Fungal Image Analysis and Classification: Using Deep Learning

A project report submitted in six months training program on

Artificial Intelligence -Data Scientist

by

Rahul Kumar Vishwakarma

(01492)

Organized by

Department of Computer Science

Institute of Science

Banaras Hindu University,

Varanasi – 221005

Supported by

Ministry of Electronics and Information Technology (MeitY), Government of India

Feb - Aug 2023

CANDIDATE'S DECLARATION

I **Rahul Kumar Vishwakarma** hereby certify that the work, which is being presented in the report, entitled **Fungal Image Analysis Classification: Using Deep Learning**, in partial fulfillment for completion of the training program on **Artificial Intelligence – Data Scientist** and is an authentic record of my own work carried out during the period *May-2023* to *July-2023* with guidance from the program team. I also cited the reference about the data, text(s) /figure(s) /table(s) /equation(s) from where they have been taken.

Date: **Signature of the Candidate**

ABSTRACT

Fungal infection also known as mycoses are caused by different fungi found in environment. They can infect different body parts such as feet, nose etcetera. During COVID-19 pandemic, which largely affected lungs and the persons whose lungs got severely affected by the virus and kept on oxygen supply, developed Mucor mycosis infections. This report analyses different fungal images and predict which type of fungi is present. There are various methods applied here to analyze and predict. I used deep learning for the classification problem which includes VGG16 and ResNet. The model with the best accuracy levels is adopted.

Keywords: CNN, VGG16, ResNet

TABLE OF CONTENTS

Title		Page No.
ABSTRACT		i
CHAPTER 1	**INTRODUCTION**	
1.1	General Introduction	1
1.2	Objectives	2
CHAPTER 2	**METHODOLOGY**	
2.1	Proposed Approach	3
2.2	Description of method/model used	4
2.3	Operating Conditions	11
2.4	Experiment Procedure	12
CHAPTER 3	**DATA DESCRIPTION**	
3.1	Data Source	13
3.2	Data Description	13
CHAPTER 4	**RESULTS AND DISCUSSION**	
4.1	Observations	15
4.2	Model Evaluation	16
4.3	Implementation Details	18
CHAPTER 5	**CONCLUSION AND FUTURE WORK**	
5.1	Conclusions	19
5.2	Scope for Future Work	20
REFERENCES		21
APPENDIX		
A.1	Appendix 1: Python Code for VGG16	22
A.2	Appendix 2: Python Code for ResNet50	25

LIST OF ABBREVIATIONS

Abbreviation	Description
ACM	Association for Computing Machinery
ADES	Advance Data Encryption Standard
CBC	Cipher Block Chaining
CFB	Cipher Feedback Mode
DDES	Double Data Encryption Standard
DES	Data Encryption Standard
ECD	Electronic Codebook Mode
IEEE	Institute of Electrical and Electronics Engineers
NIST	National Institute of Standards and Technology
OFB	Output Feedback Mode
OTP	One Time Pad
TDES	Triple Data Encryption Standard

LIST OF NOTATIONS

Notation	Description
$A < B$	A is less than B
$A > B$	A is greater than B
$A \leq B$	A is less than or equal to B
$A \geq B$	B is greater than or equal to B
E_K	Encryption key
D_K	Decryption key
F_n	*n*th Fibonacci number
I_n	the identity matrix of order n
N	Set of positive Integers
W	Set of whole numbers
Z	Set of integers
Z^+	Set of positive integers
Z_n	Set of non-negative integers less than n
μ	Static friction coefficient

CHAPTER 1

INTRODUCTION

Mucor mycosis infection is a disease caused by fungi. After the second wave of COVID-19, it came into notice. The most of the cases were reported in India where the infection rate was five times the world. Mucor mycosis is diagnosed by looking at a tissue sample in the lab. Your doctor may collect a sample of phlegm or nasal discharge if you have a suspected sinus infection. In the case of a skin infection, your doctor may also clean the wounded area in question.[2]

This type of diagnosis is very error prone and negative reporting may be done. So, to reduce the pressure and supporting the professionals involved in diagnosis.

1.1 BACKGROUND

Since we are living in the age of technological advancements, introduction in new technologies have impacted our lives not only providing leisure but also with negative social factors. One side where technology provided millions of students' quality education at very low cost on the other side it also created deep fake technologies which heavily impacted live of millions.

But it is no topic of debate, neither I am going to discuss it here. I am here to discuss how technology changed medical domain altogether. Medical domain is not an exception benefitting from technological advancement. The introduction of tech has already empowered our doctors as well as the diagnosis systems. The introduction of various technologies like X-Ray, CT-Scans, MRIs etc. have been boon to the society and people's health.

Now, with the introduction of Industry-4.0 everything is changing to computers, Internet of Things, Artificial Intelligence and Machine Learning. Recent advancements in deep learning and other sophisticated tools of artificial intelligence not made lives of people easier but diagnosis cost affordable.

In my research problem, fungi classification by using deep learning model is one of them. I used various deep learning models like VGG16, ResNet and CNN to predict class of fungi images.

1.2 OBJECTIVE

In Deep Learning models any accuracy more than 60% is considered very good and ready to deploy but there are so many other best possible methods are also possible. In this problem I used three algorithms VGG16, ResNet and CNN.

VGG16 and ResNet is an application of transfer learning while CNN stands for convolutional neural network.

By two algorithms accuracy level are achieved up to deployment level while in case of CNN the model started overfitting.

CHAPTER 2

METHODOLOGY

2.1 PROPOSED APPROACH

First of all, problem is defined for the research then dataset is obtained from the website [1]. Once the dataset is obtained the google colab environment is set up. The whole process can be picturized by the following flow chart.

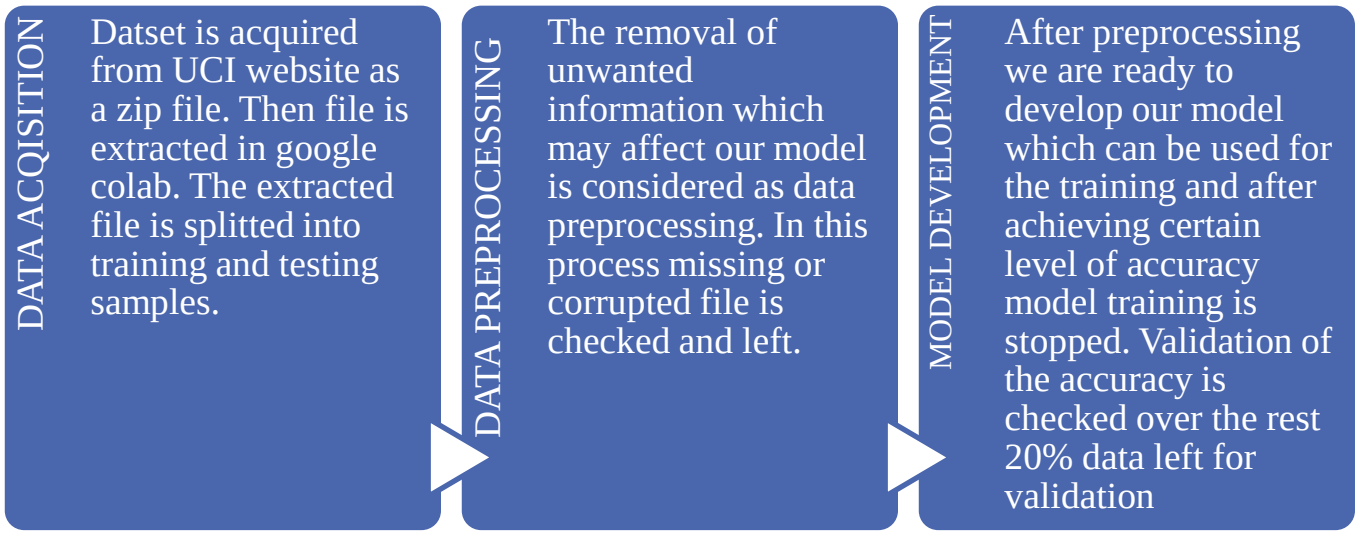


Fig-(1): Flow Diagram of the project

Once the data is loaded into google colab environment, and checked for corrupt files, data is normalized using standard scaler method. Also, it is stored in different folders and computer does not understand different folder so it the labels of the images are encoded as the class labels once class labels are finished.

The training and validation data is generated, model is created. Once the model is created it is set up for training process up to three epochs. Since the configuration I am using in my personal computer does not provide me heavy GPU support so I had to stop it in three epochs. The model performance on training data and test data is plotted against each other and checked for overfitting condition. When it is satisfied model is not overfitting. Prediction is done by the model and at last model is saved.

2.2 DESCRIPTION OF METHODS/MODELS

2.2.1 DATA ACQUISTION

Data acquisition is defined as the process by which data is obtained from the source for the research purposes.

In my research I obtained data from DeFungi, UCI Machine Learning Repository. Which is free for the acquisition.

2.2.2 DATA PREPROCESSING

The process of removal of redundant features or noises from a dataset is termed as data preprocessing. There are different data preprocessing techniques available for the datasets.

Some of the preprocessing techniques may be defined as follows

1.) Noise Reduction: - Often we see images blurred, captured while in motion and so on. These features which are not adding value to the dataset is called noise. The process of removal of these noises are called noise reduction. Some of the noise reduction methods are Gaussian smoothing, median filtering, and wavelet denoising.[3]

2.) Contrast Enhancement: - Contrast enhancement is a process of enhancement of contrast of images so that some features of images may be clearly visible. The most widely used application of this preprocessing technique is in medical imaging. Some standard contrast enhancement techniques include histogram equalization, adaptive histogram equalization, and contrast stretching. [3]

3.) Image Resizing: - Images in captured with different dimensions and sizes. But in Deep Learning standard size image is used for the training processes and sometimes the image size led to excessive amount of consumption of memory which makes hard to

train a model. Some typical image resizing techniques include nearest neighbor interpolation, bilinear interpolation, and bicubic interpolation.[3]

4.) Color Correction: - Color correction is a process of balancing color of an image which may be critical for the accuracy of the model. The most used case of this application is in photography where color of different object matters very much. Some common color correction techniques include gray world assumption, white balance, and color transfer.[3]

5.) Segmentation: - Segmentation is a process in which an image is divided into several segments so that special attention may be given to the different segments. The applications of this techniques are mostly in the medical imagery where different parts of organs are picturized and analyzed differently. Some standard segmentation techniques include thresholding, edge detection, and region growing.[3]

6.) Feature Extraction: - It is a technique where different features are identified and analyzed some of them may be corners of the object, different objects itself. Object recognition. Some standard feature extraction techniques include edge detection, corner detection, and texture analysis.[3]

2.2.3 SPLITTING DATASET

Before start developing model, we must split our dataset into training and validation datasets. The terms training and testing datasets are defined as follows: -

1.) Training Dataset: - The part of dataset which is used for the model training is termed as training dataset. It mostly consists 60%-80% of the actual dataset. The split is done not in a sequence rather it is done randomly.

2.) Validation Dataset: - There are various methods for the validation of a model. Some of them are K-Fold Cross Validation, Cross Validation etc. This is a key parameter to check whether model is overfitting or underfitting.

2.2.4 MODEL DEVELOPMENT

The process of selecting a model on the basis of its architecture and use cases are called as the model development. In this experiment I used three Deep Learning models to train. Before diving into the model let me discuss a little bit about the Deep Learning.

Deep Learning: - With the introduction of high compute power. This type of learning came into picture. Neuron shown in **Fig-(2)** has different components and function. Similarly, Artificial Neural Network (ANN) is inspired by the functioning of our brain. Simple perceptron model may be picturized in **Fig-(2).**

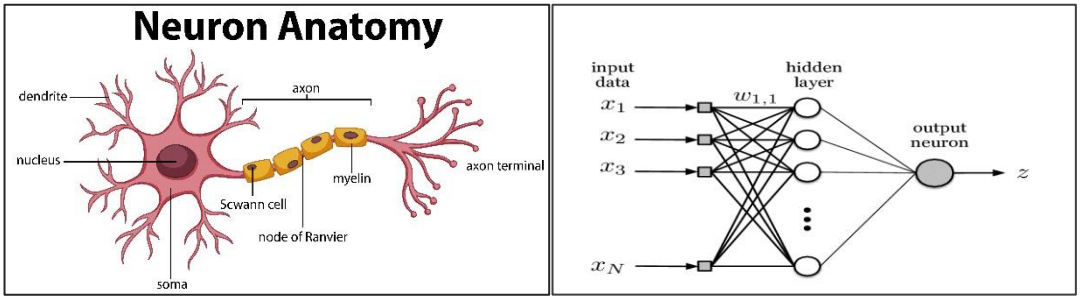


Fig-(2): Biological Neuron and simple perceptron

In **Fig-(2),** both neurons and replica of neuron are depicted. In the replica when the number of hidden layers the layer except input layer and output neuron are increased such that it becomes two or more than two then it the model structure is considered as a Deep Neural Network.

Picture of a deep neural network is given by the figure **Fig-(3),**

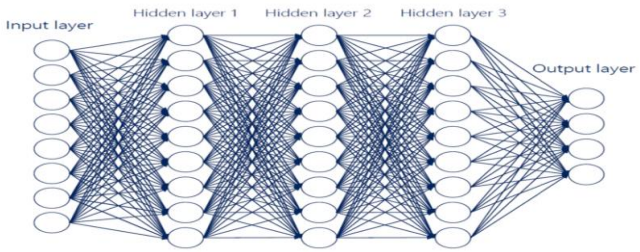


Fig-(3): Deep Neural Network

Input Layer: - Input is given by the user for the prediction. The inputs are always given either in the form of vector, matrix or tensor. In other words, tensor (rank-0,1,2). In my research problem I given the input as a 2D matrix with 3 channels. In matrix channels stands for the colors RGB. The values in matrix are in the range (0,255). Which is normalized by dividing each pixel by 255 to obtain a value between 0-1.

This input layers also assigns weight to the different nodes randomly, the randomness while assigning the weights are very important otherwise all the probabilities may be zero.

Hidden Layers: - After weighted input are calculated with some added bias it is sent to the next layer called as hidden layer. The process continues until output layers is not reached. The whole process of reaching up to the output layer is called a forward pass.

Output Layers: - After hidden layers final predictions are obtained by this layer either in the form of probabilities or a vector.

Activation Function: - To jump from one layer to another layer, after the calculation of weighted sum a function is applied for the output, which acts as an input for the next layers some of the well-known examples of activation functions are as follows: -

1.) Sigmoid- It is also called S-shape function.

$$f(x) = \frac{1}{1 + e^{x}}$$

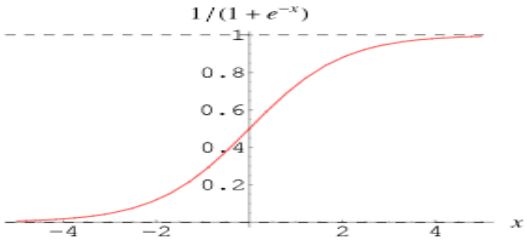


Fig-(4): Sigmoid

2.) RELU: - Rectified Linear Unit

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

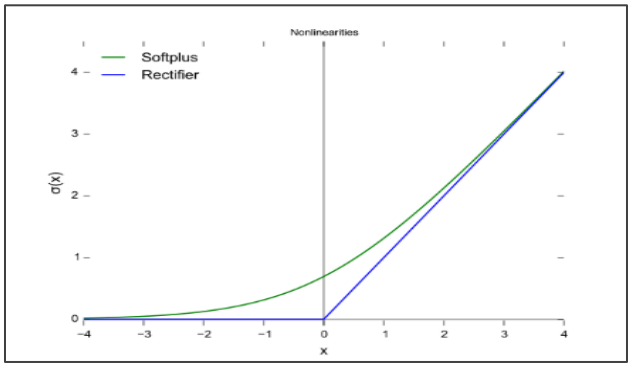


Fig-(5): Relu

3.) SOFTMAX: -

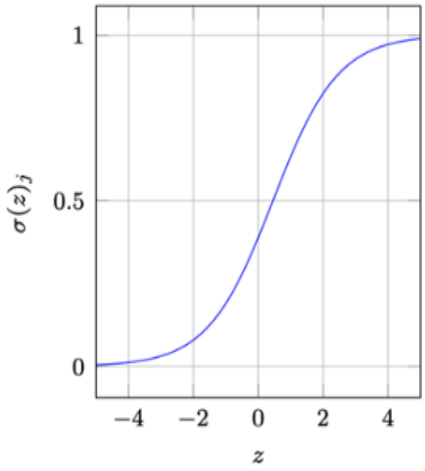


Fig-(6): Softmax function

ResNet50: -

Architecture: -

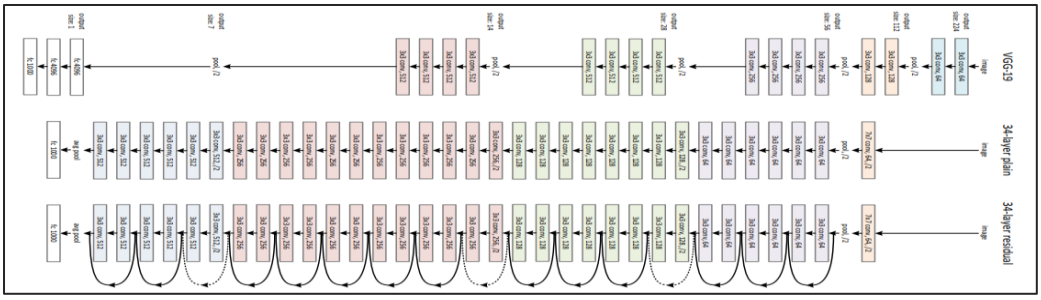


Fig-(7): ResNet50 architecture

ResNet-50 is a 50-layer convolutional neural network (48 convolutional layers, one MaxPool layer, and one average pool layer). Residual neural networks are a type of artificial neural network (ANN) that forms networks by stacking residual blocks.[4]

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2				
		$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Table-(1): ResNet information [5]

VGG16: -

ImageNet Large Scale Visual Recognition Challenge (ILSVRC) is an annual computer vision competition. Each year, teams compete on two tasks. The first is to detect objects within an image coming from *200* classes, which is called object localization. The second is to classify images, each labeled with one of *1000* categories, which is called image classification. VGG 16 was proposed by Karen Simonyan and Andrew Zisserman of the Visual Geometry Group Lab of Oxford University in 2014 in the paper "VERY DEEP CONVOLUTIONAL NETWORKS FOR LARGE-SCALE IMAGE RECOGNITION". This model won 1st and 2nd place in the above categories in the 2014 ILSVRC challenge. [6]

Configuration: The table below listed different VGG architectures. We can see that there are 2 versions of VGG-16 (C and D). There is not much difference between them except for one that except for some convolution layers, *(3, 3)* filter size convolution is used instead of *(1, 1)*. These two contain *134* million and *138* million parameters respectively.[6]

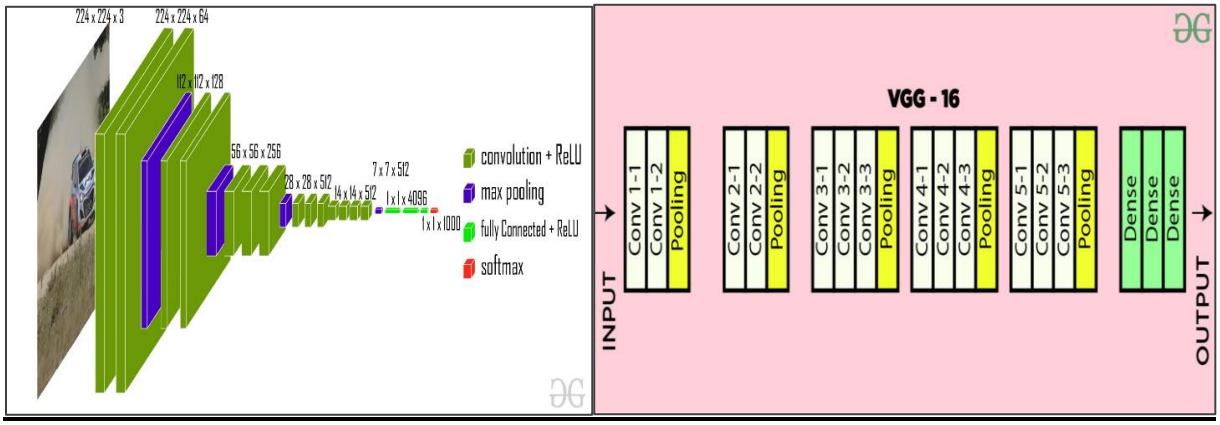


Fig-(9): VGG16 model architecture [6]

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

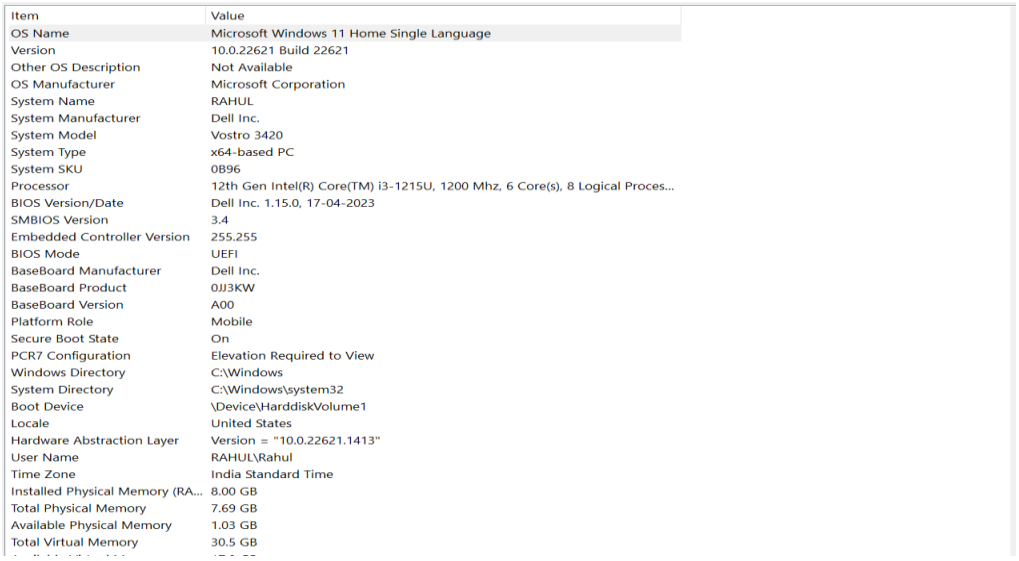
Table-(2): All VGG

2.3 OPERATING CONDITIONS

2.3.1 SYSTEM CONFIGURATION

OS	Windows 11
Environment	Google Colab
Python Version	Python 3.10.6

Table-(3): System Configuration



Item	Value
OS Name	Microsoft Windows 11 Home Single Language
Version	10.0.22621 Build 22621
Other OS Description	Not Available
OS Manufacturer	Microsoft Corporation
System Name	RAHUL
System Manufacturer	Dell Inc.
System Model	Vostro 3420
System Type	x64-based PC
System SKU	0B96
Processor	12th Gen Intel(R) Core(TM) i3-1215U, 1200 Mhz, 6 Core(s), 8 Logical Proces...
BIOS Version/Date	Dell Inc. 1.15.0, 17-04-2023
SMBIOS Version	3.4
Embedded Controller Version	255.255
BIOS Mode	UEFI
BaseBoard Manufacturer	Dell Inc.
BaseBoard Product	0JJ3KW
BaseBoard Version	A00
Platform Role	Mobile
Secure Boot State	On
PCR7 Configuration	Elevation Required to View
Windows Directory	C:\Windows
System Directory	C:\Windows\system32
Boot Device	\Device\HarddiskVolume1
Locale	United States
Hardware Abstraction Layer	Version = "10.0.22621.1413"
User Name	RAHUL\Rahul
Time Zone	India Standard Time
Installed Physical Memory (RA...	8.00 GB
Total Physical Memory	7.69 GB
Available Physical Memory	1.03 GB
Total Virtual Memory	30.5 GB

Fig-(10): System Information

2.3.2 TOOLS SETUP

TOOLS	VERSION
Matplotlib	3.7.1
Tensorflow	2.12.0
Split-folders	0.5.1

Table-(4): Tools information

2.4 EXPERIMENTAL PROCEDURE

STEP 1: Loading Dataset from google drive.

STEP 2: Splitting Dataset using Split-folder.

STEP 3: Data Preprocessing using Tensorflow

STEP 4: Model Creation (Using Tensorflow API for Deep Learning Model)

STEP 5: Starting Model Training Up to 3 epochs. (Since GPU in Google colab not Available for this PC.)

STEP 6: Plot of model accuracy, validation accuracy, loss and validation loss for Different model is plotted using Matplotlib.

STEP 7: Model Evaluation and Prediction.

CHAPTER 3

DATA DESCRIPTION

3.1 DATA SOURCE

DeFungi dataset is obtained from UCI machine learning repository. UCI Machine Learning Repository is a popular online collection of datasets for machine learning research hosted by the University of California, Irvine (UCI), and it provides a wide variety of datasets that researchers and practitioners can use to experiment with machine learning algorithms.

3.2 DATA DESCRIPTION

DeFungi is a dataset for direct mycological examination of microscopic fungi images. The images are from superficial fungal infections caused by yeasts, moulds, or dermatophyte fungi. The images have been manually labelled into five classes and curated with a subject matter expert assistance. The images have been cropped with automated algorithms to produce the final dataset.[7]

3.3.1 DATASET CHARACTERISTICS

Dataset containing only Images as datafiles.

3.3.2 SUBJECT AREA

Computer Science

3.3.3 ASSOCIATED TASK

Classification

3.3.4 ATTRIBUTE TYPE

Real

3.3.5 INSTANCES

9114

3.3.6 MISSING VALUES

There are no missing values in the dataset.

3.3.7 NUMBER OF LABELS

The data belongs to 5 classes.

SAMPLES OF SOME IMAGES:

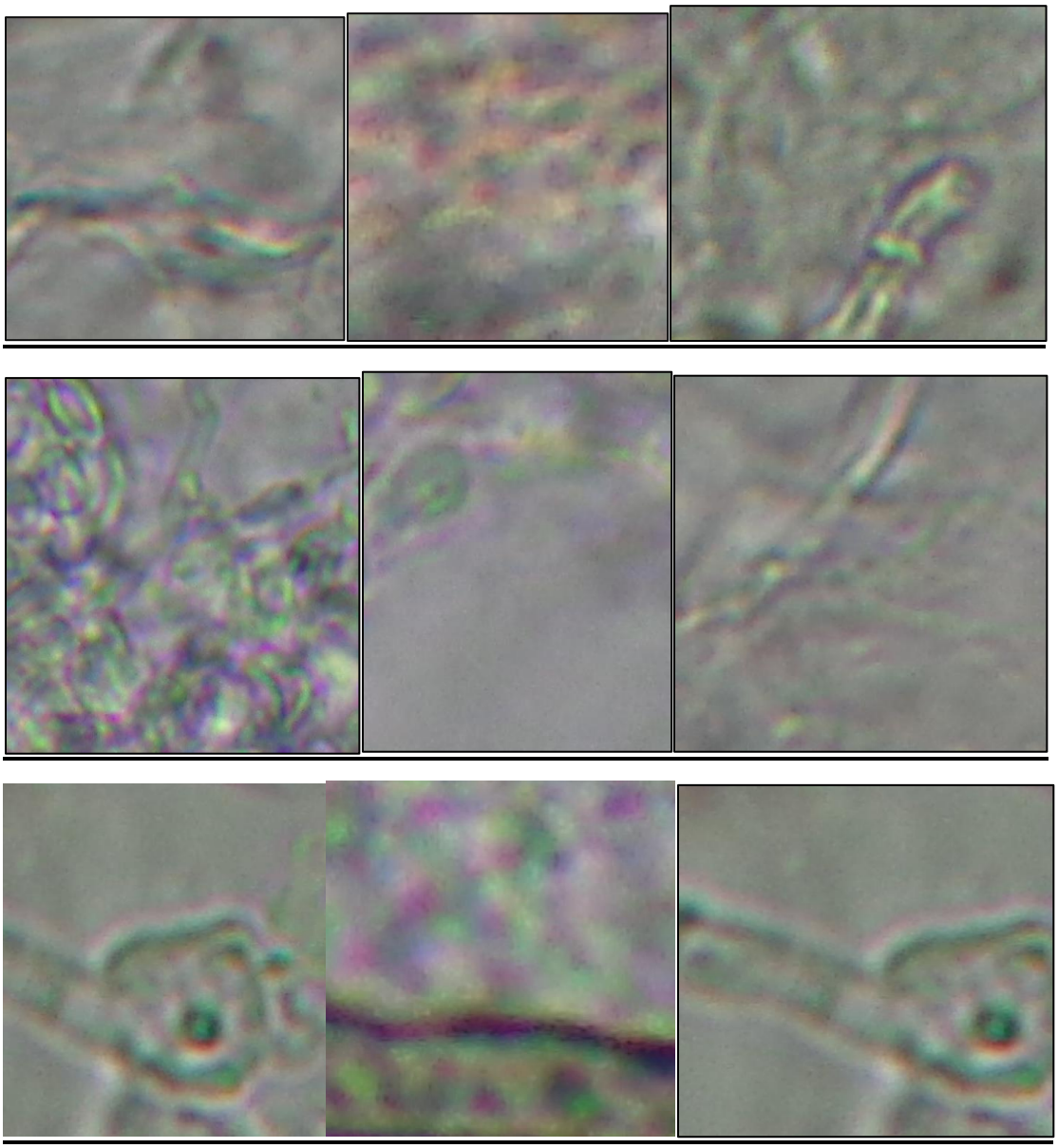


Fig-(11): Sample of some images

CHAPTER 4

RESULTS AND DISCUSSION

4.1 OBSERVATIONS

4.4.1 RESNET

Layers	50
Optimizer	Adam
Image Size	256*256
Batch Normalization	Yes
Batch Size	32
Activation Functions	ReLU, Softmax
Loss Measure	CATEGORICAL CROSSENTROPY
Learning Rate	0.001
Number of Epochs	3

Table-(5): Configuration ResNet50

Epoch 1:

There are 228 batches for the model training. Each step took $10s$ to train. And the total time taken by the model to be trained is $2358s$ with a loss of 0.7376 and accuracy 0.6996. The total validation loss is remained at 0.6690 and validation accuracy stood at 0.7368.

Epoch 2:

There are again 228 batches for the model training. Each step took $10s$ to train. The total loss now decreased to 0.6728 and accuracy increased to 0.7292. Total time taken is $2337s$. Validation loss is decreased to 0.6019 and validation accuracy stood at 0.7527.

Epoch 3:

There are again 228 batches for the model training. Each step took $10s$ to train. The total time taken by the model to be trained stood at $2339s$. Loss decreased to 0.6431 accuracy improved to 0.7413. The validation loss is slightly increased to 0.6044. But the validation accuracy is increased to 0.7675.

4.4.2 VGG16

Layers	16
Optimizer	Adam
Image Size	224*224
Batch Normalization	Yes
Batch Size	32
Activation Functions	ReLU, Softmax
Loss Measure	CATEGORICAL CROSSENTROPY
Learning Rate	0.1
Number of Epochs	15

Table-(6): Configuration of VGG16

After 15 epochs the model accuracy on training stood at 0.8347 while loss stood at 0.4182. Validation loss stood at 0.8996. While validation accuracy stood at 0.6793.
If we take average of all training models, we will find accuracy stood at 0.7943.
The average loss of the model stood at 0.47092.

In validation accuracy the model performed slightly different as a case of overfitting since model is giving us accuracy of more than 60%. In validation set also so we are accepting the model despite little overfitting.

4.2 MODEL EVALUATION

The performance of the model at different stages are given by the plots of the model performance which can be found below: -

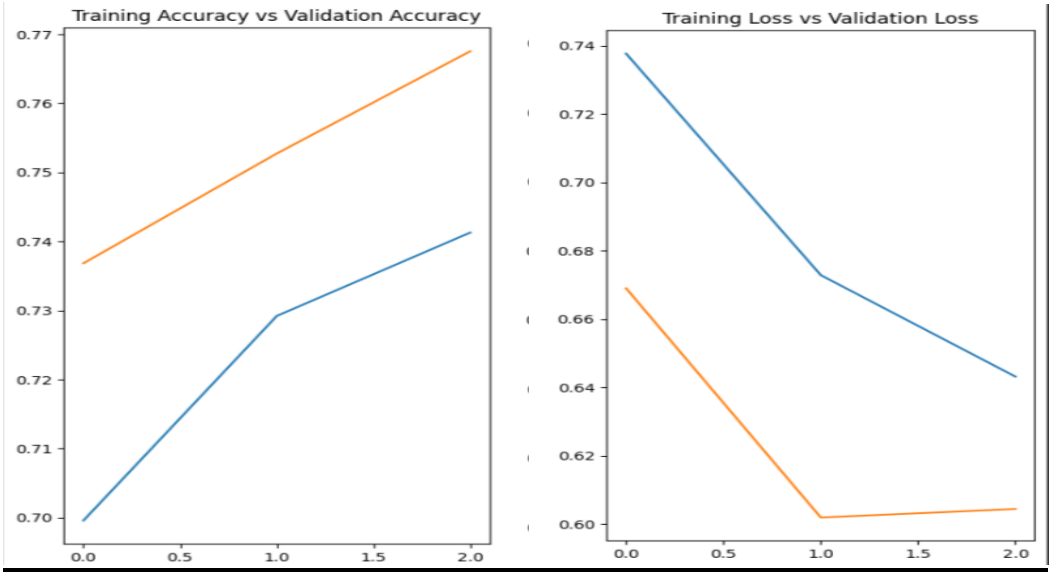


Fig-(12): Loss and Accuracy for ResNet Model

Where:

Orange color: Validation Accuracy and Loss

Blue : Model Accuracy

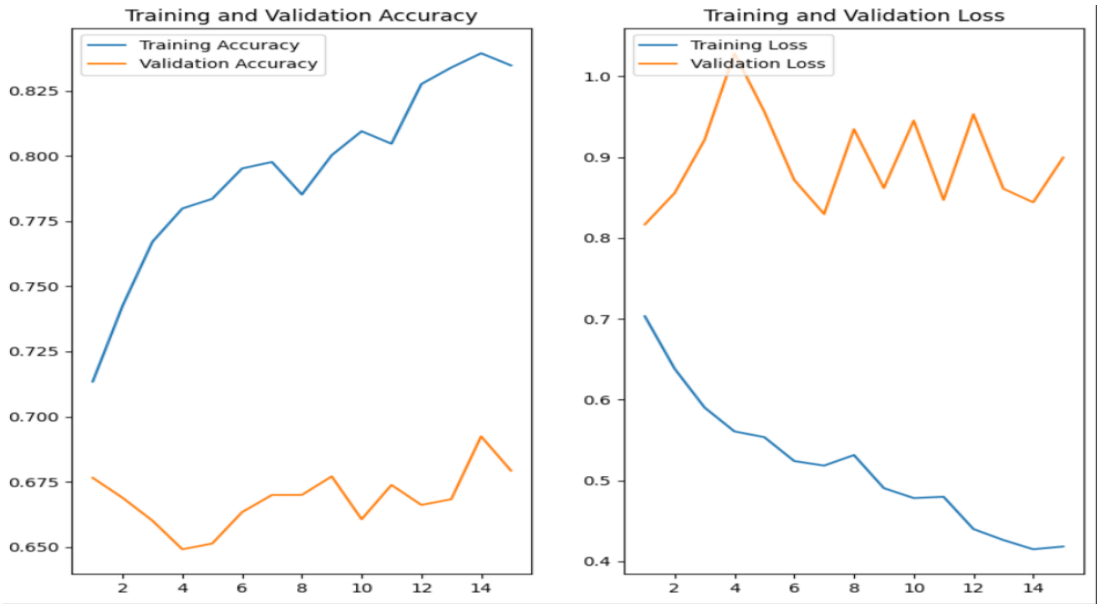


Fig-(13): Accuracy and Loss for VGG16 Model

4.3 IMPLEMENTATION DETAILS

Basic google colab is used for implementing different models. Python was the preferred language with APIs Tensorflow, Keras which constituted the architecture of the VGG16 and ResNet50.

Since the dataset was not split into training and validation set so it was split using another module split-folders.

The model performance is continuously observed so that overfitting may be avoided. Although dropout method is also implemented to prevent overfitting.

ResNet50 was taking time so model is trained only up to 3 epochs and VGG16 was trained up to 15 epochs predictions are shown in the form of numpy array which is a 2-Dimensional vector.

CHAPTER 5

CONCLUSION AND FUTURE SCOPE

From above experiments it is very clear, both the models VGG16 and ResNet50 are state of the art models which perform well in multiclass classification. Here too we can see the model with usual configuration gave us satisfactory result for classification. The summary of the both model's performance can be summarized in the following table:

Model Name	**VGG16**	**ResNet50**
Layers	16	50
Optimizer	Adam	Adam
Image Size	224*224	256*256
Batch Normalization	Yes	No
Batch Size	32	32
Activation Functions	ReLU, Softmax	ReLU, Softmax
Loss Measure	CATEGORICAL CROSSENTROPY	Categorical Crossentropy
Learning Rate	0.1	0.001
Number of Epochs	15	3
Accuracy (Validation)	67.93%	76.65%

Table (7): Final conclusion

Both the models are giving us accuracy more than 60%. So, any of the model can be used for the predictions.

Since these may be considered satisfactory but not the best, we can do there are new world out there, there are also some sophisticated models of VGG and ResNet are available which can be further used.

By the use of convolutional neural network, we can accomplish this task also. We may use other aspects deep learning in preprocessing images. Experiment could be further for the different

optimizer functions as well as activation functions also. Since the field is still developing so there are a lot of possibilities available to us.

In final words, deep learning is very vast in itself. So, we may use different models may be train them to our own purpose may be tune it as per the task requirement.

In coming days, the field is going to develop more so whatever result obtained today, may not be hold true tomorrow.

I would like to borrow a quote from the unknown “Change is the only constant.”. So adapt change.

REFERENCES

[1] Hajati,Farshid. (2023). DeFungi. UCI Machine Learning Repository. https://doi.org/10.48550/arXiv.2109.07322.

[2] https://www.healthline.com/health/mucormycosis#diagnosis

[3] https://www.analyticsvidhya.com/blog/2023/03/getting-started-with-image-processing-using-opencv/#:~:text=Image%20preprocessing%20is%20quite%20useful%20to%20improve%20the,image%20resizing%2C%20color%20correction%2C%20segmentation%2C%20feature%20extraction%2C%20etc.

[4] https://datagen.tech/guides/computer-vision/resnet-50/#

[5] https://doi.org/10.48550/arXiv.1512.03385F.

[6] https://www.geeksforgeeks.org/vgg-16-cnn-model/

[7] https://archive.ics.uci.edu/dataset/773/defungi

APPENDIX

A.1 Appendix 1: Python Code for VGG16

```
from google.colab import drive
drive.mount('/content/drive')
!pip install tensorflow
!pip install split-folders
from keras.layers import Input, Lambda, Dense, Flatten
from keras.models import Model
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.models import Sequential
import numpy as np
from glob import glob
import matplotlib.pyplot as plt
import splitfolders
input_folder='/content/drive/MyDrive/defungi'
output='output'
splitfolders.ratio(input_folder,output=output,seed=42,ratio=(0.8,0.2))
IMAGE_SIZE=[224,224]

train_path='/content/output/train'
valid_path='/content/output/test'
vgg=VGG16(input_shape=IMAGE_SIZE+[3],weights='imagenet',include_top=False)
for layer in vgg.layers:
  layer.trainable=False
folders=glob('/content/output/train/*')
from keras.layers import Dropout
x=Dropout(.2)
x=Flatten()(vgg.output)
prediction=Dense(len(folders),activation='softmax')(x)

model=Model(inputs=vgg.input,outputs=prediction)
model.summary()
model.compile(
    loss='categorical_crossentropy',
    optimizer='adam',
    metrics=['accuracy']
)
from keras.preprocessing.image import ImageDataGenerator
train_datagen=ImageDataGenerator(rescale=1./255)

test_datagen=ImageDataGenerator(rescale=1./255)
train_set=train_datagen.flow_from_directory(
```

```
    train_path,
    target_size=(224,224),
    batch_size=32,
    class_mode='categorical'
)

test_set=test_datagen.flow_from_directory('/content/output/val',
                                          target_size=(224,224),
                                          batch_size=32,
hist=model.fit(train_set,
          validation_data=test_set,
          epochs=15,
          verbose=1)
acc=hist.history['accuracy']
val_acc=hist.history['val_accuracy']

loss=hist.history['loss']
val_loss=hist.history['val_loss']

plt.figure(figsize=(10,8))
plt.subplot(1,2,1)
plt.plot(range(1,16),acc,label='Training Accuracy')
plt.plot(range(1,16),val_acc,label='Validation Accuracy')
plt.legend(loc='upper left')
plt.title('Training and Validation Accuracy')


plt.subplot(1,2,2)
plt.plot(range(1,16),loss,label='Training Loss')
plt.plot(range(1,16),val_loss,label='Validation Loss')
plt.legend(loc='upper left')
plt.title('Training and Validation Loss')
plt.show()

model.save("defungi1.hdf5")
array=model.predict(test_set)
array=(array== array.max(axis=1)[:,None]).astype(int)
tuplle=array.shape
import splitfolders
input_folder='/content/drive/MyDrive/defungi'
output='output1'
splitfolders.ratio(input_folder,output=output,seed=42,ratio=(0.7,0.2,0.1))
test_set_2=test_datagen.flow_from_directory('/content/output1/test',
                                          target_size=(224,224),
                                          batch_size=32,
```

```
                                            class_mode='categorical')
array=model.predict(test_set_2)
array=(array==array.max(axis=1)[:,None]).astype(int)
print(array)
```

A.2 Appendix 2: Python Code for ResNet50 model

```
!pip install --root-user-action=ignore split-folders
import warnings
warnings.filterwarnings("ignore")
import pandas as pd
import os
import tensorflow as tf
import numpy as np
from tensorflow.keras.applications.resnet50 import ResNet50,
preprocess_input
from tensorflow.keras.models import Model
from keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.layers import GlobalAveragePooling2D
import matplotlib.pyplot as plt
import splitfolders
input_folder = "/content/drive/MyDrive/Colab Notebooks/defungi"
output = "output"

splitfolders.ratio(input_folder, output=output, seed=42, ratio=(0.80,
0.20))
dataset=
tf.keras.preprocessing.image_dataset_from_directory('/content/output/train
')
class_names = dataset.class_names
class_names
image_size = 256

data_generator_with_aug =
ImageDataGenerator(preprocessing_function=preprocess_input,
                                    horizontal_flip=True,
                                    vertical_flip=True)

data_generator_no_aug =
ImageDataGenerator(preprocessing_function=preprocess_input)

train_generator = data_generator_with_aug.flow_from_directory(
        '/content/output/train',
        target_size=(image_size, image_size),
        batch_size=32,
        class_mode='categorical',
        shuffle=True)

validation_generator = data_generator_no_aug.flow_from_directory(
        '/content/output/val',
        target_size=(image_size, image_size),
```

```
        batch_size=32,
        class_mode='categorical',
        shuffle=True)
resnet_model = ResNet50(include_top=False,weights="imagenet")
x = resnet_model.output
x = GlobalAveragePooling2D()(x)
x = tf.keras.layers.Flatten()(x)
x = tf.keras.layers.Dense(1024,activation='relu')(x)
x = tf.keras.layers.Dropout(0.6)(x)
x = tf.keras.layers.Dense(512,activation='relu')(x)
x = tf.keras.layers.Dropout(0.5)(x)
res = tf.keras.layers.Dense(6, activation="softmax")(x)

model = Model(inputs=resnet_model.input, outputs=res)

# Freezing the pre-trained layers so that we do not have to update weights
ourselves
for layer in resnet_model.layers:
    layer.trainable = False

model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),
    loss='categorical_crossentropy',
    metrics=['accuracy'])
hist = model.fit(
    train_generator,
    validation_data=validation_generator,
    epochs=3,
    verbose=1
).history

model.summary()
acc=hist['accuracy']
val_acc=hist['val_accuracy']

loss=hist['loss']
val_loss=hist['val_loss']

plt.figure(figsize=(10,8))

plt.subplot(1,2,1)
plt.plot(range(3),acc,label='Training Accuracy')
plt.plot(range(3),val_acc,label='Validation Accuracy')
plt.title("Training Accuracy vs Validation Accuracy")
```

```
plt.subplot(1,2,2)
plt.plot(range(3),loss,label='Training Loss')
plt.plot(range(3),val_loss,label='Validation Loss')
plt.title("Training Loss vs Validation Loss")
plt.show()
model.save("defungi.h5")
array=model.predict(validation_generator)
array=(array == array.max(axis=1)[:,None]).astype(int)
array
```