Rice Disease and Pest Detection Using Deep Learning

Md. Minhaz Ul Karim

ID: 2017-3-96-003

A thesis submitted in partial fulfillment of the requirements for the degree of

Masters of Science in Computer Science

Department of Computer Science & Engineering

East West University

DECLARATION

I hereby declare that I have completed thesis on the topic entitled "Rice Disease and Pest Detection Using Deep Learning" as well as prepared the thesis report under the supervision of Dr. Ahmed Wasif Reza, Associate Professor, Department of Computer Science Engineering. This report is submitted to the department of Computer Science Engineering, East West university in partial fulfillment of the requirement for the degree of MS in CSE, Under the course "MS Thesis (CSE 599)".

I further assert that this report in question is based on my original exertion having never been produced fully and/or partially anywhere for any requirement.

Countersigned

….....

Dr. Ahmed Wasif Reza
Associate Professor,
Department of Computer Science & Engineering

Signature

….....

Md. Minhaz Ul Karim
ID: 2017-2-96-003

LETTER OF ACCEPTANCE

This thesis report "Research on Rice Disease and Pest Detection using Deep Learning" is the outcome of the original work carried out by Md. Minhaz Ul Karim, ID: 2017-2-96-003, under my supervision to the Department of Computer Science & Engineering, East West University, Dhaka–1212

Supervisor

….....

Dr. Ahmed Wasif Reza

Associate Professor,

Department of Computer Science & Engineering,

East West University

Chairperson

….....

Dr. Taskeed Jabid

Associate Professor and Chairperson,

Department of Computer Science & Engineering,

East West University

ABSTRACT

In the field of agriculture information, automatic detection and diagnosis of plant disease and pest is highly desirable. Feature extraction technologies play a critical and crucial role in leaf disease detection and diagnostic system. Researches in leaf disease detection have used many different feature detection techniques like color, texture, shape etc. Recently very promising results are found using deep learning in different types of computer vision problems. Now a days deep learning is hot research topic in pattern recognition, machine learning as well as artificial intelligence. Deep neural network-based models can be an effective solution to vegetable pathology. In this research I have proposed a novel rice disease and pest detection model which is based on deep convolutional neural networks (CNN). This model gives a training accuracy of 80.11% with 77.68% training accuracy.

….....

Md. Minhaz Ul Karim

ID: 2017-2-96-003

ACKNOWLEDGEMENT

First and foremost, with all my heartiest devotion I am grateful to almighty Allah for blessing me with such opportunity of learning and ability to successfully complete the research.

A special thanks with the honor to my respected supervisor Dr. Ahmed Wasif Reza who was kind enough to allocate his valuable time to provide me with his humble guidance, motivating thought and encouragement. Without his guidance this work would not have been possible.

Table of Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

1.1. Background of Thesis

Rice is the major food crop in Bangladesh. Rice fills almost 70 percent of the grossed crop area and 93 percent of total cereal production in Bangladesh **[1]**. It is also primary staple food in many countries. With the growth of population, demand of rice is increasing as well. Unfortunately, a great loss in yield is caused by rice diseases. When a rice disease spread out somewhere, Government appoints consultants or agriculture officers to advice the farmers. The whole process is time consuming. Farmers in outlier area sometimes do not even get the facilities in time. Plant disease is not only a threat to food supply but also bring in horrible consequences for small holder farmers. Their livelihood depends on healthy crops, which is heavily affected by any kinds of crop perishing epidemic. In developing countries like Bangladesh around 80 percent agricultural production is generated by small holder farmers and yield loss of almost 50 percent is very common due to pests and diseases **[2]**. That is why timely detection of diseases and pests is one of the major issues in agriculture sector.

On the other hand, technologies have reached every corner of the world. Using technology to solve any problem is not anything surprising anymore. Smart phones in particular can be very handy tool to detect plant diseases and pests. The devices possess tremendous computing power, high-resolution displays, and extensive built-in sets of accessories such as advanced HD camera. There is estimation that there will be around 6 billion smart phones by the end of 2020. The combined factors of widespread use of such devices and their processing power along with HD cameras can lead to a situation where disease and pests detection can be made available to an unprecedented scale.

Deep learning is a very promising technique for image classification. This technique is based on feature learning from labeled training dataset. Computer vision and object detection have achieved a great advancement in the recent years. The PASCALVOC Challenge, and more recently the Large-Scale Visual Recognition Challenge (ILSVRC) based on the ImageNet dataset have been widely used as benchmarks for numerous visualization-related problems in computer vision including object classification. In 2012, a large, deep convolutional neural network achieved a top-5 error of 16.4% for the classification of images into 1,000 possible categories.In the following three years, various advances in deep convolutional neural networks lowered the error rate to 3.57%. While training large neural networks can be very time-consuming, the trained models can classify images fairly quickly.

In recent years, deep learning techniques have been used to analyze diseases of tea **[3]**, apple **[4]**, tomato **[5]**, grapevine, peach, and pear **[6]**. Most of the cases, they have used leaves or fruits to detect the diseases from the images from homogeneous backgrounds. Two studies related to rice disease detection can be found in **[7]** and **[8]**. Lu et al conducted a study on detecting 10 different rice plant diseases using a small handmade Convolutional Neural Network architecture inspired by older deep learning frameworks such as LeNet-5 and AlexNet **[7]**.

1.2. Problem Statement

Generally, farmers in our country have low knowledge about plant diseases. It is very often that an epidemic spread over a large area and the farmers have not even idea what has affected them. Due to lack of awareness they hesitate to consult an agriculturist concerning their crop which is really troublesome for farmers in outlying area. An easy access about the wellbeing of their crop will really benefit them. If they get to know what is affecting their plant without any hassle then lots of time and money would be saved.

1.3. Objective of Thesis

The main goal of this thesis is to analyze if it is possible to detect rice leaf diseases from images. Although it will help farmers to detect rice leaf diseases without having much knowledge about plant diseases. This will also reduce the time of detecting disease as no need to wait for an agriculturist to visit physically.

1.4. Chapterization Plan

The following are the overview of chapters and contents of this report,

Chapter 2

Chapter 2 includes a detailed literature review of supervised learning, classification model, learning types, deep learning, CNN, architecture of Mobilenet, Resnet, rice diseases and insects etc.

Chapter 3

Chapter 3 gives the detailed information about methodology, sample data, architecture of the proposed model, how the data has been split for training, testing and evaluation

Chapter 4

Chapter 4 provides the detailed result of the thesis along with graphs of training, testing accuracy and loss. Also provides the table for comparison between training, test and evaluation results of the three models.

Chapter 5

Chapter 5 is the conclusion chapter which includes summary of findings along with the conclusion to the thesis followed by future work for this thesis.

CHAPTER 2

LITERATURE REVIEW

The goal of this section is to gather and include the knowledge required to understand the methodology of this work in an organized way. In that way, the problem we are working with can be understood as well as attempted to solve better.

2.1. Supervised Learning

Supervised learning, in the context of artificial intelligence (AI) and machine learning, is a type of system in which both input and desired output data are provided **[9]**. Input and output data are labeled for classification to provide a learning basis for future data processing. For example, in a supervised learning system, we input age, height, weight of a human being and provide outputs or labels to these instances whether the person is a male or a female. The machine learns from this instance. Studying the pattern of age, height and weight the machine comes up with a rule or base on which it will label unknown instances as male or female. After the machine has been trained (learned) we give it some 15 inputs and it provides us the outputs, i.e. labels the new instance. This is a high-level overview of supervised learning.

The aim of supervised learning is to build a concise model of the distribution of class labels regarding the input features provided. When provided with more observations, the machine improves its performance, i.e. learns better. The resulting classifier model is then used to assign labels to the test instances where the values of the input features are known, but the value of the label is unknown. The whole supervised learning procedure is depicted in the following diagram:

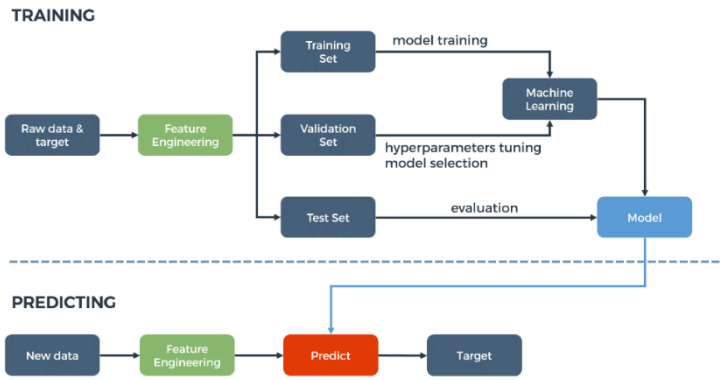

Fig 2.1: Process of Supervised Learning

2.2. Classification Model

The process of predicting the class from given data points is called classification. The classes are called as categories or targets/labels. Classification predictive modeling is the task of approximating a mapping function (f) from input variables (X) to discrete output variables (y).

For example, spam detection in email service providers can be identified as a classification problem. There are only 2 classes, spam and not spam. So, this is a binary classification problem. A classifier understands the class by utilizing the training data to understand the input variables related to the class. In this case, known spam and non-spam emails are used as training data. After the training is done, it can be used to detect an unknown email.

Classification belongs to the category of supervised learning where the targets also provided with the input data. There are many applications in classification in many domains such as in credit approval, medical diagnosis, target marketing etc.

There are two types of learners in classification as lazy learners and eager learners.

2.2.1. Lazy Learning

The computation undertaken by a learning system can be viewed as occurring at two distinct times, training time and consultation time. Consultation time is the time between when an object is presented to a system for an inference to be made and the time when the inference is completed. Training time is the time prior to consultation time during which the system makes inferences from training data in preparation for consultation time. Lazy learning refers to any machine learning process that defers the majority of computation to consultation time. Two typical examples of lazy learning are instance-based learning and Lazy Bayesian Rules. Lazy learning stands in contrast to eager learning in which the majority of computation occurs at training time. **[10]**

2.2.2. Eager Learning

In artificial intelligence, eager learning is a learning method in which the system tries to construct a general, input-independent target function during training of the system, as opposed to lazy learning, where generalization beyond the training data is delayed until a query is made to the system. **[11]** The main advantage gained in employing an eager learning method, such as an artificial neural network, is that the target function will be approximated globally during training, thus requiring much less space than

using a lazy learning system. Eager learning systems also deal much better with noise in the training data. Eager learning is an example of offline learning, in which post-training queries to the system have no effect on the system itself, and thus the same query to the system will always produce the same result.

The main disadvantage with eager learning is that it is generally unable to provide good local approximations in the target function.

2.3. Deep Learning

Deep Learning is a subfield of machine learning concerned with algorithms inspired by the structure and function of the brain called artificial neural networks. It is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible

potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

2.4. CNN

In deep learning, a convolutional neural network (CNN) is a class of deep neural networks, most commonly applied to analyzing visual imagery.

CNNs are regularized versions of multilayer perceptions. Multilayer perceptron usually means fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "fully-connectedness" of these networks makes them prone to overfitting data. Typical ways of regularization include adding some form of magnitude measurement of weights to the loss function. However, CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble more complex patterns using smaller and simpler patterns. Therefore, on the scale of connectedness and complexity, CNNs are on the lower extreme.

They are also known as shift invariant or space invariant artificial neural networks (SIANN), based on their shared-weights architecture and translation invariance characteristics. **[12]**

A convolution is the simple application of a filter to an input that results in an activation. Repeated application of the same filter to an input results in a map of activations called a feature map, indicating the locations and strength of a detected feature in an input, such as an image.

The innovation of convolutional neural networks is the ability to automatically learn a large number of filters in parallel specific to a training dataset under the constraints of a specific predictive modeling problem, such as image classification. The result is highly specific features that can be detected anywhere on input images.

2.5. Mobilenet

MobileNet is an architecture which is more suitable for mobile and embedded based vision applications where there is lack of compute power. This architecture was proposed by Google. This architecture uses depth wise separable convolutions which significantly reduces the number of parameters when compared to the network with normal convolutions with the same depth in the networks. This results in light weight deep neural networks. The normal

convolution is replaced by depth wise convolution followed by pointwise convolution which is called as depth wise separable convolution.

This results in the reduction of number of parameters significantly and thereby reduces the total number of floating-point multiplication operations which is favorable in mobile and embedded vision applications with less compute power. By using depth wise separable convolutions, there is some sacrifice of accuracy for low complexity deep neural network.

The following table shows the layer level architecture of mobilenet model,

Layer Type	**Configuration**
Input Image	Input Raw Image (224 X 224 X 3)
Convolution	k=3 X 3 X 3 X 32, s=2 X 2
Convolution dw	k=3 X 3 X 32 dw, s=1 X 1
Convolution	k=1 X 1 X 32 X 64, s=1 X 1
Convolution dw	k=3 X 3 X 64 dw, s=2 X 2
Convolution	k=1 X 1 X 64 X 128, s=1 X 1
Convolution dw	k=3 X 3 X 128 dw, s=1 X 1
Convolution	k=1 X 1 X 128 X 128, s=1 X 1
Convolution dw	k=3 X 3 X 128 dw, s=2 X 2
Convolution	k=1 X 1 X 128 X 256, s=1 X 1
Convolution dw	k=3 X 3 X 256 dw, s=1 X 1
Convolution	k=1 X 1 X 256 X 256, s=1 X 1
Convolution	k=3 X 3 X 256, s=2 X 2
Convolution	k=1 X 1 X 256 X 512, s=1 X 1
5 X (Convolution dw, Convolution)	k=3 X 3 X 512 dw, s=1 X 1 k=1 X 1 X 512 X 512, s=1 X 1
Convolution dw	k=3 X 3 X 512 dw, s=2 X 2
Convolution	k=1 X 1 X 512 X 1024, s=1 X 1
Convolution dw	k=3 X 3 X 1024 dw, s=2 X 2
Convolution	k=1 X 1 X 1024 X 1024, s=1 X 1
Average Pooling	Pool=7 X 7, s=1 X 1
Fully Connected	Output Shape = 1024 X 1000
Classifier	1 X 1 X 1000

Table 2.1: Mobilenet architecture

2.6. Resnet

According to the universal approximation theorem, given enough capacity, we know that a feedforward network with a single layer is sufficient to represent any function. However, the layer might be massive and the network is prone to overfitting the data. Therefore, there is a common trend in the research community that our network architecture needs to go deeper.

Since AlexNet, the state-of-the-art CNN architecture is going deeper and deeper. While AlexNet had only 5 convolutional layers, the VGG network **[13]** and GoogleNet (also codenamed Inception_v1) **[14]** had 19 and 22 layers respectively.

However, increasing network depth does not work by simply stacking layers together. Deep networks are hard to train because of the notorious vanishing gradient problem — as the gradient is back-propagated to earlier layers, repeated multiplication may make the gradient infinitively small. As a result, as the network goes deeper, its performance gets saturated or even starts degrading rapidly.

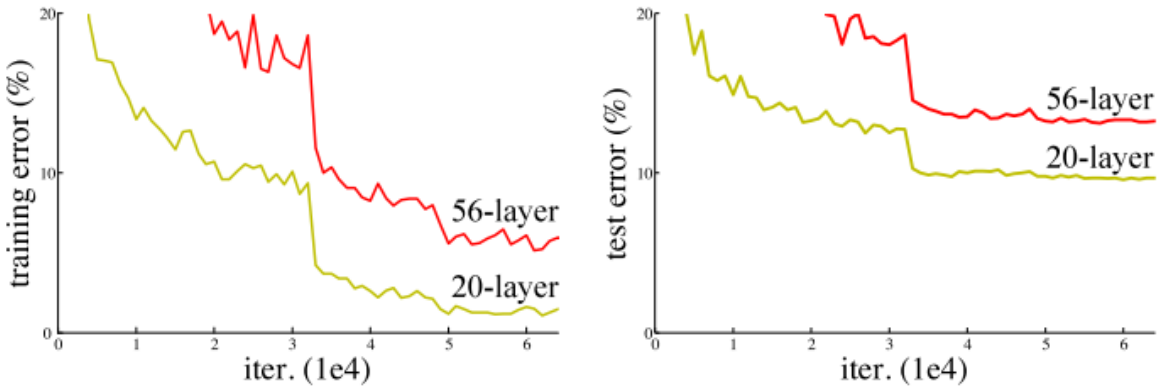

Fig 2.2: Increasing network depth leads to worse performance

Before ResNet, there had been several ways to deal the vanishing gradient issue, for instance, **[14]** adds an auxiliary loss in a middle layer as extra supervision, but none seemed to really tackle the problem once and for all.

The core idea of ResNet is introducing a so-called “identity shortcut connection” that skips one or more layers, as shown in the following figure,

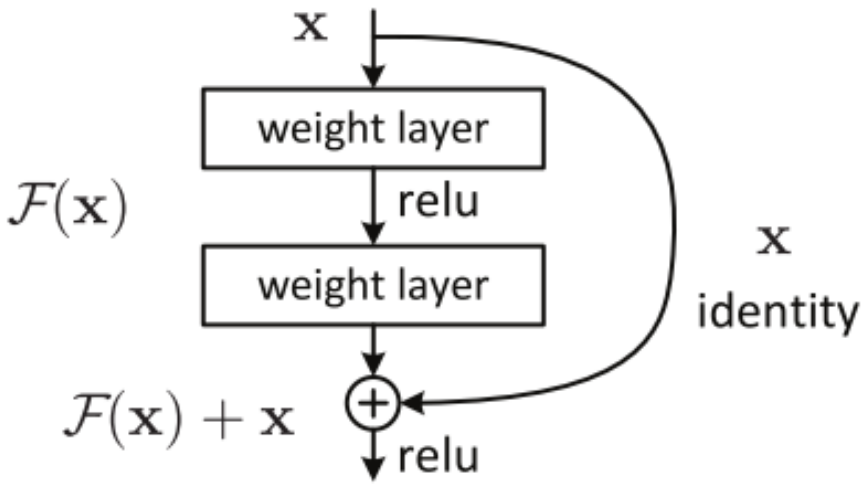

Fig 2.3: A residual block

The following table shows the architecture of resnet50,

Layer Type	**Configuration**
Input Image	Input Raw image (224 X 224 X 3)
Convolution	k=7 X 7 s=2 X 2
Max Pooling	Pool=3 X 3 s=2 X 2
Convolution Block	Convolution Block * 1
Identity Block	Identity Block * 2
Convolution Block	Convolution Block * 1
Identity Block	Identity Block * 3
Convolution Block	Convolution Block * 1
Identity Block	Identity Block * 5
Convolution Block	Convolution Block * 1
Identity Block	Identity Block * 2
Average Pooling	s=1 X 1

Table 2.2: Resnet50 architecture

The following figure shows what an identity block and convolution block consist of,

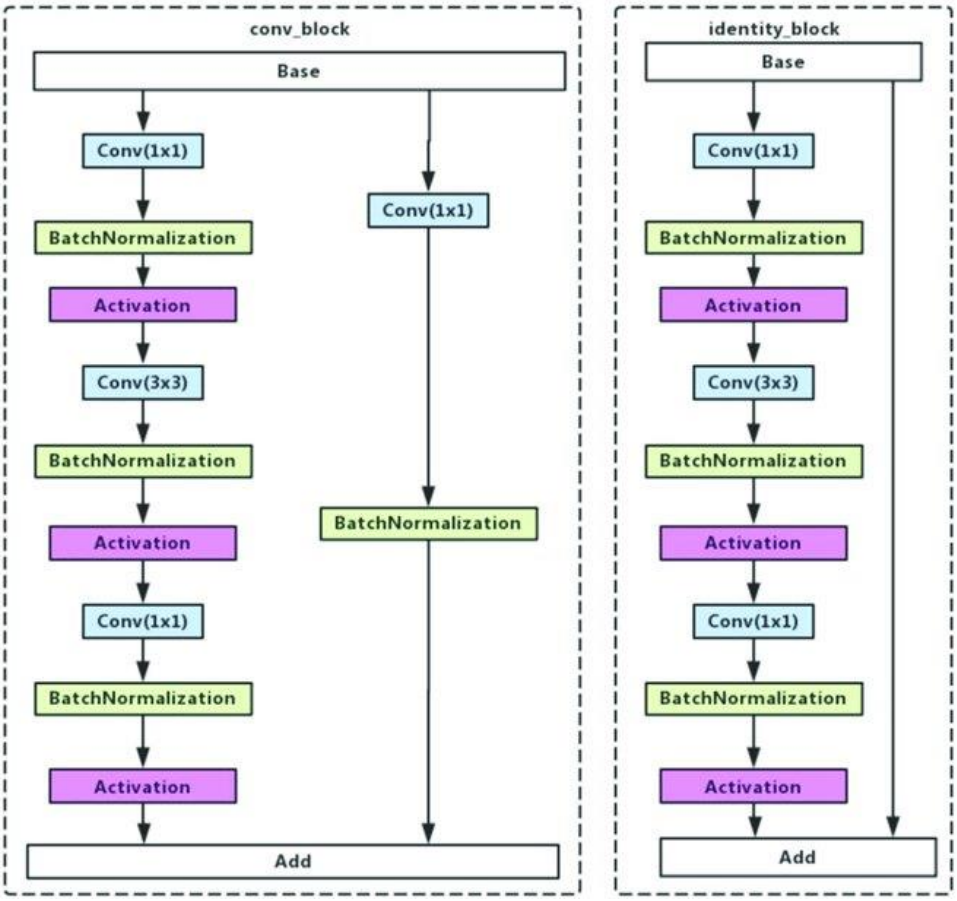

Fig 2.4: Convolution Block and Identity Block for resnet

2.8. Rectified Linear Unit (ReLU)

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

The rectified linear activation function is a piecewise linear function that will output the input directly if is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

Rectified linear units find applications in computer vision **[15]** and speech recognition using deep neural nets

2.9. Softmax

Softmax function, a wonderful activation function that turns numbers aka logits into probabilities that sum to one. Softmax function outputs a vector that represents the probability distributions of a list of potential outcomes. It's also a core element used in deep learning classification tasks.

In mathematics, the softmax function, also known as softargmax or normalized exponential function, is a function that takes as input a vector of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers. That is, prior to applying softmax, some vector components could be negative, or greater than one; and might not sum to 1; but after applying softmax, each component will be in the interval (0,1) and the components will add up to 1, so that they can be interpreted as probabilities. Furthermore, the larger input components will correspond to larger probabilities. Softmax is often used in neural networks, to map the non-normalized output of a network to a probability distribution over predicted output classes.

2.10. Rice Diseases

Disease damage to rice can greatly reduce yield. They are mainly caused by bacteria, viruses, or fungi. Planting a resistant variety is the simplest and, often, the most cost effective management for diseases.

The followings are different types of rice disease:

- **Bacterial Blight**

  Bacterial blight is caused by Xanthomonas oryzae pv. oryzae. It causes wilting of seedlings and yellowing and drying of leaves.

- **Bacterial Leaf Streak**

  Bacterial leaf streak is caused by *Xanthomonas oryzae* pv. *Oryzicola*. Infected plants show browning and drying of leaves. Under severe conditions, this could lead to reduced grain weight due to loss of photosynthetic area.

- **Blast (Leaf and Collar)**
  Blast is caused by the fungus Magnaporthe oryzae. It can affect all above ground parts of a rice plant: leaf, collar, node, neck, parts of panicle, and sometimes leaf sheath.
- **Brown Spot**
  Brown spot has been historically largely ignored as one of the most common and most damaging rice diseases.

- **False Smut**
  False smut causes chalkiness of grains which leads to reduction in grain weight. It also reduces seed germination.
- **Rice Grassy Stunt**
  Rice grassy stunt virus reduces yields by inhibiting panicle production.
- **Rice Ragged Stunt**
  Rice ragged stunt virus reduces yield by causing partially exerted panicles, unfilled grains and plant density loss. It is vector-transmitted from one plant to another by brown plant hoppers. Leaves of infected plants have a ragged appearance.
- **Sheath Blight**
  Sheath blight is a fungal disease caused by Rhizoctonia solani. Infected leaves senesce or dry out and die more rapidly, young tillers can also be destroyed.
- **Tungro**
  Tungro infects cultivated rice, some wild rice relatives and other grassy weeds commonly found in rice paddies**.**
- **Leaf Scald**
  Leaf scald is a fungal disease caused by Microdochium oryzae, which causes the scalded appearance of leaves.
- **Narrow Brown Spot**
  Narrow brown spot (also called narrow brown leaf spot, or rice Cercospora leaf spot) is caused by the fungus Sphaerulina oryzina (syn. Cercospora janseana, Cercospora oryzae) and can infect leaves, sheaths, and panicles.
- **Red Stripe**
  Red stripe causes formation of lesions on leaves.
- **Bakanae**
  Bakanae is a seedborne fungal disease. The fungus infects plants through the roots or crowns. It then grows systemically within the plant.
- **Sheath Rot**
  Sheath rot is caused by Sarocladium oryzae.
- **Blast (Node and Neck)**
  Blast is caused by the fungus Magnaporthe oryzae. It can affect all above ground parts of a rice plant: leaf, collar, node, neck, parts of panicle, and sometimes leaf sheath.
- **Stem rot**
  Stem rot leads to formation of lesions and production of chalky grains and unfilled panicles.
- **Bacterial sheath brown rot**
  Sheath brown rot is caused by Pseudomonas fuscovaginae. It causes rotting in sheaths and grains of seedlings and mature plants.
- **Rice Stripe Virus Disease**
  Rice stripe virus disease (RSVD) can cause high yield losses when severe epidemics occur.
- **Rice Yellow Mottle Virus**
  Rice Yellow Mottle Virus (RYMV) is endemic and largely restricted to the African continent, where it has been found in most of the rice-growing countries. The virus has also been reported in Turkey.

2.11. Insects in Rice Leaf

Insect pests attack all portions of the rice plant and all stages of plant growth. Feeding guilds consist of root feeders, stem borers, leafhoppers and planthoppers, defoliators, and grain sucking insects. Insects also attack rice grains in storage

The followings are different types of insect that attacks on rice:

- **Black Bug**
- **Zigzag Leafhopper**
- **Rice Skipper**
- **Rice Thrips**
- **Rice Whorl Maggot**
- **Mealy Bug**
- **Mole Cricket**
- **Ant**
- **Armyworm**
- **Green Semilooper**
- **Greenhorned Caterpillar**
- **Rice Bug**
- **Planthopper**
- **Field Cricket**
- **Cutworm**
- **Green Leafhopper**
- **Rice Caseworm**
- **Grasshopper (Short-horned) and Locust**
- **Rice Gall Midge**
- **Rice Hispa**
- **Stem Borer**
- **Root Aphid**
- **Rice Leaf Folder**
- **Root Grub**

CHAPTER 3

METHODOLOGY

The Methodology section discusses the techniques used to conduct this research work. The primary step was to aggregate training and testing data to analyze and learn from.

3.1. Tools

- Python programming language is used for coding
- scikit learn, keras, tensorflow etc machine learning based python api are used to create the model architecture
- Mobilenet and resnet50 pre-trained models are used for training

3.2. Data

Here open data of rice disease and pest image dataset is used for training the model. This dataset is a collection of total 3355 images. This collection has four different types of images. Among them two are disease, one is pest and healthy type. Brownspot and LeafBlast are the two diseases whereas Hispa is the pest. In this collection the number of images per class is as following: Brownspot – 523, LeafBlast – 779, Hispa – 565. This dataset has 1488 healthy images.

These images are divided for train and test using the train_test_split method of scikit learning api in python. The split ratio is as following: Train 80%, Test 20%. When splitting the images are chosen randomly.

The sample from dataset is given below,

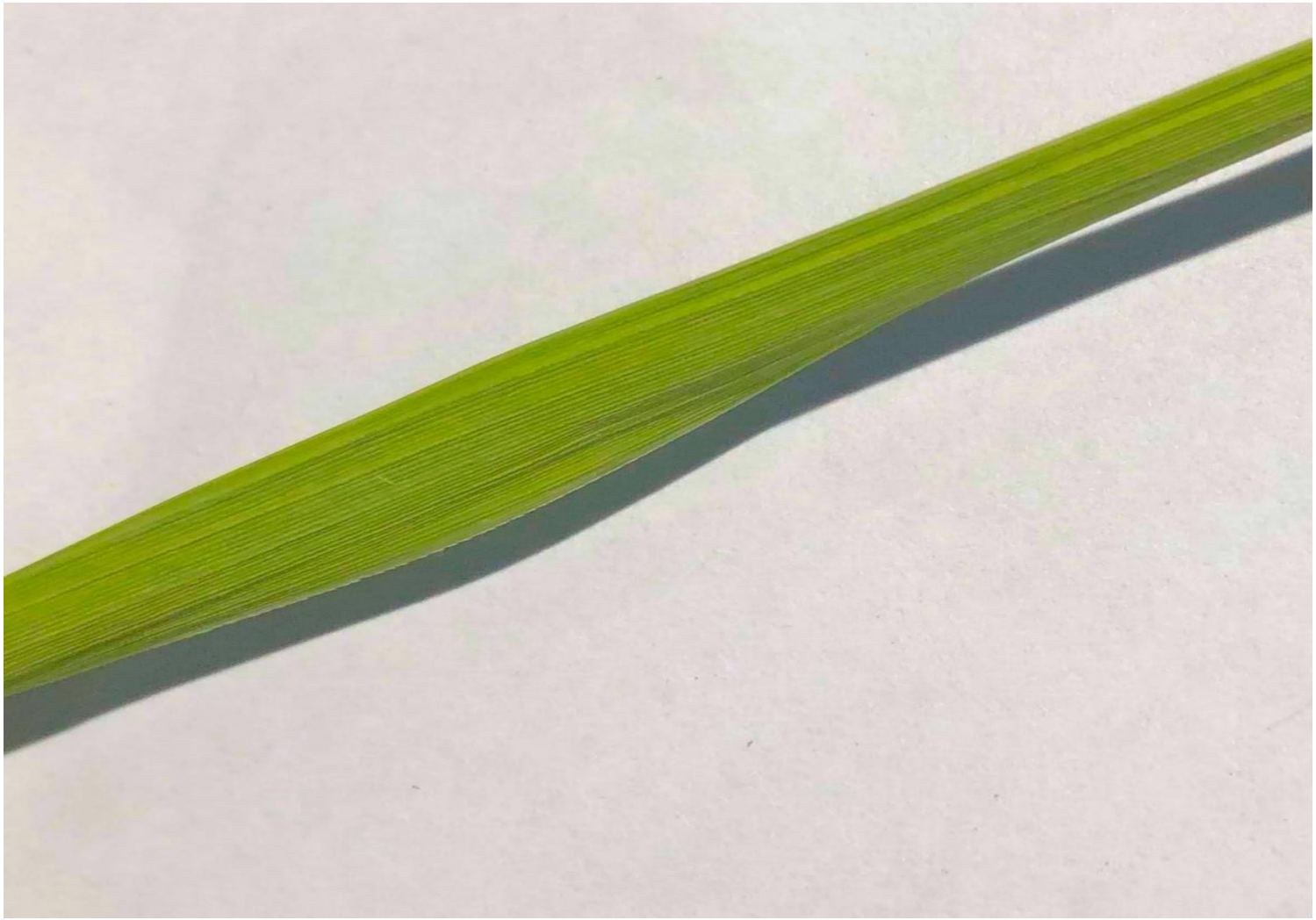

Fig 3.1: Healthy image

Fig 3.2: Brownspot disease image

Fig 3.3: LeafBlast disease image

Fig 3.4: Hispa infected image

3.3. Models for Training

For training here 3 models are used. Two of them are pre-trained. Another one model is developed by myself (Proposed Model). The pre-trained models are mobilenet and resnet 50. Both of the models are trained earlier using millions of images from the imagenet dataset. These models can classify more than 1000 types of objects.

3.4. Mobilenet

Mobilenet model consists of total 3,394,116 parameters. Among them 165,252 are trainable and 3,228,864 are non-trainable. In this configuration the number of trainable parameters are very low as here we have used pre-trained imagenet weights. Here only have trained last 2 dense layers.

3.5. Resnet50

Resnet model consists of multiples of identity and convolution blocks. Each of the identity and convolution blocks are consist of convolution layers along with normalization and activation. In resnet50 total number of layers are 50. Here this model has total 23,884,036 parameters where 296,324 are trainable and 23,587,712 non-trainable.

3.6. Proposed Model

The proposed model consists of 7 layers. Among them 5 are convolution and 2 are fully connected dense layers. The following table shows the layer level architecture of the proposed model,

Layer Type	Configuration
Input	Input Raw Image (256 X 256 X 3)
Convolution	maps=32, k=3 X 3 s=1 X 1
Max Pooling	k=3 X 3
Convolution	maps=64, k=3 X 3
Convolution	maps=64, k=3 X 3
Max Pooling	k=2 X 2
Convolution	maps=128, k=3 X 3
Convolution	maps=128, k=3 X 3
Max Pooling	k=2 X 2
Dense	Output Shape=1024
Dense	Output Shape=4

Table 3.1: Architecture of the proposed model

For all the convolution layers used activation function is Rectified Linear Unit (ReLU). Finally, after going through all the layers it has been passed through a softmax activation function to get normalized weighted values.

This model has a total of 58,091,396 parameters. 58,088,516 of the parameters are trainable and 2880 parameters are non-trainable.

As this proposed model has a less layer than both of the mobilenet and resnet50 it will take less computation power for training.

3.7. Training

Both of the 3 models have been trained using the dataset. The dataset has been randomly split accordingly 70% for training and 30% for testing.

For the mobilenet and resnet50 after 20 epochs the model converges to its optimum accuracy, and the model is not further trained.

For the proposed model it needs to run 30 epochs to get the optimum accuracy for this model.

3.8. Testing

After training the trained model has been saved for highest training accuracy. Then the saved model is tested using the 30% testing data to get the test accuracy.

CHAPTER 4

RESULTS & EVALUATION

4.1. Results

This section contains the result found by training the models. These results as well as plots are based on training, validation accuracy and training, validation loss.

The following table shows detailed comparison of accuracy and loss for training and validation along with testing accuracy for the trained 3 models,

Model	Training Accuracy (%)	Training Loss	Validation Accuracy (%)	Validation Loss	Test Accuracy (%)
Proposed Model	**80.11**	**0.4321**	**77.68**	**0.4790**	**77.6812**
Mobilenet	76.64	0.5017	74.38	0.5277	74.45
Resnet50	82.27	0.3867	75.00	0.5332	73.34

Table 4.1: Table for result comparison

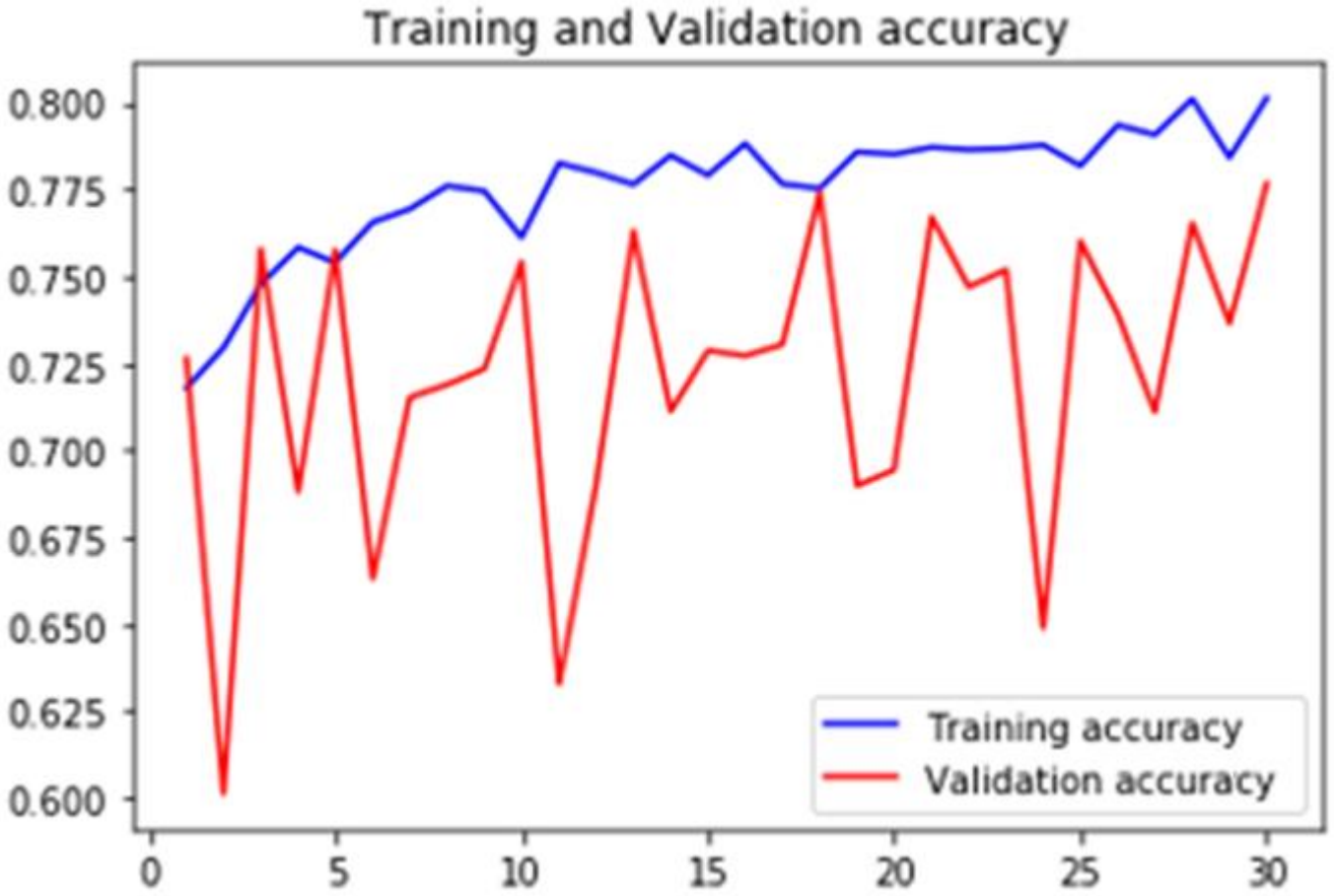

Fig 4.1: Training and validation accuracy for the proposed model

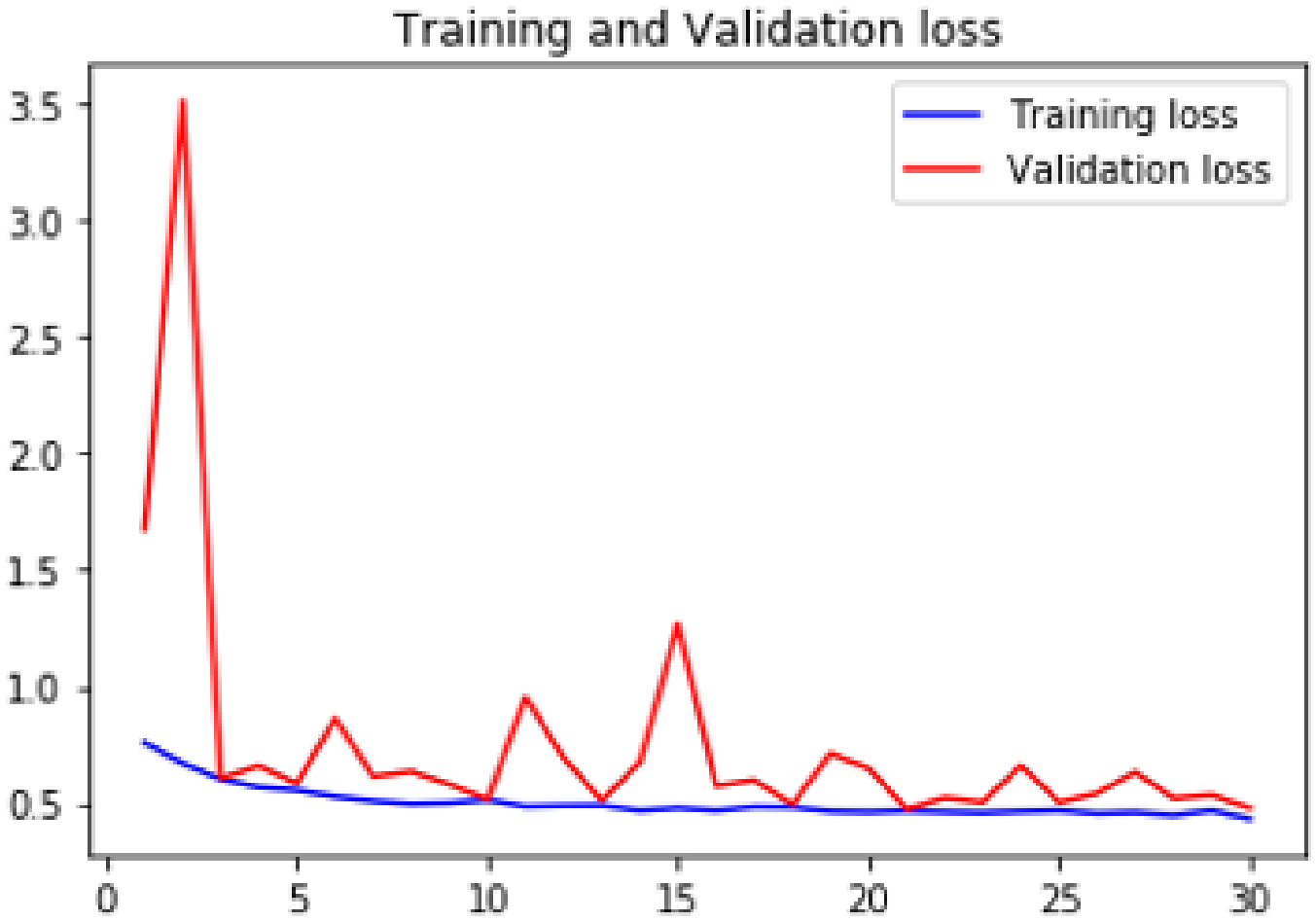

Fig 4.2: Training and validation loss for the proposed model

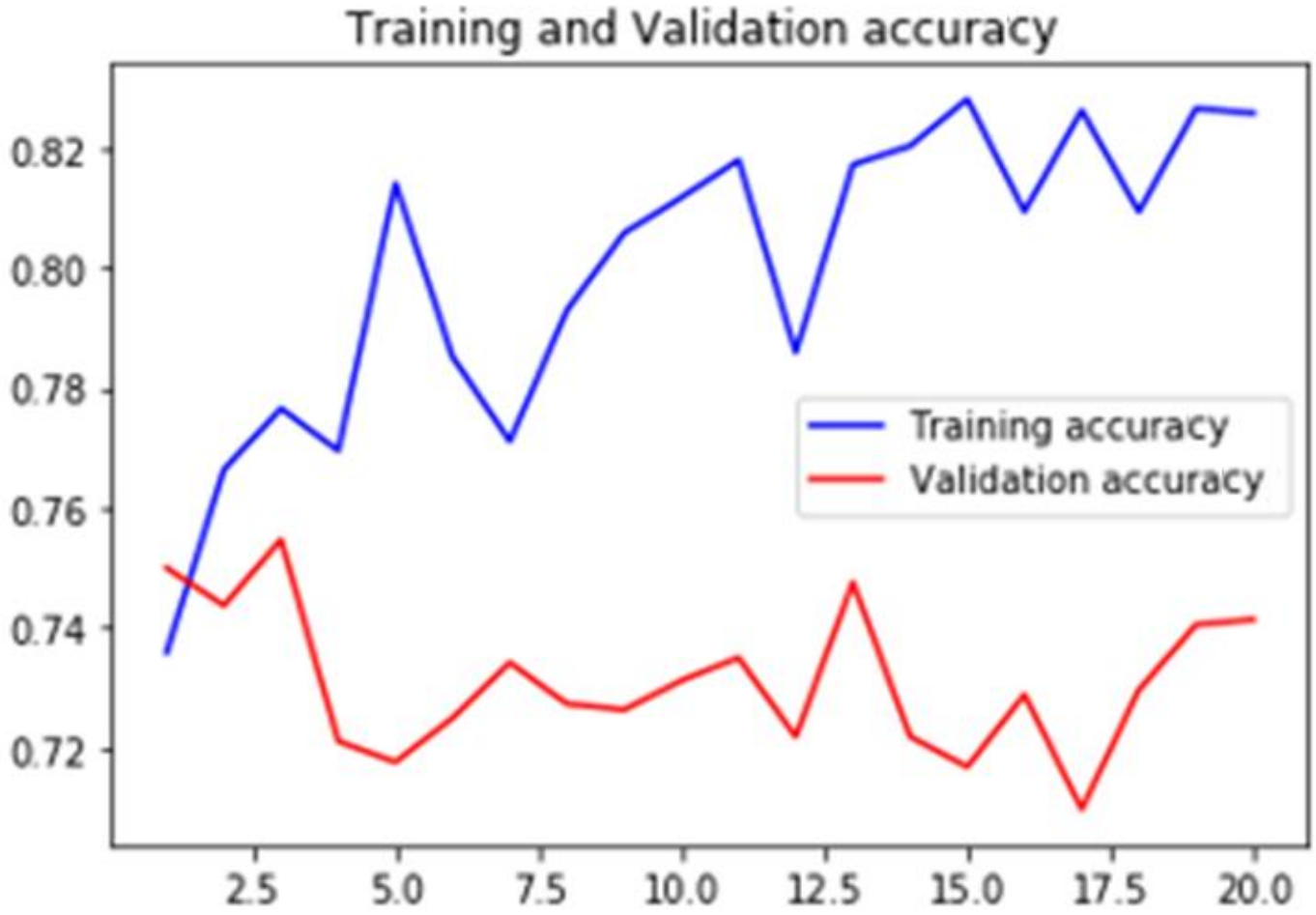

Fig 4.3: Training and validation accuracy for mobilenet

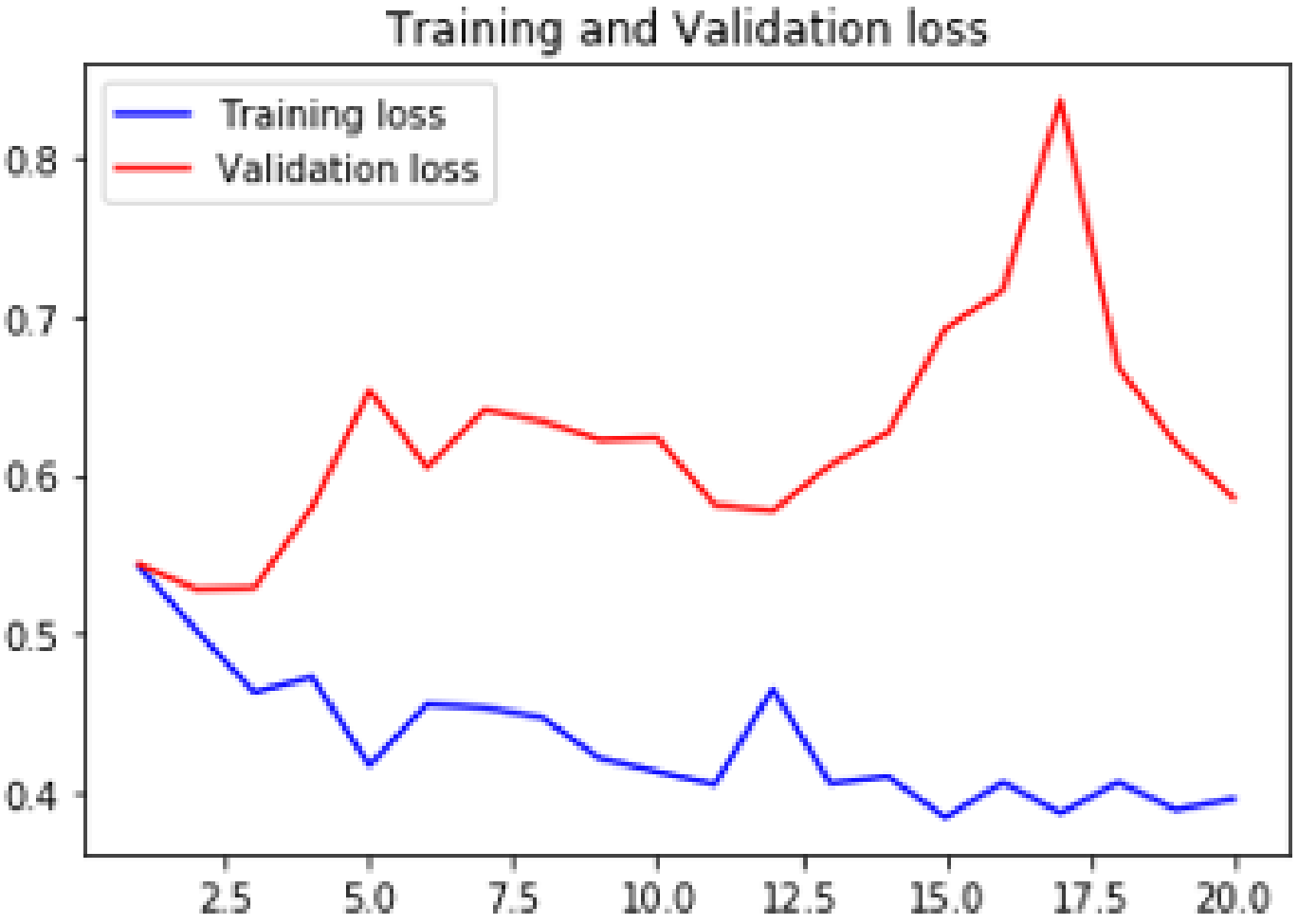

Fig 4.4: Training and validation loss for mobilenet

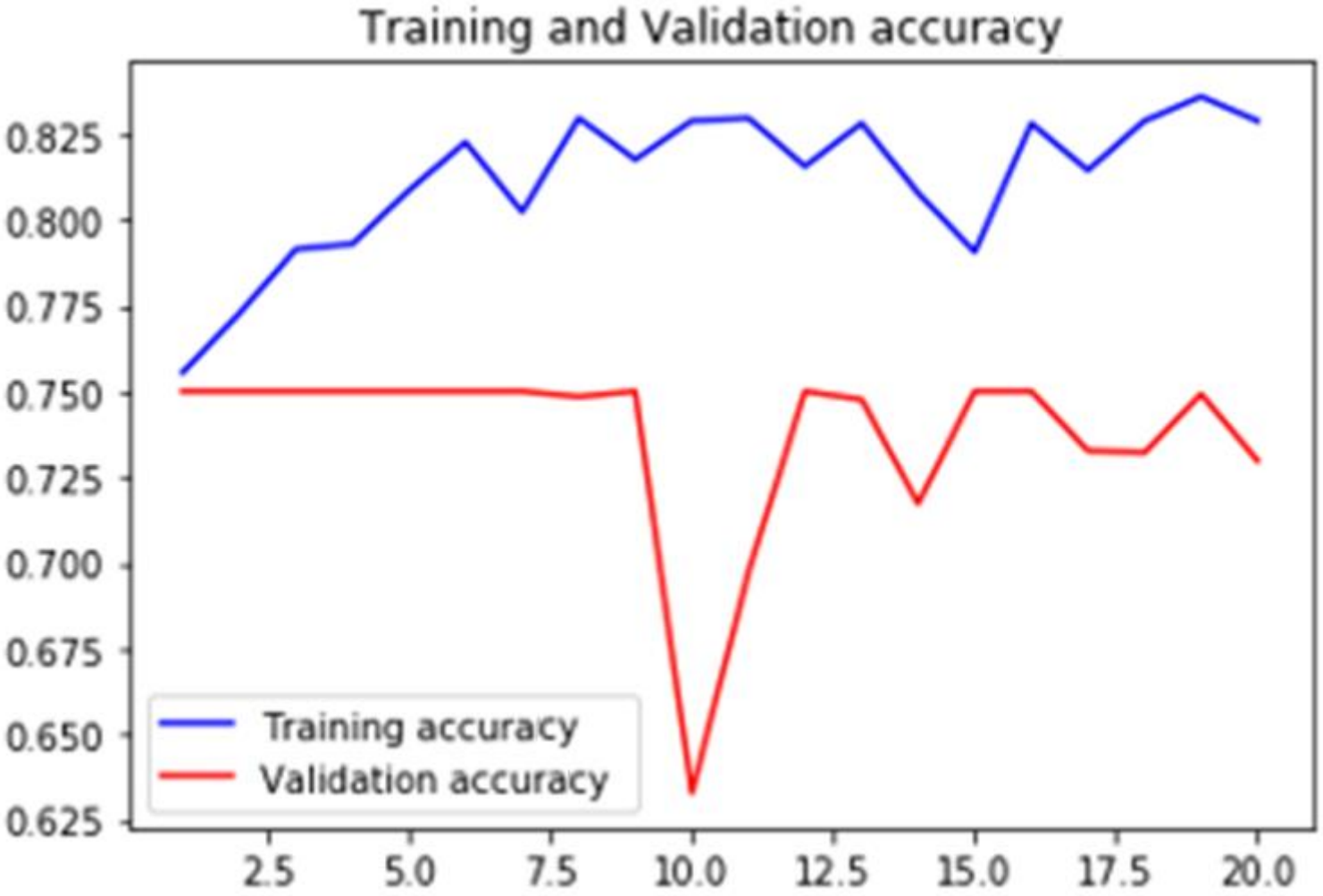

Fig 4.5: Training and validation accuracy for resnet50

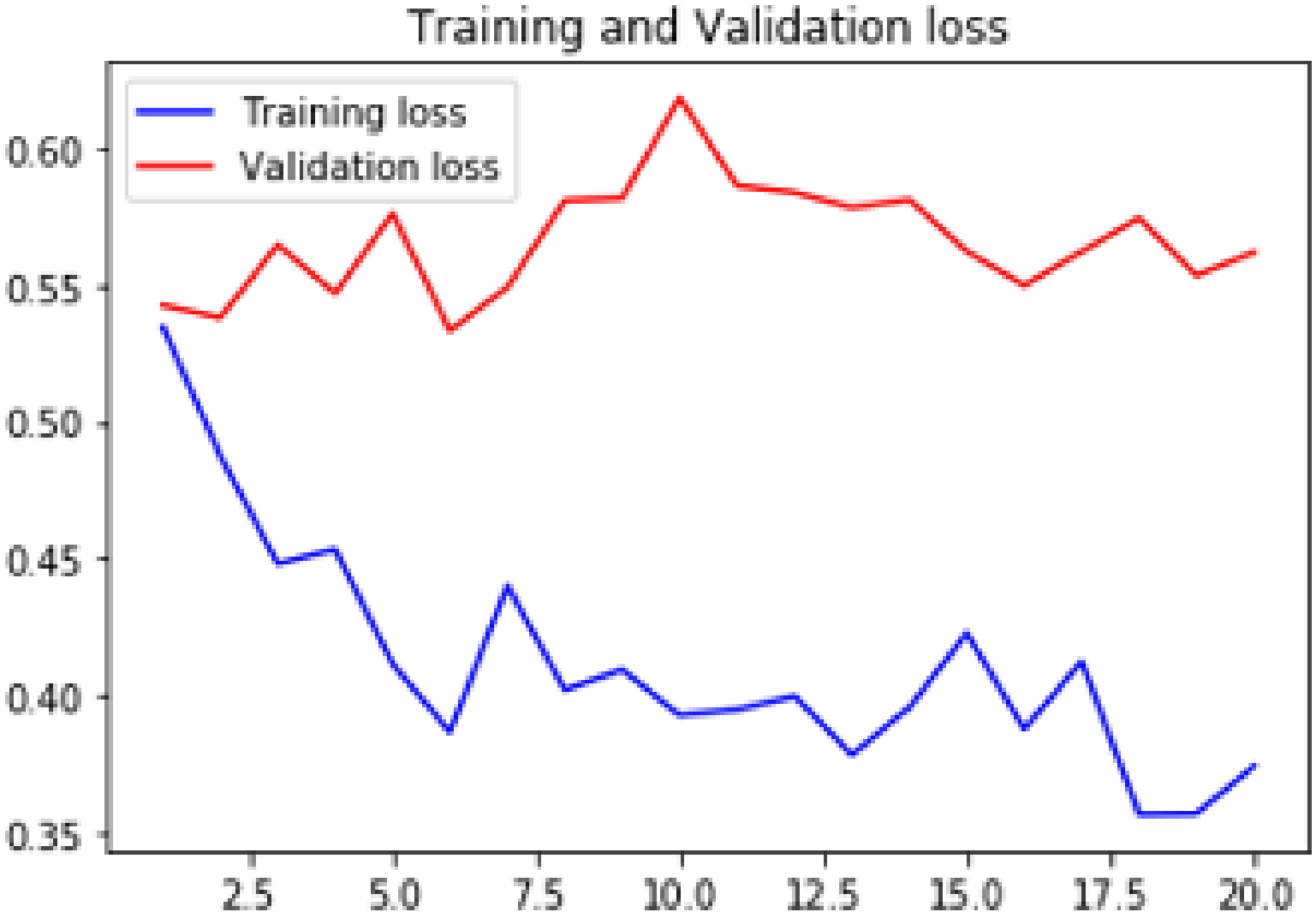

Fig 4.6: Training and validation loss for resnet50

4.2. Evaluation

From the results it is found that when using resnet50 model it gives the best training accuracy (82.27%) along with a minimum training loss of 0.3867. But in terms of validation this model gives 75% accuracy with a maximum of 0.5332 loss. When testing this model gives 73.34% accuracy. This model overfits according to this dataset because it gives better accuracy in training. But when testing this model gives the lower accuracy among these three models.

The mobilenet model gives 76.64% training accuracy, 0.5017 training loss, 74.38% validation accuracy and 0.5277 validation loss. For the evaluation this model gives 74.45% test accuracy. This model captures the data well enough with a very little difference in training and test accuracy.

The proposed model gives 80.11% training accuracy with 0.4321 loss. In terms of validation the accuracy and loss are 77.68% and 0.4790 respectively for this model. This model evaluates with a test accuracy of 77.6812%. This model captures the pattern of data precisely with higher training, validation and test accuracy along with lower validation loss

Therefore, we can conclude that the proposed model is by far best model in terms of training, test accuracy for this dataset.

CHAPTER 5

CONCLUSION

5.1. Conclusion

We have proposed deep CNN based classifier for real time rice disease and pest recognition. We have conducted a comprehensive study on rice disease and pest recognition, incorporating four classes of rice diseases, pests and healthy plant. A dataset of 3355 images were used for training. The knowledge of agriculture in solving rice disease classification problem is used here. Three types of models are implemented here for training. Proposed model is trained by ourselves. Pre-trained mobilenet and resnet50 models are used by transfer learning. From the results section we evaluate that the proposed model is best among these models with an training accuracy of 80.11%. This model will facilitate automated and accurate disease detection using mobile devices.

5.2. Future Work

This research has a future potential of being used as a basis for rice disease detection. Different mobile applications for rice disease and pest detection can be created using this model. Farmers can detect if the plant is infected or not by taking an image only. A number rice can be saved if the disease can be detected timely. If apps can be developed there is a minimum need of agriculture consultants to visit the infected area. This can save a lot more time and save the disease infected plants.

This model can also be used for detection of diseases of other plants. Model need to be trained using the that plants disease image dataset. Finally, model should be evaluated. Although without training model can be used for detection, which may not give a good accuracy.

REFERENCES

1. T. Coelli, S. Rahman, C. Thirtle, Technical, allocative, cost and scale efficiencies in Bangladesh rice cultivation: A non-parametric approach, *Journal of Agricultural Economics* 53 (3): 607–626, 2002.
2. Harvey CA et al., Extreme vulnerability of smallholder farmers to agricultural risks and climate change in madagascar, *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 369(1639), 2014.
3. B. C. Karmokar, M. S. Ullah, M. K. Siddiquee, K. M. R. Alam, *Tea leaf diseases recognition using neural network ensemble, International Journal of Computer Applications*, 114 (17).
4. G. Wang, Y. Sun, J. Wang, Automatic image-based plant disease severity estimation using deep learning, *Computational intelligence and neuroscience*, 2017.
5. A. Fuentes, S. Yoon, S. C. Kim, D. S. Park, A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition, *Sensors* 17 (9):2022, 2017.
6. S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, D. Stefanovic, Deep neural networks-based recognition of plant diseases by leaf image classification, *Computational intelligence and neuroscience*, 2016.
7. Y. Lu, S. Yi, N. Zeng, Y. Liu, Y. Zhang, Identification of rice diseases using deep convolutional neural networks, *Neurocomputing*, 267: 378–384, 2017.
8. R. R. Atole, D. Park, A multiclass deep convolutional neural network classifier for detection of common rice plant anomalies, *International Journal of Advanced Computer Science and Applications*, 9 (1): 67–70, 2018.
9. What is the difference between supervised and unsupervised learning? https://medium.com/@gowthamy/machine-learning-supervised-learningvs-unsupervised-learning-f1658e12a780.
10. G.I. Webb, Lazy Learning. In: Sammut C., Webb G.I. (eds) Encyclopedia of *Machine Learning. Springer*, Boston, MA, 2011.
11. Iris Hendrickx, Antal Van den Bosch, "Hybrid algorithms with Instance-Based Classification", *Machine Learning: ECML, Springer*. pp. 158–169, 2005.
12. Wei Zhang, "Parallel distributed processing model with local space-invariant interconnections and its optical architecture", *Applied Optics*. 29 (32): 4790-7, 1990.
13. K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition”, arXiv:1409.1556, 2014.

14. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. “Going deeper with convolutions”. *In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015.
15. Xavier Glorot, Antoine Bordes and Yoshua Bengio, Deep sparse rectifier neural networks, *14th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2011.

APPENDIX

APPENDIX A

Abbreviations

AI – Artificial Intelligence

ANN – Artificial Neural Network

CNN – Convolutional Neural Network

ML – Machine Learning

Resnet – Residual Network

APPENDIX B

Essential Source Code of This Research

```
from os import listdir

import cv2
import pickle
import numpy as np
import matplotlib.pyplot as plt
from keras import backend as K
from keras.optimizers import Adam
from keras.callbacks import ModelCheckpoint
from keras.models import Sequential, load_model
from sklearn.preprocessing import LabelBinarizer
from sklearn.model_selection import train_test_split
from keras.layers.normalization import BatchNormalization
from keras.layers.convolutional import Conv2D, MaxPooling2D
from keras.layers.core import Activation, Flatten, Dropout, Dense
from keras.preprocessing.image import img_to_array, ImageDataGenerator

# Initializing necessary parameters
EPOCHS = 30
initial_learning_rate = 1e-3
batch_size = 32
default_image_size = tuple((256, 256))
directory_root = '../input/rice-diseases-image-dataset/labelledrice/'
input_width = 256
input_height = 256
input_depth = 3



# Converting images to np arrays
def convert_image_to_array(image_dir):
    try:
        image_from_directory = cv2.imread(image_dir)
        if image_from_directory is not None:
            image_from_directory = cv2.resize(image_from_directory, default_image_size)
            return img_to_array(image_from_directory)
        else:
            return np.array([])
    except Exception as e:
        print(f"Error : {e}")
        return None



# Creating image list and label list
image_list, image_label_list = [], []
try:
    print("[INFO] Loading images ...")
    root_dir = listdir(directory_root)

    for plant_folder in root_dir:
        plant_disease_folder_list = listdir(f"{directory_root}/{plant_folder}")

        for plant_disease_folder in plant_disease_folder_list:
            print(f"[INFO] Processing {plant_disease_folder} ...")
            rice_disease_image_list = listdir(f"{directory_root}/{plant_folder}/{plant_disease_folder}/")
```

```
            for image_from_list in rice_disease_image_list:
                image_directory = f"{directory_root}/{plant_folder}/{plant_disease_folder}/{image_from_list}"
                if image_directory.endswith(".jpg") or image_directory.endswith(".JPG"):
                    image_list.append(convert_image_to_array(image_directory))
                    image_label_list.append(plant_disease_folder)
    print("[INFO] Image loading completed")
except Exception as e:
    print(f"Error : {e}")

# Encoding string labels to integers
label_binarizer_for_image_labels = LabelBinarizer()
binarized_image_labels = label_binarizer_for_image_labels.fit_transform(image_label_list)
pickle.dump(label_binarizer_for_image_labels, open('label_transform.pkl', 'wb'))
n_classes = len(label_binarizer_for_image_labels.classes_)

# Normalizing images
normalized_image_list = np.array(image_list, dtype=np.float16) / 225.0

# Splitting image list in train and test
print("[INFO] Spliting data to train, test")
x_train, x_test, y_train, y_test = train_test_split(normalized_image_list, binarized_image_labels, test_size=0.30,
random_state=42)

# Creating augmentation object
image_augmentation = ImageDataGenerator(
    rotation_range=30, width_shift_range=0.15,
    height_shift_range=0.15, shear_range=0.15,
    zoom_range=0.2, horizontal_flip=True,
    fill_mode="nearest")

# Proposed model building
proposed_model = Sequential()
input_shape = (input_height, input_width, input_depth)
channel_dimension = -1
if K.image_data_format() == "channels_first":
    inputShape = (input_depth, input_height, input_width)
    channel_dimension = 1
proposed_model.add(Conv2D(32, (3, 3), padding="same", input_shape=input_shape))
proposed_model.add(Activation("relu"))
proposed_model.add(BatchNormalization(axis=channel_dimension))
proposed_model.add(MaxPooling2D(pool_size=(3, 3)))
proposed_model.add(Dropout(0.25))
proposed_model.add(Conv2D(64, (3, 3), padding="same"))
proposed_model.add(Activation("relu"))
proposed_model.add(BatchNormalization(axis=channel_dimension))
proposed_model.add(Conv2D(64, (3, 3), padding="same"))
proposed_model.add(Activation("relu"))
proposed_model.add(BatchNormalization(axis=channel_dimension))
proposed_model.add(MaxPooling2D(pool_size=(2, 2)))
proposed_model.add(Dropout(0.25))
proposed_model.add(Conv2D(128, (3, 3), padding="same"))
proposed_model.add(Activation("relu"))
proposed_model.add(BatchNormalization(axis=channel_dimension))
proposed_model.add(Conv2D(128, (3, 3), padding="same"))
proposed_model.add(Activation("relu"))
proposed_model.add(BatchNormalization(axis=channel_dimension))
proposed_model.add(MaxPooling2D(pool_size=(2, 2)))
proposed_model.add(Dropout(0.25))
proposed_model.add(Flatten())
```

```
proposed_model.add(Dense(1024))
proposed_model.add(Activation("relu"))
proposed_model.add(BatchNormalization())
proposed_model.add(Dropout(0.5))
proposed_model.add(Dense(n_classes))
proposed_model.add(Activation("softmax"))

# Compiling model
optimizer = Adam(lr=initial_learning_rate, decay=initial_learning_rate / EPOCHS)
proposed_model.compile(loss="binary_crossentropy", optimizer=optimizer, metrics=["accuracy"])
print("[INFO] training network...")

# Training model
checkpoint_for_best_model = ModelCheckpoint('best_model.h5',
                                verbose=1, monitor='acc',
                                save_best_only=True, mode='auto')

model_history = proposed_model.fit_generator(
    image_augmentation.flow(x_train, y_train, batch_size=batch_size),
    validation_data=(x_test, y_test),
    steps_per_epoch=len(x_train) // batch_size,
    epochs=EPOCHS, verbose=1,
    callbacks=[checkpoint_for_best_model])

# Getting accuracy and loss list
train_acc = model_history.history['acc']
val_acc = model_history.history['val_acc']
train_loss = model_history.history['loss']
val_loss = model_history.history['val_loss']
epochs = range(1, len(train_acc) + 1)

# Plotting train and validation accuracy
plt.plot(epochs, train_acc, 'b', label='Training accuracy')
plt.plot(epochs, val_acc, 'r', label='Validation accuracy')
plt.title('Training and Validation accuracy')
plt.legend()
plt.show()

# Plotting train and validation loss
plt.plot(epochs, train_loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and Validation loss')
plt.legend()
plt.show()

# Loading saved model and evaluating
best_trained_model = load_model('best_model.h5')
print("[INFO] Calculating model accuracy")
scores = best_trained_model.evaluate(x_test, y_test)
print(f"Test Accuracy: {scores[1] * 100}")
```