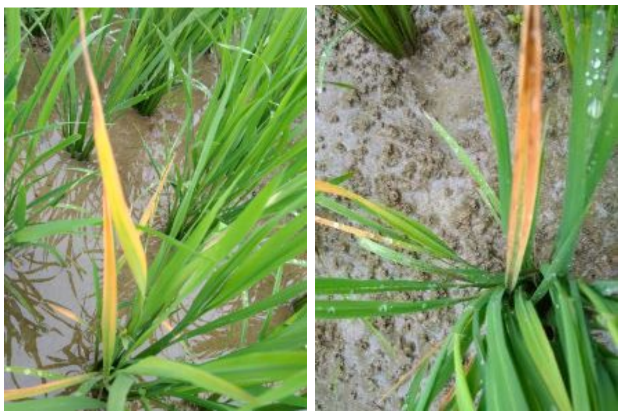


Figure 1.4: Rice Tungro [2]

1.3 Motivation

Bangladesh is an agricultural country. Rice is a major food for the people of our country. Rice cultivation provides about 48%citeb35 of rural development. Around 75%citeb35 of the overall harvested area and above 80% of the overall irrigated zone are cultivated through rice plants. Consequently, Rice plays an important role for the people of Bangladesh. But the quality, as well as quantity of rice production, may be decreased because of Rice plant Diseases. As having the disease in plants is quite natural. If proper step is not taken in this regard then it causes serious effects on Rice plants. Therefore, it is the major task to identify Rice plant Disease in the early stages. Early detection of Rice plant diseases is very crucial for the crop protection system of our country. Most of the farmers of our country use their own experience to detect diseases, farmers use pesticides in excessive quantities which can not help in the prevention of disease, but can have a malignant effect on plants.Thus their misclassification creates bad impact on rice cultivation. So, they need advice from rice disease specialists. In remote or rural areas, rice disease specialists are not able to give quick remedies or advice to the farmers in the right time and they also require expensive equipment and a large amount of time for manually identifying and classifying rice diseases. Moreover, traditional visual observation methods are mostly inaccurate. Besides that laboratory testing requires time and can be expensive. The research that has been done on Bangladeshi Rice plant disease is not sufficient. Nowadays smartphones are available. If the farmers can somehow afford a smartphone it will be easier for them to use image-processing methods to detect rice plant diseases.

Chapter 2

Literature Review

2.1 Overview

Numerous Work related to rice plant disease classification has been carried out over the years. But they have had their own methods and approaches. A selection of papers related to our study has been discussed in this chapter that is the most relevant and important to the research.

2.2 Related Works

Md Jahid Hasan et al. [3] proposed a paper for rice disease identification and classification by integrating a support vector machine with Deep Convolutional Neural Network.

A hybrid network integrating Deep CNN with SVM for classification. Transfer Learning techniques have been used in order to improve the proposed model. Features are extracted using D-CNN and SVM classifier is trained with the features.

The paper uses a complex AI system that integrates SVM and D-CNN for identifying rice plant diseases. The proposed model achieves an accuracy of 97.5%. However, the dataset used for the training is a small collection of about 1080 images and for the purpose of testing, it was 270 images. The proposed methodology seems to be best suited for a dataset with a smaller number of images.

Tejas Tawde et al. [4] proposed a paper for rice plant disease detection and classification using Convolutional Neural Network (CNN).

Focuses to distinguish different methods based on classifier used. Gives insight of the different techniques used for the identification of Rice plant Diseases. A hardware prototype and model using CNN is proposed.

The proposed model has achieved an accuracy of 96% using Deep Learning models of CNN. However it is using a small dataset where the latest CNN models have not been implemented to yeild better results.

kawcher Ahmed et al. [5] proposed a model for rice leaf disease detection using machine learning techniques.

They applied four supervised classification algorithm to detect a total of three diseases. Disease detection model was developed using CNN. Affected parts were separated using K-means clustering and SVM. For extracting the features of an image, HOG(Histogram of Oriented Gradients) was used.

They achieved 96.77% accuracy when the dataset was trained using KNN, Logistic Regression, j48, and Naive Bayes. However, only 400+ images were used for the training process. Three plant diseases were detected and no deep-learning models were used. Image classification gives better accuracy with faster performance with deep learning models. So machine learning models used in this proposed model generally give a slower performance.

S Ramesh et al. [6] proposed a model for Rice Blast Disease detection and classification using machine learning algorithm.

An appropriate number of features were extracted. Images have been classified using ANN. K-Means Clustering is used for Image Segmentation.

An impressive 99% accuracy has been achieved here for the blast-infected images. A near-perfect accuracy has been achieved here for other normal images. But, it is only a binary classification of only one disease. It shows 90% accuracy for infected images and 86% for healthy images. The model shows a certain amount of over-fitting tendencies since the accuracy is very high for only 300 leaf samples used as the dataset.

Anam Islam et al. [7] proposed a model for rice leaf disease recognition using Local Threshold Based segmentation and Deep CNN.

Local Threshold method is used for image segmentation. Three CNN architecture models VGG16, ResNet50, DenseNet121 are used for classification. CNN is trained with the segmented images

The dataset used for this work was created by the authors themselves. The segmented dataset was used for the training process for the model. However, a small dataset of only 786 images has been used with a testing accuracy of 78.84%.

S Ramesh et al. [8] proposed a model for the application of machine learning in the detection of blast disease in South Indian rice crops.

Detected Rice Blast disease using KNN and ANN classification techniques. K-means Clustering is used for image segmentation. The extracted features are applied to a classifier to determine whether it is an image of an infected crop or not.

The ANN classifier used in this model gave 99% accuracy for normal images and 100% for blast-infected images. But it only detects a single type of disease (Rice Blast). Furthermore, it has a small image dataset of 451 images. The k-means Clustering segmentation method is used whereas there are other methods that give a more accurate result.

Md. Sazzadul Islam et al. [9] proposed a model of the Identification of Various Rice Plant Diseases Using Optimized Convolutional Neural Networks.

An optimized CNN architecture based on depthwise separable convolutions. Along with the proposed CNN model different lightweight state-of-the-art CNN architectures have been used and results were analyzed. 1677 images were used which was further augmented to increase the size of the dataset.

The model uses a relatively small parameter size of 2.4 million. The depth-wise convolution is known to reduce computational cost and parameter size. Here around 12 classes of diseases are being used for the training images of the dataset. However, a large number of classes is causing bias and misclassifications among some of the diseases with similar characteristics.

Vimal K. Shrivastava et al. [10] proposed a model for Rice Plant Disease Classification Using Transfer Learning Of Deep Convolutional Neural Networks.

Deep Convolutional Neural Network (CNN) as feature extractor Support Vector Machine (SVM) as a classifier. AlexNet deep CNN pre-trained on a large ImageNet dataset was used for feature extraction.

Here the accuracy achieved was around 91.37%. The AlexNet model pre-trained on a large ImageNet dataset of 1.2 Million Images 1000 classes were used in this case. But a relatively small dataset of 619 images has been used in this case for identifying 3 types of rice diseases. The paper also mentioned the unavailability of labeled rice disease images.

Chapter 3

Background Study

3.1 Feature Extraction

3.1.1 Gabor Filter

The Gabor filter [36] is an application of the Gabor transform, which combines a Gaussian window and a short-term Fourier transformation for analysis in the spatial domain. The texture information of the image is incorporated into the distortion information of content-adaptive image steganography. There creates a textural anomaly [36] in an image when embedded. The result of the 2D Gabor filtering on the Gabor signal can be used to identify this textural anomaly. Due to its spatial selectivity and direction, the two-dimensional Gabor filter represents the texture information. Because of its spatial representation, a filter represents texture information.

$$g_{\lambda,\theta,\varphi,\sigma,\gamma}(x,y) = exp\left(-\frac{(x^2+\gamma^2 y^2)}{2\sigma^2}\right)cos(2\pi\frac{x}{\lambda}+\varphi) \tag{3.1}$$

where, the value of x and y denote the following

$$x = acos\theta + bsin\theta \tag{3.2}$$

$$y = -acos\theta + bsin\theta \tag{3.3}$$

λ – Wavelength of Gabor function cosine factor.

θ – Orientation of Gabor function normal to the parallel stripes.
φ – Phase offset of the Gabor function cosine factor.
σ – Standard deviation sigma of Gaussian factor.
γ – Ellipticity of the Gaussian factor.

Based on different orientation parameters, different types of filters may be generated

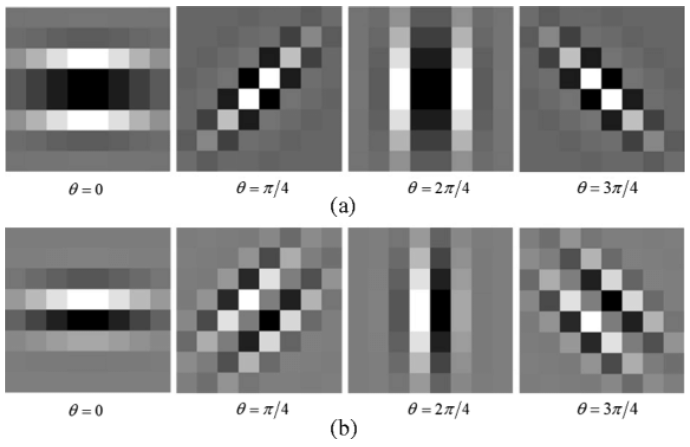


Figure 3.1: Different Orientations for Gabor Filter [37]

3.1.2 Sobel Filter

Two 3 x 3 kernels are used in the Sobel filter [38]. There are two changes in the horizontal and vertical directions, respectively. To compute the approximate derivatives, the two kernels are convolved with the original image. The calculations are as follows if Gx and Gy are two images that, respectively, include the horizontal and vertical derivative approximations:

$$G_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} * A$$

$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} * A$$

Where A is the original source image. The x coordinate is defined as an increase in the right direction and the y coordinate is defined as an increase in the down direction. To compute Gx and Gy we move the appropriate kernel (window) over the input image, computing the value for one pixel and then shifting one pixel to the right. Once the end of the row is reached, we move down to the beginning of the next row.

At each pixel in the image, the gradient approximations given by Gx and Gy are combined to give the gradient magnitude, using:

$$G = \sqrt{G_x^2 + G_y^2} \tag{3.4}$$

The gradient's direction is calculated using:

$$\theta = arctan\left(\frac{G_y}{G_x}\right) \tag{3.5}$$

A θ value of 0 would indicate a vertical edge that is darker on the left side.

3.2 Traditional Machine Learning Classifiers

A machine learning algorithm is a method by which the AI system conducts its task, generally predicting output values from given input data. The two main processes of machine learning algorithms are classification and regression. Machine learning (ML) algorithms are broadly categorized as either supervised or unsupervised. Supervised learning algorithms have both input data and desired output data provided for them through labeling, while unsupervised algorithms work with data that is neither classified nor labeled. An unsupervised algorithm might, for example, group unsorted data according to similarities and differences.Some of the common traditional ML Algorithms are : Adaboost, Random Forest, Decision Tree, Support Vector Machine etc.

3.2.1 KNN

KNN [23]is one of the simplest classification algorithms available for supervised learning. The idea is to search for the closest match of the test data in feature space. Typically the object is classified based on the labels of its k nearest neighbors by majority vote. If k is 1, the object is simply classified as the class of the object nearest to it. When there are only two classes, k must be an odd integer. However, there can still be times when k is an odd integer when performing multi-class classification. After we convert each image to a vector of fixed length with real numbers, we used the most common distance function for KNN which is Euclidean distance between two points x and y and it can be written as:

$$d(x,y) = |x-y| = \sqrt{(x-y).(x-y)} = \sqrt{(\sum_{i=1}^{m}(x_i - y_i)^2)} \tag{3.6}$$

3.2.2 Support Vector Machine (SVM)

SVM [22] are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.
Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. An SVM model is a representation of the examples as points in space, mapped so that the examples of the separate categories are divided by a clear gap that is as wide as possible. New examples are then mapped into that same space and predicted to belong to a category based on which side of the gap they fall.
In addition to performing linear classification, SVM can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

3.2.3 AdaBoost

AdaBoost [24], also known as Adaptive Boosting, is a machine learning method used in an ensemble setting. Decision trees with one level, or Decision trees with only one split, are the most popular algorithm used with AdaBoost. Another name for these trees is Decision Stumps. This algorithm creates a model while assigning each data piece an equal weight. Then, it gives points that were incorrectly categorized as larger weights. The next model now gives more weight to all the points with higher weights. If a low error is not reported, it will continue to train models.

3.2.4 Random Forest

Random forest [21] is a type of supervised machine learning algorithm based on ensemble learning. Ensemble learning is a type of learning where we join different types of algorithms or the same algorithm multiple times to form a more powerful prediction model. The Random Forest algorithm combines multiple algorithms of the same type i.e. multiple decision trees. Random decision forests correct for decision trees' habit of over-fitting to their training set. It operates by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

3.2.5 Decision Tree

A decision tree [25] is a tool that can be used in many different contexts. Both classification and regression issues can be solved using decision trees. The term itself implies that it displays the predictions that come from a sequence of feature-based splits using a flowchart that resembles a tree structure. The decision is made by the leaves at the end which follow the root node.

3.3 Deep Learning

Deep Learning is a subset of machine learning algorithms that is very good at recognizing patterns but typically requires a large number of data. Deep learning excels in recognizing objects in images as it is implemented using three or more layers of artificial neural networks where each layer is responsible for extracting one or more features of the image.

3.3.1 Convolutional Neural Network (CNN)

Deep Learning has established itself as a very potent tool over the last few decades due to its capacity for handling massive amounts of data. Hidden layer technology is much more popular than conventional methods, particularly for pattern recognition. Convolutional Neural Networks are among the most widely used deep neural networks.
ANN takes inputs and transforms them by putting them through a series of hidden layers. Every layer is made up of a set of neurons, where each layer is fully connected to all neurons of the immediate previous layer. Finally, the output layer is also fully connected which represents the predictions. It is a bit different in Convolutional neural networks and ANN. Alike ANN the neurons of a layer of a Convolutional neural network does not connect to all the neurons in the next layer. Only a small region of a layer connects to the next layer. The layers of a Convolutional neural network are organized in 3- dimensions, width, height, and depth. The output in the Convolutional neural network will be reduced to a single vector of probability scores organized along the depth dimension.
To classify an object, CNN takes an input image, processes it, and classifies it under certain categories (e.g., cat, dog). Each input image will pass through a series of convolutional layers with filters (Kernels), Pooling, fully connected layer (FC) and apply activation functions

to classify an object using probabilistic values between 0 and 1. There are two components of CNN. They are feature extraction/hidden layers and the classification part.

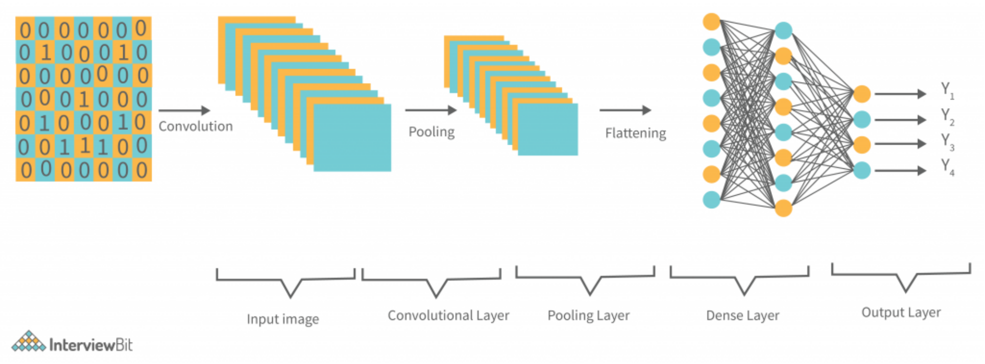


Figure 3.2: Convolutional Neural Network [26]

- **Feature extraction:** CNN performs a series of convolutions, and pooling operations to extract features from the input.
- **Classification:** In this part, the features from the feature extraction part will go through a fully connected layer. From the fully connected layer, we will finally get an output.

Convolutional Layers

Convolutional layers are the building block of CNN. In the first layer, where feature extraction of an input image is performed. Convolutional layers preserve the relationship between pixels by learning image features using small squares of input data. This is a mathematical operation that takes an image matrix and a filter (kernel) matrix as input.

- **Input:**An image matrix (volume) of dimension ($h \times w \times d$), a filter matrix ($f\,h \times f\,w \times d$)
- **Output:** A volume dimension (h - $f\,h + 1$)(w - $f\,w + 1$) $\times$ 1
  Considering a 5x5 image with pixel values of only 0 and 1 and a filter matrix 3x3 which is shown in figure 3.3 :

  Now convolution will be performed by placing the filter matrix at every position of the input image and finally we will get a Feature map. When the filter matrix is placed at every position of the input image given in Figure 3.4(Input$\times$Filter), matrix multiplication is performed, and sum up the result onto the feature map. The result after performing convolution is shown in Figure 3.4(Feature Map).

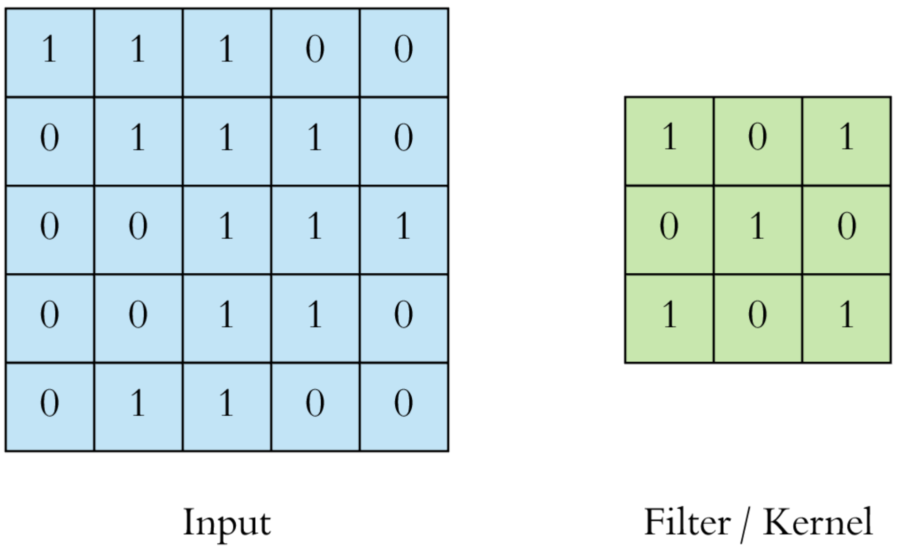


Figure 3.3: An Input image and 3×3 Kernel [27]

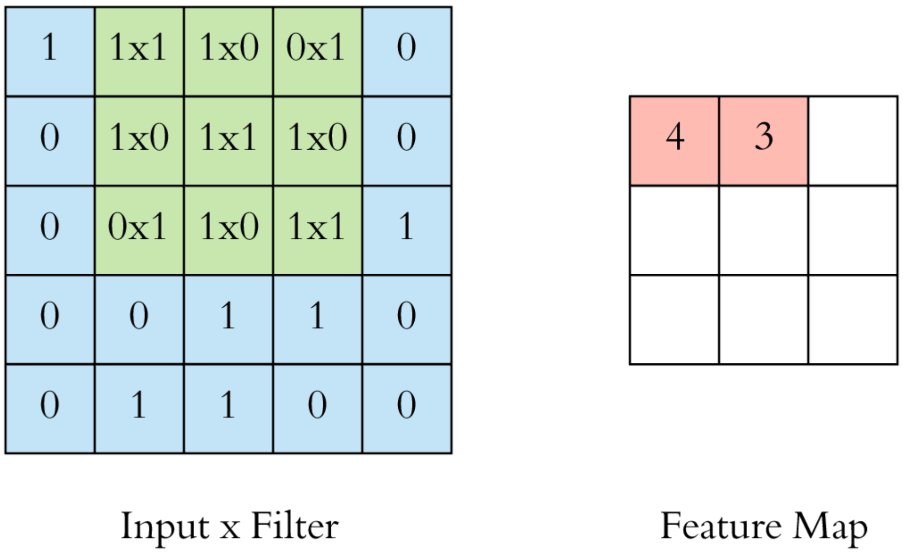


Figure 3.4: Performing convolution [27]

Pooling layer

When convolution is done once, it is very common to add a pooling layer in between the CNN layer. Pooling layers minimize the number of parameters when the input image size is too large. As a result, the computational complexity is reduced. This also shortens the training time and prevents the model from being overfitted. Spatial pooling reduces the dimensional and keeps the important information. There are three types of spatial pooling.

- Max pooling
- Average pooling
- Sum pooling

Among them, max pooling is the most frequent type. Max pooling takes the maximum value from the rectified feature map. This decreases the feature map size but keeps the important part of the image.

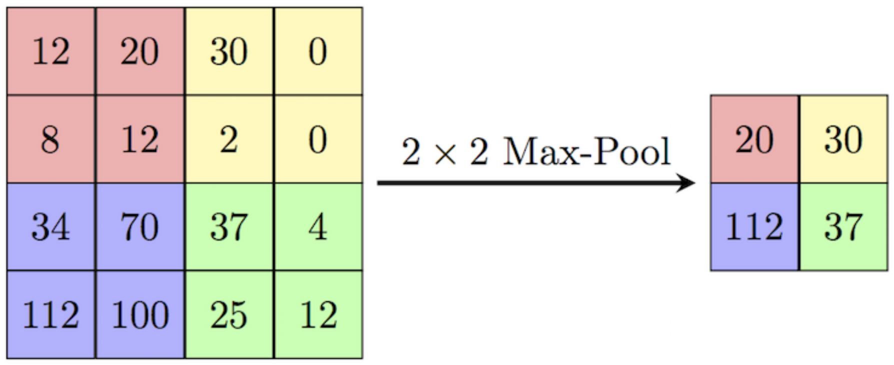


Figure 3.5: Max Pooling Layer [28]

Flattening

Flattening is the process that converts two-dimensional arrays into 1 Dimensional array or single linear vector of a long chain. This step is needed in CNN as it can provide input for the fully connected layer. In Convolutional Neural Networks, convolution gives a series of filters which is then max-pooled. This gives a grid-shaped 2-dimensional array. A fully connected layer does not work with two-dimensional input. Flattening basically breaks the spatial structure of the data from a tri-dimensional tensor into a monodimensional (or dimensional) tensor which makes the structure understandable and unstructured as a fully connected layer takes a vector as input. So flattening step is important in Convolutional Neural Networks as it works with complex data and that data is needed to be normalized. The pooled feature image is flattened into a column of a one-dimensional vector in our model to get special information around the images' pixels. Our heartbeat images are flattened for the fully connected layer to get the important feature for detection.

Dropout

A dropout is an approach to regularization in neural networks which helps reduce interdependent learning amongst the neurons. Dropout forces a neural network to learn more robust features that are useful in conjunction with many different random subsets of the other neurons. During the forward pass, Dropout temporarily removes some of the neurons. For example, if we fix dropout to 20%, then every 1 out of 5 neurons will be inactivated during the forward pass. During the backward pass, any weigh update will not be applied to these neurons.

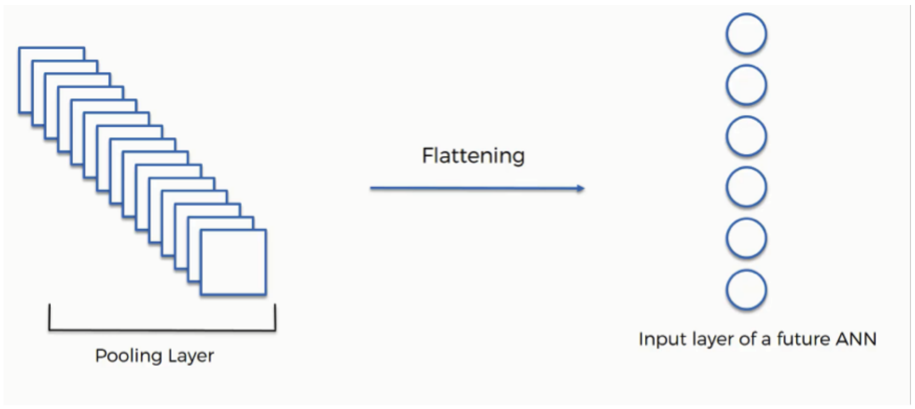


Figure 3.6: Flattening [29]

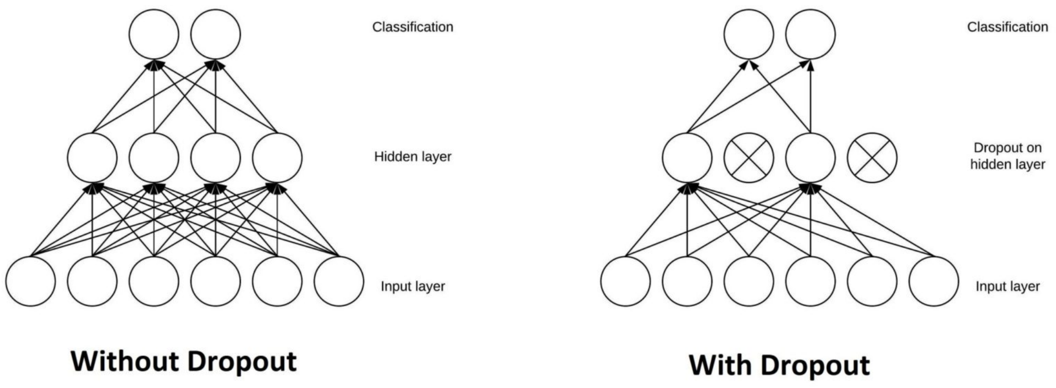


Figure 3.7: Network before and after Dropout [30]

Fully Connected Layer

The main purpose of a Neural Network is to predict more accurately by combining the features in such a way that can give a more rational prediction. The fully connected layer provides the feature combination that gives the result. A whole Artificial Neural Network is added to Convolutional Neural Network. The full Connection layer is the hidden layer of CNN. The difference is that in ANN, the hidden layers are not connected with the input layer. But in CNN, Full Connection layer is fully connected to the input layer or all the previous layers. The fully connected layer is used to get the feature combined. This combination of features makes the prediction more accurate. Full connection works with the entire input. Whenever data is needed to be classified, its features are stored in the neuron of the full connection and it is connected with the output classes. When a feature is matched, that neuron is activated and the class gets more prediction value.
After feature extraction, classification can take images in the first layer or input layer. The classification part consists of a fully connected layer. That means all the neurons of a layer are connected to all the neurons of the next layer. The fully connected layer is a traditional multi-layer perception that uses the Softmax activation function. The output from the convolution and pooling layer represents high-level input data. The aim of the fully connected

layer is to use the produced feature data and classify them into various classes.

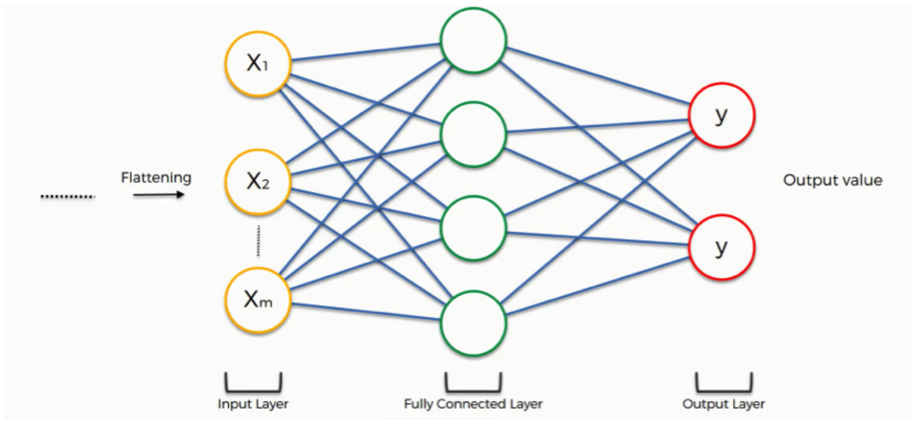


Figure 3.8: Fully Connected Layer [31]

Activation Function

The Activation function [39] is a non-linear function that is used in a hidden layer or in the output layer. It is used to define the output of the kernel weights for the input of the next neuron. A non-linear function is used to make a neural network act like a normal linear regression with limited learning power. There are various types of activation functions can be used in a CNN model, such as:

- Sigmoid or Logistic Activation Function
- Tanh or Hyperbolic Tangent Activation Function
- Rectified Linear Unit (ReLU) Activation Function
- Leaky ReLU

We have used ReLU and Softmax activation functions in our custom model.

- **Softmax:** Softmax function [40] is used to normalize the neural network output to fit between zero and one. It turns arbitrary real values into probabilities. Softmax performs the following transforms which are given in eq. (3.2)

$$\sigma(\vec{z}_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_i}} \tag{3.7}$$

  The structure of the neural network is designed to output two real numbers, say, one representing a dog and another is a cat. If the output values for a dog is -1 and for a cat is 2 then we can calculate the probability using the Softmax function.

Animal	**Z**	e^z	**Probability**
Dog	-1	0.368	0.47
Cat	-2	7.39	0.953

Table 3.1: Cat & Dog example [42]

We can see that using the softmax function we can calculate the probability of the output of the neural network. Here we can take the decision that the output means a cat as the probability is 95.3% of being a cat.

- **Rectified Linear Unit (ReLU):**
  Most used activation function in CNN is the Rectified Linear Unit or ReLU function [41]. One is a nonlinear function. CNN models need positive data to learn. This function changes the negative values with zeros and always provides positive outputs. There are other activations functions like the sigmoid function, tanh. But ReLU is used since it provides better results than the others.

$$f(x) = max(0, x) \tag{3.8}$$

3.3.2 VGG16

In their publication "Very Deep Convolutional Networks for Large-Scale Image Recognition" [12], K. Simonyan and A. Zisserman from the University of Oxford proposed the convolutional neural network model known as VGG16. In the top five tests, the model performs 92.7% accurately in ImageNet, a dataset of over 14 million images divided into 1000 classes. It creates the enhancement over AlexNet [13] by sequentially replacing several 11 and 5 kernel-sized filters in the first and second convolutional layers with multiple 33 kernel-sized filters. Weeks of training were put into VGG16 using NVIDIA Titan Black GPUs.

A 224x224 RGB image serves as the input to the VGG-based convNet. The preprocessing layer subtracts the mean image values derived for the complete ImageNet training set from the RGB image with pixel values in the range of 0-255.

These weight layers are applied after preprocessing the input photos. A stack of convolution layers is applied to the training images. The VGG16 architecture consists of 3 fully linked layers and a total of 13 convolutional layers. Instead of using huge filters, VGG uses smaller (3x3) filters with greater depth. The effective receptive field ended up being the same as if there were just one 7x7 convolutional layer.

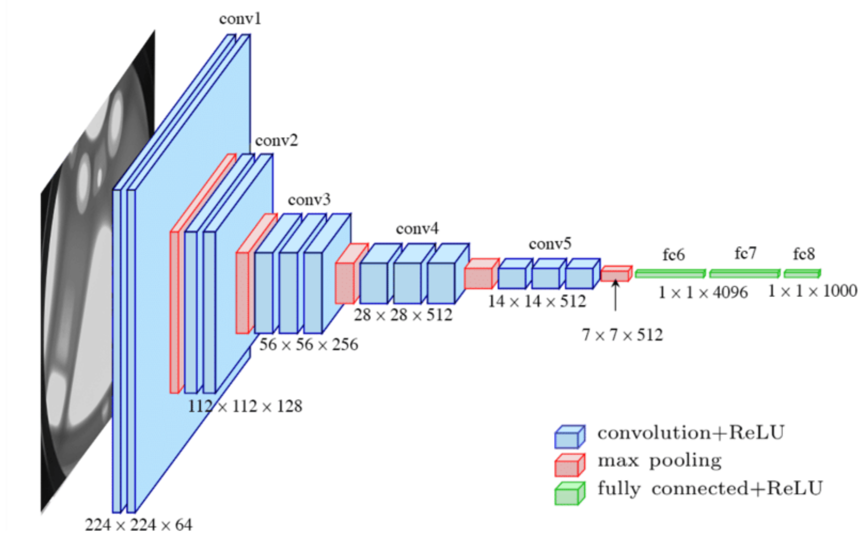


Figure 3.9: VGG-16 [14]

3.3.3 VGG19

"A Convolutional Neural Network Classifier VGG-19 Architecture for Lesion Detection and Grading in Diabetic Retinopathy Based on Deep Learning" [15], V. Sudha, Dr. T. R. Ganeshbabu proposed another version of the VGGNet contains 19 weight layers, including the same 5 pooling layers and 16 convolutional layers with 3 fully connected layers each. There are two Fully Connected layers with 4096 channels each in both variations of the VGGNet, followed by another layer.

1000 channels on a fully connected layer with 1000 predicted labels Softmax layer is used as the final fully connected layer for categorization.

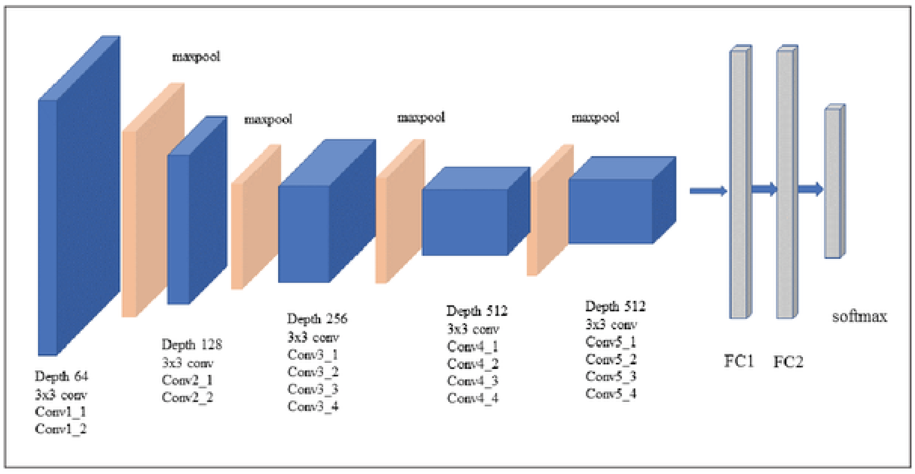


Fig. 3. VGG-19 network architecture

Figure 3.10: VGG-19 [16]

3.3.4 Inception v3

The Inception V3 model enables [17] the expansion of the depth and breadth of the deep learning network while simultaneously keeping the computational cost constant. Over a million training photos from the original ImageNet dataset were used to train this model. By computing 1*1, 3*3, and 5*5 convolutions, it functions as a multi-level feature generator. It enables the model to apply various kernels to the image and receive output from each one. The channel dimension is utilized to stack all of these outputs and serve as the next layer's input. This model utilized various cutting-edge strategies to obtain top performance for computer vision tasks.

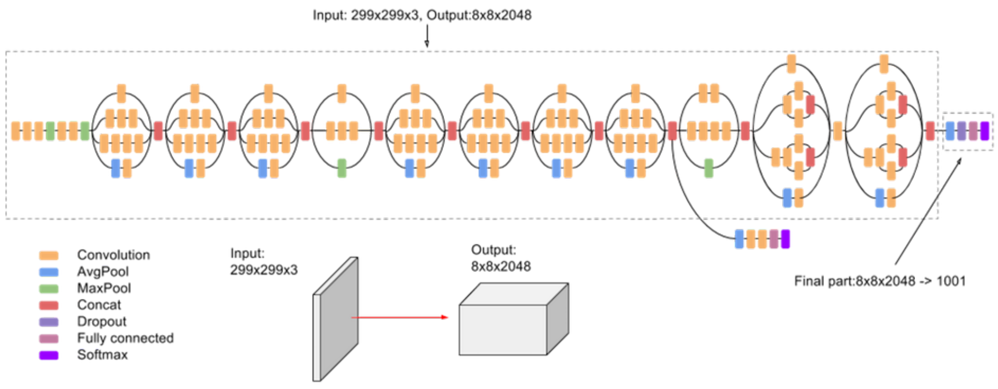


Figure 3.11: Inception-V3 [18]

3.3.5 ResNet50

Resnet50 [19] is a subclass of convolutional neural networks used for image categorization and is a deep residual network. The introduction of the new architecture, network-in-network utilizing additional layers, is the main innovation. There are five steps in the Renset50.

each with a convolution block, an identity block, and three convolution layers on each of the convolution and identity blocks. Resnet50 accepts 224x224 pixel pictures and has 50 residual networks. A residual learning framework is included with the ResNet model to make it easier to train deeper networks. The architecture is based on reinterpreting network layer inputs as learning residual functions. The remaining network is eight times as deep as VGG networks, but it is far more complex

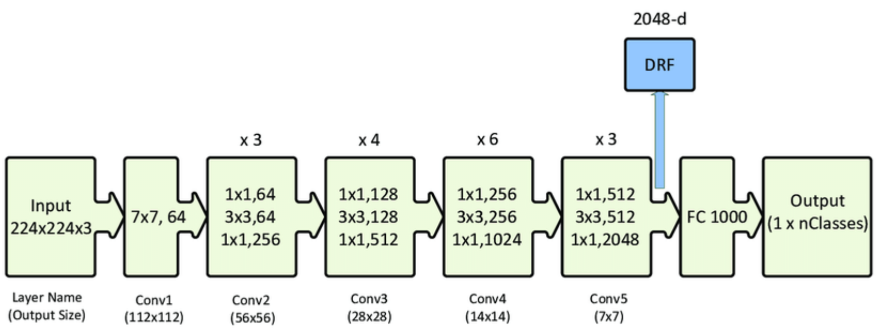


Figure 3.12: ResNet-50 [20]

3.4 Performance Measures

Different performance metrics are used for evaluating different Machine Learning Algorithms. For example, we can use performance metrics Accuracy, AUC, etc. for the classification of images. The RMSE can be used to determine how well a machine learning model is at predicting stock prices. Precision, recall, and F-score are other metrices for evaluating machine learning algorithms. Therefore, we see that different metrics are required to measure the efficiency of different algorithms, also depending upon the dataset at hand.

3.4.1 Confusion Matrix

A confusion matrix is a widely used measurement when attempting to solve classification problems. It can be applied to both binary classification and multiclass classification problems. The Confusion Matrix consists of four parameters which are described below-

- **True Positive**: When the model accurately predicts the positive class, the result is a true positive.
- **True Negative**: True negative results occur when the model predicts the negative class correctly.
- **False Positive**: A false positive is an outcome where the model incorrectly predicts the positive class.
- **False Negative**: A false negative is an outcome where the model incorrectly predicts the negative class.

3.4.2 Accuracy

Accuracy is a metric for evaluating classification models. It is useful when all classes are of equal importance. Informally, accuracy is the fraction of accurate predictions made by our model. Mathematically accuracy is represented as:

$$Accuracy = \frac{Total\ correct\ predictions}{Total\ number\ of\ predictions} = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

3.4.3 Precision

The precision is calculated as the ratio of Positive samples that were correctly classified to all samples that were classified as Positive (either correctly or incorrectly). The precision reflects how reliable the model is in classifying samples as Positive. When the precision rate is higher, the model is more effective. Precision is defined as follows:

$$Precision = \frac{TP}{TP + TN} \tag{3.10}$$

So, the precision rate is higher when:

- Many accurate Positive classifications are made by the model (maximizes True Positive)
- Fewer inaccurate Positive classifications are made by the model (minimizes False Positive)

3.4.4 Recall

The recall is determined as the proportion of positive samples that were correctly classified as positive to a number of all positive samples. The recall measures how well the model can classify positive samples. It can also be called a True Positive Rate. The more positive samples that are correctly classified(TP), the larger the Recall. Also the lower the False Negative the higher the Recall. Mathematically, recall is defined as follows:

$$Recall = \frac{TP}{TP + FN} \tag{3.11}$$

3.4.5 F1-Score

One of the most important evaluation metrics in machine learning is F1-score. It elegantly sums up the predictive performance of a model by combining two other competing metrics-precision and recall. By definition, F1-score is the harmonic mean of precision and recall. It combines precision and recall into a single number using the following formula:

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{3.12}$$

3.4.6 Specificity

Specificity is the True Negative Rate (TNR) of the model and the statistical measure of the binary classification test. As we are dealing with a binary classification (tumor or nontumor) so we can use this as the performance evaluation. It is the ratio of the number of non-tumor images that are correctly classified (TN) and the number of images that are classified or misclassified as non-tumor (TN + FP). The lower the false positive (FP) the higher the specificity or selectivity.

$$Specificity = TNR = \frac{TN}{TN + FP} \tag{3.13}$$

3.4.7 Receiver Operating Characteristic (ROC)

ROC curve [45] is another performance measure for machine learning algorithms. It is a probability curve that plots the TPR (True Positive Rate) against FPR (False Positive Rate) at various threshold values and essentially separates the 'signal' from the 'noise'. The Area Under the Curve (AUC) is the measure of the ability of a classifier to distinguish between classes and is used as a summary of the ROC curve.
The higher the AUC, the better the performance of the model at distinguishing between the positive and negative classes.

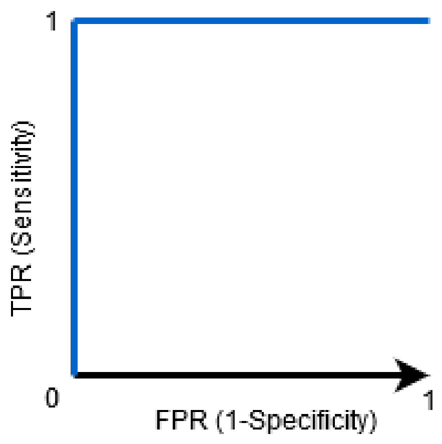


Figure 3.13: When AUC=0 or 1 [46]

When AUC = 1, then the classifier is able to perfectly distinguish between all the Positive and the Negative class points correctly. If, however, the AUC had been 0, then the classifier would be predicting all Negatives as Positives, and all Positives as Negatives.

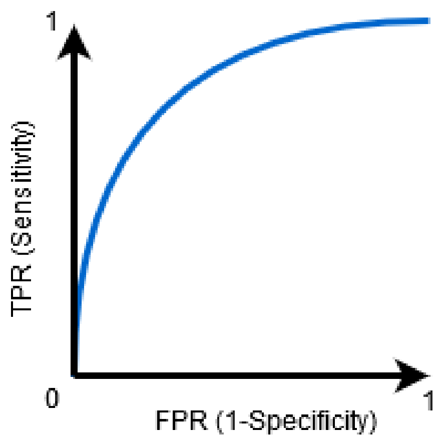


Figure 3.14: When 0.5<AUC<1 [46]

When 0.5<AUC<1, there is a high chance that the classifier will be able to distinguish the positive class values from the negative class values. This is so because the classifier is able to detect more numbers of True positives and True negatives than False negatives and False positives.

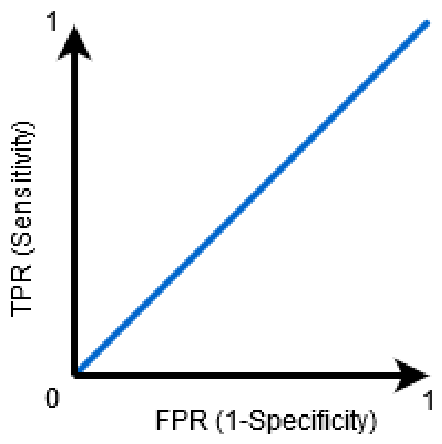


Figure 3.15: When AUC=0.5 [46]

When AUC=0.5, then the classifier is not able to distinguish between Positive and Negative class points. Meaning either the classifier is predicting random class or constant class for all the data points.

So, the higher the AUC value for a classifier, the better its ability to distinguish between positive and negative classes.

Chapter 4

Proposed Methodology and Implementation

4.1 Overview

Traditional machine learning models and deep learning models as classifiers for comparison have both been used to detect plant diseases. The proposed model has been tested using five machine learning algorithms, including KNN, Adaboost, Decision Tree, Random Forest, etc in the classification step of classical machine learning. For classifiers using deep learning, for greater model accuracy, a customized CNN model and transfer learning approaches have been applied. Augmentation techniques have been applied to train the model with deep learning models. And in the case of machine learning, Gabor Filter followed by a Sobel Filter have been used for feature extraction purposes.

4.2 Experimental Setup

All of these experiments were performed on a computer with a Windows 10-based system with Intel R Core (TM) i7-6700HQ CPU @2.60GHz processor, 1 TB HDD, 500 GB NVMe M.2 SSD, 8 GB RAM with 2133 MHz, a CUDA-enabled Nvidia R GTX 960M 4GB graphical the processing unit (GPU), Python R 3.7.6, Keras R 2.2.4-tf with TensorFlow R 2.1.0 backend, and CUDA compilation tools, release 10.2, cuDNN 7.6.0 dependencies for GPU acceleration.

4.3 Working Approach

Our work can be divided into two approaches. The two approaches are :

- Approach-1: Rice plant disease classification using deep learning approach
- Approach-2: Rice plant disease classification using traditional machine learning approach

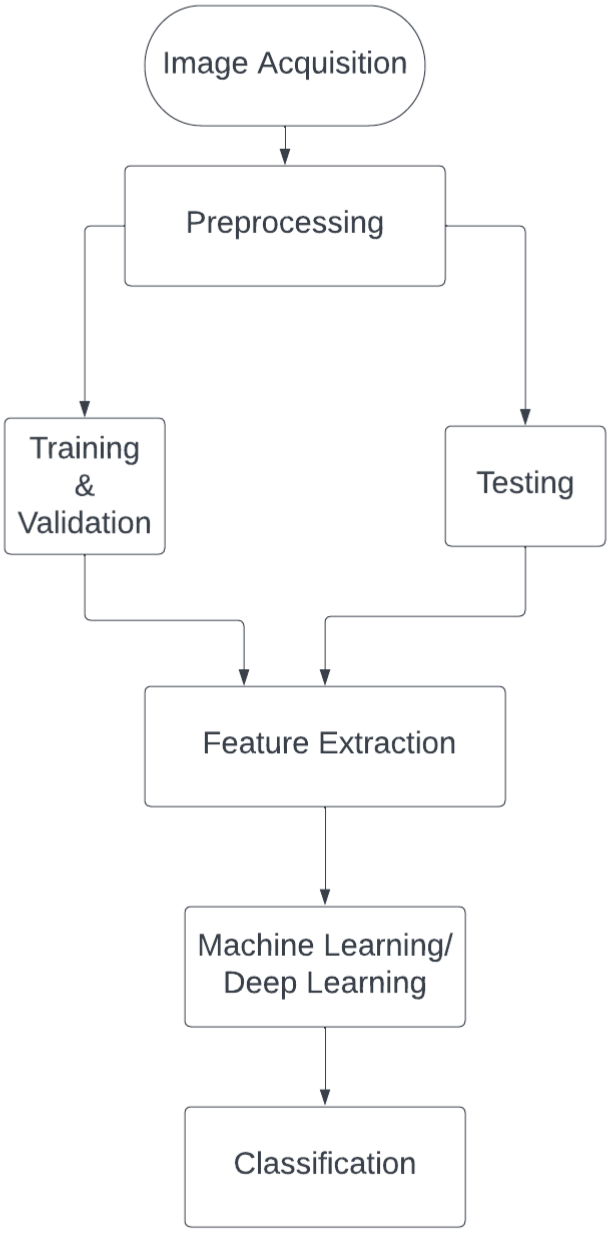


Figure 4.1: Flowchart of Working Procedure

For both approaches, there are some steps that are followed. The steps are:

- Step-1: Dataset acquisition
- Step-2: Dataset preprocessing
- Step-3: Feature extraction
- Step-4: Training the model

4.4 Dataset Acquisition

At first, we collected a suitable dataset [2]. The dataset contains 5992 images of 4 major Rice plant diseases of Bangladesh as well as other countries. The diseases are Bacterial Blight (figure 1.1) , RiceBlast (figure 1.2), RiceTungro (figure 1.4), and BrownSpot 1.3.

4.4.1 Dataset Preprocessing

After selecting an appropriate dataset for the work, changes and modifications were made based on the needs of our objectives. The dataset of 4 different rice plant diseases was divided into 2 sets of data. The training set, the testing set, or the validation set. Feature scaling or normalization was done for the data to decrease the load of the training and testing process. To improve the training process the dataset was significantly augmented (horizontal and vertical flipping, zooming, resizing) figure 4.2 to bring well-needed variation in the training data to avoid overfitting.

Data Normalization

One of the most popular methods for preparing data is normalization, which enables us to alter the values. By generating new values and retaining the general distribution as well as a ratio in the data, normalization prevents raw data and other problems with datasets. Additionally, it makes use of a number of approaches and algorithms to enhance the efficiency and accuracy of the machine or deep learning models. In image processing, normalization is a process that changes the range of pixel intensity values. The possible values for each pixel are 0 to 256. A color code is represented by each digit. The computation of high numeric values may become more difficult when using the image as it is and running it through a Deep Neural Network. We can normalize the data to fall between 0 and 1 to lessen this. In this way, the numbers will be small and the computation becomes easier and faster. As the pixel values range from 0 to 256, apart from 0 the range is 255. So dividing all the values by 255 will convert it to a range from 0 to 1.

Image Augmentation

For deep networks to perform well, a lot of training data is required. Image augmentation is typically needed to improve deep network performance in order to create a powerful image classifier with little training data. Image augmentation artificially generates training images by combining several processing techniques, such as random rotation, shifts, shear, flips, etc.

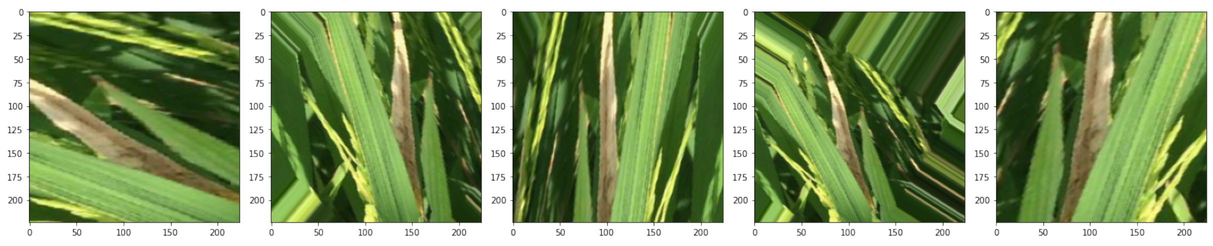


Figure 4.2: Augmented image

Dataset Splitting

The datasets are split into two sets which are training and validation/testing sets having a ratio of 80:20 respectively. The augmentation techniques are applied to the training dataset only.

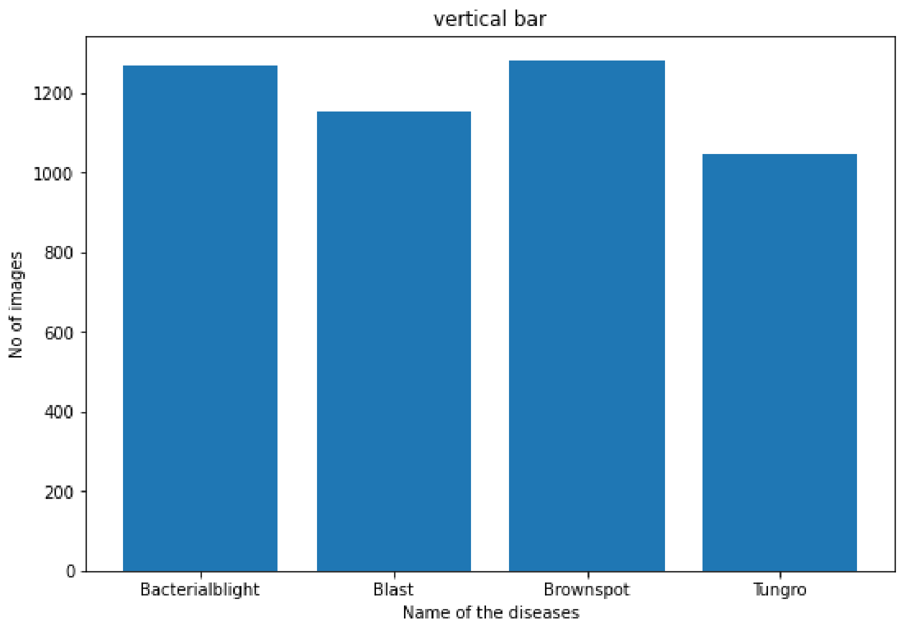


Figure 4.3: Train Data Distribution

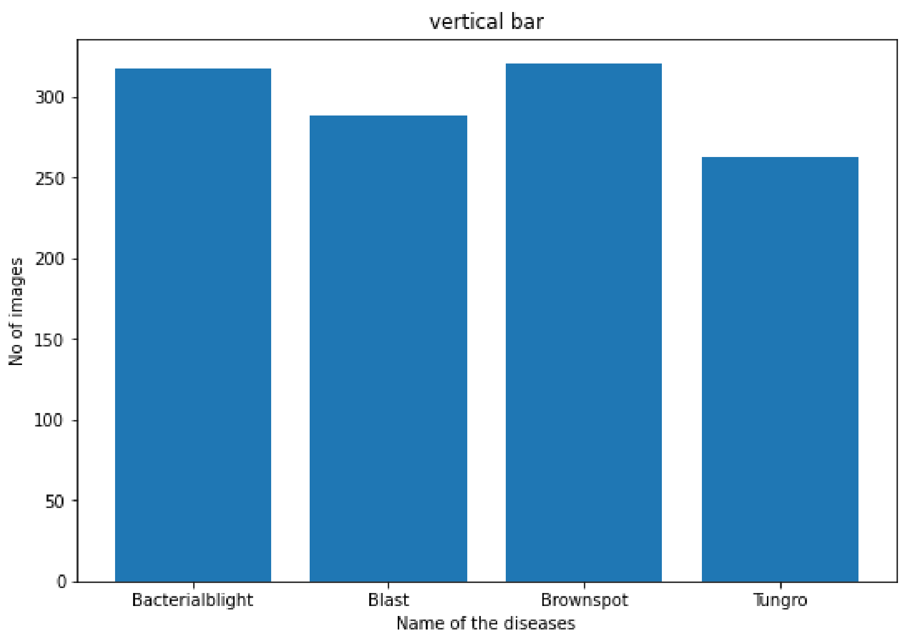


Figure 4.4: Test Data Distribution

There is total data of 5932 images where 4745 images are used for training and 1187 images for training. In training distribution 1267 images for Bacterial Blight, 1152 for Blast, 1280 for Brown Spot, and 1046 for Tungro are used. In test distribution 317 images for Bacterial Blight, 288 for Blast, 320 for Brown Spot, and 262 for Tungro are used.

4.4.2 Feature Extraction

For feature extraction, the Gabor filter (figure 3.1) followed by a Sobel filter [38] of size 3X3 is used. Here 256 filters are generated and convoluted over the training and testing images. For Gabor kernels, the kernel size is set to 5X5.

Gabor Features	Kernel Size	Theta	Sigma	Lambda	Gamma
256	5×5	0 - .78	1 & 3	0 - .78	.05 & .5

Table 4.1: Parameters of Gabor Kernels

The Gabor features have been used over the training images for a total of 256 times for the Machine Learning model. With each iteration, the images undergo a feature extraction process based on the Gabor filter's different parameters. The filter uses parameters that control orientation(Theta), Phase Offset(Lambda), Gaussian Factor (Sigma), and Ellipticity(Gamma). All of these properties dictate the way how the information is extracted from the training images. This allows for the training to have images with highly crucial features required to detect the particular type of disease for a specific label. Figure 4.5 shows different stages of feature extraction with each iteration.

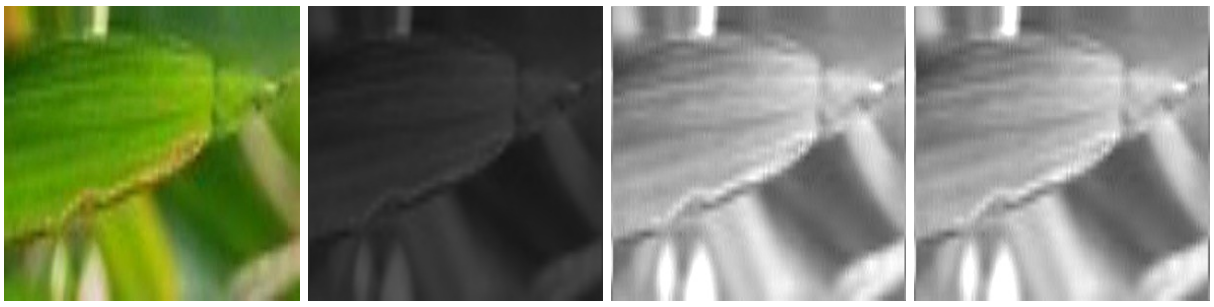


Figure 4.5: (From left to right)- Normal image, Seventh Stage of, 63th stage and 256th stage of Gabor Features

Deep Learning Model Architecture

A custom CNN model has been proposed by our own to train the preprocessed dataset. The model consists of 11 layers of which 10 are hidden layers and 1 input and 1 out layer. The layers include a convolutional layer, pooling layer, flattened layer, and fully-connected or dense layer. The activation functions which are used are Relu and Softmax activation functions. The 4.5 shows the model architecture of the custom model.

- Layer 1: At first a convolutional layer has been used with an input of 224x224x3.Here 224x224 is the input size of the image and 1 is the number of channels or depth. The number of filters used here is 16 and there is no padding in the layer. stride, S=1, and filter size, F=3. So from the input, we get the following output image size,
  - output depth, D2=number of filters=64
  - output width, W2 = (224-3+2x0)/1+1=222 and
  - output height, H2 = (224-3+2x0)/1+1=222
  - output volume is 222x222x16
  - number of parameters = 3x3x3x16+16=448
- Layer 2: Here, a pooling layer has been used to re-size the input and extract the important pixel values. The Max pooling technique is applied here. The input volume is $222 \times 222 \times 16$. For pooling, we need the value for spatial extend, filter and stride, S. We have used F = 2 and S = 2 which is a very commonly used size. So from the input volume, we get the following output image size,
  - output depth, D2=number of filters=16
  - output width, W2 = $\lfloor 222/2 \rfloor$=111 and
  - output height, H2 = $\lfloor 222/2 \rfloor$=111 and
  - output volume is 111x111x16

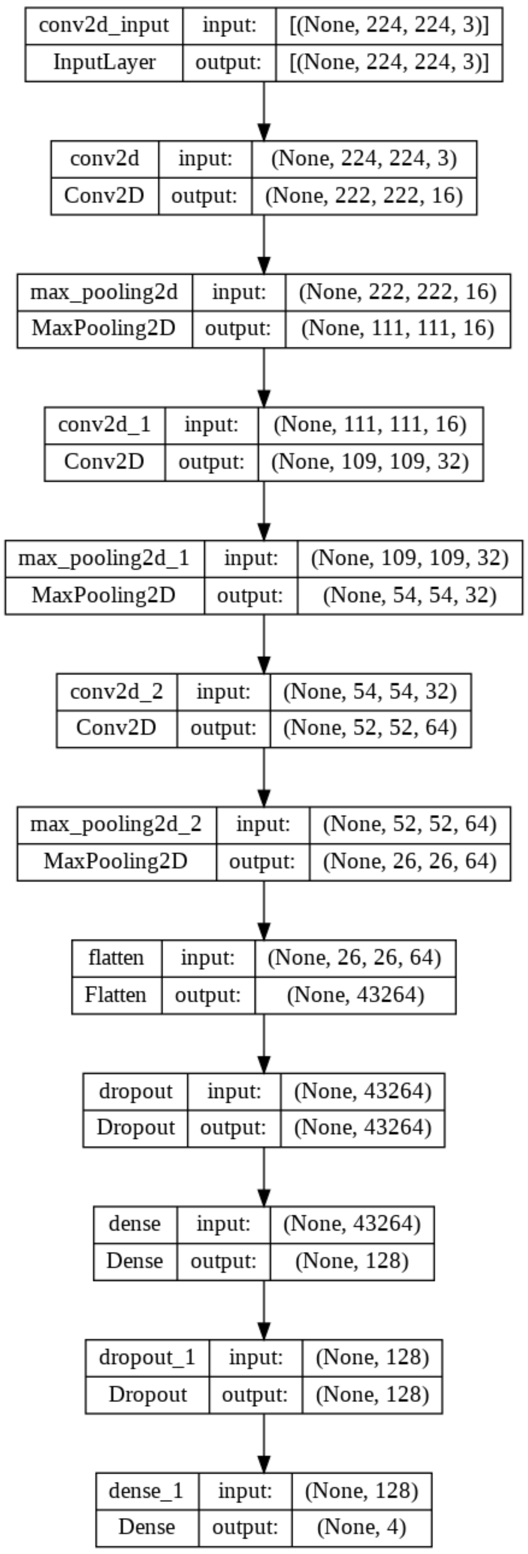

conv2d_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
conv2d | input: | (None, 224, 224, 3)
Conv2D | output: | (None, 222, 222, 16)
max_pooling2d | input: | (None, 222, 222, 16)
MaxPooling2D | output: | (None, 111, 111, 16)
conv2d_1 | input: | (None, 111, 111, 16)
Conv2D | output: | (None, 109, 109, 32)
max_pooling2d_1 | input: | (None, 109, 109, 32)
MaxPooling2D | output: | (None, 54, 54, 32)
conv2d_2 | input: | (None, 54, 54, 32)
Conv2D | output: | (None, 52, 52, 64)
max_pooling2d_2 | input: | (None, 52, 52, 64)
MaxPooling2D | output: | (None, 26, 26, 64)
flatten | input: | (None, 26, 26, 64)
Flatten | output: | (None, 43264)
dropout | input: | (None, 43264)
Dropout | output: | (None, 43264)
dense | input: | (None, 43264)
Dense | output: | (None, 128)
dropout_1 | input: | (None, 128)
Dropout | output: | (None, 128)
dense_1 | input: | (None, 128)
Dense | output: | (None, 4)

Figure 4.6: Model Architecture

  – number of parameters=0

• Layer 3: And Then a convolutional layer is used with the output from the previous layer which is the input of this layer, 111x111x16.Here 111x111 is the input size of the image and 16 is the number of channels or depth. The number of filters used here is 32 and there is no padding in the layer. Stride, S=1, and filter size, F=3. So from the input, we get the following information

  - output depth, D2=number of filters=32
  - output width, W2 = (111-3+2x0)/1+1=109 and
  - output height, H2 = (111-3+2x0)/1+1=109 and
  - output volume is 109x109x32
  - number of parameters=3x3x16x32+32=4640

- Layer 4: Then, another pooling layer has been used to resize the input and extract the important pixel values. The Max pooling technique is applied here. The input volume is 109×109×32. For filter or pooling, we need the value for spatial to extend, filter, and stride, S. We have used F = 2 and S = 2 which is a very commonly used size. So from the input volume, we get the following output image size

  - output depth, D2=number of filters=16
  - output width, W2 = $\lfloor 109/2 \rfloor$=54 and
  - output height, H2 = $\lfloor 109/2 \rfloor$=54 and
  - output volume is 54x54x32
  - number of parameters = 0

- Layer 5: And then a convolutional layer has been used with the output from the previous layer which is the input of this layer, 54x54x32. Here 54x54 is the input size of the image and 32 is the number of channels or depth. The number of filters used here is 64 and there is no padding in the layer. Stride, S=1, and filter size, F=3. So from the input, we get the following information

  - output depth, D2=number of filters=64
  - output width, W2 = (54-3+2x0)/1+1=52 and
  - output height, H2 = (54-3+2x0)/1+1=52 and
  - output volume is 52x52x64
  - number of parameters=3x3x32x64+64=18496

- Layer 6: Then, another pooling layer has been used to resize the input and extract the important pixel values. The Max pooling technique is applied here. The input volume is 52×52×64. For filter or pooling, we need the value for spatial to extend, filter, and stride, S. We have used F = 2 and S = 2 which is a very commonly used size. So from the input volume, we get the following output image size

- output depth, D2=Number of filters=16
- output width, W2 = $\lfloor 52/2 \rfloor$=26 and
- output Height, H2 = $\lfloor 52/2 \rfloor$=26 and
- output volume is 26x26x64
- number of parameters=0

• Layer 7: Then, another flattened layer has been used to the input size of 26x26x64 from the input volume we get the following output image size

  - output image size D2=26x26x64=43264
  - number of parameters=0

• Layer 8: Then, a dropout layer has been added to the input size of 43264 from the input volume we get the following output image size

  - output image size D2=26x26x64=43264
  - number of parameters=0

• Layer 9: Then, a dense layer has been added to the input size of 43264 from the input volume we get the following output image size

  - output image size D2=128
  - number of parameters=43264x128+128=5537920

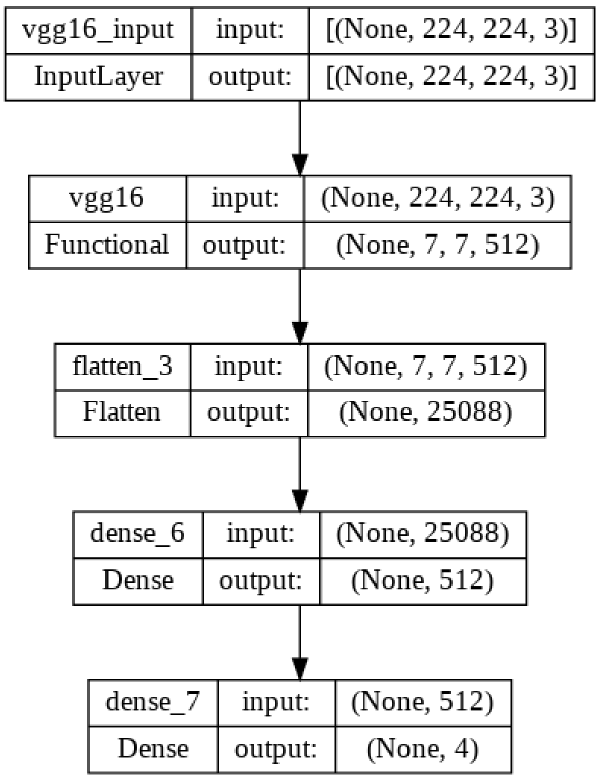
• Layer 10: Then, another dropout layer has been added to the input size of 5537920 from the input volume, and using dropout of (.2) ratio we get the following output image size

  - output image size D2=128
  - number of parameters=0

• Layer 11: Then, another dense layer has been added to the input size of 128 from the input volume and we get the following output information

  - number of classes D2=4
  - number of parameters=128X4+128=516

Here softmax activation has been used for classification purposes.



vgg16_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
vgg16 | input: | (None, 224, 224, 3)
Functional | output: | (None, 7, 7, 512)
flatten_3 | input: | (None, 7, 7, 512)
Flatten | output: | (None, 25088)
dense_6 | input: | (None, 25088)
Dense | output: | (None, 512)
dense_7 | input: | (None, 512)
Dense | output: | (None, 4)

Figure 4.7: VGG-16 Model Architecture

VGG-16

A pre-trained VGG-16 model figure no 4.7 is used by changing the input size to 224X224X3. Then all other layers are kept frozen and those layers' weights will not be updated. After that, a flattened layer and two consecutive dense layers are added. In the last layer, the Softmax classifier is used.

VGG-19

A pre-trained VGG-19 model figure no 4.8 is used by changing the input size to 224X224X3. Then all other layers are kept frozen and those layers' weights will not be updated. After that, a flattened layer and two consecutive dense layers are added. In the last layer, the Softmax classifier is used.

Inception-V3

A pre-trained InceptionV3 model figure no 4.9 is used by changing the input size to 299X299X3. Then all other layers are kept frozen and those layers' weights will not be updated. After that, a flattened layer and two consecutive dense layers are added. In the last layer, the Softmax classifier is used.

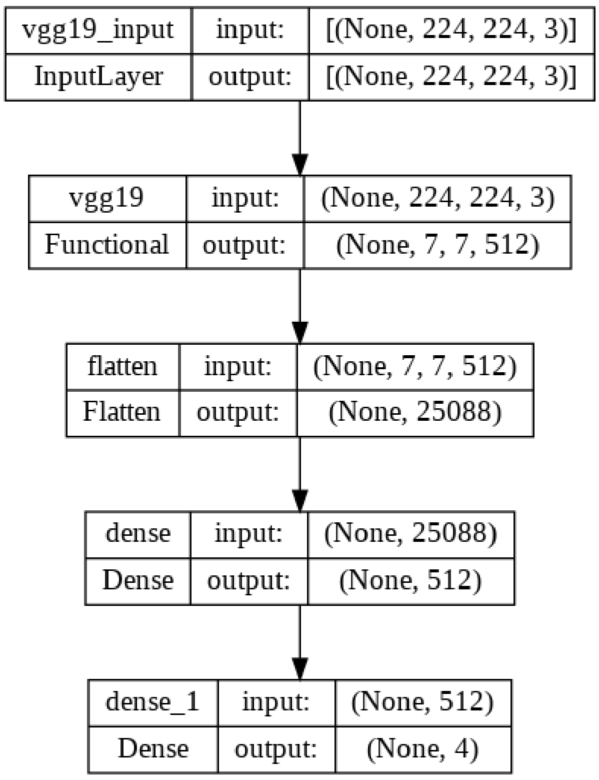

vgg19_input | InputLayer | input: [(None, 224, 224, 3)] | output: [(None, 224, 224, 3)]
vgg19 | Functional | input: (None, 224, 224, 3) | output: (None, 7, 7, 512)
flatten | Flatten | input: (None, 7, 7, 512) | output: (None, 25088)
dense | Dense | input: (None, 25088) | output: (None, 512)
dense_1 | Dense | input: (None, 512) | output: (None, 4)

Figure 4.8: VGG-19 Model Architecture

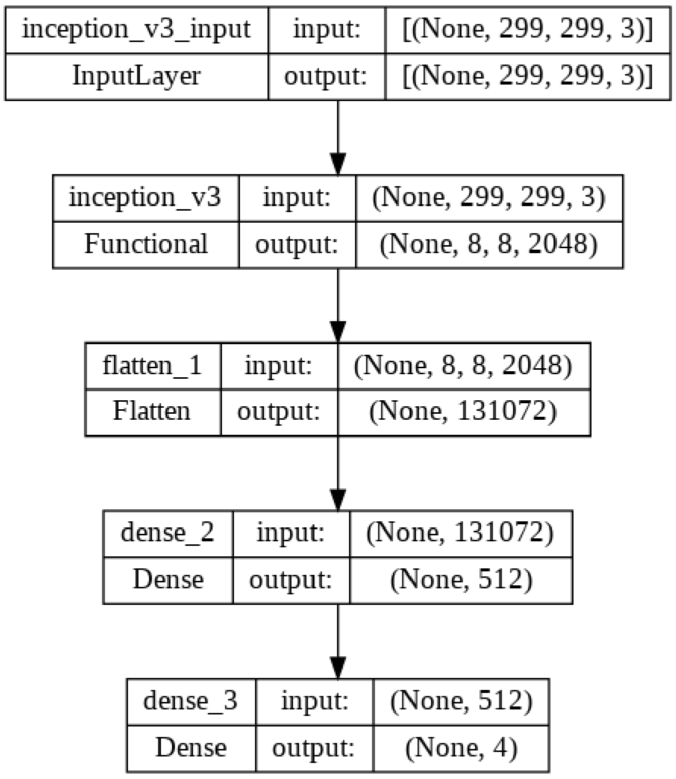

inception_v3_input | InputLayer | input: [(None, 299, 299, 3)] | output: [(None, 299, 299, 3)]
inception_v3 | Functional | input: (None, 299, 299, 3) | output: (None, 8, 8, 2048)
flatten_1 | Flatten | input: (None, 8, 8, 2048) | output: (None, 131072)
dense_2 | Dense | input: (None, 131072) | output: (None, 512)
dense_3 | Dense | input: (None, 512) | output: (None, 4)

Figure 4.9: InceptionV3 Model Architecture

Resnet-50

A pre-trained Resnet model (figure no) 4.10 is used by changing the input size to 224X224X3. Then all other layers are kept frozen and those layers' weights will not

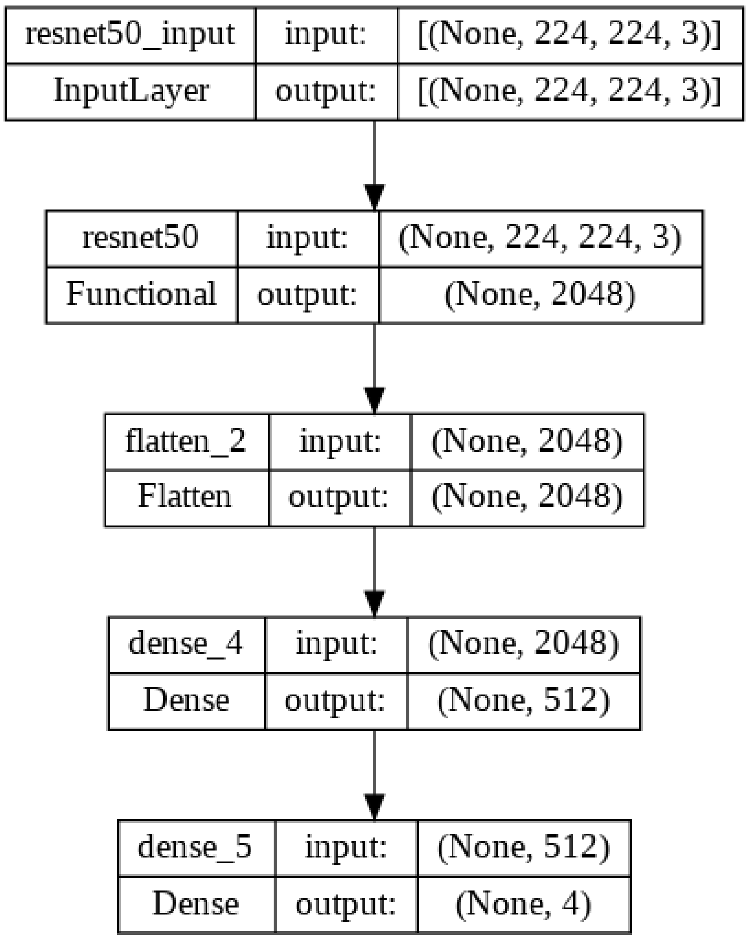

resnet50_input | input: | [(None, 224, 224, 3)]
InputLayer | output: | [(None, 224, 224, 3)]
resnet50 | input: | (None, 224, 224, 3)
Functional | output: | (None, 2048)
flatten_2 | input: | (None, 2048)
Flatten | output: | (None, 2048)
dense_4 | input: | (None, 2048)
Dense | output: | (None, 512)
dense_5 | input: | (None, 512)
Dense | output: | (None, 4)

Figure 4.10: resnet50 Model Architecture

be updated. After that, a flattened layer and two consecutive dense layers are added. In the last layer, the softmax classifier is used.

4.4.3 Parameters and Hyperparameters

Besides the inceptionV3 model, all other models have the same input size of 224x224x3. The inceptionV3 model needs to have an image input size of 229x229x3. After many trials and errors, a batch size of 8 for these images has been selected for the best possible accuracy.
The Adam optimizer [44] has a faster computation time, and requires fewer parameters for tuning. This makes it the best option for all the deep learning models. The Loss Function used for all the models is the Sparse Categorical Crossentropy (SCC). Since the truth labels are integer encoded, the loss function for the deep learning models has been fixed to Sparse Categorical Crossentropy (SCC). Further hyperparameter tuning has been explored in table 5.7

Hyperparameters	**Value**				
	Custom	**VGG16**	**VGG19**	**InceptionV3**	**ResNet50**
Input Size	224x224x3	224x224x3	224x224x3	299x299x3	224x224x3
Optimizer	Adam	Adam	Adam	Adam	Adam
Loss Function	SCC	SCC	SCC	SCC	SCC
Learning Rate	0.001	0.001	0.001	0.001	0.001
Epochs	40	40	40	40	40
Batch Size	8	8	8	8	8

Table 4.2: Parameters and Hyperparameters for deep learning models

Machine Learning Model

For Machine Learning Models, SVM, Random Forest, Adaboost, etc are used. And last, their comparison Analysis is shown in table 5.6

Chapter 5

Experiments and Result Analysis

5.1 Performance Analysis for Machine Learning Models

To start out with the comparative analysis of this study, the machine learning models were taken into consideration. This part of the study includes the use of models such as KNN (K-Nearest Neighbour), Random Forest, Adaboost, SVM & Decision Tree. Most of these models have achieved satisfactory results. The KNN model gives a result of 75.48% accuracy. However, the Random Forest model shows the highest accuracy of around 96.61%

5.1.1 Random Forest

The model used here for the performance analysis using machine learning is Random Forest. The dataset used for this study has already gone through the process of feature extraction using Gabor and Sobel filters. These images are then fed into the model for training. Since random forest combines multiple decision tree algorithms for correcting the over-fitting problem with the dataset. The model outputs the accuracy based on the multiclass classification into 4 of the aforementioned disease groups.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.95	0.99	0.97
Blast	0.99	0.90	0.94
Brown spot	0.96	0.99	0.98
Tungro	1.00	0.93	0.96
weigted avg	0.97	0.97	0.97

Table 5.1: Result of Evaluation Metrics for Random Forest

The above table shows the precision, recall, and overall F1-score achieved for every individual disease type. Observing this data gives the precise identification of disease in the case of almost every disease type. Here, Rice Tungro shows perfect precision which means there are no false positives detected here. Similarly, Brown spot shows near-perfect recall which means there are little to no false negatives present here. Random Forest gives an accuracy of **96.61%** which is the best accuracy achieved out of all the machine learning models.

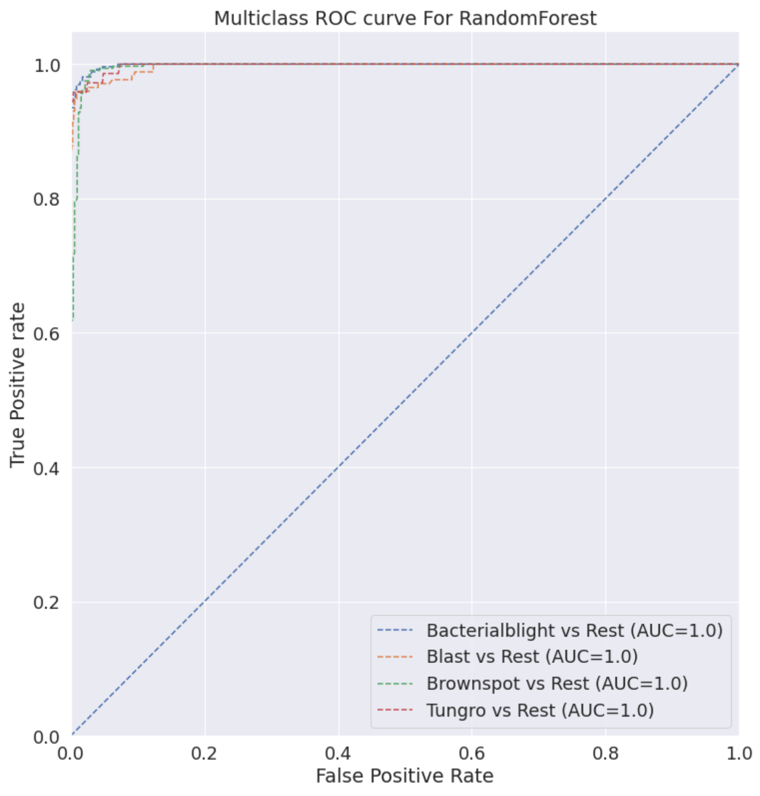


Figure 5.1: ROC curve for Random Forest

The ROC curve for random forest shows that in the case of almost every disease, this model acts as a near-perfect classifier since the AUC for every disease shows the maximum area.

5.1.2 KNN (K-Nearest Neighbour)

KNN is one of the more simpler machine learning algorithms, which is classified by determining the neighbors around a certain label or class and the maximum euclidean distance is considered for the label. The 4 diseases have shown diversity in their results in the case of KNN. The table depicts how in some instances, disease detection sometimes yields dissimilar results.

The table depicts that the bacterial blight and Rice Tungro have very high precision values, which means they yield low amounts of false positive classifications. However, in both cases, there are a high amount of false negatives. Rice Tungro seems to be displaying that behavior

	Precision	**Recall**	**F1-score**
Bacterial blight	0.98	0.68	0.80
Blast	0.68	0.69	0.69
Brown spot	0.67	0.91	0.77
Tungro	0.95	0.50	0.65
weigted avg	0.80	0.75	0.75

Table 5.2: Result of Evaluation Metrics for KNN

the most in this case. The accuracy for KNN is **75.48%** which is the lowest accuracy achieved out of all the machine learning models.

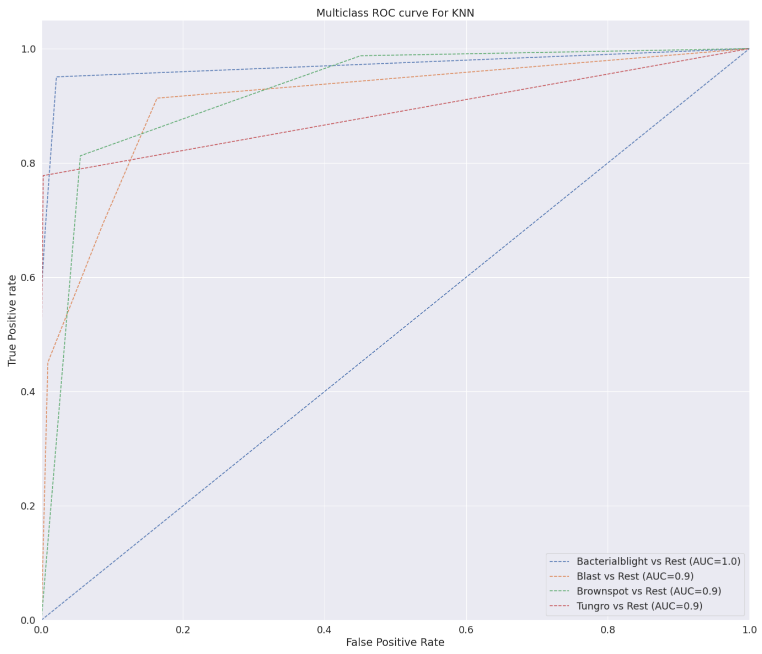


Figure 5.2: ROC curve for KNN

The ROC curve shows diversity for each disease label the disease classification has shown both positive and negative. Bacterial Blight shows the best accuracy compared to all the other diseases in the dataset. Rice Blast seems to be the least accurate of the bunch since both its precision and recall values are relatively low. This is why the AUC is maximum for Bacterial Blight and least for Rice Blast.

5.1.3 Adaboost

Adaboost being an Adaptive boosting algorithm is used as an Ensemble method. Since AdaBoost is generally resistant to overfitting, it has been considered as a model for testing the dataset. Now AdaBoost is known for being sensitive to noisy data and outliers. The effects of this have been observed in the evaluation metrics of in the given table. But since the dataset used for this study proves to be efficient for detecting and classifying rice diseases, so the results show relatively high accuracy.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.86	0.94	0.90
Blast	0.91	0.80	0.85
Brown spot	0.92	0.94	0.93
Tungro	0.95	0.86	0.91
weigted avg	0.90	0.90	0.90

Table 5.3: Result of Evaluation Metrics for Adaboost

The table shows that the precision achieved for rice Tungro is the highest whereas the Brown spot gives the best recall score. Brown spot is observed to have the overall best score out of all other diseases. So the accuracy for AdaBoost is **90.21%**. It may not be as much as a random forest but it is still a satisfactory score in terms of a machine-learning model.

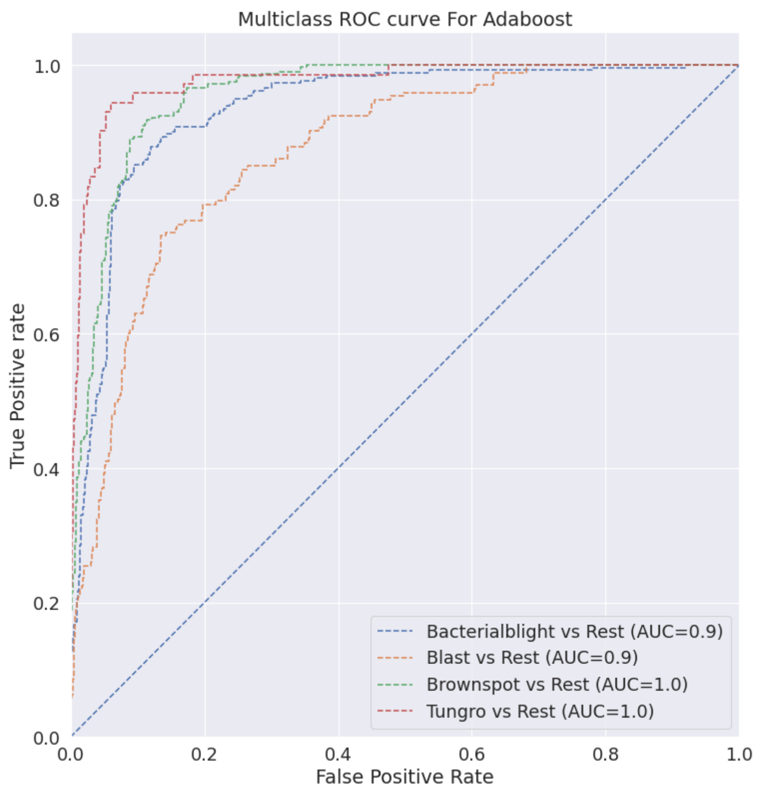


Figure 5.3: ROC curve for Adaboost

The ROC curve seems to show the most AUC underneath Brown spot and Tungro compared to the other two. This gives an idea of how the model is better at classifying some diseases more than others.

5.1.4 SVM (Support Vector Machine)

The SVM algorithm is effective in cases where the number of dimensions is greater than the number of samples. However this machine learning algorithm works by putting data points, above and below the classifying hyperplane, so this means there is no probabilistic explanation for the classification. furthermore, SVM does not perform very well when the data set has more noise i.e. target classes are overlapping. Since the dataset used for this study is large. The algorithm doesn't perform as well compared to the random forest algorithm.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.81	0.94	0.87
Blast	0.90	0.47	0.62
Brown spot	0.78	0.93	0.85
Tungro	0.89	0.58	0.71
weigted avg	0.82	0.81	0.79

Table 5.4: Result of Evaluation Metrics for SVM

The table shows that Rice Blast has the most precise classification since there are lesser number of false positives. However in terms of recall, it shows an unsatisfactory value. This means there are a large number of false negatives in this case. The other diseases show a relatively sound balance between the evaluation metrics. An accuracy of **80.79%** has been achieved here.

5.1.5 Decision Tree

Compared to other algorithms decision trees requires less effort for data preparation during pre-processing. Missing values in the data also do not affect the process of building a decision tree to any considerable extent.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.97	0.89	0.93
Blast	0.91	0.80	0.85
Brown spot	0.94	0.93	0.94
Tungro	0.53	0.89	0.66
weigted avg	0.91	0.89	0.89

Table 5.5: Result of Evaluation Metrics for Decision Tree

This goes to show the durability of the decision tree algorithm compared to others like

SVM or KNN. The non-Parametric method is defined as the method in which there are no assumptions about the spatial distribution and the classifier structure. So, in the end, it performs relatively well, although it still underperforms compared to the random forest which is known to use multiple decision trees. Aside from the rice tungro disease class, all other diseases show satisfactory f1 scores. The accuracy achieved through this process was **89%** which is very good in terms of a machine learning algorithm.

5.1.6 Comparison Analysis Between Machine Learning Models

The above discussion illustrates the impact of multiple machine learning models along with their ROC (Receiver operating characteristic) curves on the rice disease image dataset. It shows how besides some models, most of the others show a relatively consistent trend in their performance metrics. Here it is observed that the values are relatively less consistent compared to the deep learning models as shown in table 5.13.

Classifiers	**Accuracy**	**Precision**	**Recall**	**F1-score**
Random Forest	0.966	0.97	0.97	0.97
KNN	0.754	0.80	0.75	0.75
AdaBoost	0.902	0.90	0.90	0.90
SVM	0.81	0.82	0.81	0.79
Decision Tree	0.89	0.91	0.89	0.89

Table 5.6: Result of Evaluation Metrics for Machine Learning Models

It is observed that Random Forest, Adaboost and Decision Tree performs the best out of all the machine learning models. Their precision and recall scores are also consistent showing relatively low numbers of false detections. SVM is a relatively sound and its metrics show similar accuracy. However KNN seems to have disparity in it's precision and recall scores. This means KNN yields a high number of false negatives compared to false positives. This means the algorithm falsely detects healthy leaves which leads to a problematic results. So, in terms of machine learning models, Random Forest shows impressive accuracy of **96.6%** and also shows the best metrics. But it is unable to beat the deep learning Custom CNN model discussed in section 5.2.2

5.2 Performance Analysis for Deep Learning Models

Since the primary goal of this study was to analyze and compare the best possible approaches to classify rice plant diseases accurately and effectively, the deep learning approach has been preferred due to its better performance in terms of image datasets. Convolutional Neural networks played a key role in this since they are the most effective approach for image classification problems. Rice plant disease studies have been conducted in many of the papers in the literature reviews. The deep learning techniques have proven better than most other methods discussed.

5.2.1 Hyperparameters Tuning

The table displays the different combinations for batch sizes we have considered for better hyperparameter tuning. It shows a different accuracies for different batch sizes. Batches of 32, 16 and 8 have been considered. An epoch number of 40 has been considered based on optimal condition and our limited number of resources. The rest of the hyperparaneters have been used according to the table 4.2

	Batch Size	**Epoch**	**Accuracy (%)**
Custom CNN	32	40	85
	16		88
	8		97
InceptionV3	32	40	89
	16		90
	8		98
VGG-16	32	40	91
	16		94
	8		97
VGG-19	32	40	89
	16		95
	8		98
ResNet-50	32	40	69
	16		65
	8		71

Table 5.7: Hyperparameter Tuning

It can be observed that the accuracy for all the deep learning models, shows better results for a batch size of 8. The results concluded that a higher batch size does not usually achieve high accuracy, and the learning rate and the optimizer used will have a significant impact as well. Lowering the learning rate and decreasing the batch size will allow the network to train better, especially in the case of fine-tuning [43] . In the case of this study, the lower batch size of 8 removes the chances of noisy data, improving the accuracy for the models.

5.2.2 Custom Model

The custom model proposed for the deep learning approach outperforms the pre-trained CNN model resnet=50 and gives equivalent results for all the other CNN models used for the deep learning approach. But the parameter size for our custom model is around 5 million. This parameter size is much smaller compared to the pre-trained models used in this study. This makes the proposed model much more lightweight and easy to process compared to the other heavy CNN models.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.91	1.00	0.95
Blast	1.00	0.88	0.94
Brown spot	1.00	1.00	1.00
Tungro	1.00	1.00	1.00
weigted avg	0.97	0.97	0.97

Table 5.8: Result of Evaluation Metrics for Custom Model

The table shows the superiority of the deep learning models compared to every machine learning model used in this study. The proposed custom model outperforms the random forest model and gives an accuracy of **97%**. Since this is a deep learning model it also performs much faster compared to the machine learning model.

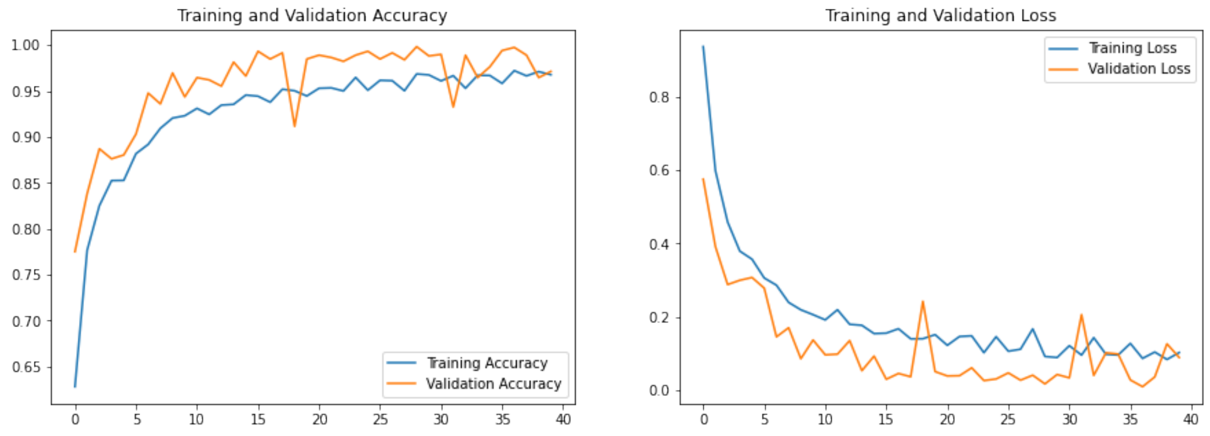


Figure 5.4: Training vs Validation (Accuracy & Loss) for **Custom Model**

Both the Accuracy and Loss graphs show that the training and validation move uniformly. The accuracy steadily increases and loss steadily declines. This is the desired behavior for a deeo learning model. Besides some very minor fluctuations, the model performs very well throughout the training and validation process.

5.2.3 VGG-19

The pre-trained CNN model VGG-19 was chosen for the deep learning models in this study. Since this is a CNN model 19 layers deep, it has a large number of parameters (around 138 Million). Since every layer isn't important, they can be dropped. It is still a large number of parameters even after adding dropout. Since this is a model trained by millions of images, it performs very well as a CNN model.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.99	0.98	0.99
Blast	0.99	0.94	0.97
Brown spot	0.95	1.00	0.98
Tungro	1.00	1.00	1.00
weigted avg	0.98	0.98	0.98

Table 5.9: Result of Evaluation Metrics for VGG-19

The evaluation metrics show that the precision and recall achieved for almost every disease are near perfect for this model. It is observed that Rice Tungro gives a perfect score which means every image used in the validation set has been successfully classified for that disease. Thus giving an accuracy of **98%** which is the highest accuracy achieved for a deep learning model.

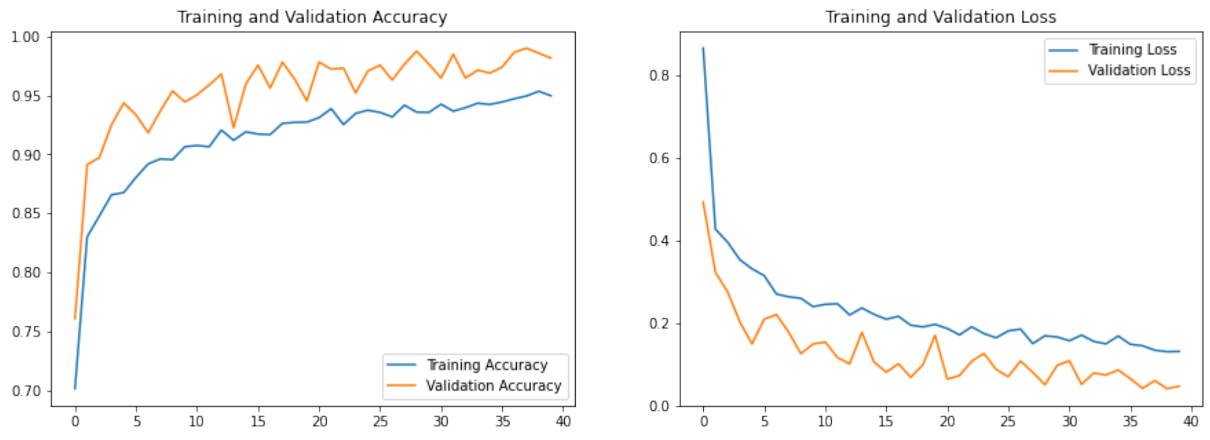


Figure 5.5: Training vs Validation (Accuracy & Loss) for **VGG-19**

The training vs validation graphs shows that the process does not yield any overfitting results. The validation accuracy and loss are uniformly increasing and decreasing respectively. This shows the result is near perfect with little to no false positives or negatives.

5.2.4 InceptionV3

InceptionV3 is one of the more preferred CNN models that are pre-trained. It has a deeper network compared to the Inception V1 and V2 models, but its speed isn't compromised. The efficiency of this model is really impressive. It is also computationally less expensive thus giving it a better edge against other CNN models.

	Precision	Recall	F1-score
Bacterial blight	0.99	0.99	0.99
Blast	0.92	1.00	0.96
Brown spot	1.00	0.93	0.96
Tungro	1.00	1.00	1.00
weigted avg	0.98	0.98	0.98

Table 5.10: Result of Evaluation Metrics for InceptionV3

The table shows a clear idea of there is almost no false positives in terms of all the diseases. Rice Tungro seems to give perfect results for both precision and recall. So the accuracy for the InceptionV3 model is **98%** which is also the highest accuracy achieved out of all models.

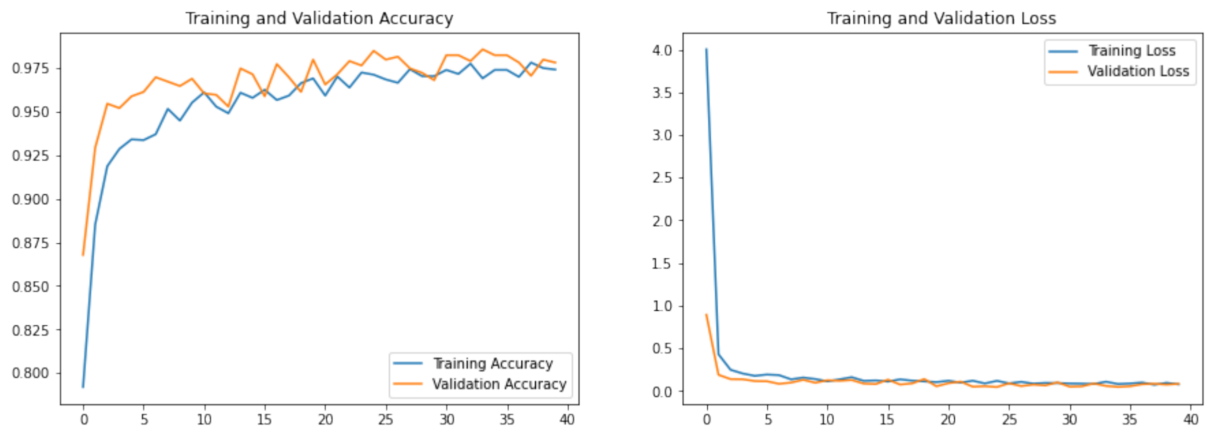


Figure 5.6: Training vs Validation (Accuracy & Loss) for **InceptionV3**

The training vs validation graphs shows that the process does not yield any overfitting results. The validation accuracy and loss are uniformly increasing and decreasing respectively. This shows the result is near perfect with little to no false positives or negatives.

5.2.5 Resnet-50

This model has a very deep neural network of 50 layers as the name suggests. Its parameter size however is not as deep as the VGG network models. The resnet-50 model is a complex network due to its depth. The main disadvantage of ResNets is that for a deeper network, the detection of errors becomes difficult.

	Precision	**Recall**	**F1-score**
Bacterial blight	0.70	0.75	0.73
Blast	0.73	0.35	0.47
Brown spot	0.76	0.80	0.78
Tungro	0.67	0.96	0.79
Weighted avg	0.72	0.71	0.69

Table 5.11: Result of Evaluation Metrics for Resnet-50

When deeper networks are able to start converging, a degradation problem has been exposed: with the network depth increasing, accuracy gets saturated (which might be unsurprising) and then degrades rapidly. This behavior of resnet networks has been observed in papers [11].

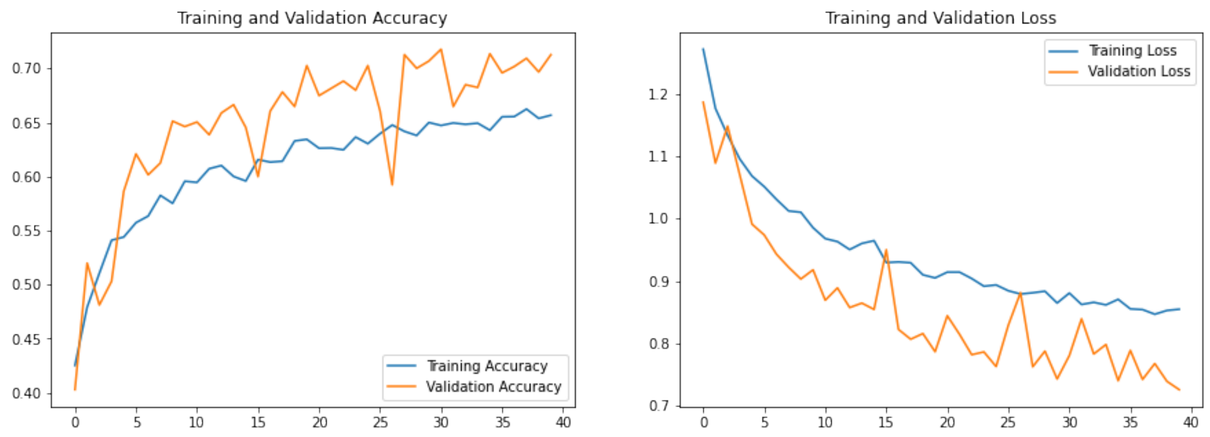


Figure 5.7: Training vs Validation (Accuracy & Loss) for **Resnet-50**

The training and validation curves for both accuracy loss seem to be showing an increasing and decreasing trend. However, this seems to have lots of Fluctuations throughout the process. This means lots of false positives and false negatives have been classified here. This is why resnet-50 gives the least satisfactory result of **71%**

5.2.6 VGG-16

The other VGG network model used in our work is the VGG-16. So, in this case the deep learning model, the VGG-16 performs similarly to the previously used vgg-19 model.

However, The total number of parameters in this model is over 138M, and the size of the model is over 500MB. This puts some serious limitations on the usage of the model, specifically in edge computing, as the inference time required is higher. So, the model itself

	Precision	**Recall**	**F1-score**
Bacterial blight	1.00	0.93	0.96
Blast	1.00	0.94	0.97
Brown spot	0.89	1.00	0.94
Tungro	1.00	1.00	1.00
weigted avg	0.97	0.97	0.97

Table 5.12: Result of Evaluation Metrics for VGG-16

performs very well. But, the resources needed to achieve this result is relatively higher. The accuracy achieved here is around **97%**.

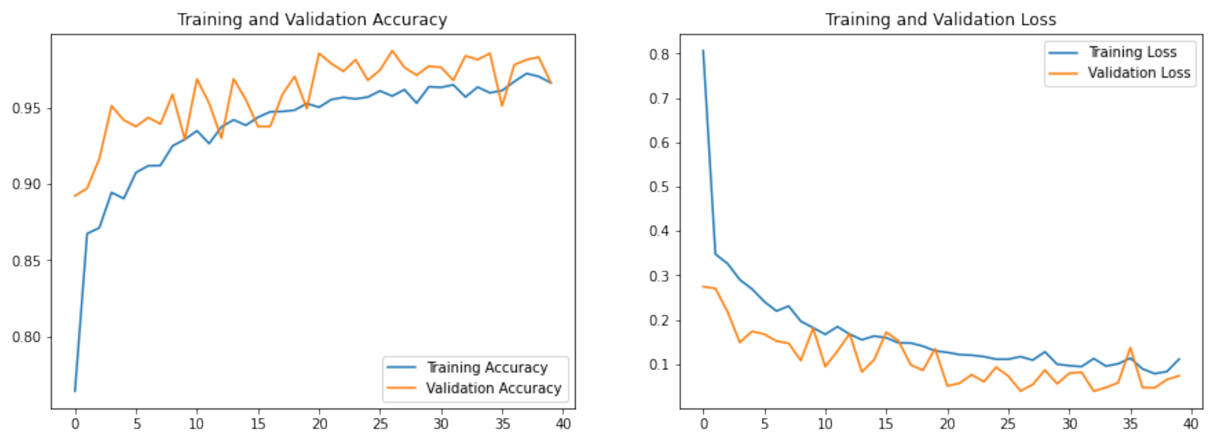


Figure 5.8: Training vs Validation (Accuracy & Loss) for **VGG-16**

The training vs validation graphs shows that the process does not yield any overfitting results. The validation accuracy and loss are uniformly increasing and decreasing respectively. This shows the result is near perfect with little to no false positives or negatives. The VGG network keeping similarity with other models upholds the uniformity of the training vs validation in both cases.

5.2.7 Comparison Between Deep Learning Models

The above discussion of the deep learning models and their training vs validation graphs illustrate the different accuracies achieved for the pre-trained models in comparison to the proposed custom CNN model. It also helps associate the relationship of the precision and recall values of the different models. It shows how training and testing may sometimes yield results that contradict the desired outcome.

However, in this case, almost every model shows that the precision and recall values are similar. Besides ResNet-50, all the other models yield results **97%** and above. So, the results show minimal numbers of false positives and false negatives. This means the validation for

Classifiers	**Accuracy**	**Precision**	**Recall**	**F1-score**
Custom CNN	0.97	0.97	0.97	0.97
VGG-16	0.97	0.97	0.97	0.97
VGG-19	0.98	0.98	0.98	0.98
Inception-V3	0.98	0.98	0.98	0.98
ResNet-50	0.71	0.72	0.71	0.69

Table 5.13: Result of Evaluation Metrics for Deep Learning Models

all the trained images do not deviate from the original training accuracy. Hence, it shows minimal false detections and high accuracy for the rice disease dataset used in this study.

Chapter 6

Future Work and Conclusion

6.1 Conclusion

Performance analysis of automated Rice plant disease classification is performed in our work. Moreover, five traditional classifiers are applied to classify Rice plant diseases in our images. Then a custom CNN model is designed on our own and applied to include deep learning in our work. Transfer Learning Techniques are applied in our work to get higher accuracy. Here, a dataset consisting of 5932 images used of four different Rice plant diseases. They have split into 80 and 20 ratios. Various experiments are applied over the datasets by tunning the parameters and hyperparameters of the model. And among them, the best ones are represented. Our custom CNN model produced an accuracy of 97 percent.

6.2 Limitations

There are some limitations of our thesis work that we have listed in this section which we are leaving to improve in our future works.

- Minimize the number of parameters to optimize the custom CNN model to make it lightweight
- The dataset only has four types of Rice plant diseased images
- Use other plant diseases for classification
- Deep Learning is very hardware dependent and takes a lot of time to prepare in a larger dataset as it is not possible to get a good result without tuning hyperparameter a reasonable amount of time. Therefore, a powerful Quadro GPU is needed to reduce this time-consuming problem, which is economically very expensive.

- We could have tried more feature extraction methods to increase the accuracy of the machine learning classifiers.

6.3 Future Works

There are more opportunities for improvement or research on our work in the future.

- First of all, dataset size can be increased. The bigger the number of images the better the model is trained
- More types of Rice plant diseases can be added to classify a variety of Rice plant diseases
- the system can be transformed into a web application or mobile Application. Using the mobile application the farmers can automatically detect Rice plant diseases

References

[1] S.A Miah, A.K.M Shahjahan, " A Survey of rice diseases in Bangladesh", International Journal of Pest Management, 2008. https://www.tandfonline.com/doi/abs/10.1080/09670878509370984. Accessed: December 11, 2022

[2] sethy, prabira Kumar, “Rice Leaf Disease Image Samples”, Mendeley Data, V1, doi: 10.17632/fwcj7stb8r.1, 2020. https://data.mendeley.com/datasets/fwcj7stb8r/1. Accessed: December 12, 2022

[3] Hasan, Md Jahid ,Mahbub, Shamim Alom, Md. Shahin Nasim, Md., "Rice Disease Identification and Classification by Integrating Support Vector Machine With Deep Convolutional Neural Network". 1-6. 10.1109/ICASERT.2019.8934568.,2019 https://ieeexplore.ieee.org/abstract/document/8934568. Accessed: December 11, 2022

[4] Tejas Tawde , Kunal Deshmukh , Lobhas Verekar , Ajay Reddy, Shailendra Aswale, Pratiksha Shetgaonkar, 2021, "Rice Plant Disease Detection and Classification Techniques : A Survey", INTERNATIONAL JOURNAL OF ENGINEERING RESEARCH TECHNOLOGY (IJERT) Volume 10, Issue 07, 2021, https://www.ijert.org/rice-plant-disease-detection-and-classification-techniques-a-survey. Accessed: December 12, 2022

[5] K. Ahmed, T. R. Shahidi, S. M. Irfanul Alam and S. Momen, "Rice Leaf Disease Detection Using Machine Learning Techniques," 2019 International Conference on Sustainable Technologies for Industry 4.0 (STI), 2019, pp. 1-5, doi: 10.1109/STI47673.2019.9068096., 2019 https://ieeexplore.ieee.org/abstract/document/9068096. Accessed: December 11, 2022

[6] S, Ramesh. "Rice Blast Disease Detection and Classification Using Machine Learning Algorithm". 255-259. 10.1109/ICMETE.2018.00063., 2018 https://ieeexplore.ieee.org/abstract/document/8742850. Accessed: December 11, 2022

[7] Anam Islam, Redoun Haque, S M Rafizul Islam, S.M. Khan, Mohammad. "Rice Leaf Disease Recognition using Local Threshold Based Segmentation and Deep CNN". International Journal of Intelligent Systems and Applications. 13. 35-45. 10.5815/ijisa.2021.05.04., 2021 https://www.semanticscholar.org/paper/Rice-Leaf-Disease-Recognition-using-Local-Threshold-Islam-Islam/cbf4b20aa8d8d000b23a1ac1599b92aa3c535307 Accessed: December 12, 2022

[8] S. Ramesh, D. Vydek, "Application of machine learning in detection of blast disease in South Indian rice crops", 2019 https://updatepublishing.com/journal/index.php/jp/article/view/5476/4893 Accessed: December 12, 2022

[9] Md. Sazzadul Islam Prottasha, A. B. M. Kabir Hossain, Md. Zihadur Rahman, S M Salim Reza, Dilshad Ara Hossain, "Identification of Various Rice Plant Diseases Using Optimized Convolutional Neural Network", 2021 https://journal.uob.edu.bh/handle/123456789/4387?show=full Accessed: December 12, 2022

[10] V. K. Shrivastava, M. K. Pradhan, S. Minz, and M. P. Thakur, "Rice Plant Disease Classification Using Transfer Learning of Deep Convolutional Neural Network", 2019 https://ui.adsabs.harvard.edu/abs/2019ISPAr.423..631S/abstract. Accessed: December 12, 2022

[11] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition", 2015 https://arxiv.org/abs/1512.03385. Accessed: December 12, 2022

[12] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition", 2014 https://arxiv.org/abs/1409.1556. Accessed: December 12, 2022

[13] Paper.pdf%3Ffbclid%3DIwAR1AA3vqQxsaMohPhzkeVf3CRFg5v-KvGU9xUcG5G4k9B8WP3JI28K8isNkh. Accessed: December 12, 2022

[14] https://pub.towardsai.net/the-architecture-and-implementation-of-vgg-16-b050e5a5920b. Accessed: December 12, 2022

[15] V. Sudha, Dr. T. R. Ganeshbabu, "A Convolutional Neural Network Classifier VGG-19 Architecture for Lesion Detection and Grading in Diabetic Retinopathy Based on Deep Learning", 2020
https://www.researchgate.net/publication/346716209. Accessed: December 12, 2022

[16] https://blog.techcraft.org/vgg-19-convolutional-neural-network. Accessed: December 12, 2022

[17] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, "Rethinking the Inception Architecture for Computer Vision" https://www.cv-foundation.org/openaccess/content$_{c}vpr_{2}016/papers$. Accessed: December 12, 2022

[18] https://iq.opengenus.org/inception-v3-model-architecture

[19] Kaiming, He Xiangyu Zhang, Shaoqing Ren, Jian Sun, "Deep Residual Learning for Image Recognition" https://arxiv.org/pdf/1512.03385.pdf. Accessed: December 12, 2022

[20] www.researchgate.net%2Ffigure%2FResNet-50-architecture-26-shown-with-the-residual-units-the-size-of-the-filter. Accessed: December 12, 2022

[21] Ho TK (1998). “The Random Subspace Method for Constructing Decision Forests”, IEEE Transactions on Pattern Analysis and Machine Intelligence. 20 (8): 832-844. https://ieeexplore.ieee.org/document/709601. Accessed: December 12, 2022

[22] Derek A. Pisner, David M. Schnyer, "Support vector machine", 2020 https://www.sciencedirect.com/topics/neuroscience/support-vector-machine?fbclid=IwAR1rZw-9JuabKcxx$_{U}0pkqQTzXdvum6pOi3WrVs7LZBAGi6DM_{5}0uEtsfLA$. Accessed: December 12, 2022

[23] Jin-Ho Kim, Byung-soo Kim, S. Savarese, "Comparing image classification methods: K-nearest-neighbor and support-vector-machines", 2012 https://www.semanticscholar.org/paper/Comparing-image-classification-methods%3A-and-Kim-Kim/6bad2168f4f3a0d3f51d33a2d85a7efb4f76f412. Accessed: December 12, 2022

[24] Yanqiu Zhang; Ming Ni, Chengwu Zhang, Shuang Liang, Sheng Fang, Ruijie Li, Zhouyu Tan"Research and Application of AdaBoost Algorithm Based on SVM",2019 https://ieeexplore.ieee.org/document/8785556. Accessed: December 12, 2022

[25] J.R. QUINLAN, "Induction of Decision Trees",1986

https://hunch.net/ coms-4771/quinlan.pdf. Accessed: December 12, 2022

[26] https://www.interviewbit.com/blog/cnn-architecture/. Accessed: December 12, 2022

[27] https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2. Accessed: December 12, 2022

[28] https://paperswithcode.com/method/max-pooling. Accessed: December 12, 2022

[29] https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening. Accessed: December 12, 2022

[30] https://wenkangwei.github.io/2020/11/13/DL-DropOut/. Accessed: December 12, 2022

[31] https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection. Accessed: December 12, 2022

[32] http://www.knowledgebank.irri.org/decision-tools/rice-doctor/rice-doctor-fact-sheets/item/bacterial-blight. Accessed: December 12, 2022

[33] http://www.knowledgebank.irri.org/training/fact-sheets/pest-management/diseases/item/blast-leaf-collar. Accessed: December 12, 2022

[34] http://www.knowledgebank.irri.org/training/fact-sheets/pest-management. Accessed: December 12, 2022

[35] http://www.knowledgebank.irri.org/training/fact-sheets/pest-management/diseases/item/tungro. Accessed: December 12, 2022

[36] A Review Paper on Gabor Filter Algorithm for The application of Texture Segmentation. Accessed: December 12, 2022

[37] Gabor Filter and Rough Clustering Based Edge Detection . Accessed: December 12, 2022

[38] Sobel Edge Detection. Accessed: December 12, 2022

[39] Shiv Ram Dubey, Satish Kumar Singh, Bidyut Baran Chaudhuri, " Activation Functions in Deep Learning: A Comprehensive Survey and Benchmark" https://arxiv.org/abs/2109.14545. Accessed: December 12, 2022

[40] Meiqi Wang, Siyuan Lu, Danyang Zhu, Jun Lin, Zhongfeng Wang, "A High-Speed and Low-Complexity Architecture for Softmax Function in Deep Learning" https://ieeexplore.ieee.org/abstract/document/8605654. Accessed: December 12, 2022

[41] Abien Fred Agarap, "Deep Learning using Rectified Linear Units (ReLU)", 2018 https://arxiv.org/abs/1803.08375. Accessed: December 12, 2022

[42] https://towardsdatascience.com/softmax-activation-function-how-it-actually-works-d292d335bd78. Accessed: December 12, 2022

[43] Ibrahem Kandel, Mauro Castelli,"The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset", 2020 https://www.sciencedirect.com/science/article/pii/S2405959519303455 ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION. Accessed: December 12, 2022

[44] Diederik P. Kingma, Jimmy Ba,"Adam: A Method for Stochastic Optimization", 2017. https://arxiv.org/abs/1412.6980. Accessed: December 12, 2022

[45] Andrew P.Bradley,"The use of the area under the ROC curve in the evaluation of machine learning algorithms",1997. https://www.sciencedirect.com/science/article/abs/pii/S0031320396001422. Accessed: December 12, 2022

[46] https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning. Accessed: December 12, 2022