International Journal on Engineering Artificial Intelligence Management, Decision Support, and Policies
Vol. 01, Iss. 01, S. No. 001, pp. 41-60, July 2024
ISSN(E): 3048-8788

Classification and Validation of Tomato Leaf Disease Using Deep Learning Techniques

Aakash[1], S. Amutha[1]*, D. Nandhini[2,3], Ansh[1]

1School of Computer Science and Engineering, Vellore Institute of Technology, Chennai-600127, India amutha.s@vit.ac.in, aakash.kumar2021a@vitstudent.ac.in , ansh.saraiya2021@vitstudent.ac.in.
2Department of Ocean Engineering, Indian Institute of Technology Madras, Chennai-600036, India, nandhiniduraimurugan.iitm@gmail.com.
3Institute of Textile Technology, RWTH Aachen University, 52074 Aachen, Germany,
*Corresponding author: amuthabenziker@gmail.com

How to cite this paper: Aakash, S. Amutha*, D. Nandhini, Ansh "Classification and Validation of Tomato Leaf Disease Using Deep Learning Techniques," *International Journal on Engineering Artificial Intelligence Management, Decision Support, and Policies* , Vol. no. 1, Iss. No 1, S No. 001, pp.41-60, July 2024.

Received: 10/07/2024
Revised: 20/07/2024
Accepted: 25/07/2024
Published: 31/07/2024

Abstract

Tomatoes are regarded as fruits since they fit the botanical definition of a fruit because they are the fleshy parts of a plant that enclose its seeds. There are approximately 10 different kinds of diseases for a tomato plant, which is huge in number and can create huge losses for the farmers. This paper focuses on the classification of tomato plant leaf diseases using Convolution Neural Network (CNN) a deep learning technique that is especially employed for image recognition and pixel data processing activities. CNN has been used to identify whether the given photo of the plant is of a healthy or unhealthy part. The secondary dataset that was taken was trained using two algorithms Inception-v3 and VGG19. VGG19 is a 19 layered algorithm, comprising 16 convolutional layers and 3 fully linked layers. Inception-v3 is the algorithm is a CNN-based algorithm that has 48 layers in it. The pre-trained networks can categorize photos into several different item categories, including several animals, types, diseases and much more. The dataset that is being used to train and validate the model is secondary data taken from Kaggle. The models were trained with 1900 images of healthy tomato plants, and 1800 images of unhealthy tomato plants. The model was also validated with 500 images of healthy and 1080 images of unhealthy tomato plants. The Inception V3 model was able to achieve an accuracy of 98% and a validation accuracy of 89% and VGG19 achieved an accuracy of 94%. Inception-v3 was the chosen model for the paper.

Keywords

Tomato leaf disease, VGG19, InceptionV3, Colorectal, lycopene

1. Introduction

With an annual production of over 18,399,000 tons, India ranks second among tomato producers. In the year 2020, tomatoes generated over 232 billion Indian rupees for the Indian economy [2]. This crop is always under threat from various diseases that may reduce its yield and quality and thus reduce its economic value. The range of diseases in tomatoes that can lead to heavy losses in quantity and quality are caused by bacteria, fungi, viruses, or pests. For instance, some of these devastating diseases include early blight, late blight, and bacterial spot, which may result in severe defoliation, fruit rot, and loss of plants if not well managed. This has not only affected the farmers' income but also threatened food security in regions that are heavily reliant on tomatoes either as staple or economic crops. After sowing the tomato seeds, it typically takes 65 to 70 days for the tomato to fully develop into a tomato. Due of their nutritional value and culinary use, tomatoes are considered vegetables. Tomatoes are low in calories and also include potassium and vitamin C, two important nutrients. A powerful antioxidant known as lycopene, which gives tomatoes their distinctive color, it also provides with 7% to 10%, of the Recommended Dietary Allowance (RDA) for iron for women. Lycopene has been associated with numerous health benefits, among them a rapider decreased risk for coronary heart disease, a variety of cancers such as lung, prostate, stomach, cervical, breast, oral, colorectal, esophageal, pancreatic, among several other types [1]. Having these many benefits of tomatoes, Production is adversely affected by diseased tomato plant [25]. Therefore, this paper aims to identify the picture of a leaf as a healthy or an unhealthy tomato plant using the CNN-based algorithm the Inception V3 from the input and will alert the user.

Figure 1. Healthy leaves of a tomato plant [6]

The table1 given below gives comprehensive information about nine illnesses that typically harm tomato's growth. Bacterial spot, early blight and leaf curl virus are commonly observed diseases in the leaves of tomato. On the other hand, late blight, leaf mold, Septoria leaf spot, Spider mites, target spot and Tomato mosaic are diseases found in the leaf tomato. This research focuses on mainly illness identified on the leaves of the tomato plant.

Table 1. Commonly affected Tomato diseases

Name of the disease	Image of disease	Symptoms
Bacterial_spot		• Small, dark, water-soaked circular patches with a yellowish halo can be seen on infected leaves. • Older leaves are primarily affected by the leaflet infection which can result in significant defoliation. • Small, wet patches first emerge, then they rise and grow until they are between an eighth and a fourth of an inch in diameter. [3]
Early_blight		• The skin around the patches may become yellow. Much of the foliage is destroyed if excessive temperatures and humidity are experienced at this time. • This typical tomato disease can affect the leaves at any stage of growth. [5]

Late_blight		• When planted in the field, transplants that have been infected by the late blight fungus frequently perish. • Concentric rings are also visible on the fruit, and lesions can reach considerable proportions, engulfing practically the whole fruit. [5]
Leaf_mold		• In general, lower leaves are attacked first. • On the top of the leaf, yellow spots appear. • On the matching bottom surface, there is a faint, greyish-brown mould development. [5]
Septo-ria_leaf_spot		• On leaves, there are tiny, rounded to irregular dots with a black border and a grey centre. • Spots usually start on lower leaves and gradually travel up the leaves. • Spots congregate, and the leaves become wilted[5]
Spider_mites		• Severely impacted stems may die, and occasionally the entire plant perishes. • Infected tuber plants have severely stunted growth that takes the appearance of a rosette and dark green leaves [4].
Target_Spot		• Small circular to oval, dark brown to black dots on leaves are the first symptoms of the illness. • They take on a leathery look. [5]
Tomato_mo-saic_virus		• Affected leaves often have smaller-than-normal, twisted, and puckered leaflets. • The leaflets can occasionally become recessed, exhibiting "fern leaf" characteristics. [5]
Leaf_Curl_Virus		• Older leaves become brittle and leathery. • The internodes and nodes have much smaller sizes.[5]

2. Related Work

This section provides a summary of past literature which were used as a guideline for creating the model. The urgent requirement in agriculture is to detect and prevent tomato leaf diseases (TLD). The paper [7] proposes an image segmentation method called Cross-layer Attention Fusion Mechanism combined with a Multi-scale Convolution Module (MC-UNet). It highlights the feature information on the edge of the leaves. It deploys 3 convolution kernels of various sizes. To retain the valid information from tomato leaves a SoftPool was used. This MC-UNet gives an accuracy of 91.32%. A deep learning (DL) framework [8] was implemented for the set of tomato leaves to extract the features in hierarchical order. The neural network (NN) is applied to classify the leaves into healthy, septoria leaf spots and bacterial spots.

The research [12] proposes a CNN model for classifying all nine types of TLD. Four modules are implemented to extract the features accurately. The performance of this model is measured as 99.9% for accuracy and 99.64% for validation accuracy. The value of the recall measure is 0.99 during the classification of TLD. For classifying the TLD, the paper [14] uses Deep CNN as well as transfer learning. The backbone of CNN comprises AlexNet, ResNet,

VGG-16, and DenseNet. techniques like Adam and RmsProp. Here, the accuracy is 99.9% thanks to the RmsProp optimization on the DenseNet model.

A precise image-based TLD detection approach PLPNet [19] was used. Initially, disease defining characteristics are extracted. Then at the neck of the network, a location reinforcement attention mechanism was used to prevent extraneous information from accessing the network's feature fusion phase. The network provides a solution to the interclass similarity of disease issue. The experimental findings further demonstrate that PLPNet obtained 54.4% average recall (AR) and 94.5% mean average precision (mAP50) with 50% thresholds. Also DenseNet-121 architecture [20] is useful to identify the TLD. This method achieves the evaluation metrics such as accuracy, precision, recall, and F1 Score as 98.9%, 0.98, 0.99, and 0.99 respectively.

The EfficientNetB5 model was implemented [21] for the TLD dataset. This model does not use any segmentation concept. This model achieved an average training accuracy, average validation accuracy, and average test accuracy of 99.84% ± 0.10%, 98.28% ± 0.20%, and of 99.07% ± 0.38% over 10 cross folds respectively. It is suggested that the gradient-weighted class activation mapping (GradCAM) and local interpretable model-agnostic explanations be used to provide the model's interpretability, which is necessary for performance prediction and beneficial for fostering confidence.

A three-compact CNN was used in [22]. This algorithm provides transfer learning to retrieve features from the fully connected layer of CNN. Once the features were extracted, the merging of features was carried out from each CNN. At last, a hybrid feature selection method was used to generate a comprehensive feature set with lower dimensions. This model gives a good accuracy of 99.92% for K-nearest neighbor classifiers when compared with SVM. In [24], the Gray Level Co-Occurrence Matrix (GLCM) algorithm was used to detect the TLD. This algorithm identifies and calculates 13 various features from the input data set. Now these derived features are used to classify the different types of diseases using a Support Vector Machine (SVM). The accuracy measured for the GLCM method is 100% for healthy leaves, 95% for early blight, 90% for septoria leaf spot and 85% for late blight. Table 2 gives the details of various recent algorithms used for identifying TLD in recent years.

Table 2. Recent Research on TLD

Author	Algorithm	No of Images	Output Parameters	Key findings
Nagamani et al. [13]	Fuzzy-SVM, CNN, RCNN	735	Accuracy	Compared many classifiers and found out the best classifier for effectively classifying the TLD with an accuracy of 96.735%
Mohamed Bouni et al.[14]	AlexNet, ResNet, VGG-16, DenseNet	7301	Accuracy, Precision, Recall, F1	Compared the pretrained models VGG-16, DenseNet, ResNet, and AlexNet with adam and RMSprop as optimizers, and identified RMSprop proved to have an edge over Adam
Sunil et al. [15]	KNN, CNN	600	Accuracy, Precision, Recall, F1	The proposed method in the paper used computer vision techniques and GLCM are used to extract the important features from the leaves
Prabhjot Kau [16]	RCNN	1610	Accuracy, F1 Score	The paper has proposed with mask R-CNN

				method for diagnosing tomato leaf disease
Thanjai Vadivel et al. [17]	GLCM+ VGG 16	10000	Accuracy	A CNN based model was proposed in the paper with 16 different kind of tomato diseases
Seongho Jeong at el. [18]	DNN with ResNet, DNN + Mask RCNN	6456	precision, recall, and F1-score	The paper proposed two DNN models, which will identify if the leaf is healthy or infected by tomato leaf miner
Xibei Huang at el. [23]	FC-SNDPN	82161	Accuracy	FCN was used for foreground segmentation and SNDPN for identification and 97.59% was the accuracy obtained
Zhou et al. [26]	Deep Convolutional Neural Network (DCNN)	54000	Accuracy	Developed a deep CNN model for the identification and classification of leaf diseases of a rice crop, achieving more than 98% accuracy—showing capability in the concerned domain of deep learning models.
Kamilaris et al. [27]	Transfer Learning with VGG16, ResNet50	5000	Accuracy, Precision, Recall, F1	Applied transfer learning to a variety of plant diseases, achieving high accuracy with ResNet50. This study emphasized the efficiency of transfer learning in agricultural applications where labeled data is scarce.
Ferentinos et al. [28]	CNNs (VGG, AlexNet, GoogleNet)	87158	Accuracy	In the research, it was proposed that a CNN model could be used for the identification of 25 different diseases on a variety of crops with an accuracy of 99.53%, thereby showing the ability of the CNN model for such parameters in agricultural use.
Ramcharan et al. [29]	MobileNet, InceptionV3	4000	Accuracy	Embedded deep learning models on mobile devices for real-time cassava disease detection. This study

				showed that deep learning models can be trans-located into the field for real-time disease diagnosis.
Brahimi et al. [30]	VGG16, InceptionV3	50000	Accuracy, Precision, Recall	The technique was applied in the present research to detect diseases of olive trees with an accuracy of more than 98%. It further showed the role of deep learning in detecting diseases with respect to different contexts in agriculture.
Brahimi et al. [31]	DenseNet, SqueezeNet	5000	Accuracy, F1 Score	In this paper, a lightweight deep learning model is proposed for the detection of apple leaf diseases. Since the model is optimized to work on edge devices, its deployment in real-world applications within the agricultural domain becomes quite promising.

In the literature review, it becomes evident that the early diagnosis of plant diseases is of paramount importance in the plant production process. To assess the health of a provided leaf image, this particular project utilizes CNN and deep learning concepts, which have proven to be valuable in the field of agriculture. In this research, the Inception-v3 algorithm was employed, achieving an impressive 98% accuracy with a dataset of slightly over 3500 images. It is apparent that increasing the number of images or expanding the dataset size will lead to improved accuracy and validation accuracy. The presence of 48 layers in Inception V3 plays a crucial role in accurately identifying unhealthy leaves when processing uploaded images.

3.Methodology

3.1. Dataset

The model is based on deep learning algorithms, and all the deep learning algorithms depends on the image data set that is fed into the model. The data set that was used to train the model had a total of 3326 pictures out of which 1563 pictures belong to unhealthy leaves and 1926 pictures belong to the healthy leaves. The model was also validated using 1561 images out of which 1080 images were of unhealthy and 481 images were of healthy tomato leaves in the data set. The model was tested using 100 images of unhealthy and 70 image of unhealthy plant. The summary of the total number of images used in each set is listed in the Table 3. The dataset used for training, testing and validation dataset from the below given website:

i.) https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf

Table 3. Number of images taken for training, validating and testing the data

S. No.	Data	No. of Healthy Images	No. of Unhealthy Images
1.	Training data	1563	1926
2.	Validation data	481	1080
3.	Testing data	100	70

3.2. VGG - 19

The University of Oxford's Visual Geometry Group (VGG) created the deep convolutional neural network architecture known as VGG19. Its 19 layers, comprising 16 convolutional layers and 3 fully linked layers. In this section, the model was trained with the dataset using VGG-19. Fig 2 illustrates the VGG19 network architecture and Fig 3 illustrates the flow chart of the model.

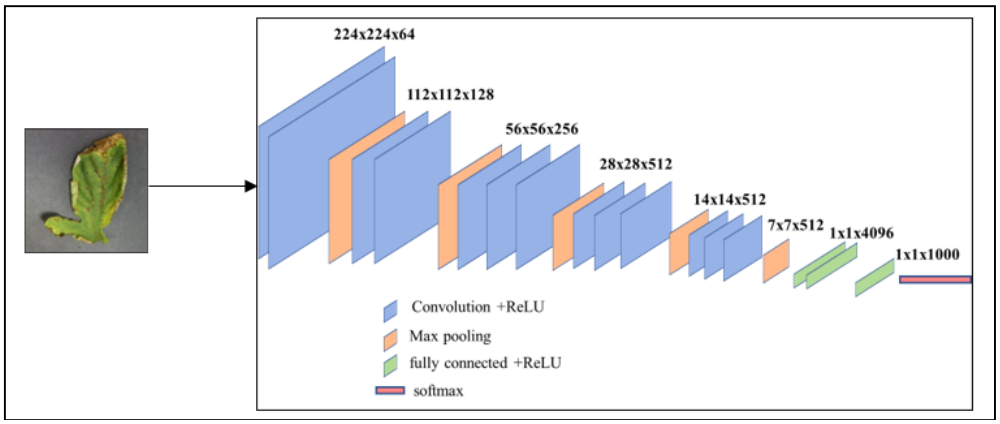


Figure 2. VGG 19 network Architecture

The model while training takes in the image of the tomato leaf in the 224x224x64. The model reduces the pixel size to 1x1x1000 going through various convolution and pooling layers. The data for training the model was gathered from Kaggle, and the photographs were sorted and pre-processed. The model was then chosen, and the data was divided into three categories: training, testing, and validation. The model was trained and validated with the training and validation data respectively. The confusion matrix is then used to evaluate the model, and the accuracy, precision, recall value, and F1 score are calculated with the values obtained from the confusion matrix. The model is then used to accept a photo as input and decide whether or not the uploaded image is healthy.

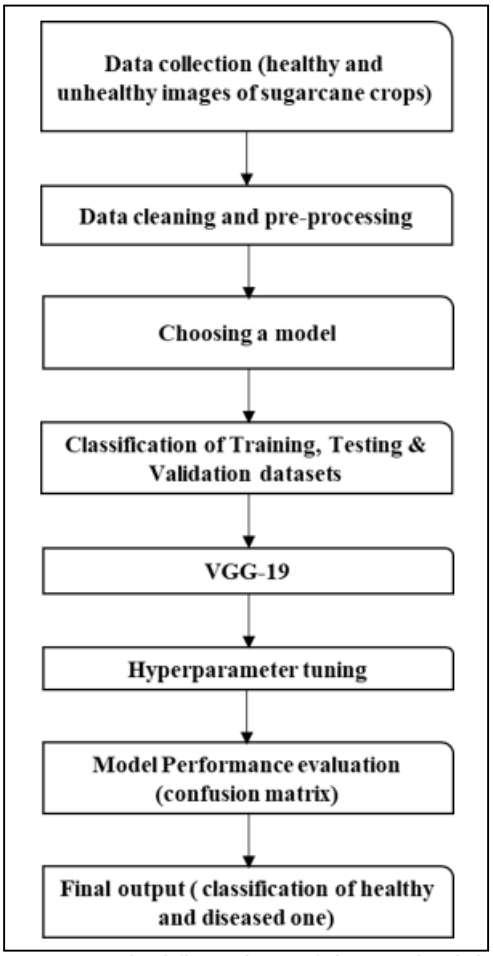


Figure 3. Typical flow chart of the methodology

3.3. Algorithm for VGG19

The process followed to simulate the VGG19 algorithm is as follows.

i. Import the dependencies and keras applications

ii. Import VGG19, preprocess input, and decode_predictions

iii. Set up the drive and the program's connection

iv. Create ImageDataGenerator with all the zoom, shear, rescale, and horizontal characteristics

v. Create the input shape of the picture using VGG19

vi. Flatten and dense the model and compile the model with the Adam optimizer

vii. Create fitgenerator with the required epochs values, verbose, validation steps.

viii. Visualize the results using graphs and print the accuracy level

ix. Create the training and testing generator with required parameters

x. Print the true positive, false positive, false negative, false positive values and the precision, recall values, F1score of the model.

The following calculations (Eqs. (1) - (6)) are done in different layers while simulating the VGG19 algorithm.

1. Convolution Layer (with ReLU activation)

The output feature map size of the layer is given by

$$H' = \frac{H - F + 2P}{S} + 1 \quad (1)$$

$$W' = \frac{W - F + 2P}{S} + 1 \quad (2)$$

2. Max Pooling Layer
   The output size of this layer is given by

$$H' = \frac{H - F}{S} + 1 \quad (3)$$

$$W' = \frac{W - F}{S} + 1 \quad (4)$$

3. Fully Connected Layer (Dense Layer)
   The output calculation function without activation is given by

$$Output = Activation(Input * Weights + Bias) \quad (5)$$

4. SoftMax Function (for multi-class classification)

$$P(y = i|x) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_i}} \quad (6)$$

The parameters used in the equation represents H' and W' are the output feature map height and weight respectively, H and W are the input feature map height and width, F is the filter size, P is the amount of zero padding, S is the stride, $P(y = i|x)$ is the probability that the input belongs to class i, z_i is the input score for class I and K is the total number of classes.

3.4. Inception V3

This study suggests a hybrid deep-learning Inception V3 model, based on a deep-learning hybrid model, for classifying tomato ailments using photos. Szegedy et al., introduced the 48-layered CNN network. It is employed in this study to determine whether or not a tomato leaf that is uploaded is infected. The model is pre-trained and kept available to determine if a picture is healthy or not. For this work, a stochastic gradient descent approach called an adaptive learning rate optimizer (Adam) is applied. Adam aids in streamlined calculation and fewer tuning parameters. To prevent overfitting issues, an early halting approach is used on the validation dataset.

The Figure 4 shows how the inception v3 uses its layers to process the image to generate the output for the given input.
Convolution: Inception-v3 employs convolutions to analyze input data. Its distinctive Inception modules blend different filter sizes (1x1, 3x3, 5x5) to capture diverse features, enhancing its capacity for extracting intricate patterns.
Convolution Padded: Padded convolution refers to the convolution operation applied to an input image with additional padding added around its borders before applying the filter.
Pooling: Pooling is employed as a down sampling technique. It reduces the spatial dimensions of feature maps, aiding in extracting higher-level features while maintaining important information

i. Patch 3x3: Considering 9 pixels at a time from the image and taking out the brightest one and learning from it
ii. Patch 8x8: Similar to 3x3, in this patch 64 pixels at a time from the image. Taking out the brightest one and learning from it

The tomato image is first fed into the architecture during the training phase as shown in Fig. 3, the image is first convoluted into 3x3 twice and additionally the convolution padding is added to analyze the image of the tomato leaf even at the edges more accurately. For down sampling the image that was convoluted in the previous steps, pooling layer is used in the architecture. Pooling helps in identifying the higher-level features of the image. The image is once again convoluted and padded for analyzing the image more effectively.

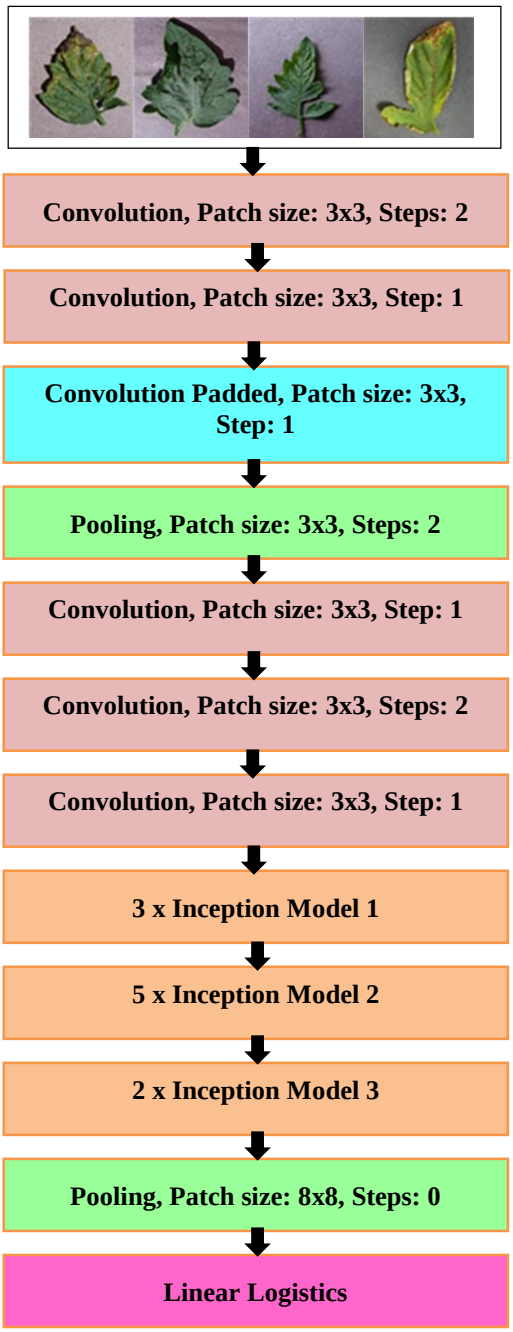


Figure 4. Flow chart demonstrating the architecture of Inception V3

3.5. Algorithm for Inception V3

The process followed to simulate the Inception V3 algorithm is as follows.

i. Import the dependencies
ii. Import Inception V3, preprocess input, decode_predictions from keras. applications.vgg19
iii. Establish the connection between the program and drive
iv. Create the input shape of the picture using inceptionv3
v. Flatten and dense the model and compile the model with the Adam optimizer
vi. Create the fitgenerator with the required epoch's values, verbose, validation steps.
vii. Visualize the results and print the accuracy levels

viii. Create the training and testing generator with the required parameters
ix. Print the true positive, false positive, false negative, false positive values and the precision, recall values, F1score of the model.

The following calculations from Eq. (7) – (12) are done while simulating the Inception V3 algorithm.
1. 1x1 Convolution:
The output feature map calculation for this layer is given by

$$H' = \frac{H-1}{S} + 1 \tag{7}$$

$$W' = \frac{W-1}{S} + 1 \tag{8}$$

2. 3x3 Convolution (with padding):

The output feature map size of this layer is given by

$$H' = \frac{H-F+2P}{S} + 1 \tag{9}$$

$$W' = \frac{W-F+2P}{S} + 1 \tag{10}$$

3. Pooling Layer:

The output size of this layer is given by

$$H' = \frac{H-F}{S} + 1 \tag{11}$$

$$W' = \frac{H-F}{S} + 1 \tag{12}$$

3.6. Evaluation Metrics

The evaluation metrics precision, recall, accuracy and F1 score are computed using Eqs. (13) – (16).

i. Precision

It refers to the measure of how well a model correctly identifies the positive instances out of all the instances it classified as positive, indicating its ability to minimize false positives.

$$Precision = \frac{TP}{TP+FP} \tag{13}$$

ii. Recall

It is a measure of how well a model identifies all the positive instances out of the total actual positive instances, reflecting its ability to minimize false negatives and capture all relevant results.

$$Recall = \frac{TP}{TP+FN} \tag{14}$$

iii. Accuracy

It refers to the measure of how well a model correctly classifies instances across all classes, indicating the overall correctness of predictions.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \tag{15}$$

iv. F1 Score

It considers both the model's ability for identifying positive instances (precision) and its ability to find all positive instances (recall).

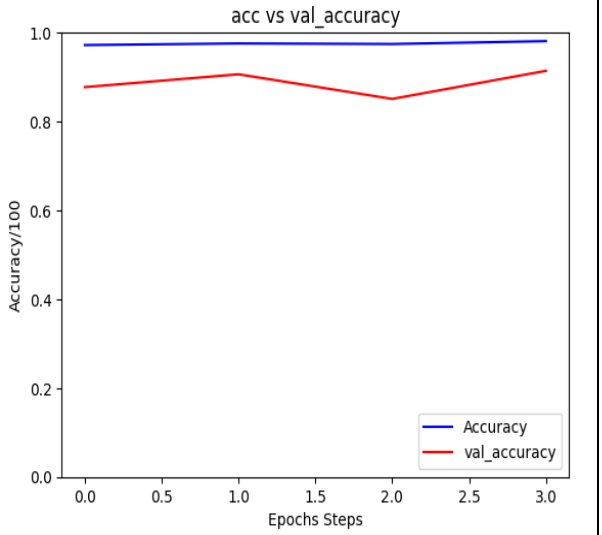
$$F1\ \ Score = 2\left(\frac{Precision * Recall}{Precision + Recall}\right) \tag{16}$$

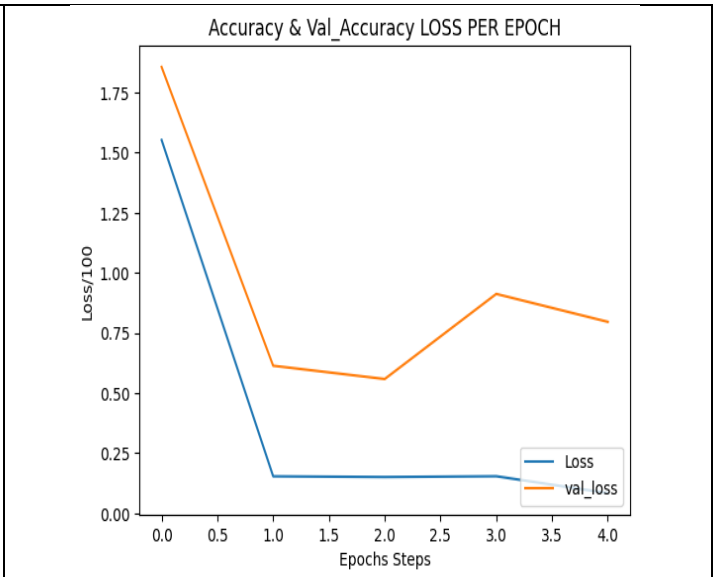
4. HYPERPARAMETER TUNING

Within the domain of training algorithms, our study aims to meticulously investigate the key parameters that exert a direct and discernible influence on the behavioural characteristics of these algorithms. Moreover, we underscore the significant role played by hyperparameter tuning in enhancing both model accuracy and performance. The discussed parameters for the hyperparameter tuning in this paper are learning rate and activation function

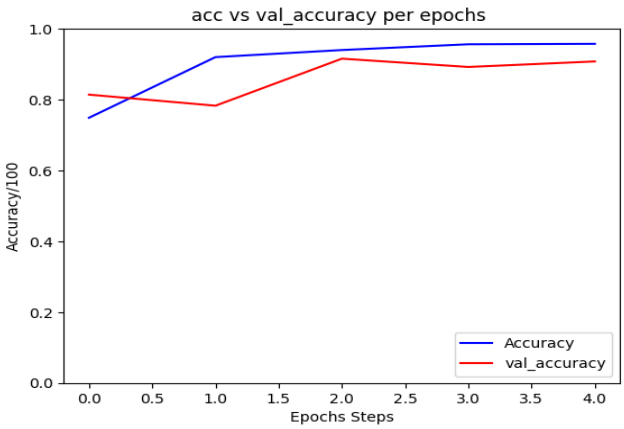
4.1 Learning Rate

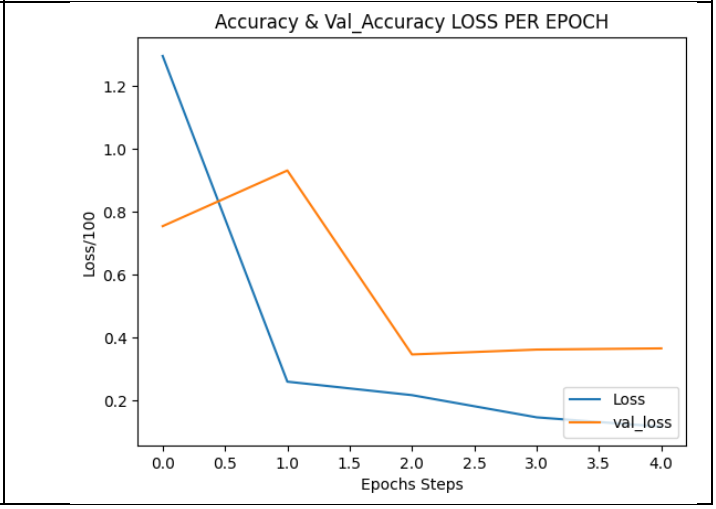
a.)





b.)





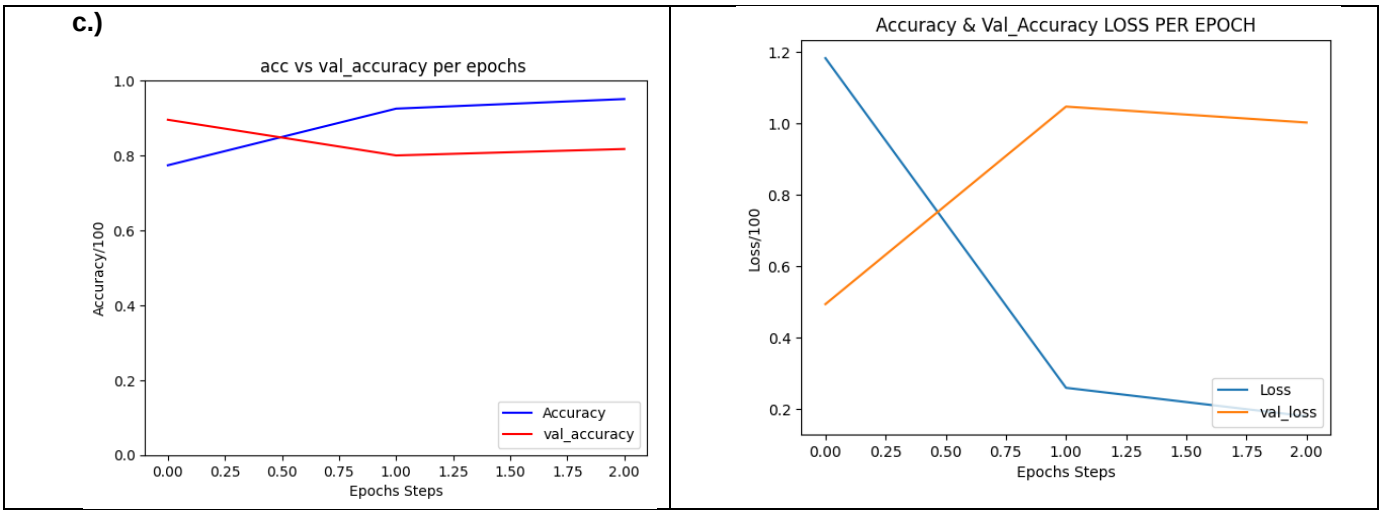
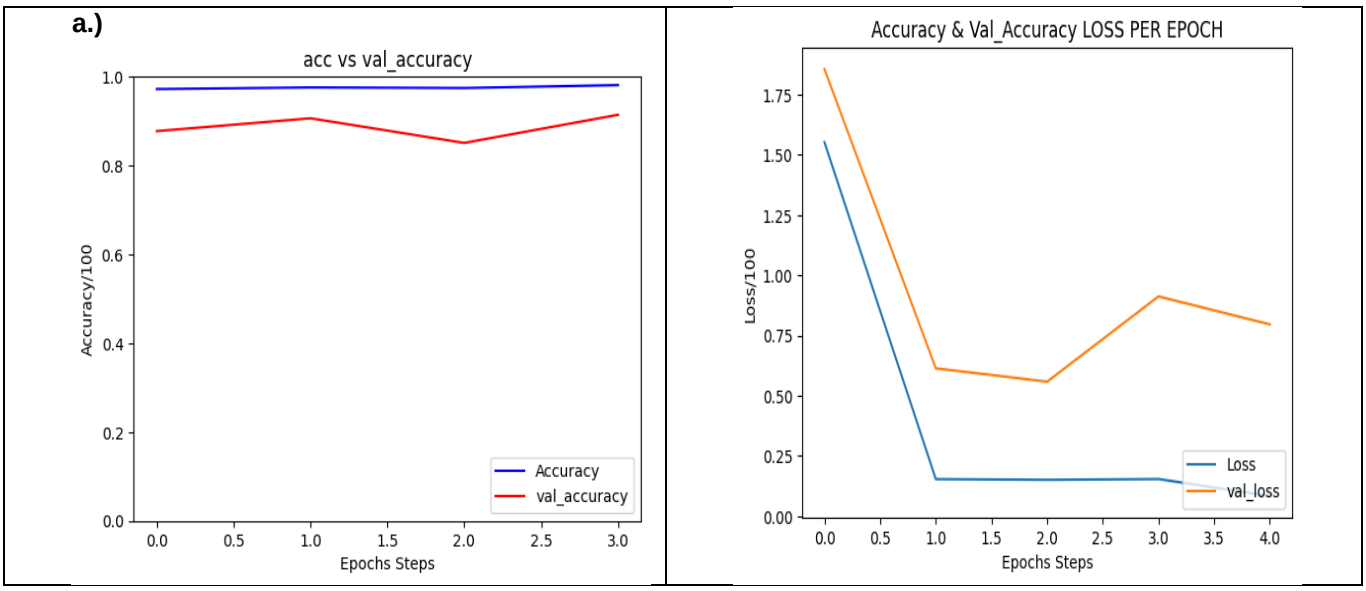


Figure 5. Typical accuracy and loss comparison plots for different learning rates (a)learning rate= 0.0001(Accuracy: - 98%) (b) learning rate= 0.001(Accuracy :- 95%) (c) learning rate= 0.0015(Accuracy :- 92%)

Upon a comprehensive analysis of the graphs presented in Figure 5, one can discern several critical observations. Firstly, in the case of (a) where the learning rate is set to 0.0001, the accuracy graph exhibits remarkable stability, with minor fluctuations within a consistent range. The corresponding loss curve indicates a notable exponential decrease, suggestive of effective training. On analyzing (b), where the learning rate is increased to 0.001, the accuracy graph displays slightly more fluctuations, commencing from a marginally lower point compared to the 0.0001 setting. Although it reaches a commendable 95% accuracy, it falls short of the stability demonstrated by the previous configuration. From (c) with a learning rate of 0.0015, the accuracy graph plateaus at around 92%, notably lower compared to other learning rates. Notably, an examination of the loss graph reveals that the validation loss is on an upward trajectory, implying an inadequate training process. From the above all learning rate, we can conclude that 0.0001 is the best learning rate for training the Inception V3 model with the dataset.

4.2 Activation Function



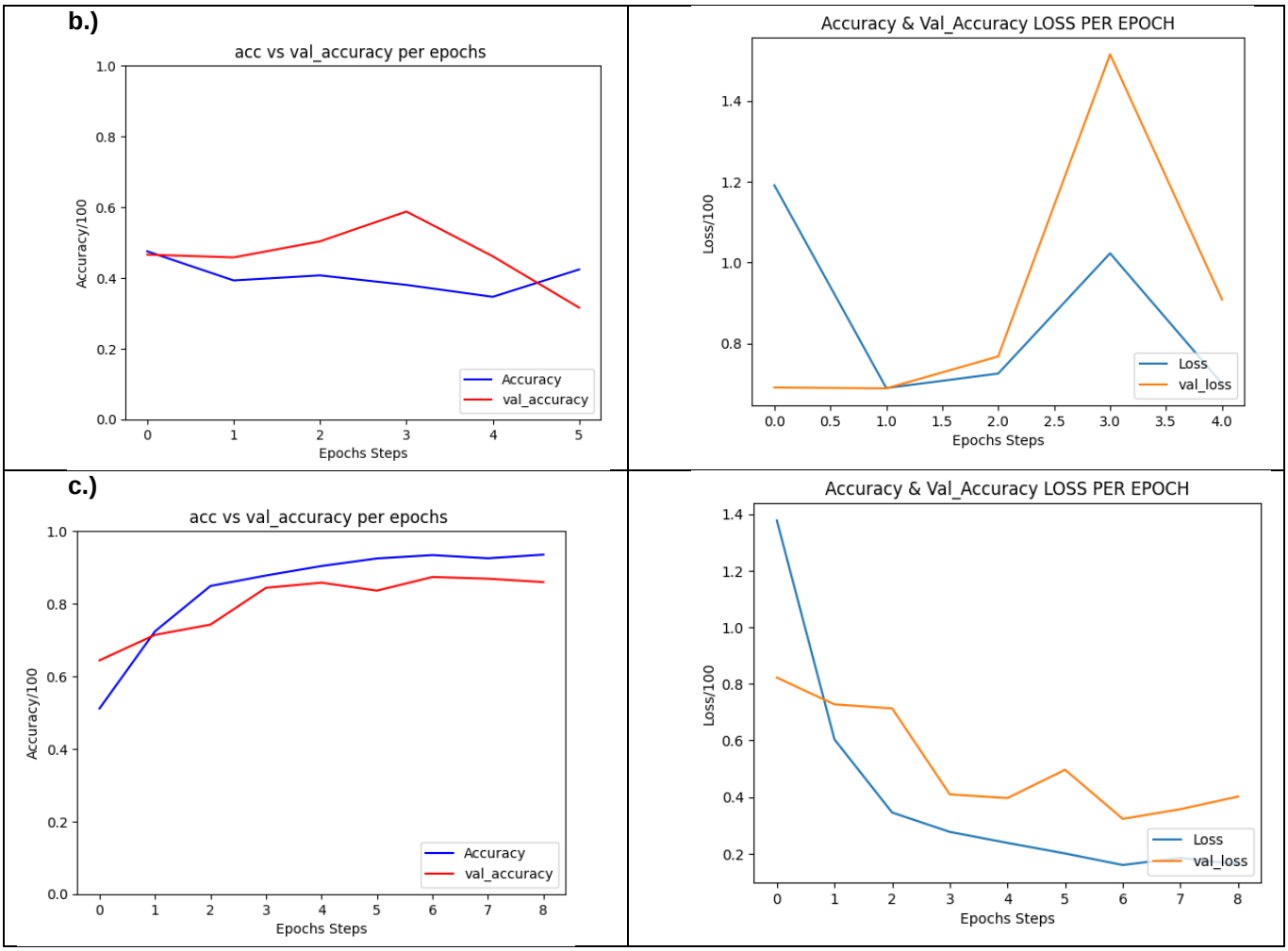


Figure 6. Typical accuracy and loss comparison plots for different Activation functions (a)Sigmoid(Accuracy: - 98%) (b) tanh(Accuracy :-52%) (c) SoftMax(Accuracy :- 92%)

Upon analyzing the graphs of three distinct activation functions in Figure 6, the following observations emerge. In Figure 6a, the sigmoid activation function demonstrates remarkable performance in terms of both accuracy and validation accuracy, with a noticeable exponential decrease in both loss and validation loss. Meanwhile, in Figure 6b, the tanh activation function exhibits accuracy and validation accuracy levels comparable to those of the sigmoid function. Lastly, in Figure 6c, SoftMax performs exceptionally well in both accuracy and validation accuracy graphs, as well as in the loss and validation loss graphs. Interestingly, when comparing the accuracy between sigmoid and SoftMax, it's worth noting that the sigmoid function outperforms SoftMax in terms of accuracy. These findings underscore the significance of selecting the right activation function to optimize the performance of your model.

5. Results and Analysis

This section discusses the experimental result for models such as VGG19 and Inception V3. The accuracy and validation accuracy were shown in Fig 7. The accuracy and validation accuracy were continuously increasing for VGG19 and Inception V3 which is shown in Figure 7 a) and b) respectively. It shows that Inception V3 is giving good accuracy and validation accuracy than VGG19.

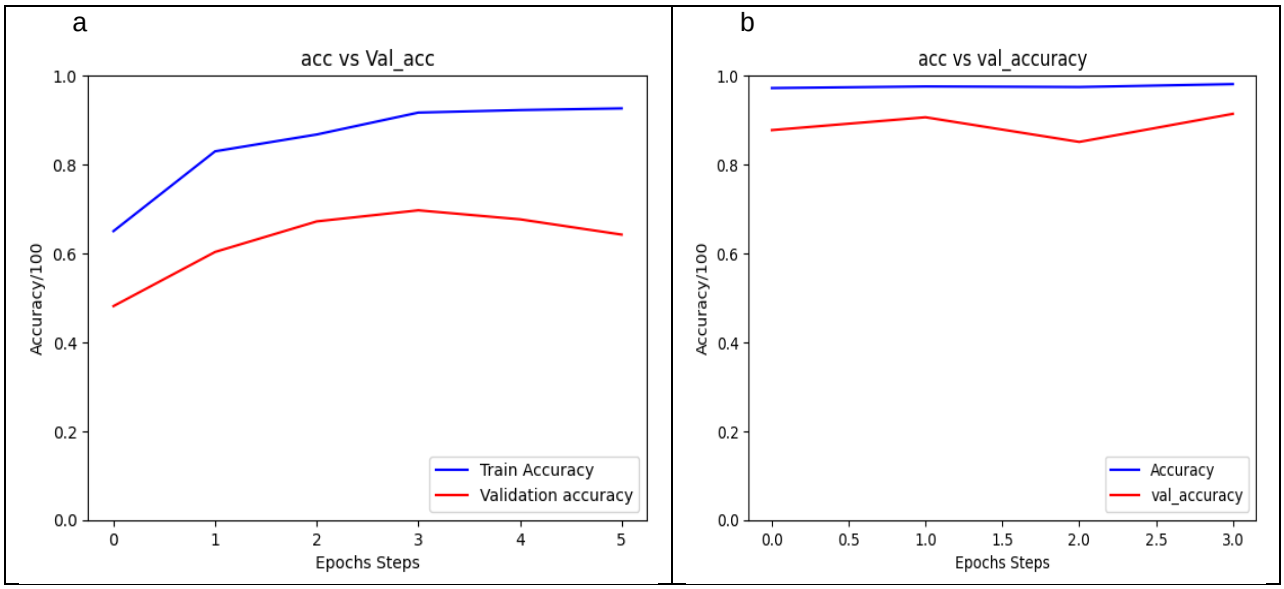


Figure 7. Accuracy and Validation accuracy per epoch for a) VGG19 b) Inception V3

The progress of the model determined by continuous decreasing of trained losses. The Fig 7a) and b) shows the visual representation of accuracy and validation accuracy loss per epoch for VGG19 and Inception V3 respectively. Inception V3 model is giving better loss per epoch than VGG 19.

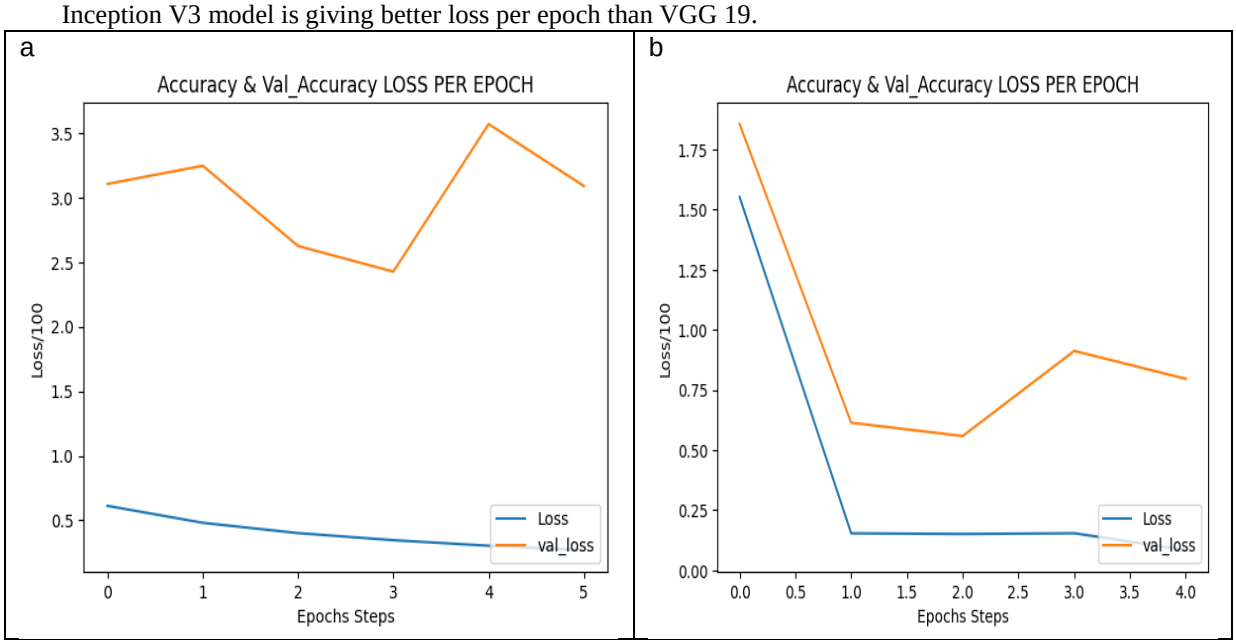


Figure 8. Loss per epoch for the model a) VGG19 b) Inception V3

The layers used in VGG19 and Inception V3 are tabulated in Table 4. VGG19 orchestrates its design around a sequence of convolutional layers followed by pooling layers and cyclic activation functions, culminating in the judicious application of batch normalization for regularization. This orchestration induces a gradual reduction in spatial dimensions of the input images, effectively managed through strategically placed max-pooling layer. On the other hand, the Inception V3 model undertakes a more intricate approach. It assembles an intricate arrangement of convolutional layers, with its standout feature being the inception blocks. These inception blocks intricately fuse diverse convolutional operations, resulting in feature maps of augmented complexity. Intermittently woven into this fabric of convolutions is the pivotal batch normalization mechanism, serving to stabilize and accelerate the learning process. In essence, while VGG19 adheres to a step-by-step reduction of image dimensions, Inception V3 employs a multidimensional strategy by weaving together intricate convolutional layers, augmented by the fusion of diverse

convolutions within inception blocks. In both networks, the normalization factor of batch normalization plays a crucial role in the quest for optimized and efficient learning outcomes.

Table 4. Layers in VGG19 and Inception V3

Layer (type)	VGG19	Inception V3
input_1 (InputLayer)	(None, 128, 128, 3)	(None, 128, 128, 3)
block1_conv1 (Conv2D)	(None, 128, 128, 64)	(None, 63, 63, 32)
batch_normalization	/	(None, 63, 63, 32)
activation_94 (Activation)	/	(None, 63, 63, 32)
block1_conv2 (Conv2D)	(None, 128, 128, 64)	None, 61, 61, 32
batch_normalization	/	(None, 61, 61, 32)
activation_95 (Activation)	/	(None, 61, 61, 32)
block2_conv1 (Conv2D)	(None, 64, 64, 128)	(None, 30, 30, 80)
batch_normalization	/	(None, 61, 61, 64)
activation_96 (Activation)	/	(None, 61, 61, 64)
max_pooling2d_4 (MaxPooling2D)	/	(None, 30, 30, 64)
block3_conv1 (Conv2D)	(None, 32, 32, 256)	(None, 13, 13, 64)
block3_conv2 (Conv2D)	(None, 32, 32, 256)	(None, 13, 13, 48)
block3_conv3 (Conv2D)	(None, 32, 32, 256)	(None, 13, 13, 96)
block3_conv4 (Conv2D)	(None, 32, 32, 256)	(None, 13, 13, 64)
block3_pool (MaxPooling2D)	(None, 16, 16, 256)	(None, 13, 13, 64)
block4_conv1 (Conv2D)	(None, 16, 16, 512)	(None, 13, 13, 96)
block4_conv2 (Conv2D)	(None, 16, 16, 512)	(None, 13, 13, 32)
block4_conv3 (Conv2D)	(None, 16, 16, 512)	(None, 13, 13, 64)
block4_conv4 (Conv2D)	(None, 16, 16, 512)	(None, 13, 13, 48)
block4_pool (MaxPooling2D)	(None, 8, 8, 512)	(None, 13, 13, 256)
block5_conv1 (Conv2D)	(None, 8, 8, 512)	(None, 13, 13, 64)
block5_conv2 (Conv2D)	(None, 8, 8, 512)	(None, 13, 13, 64)
block5_conv3 (Conv2D)	(None, 8, 8, 512)	(None, 13, 13, 96)
block5_conv4 (Conv2D)	(None, 8, 8, 512)	(None, 6, 6, 384)
block5_pool (MaxPooling2D)	(None, 4, 4, 512)	(None, 6, 6, 96)
concatenate_3 (Concatenate)	/	(None, 2, 2, 768)
batch_normalization_187	/	(None, 2, 2, 192)
activation_187 (Activation)	/	(None, 2, 2, 192)
mixed10 (Concatenate)	/	(None, 2, 2, 2048)
flatten_3 (Flatten)	(None, 8192)	(None, 8192)
dense_3 (Dense)	(None, 2)	(None, 2)
batch_normalization	(None, 2)	(None, 2)

The confusion matrix contains true positive, true negative, false positive, and false negative values. This matrix is used for identifying the prediction of how many healthy and unhealthy tomato leaves in the given dataset. The illustration of confusion matrix is shown in Fig 9a) and b) for VGG19 and Inception V3 respectively.

a	b

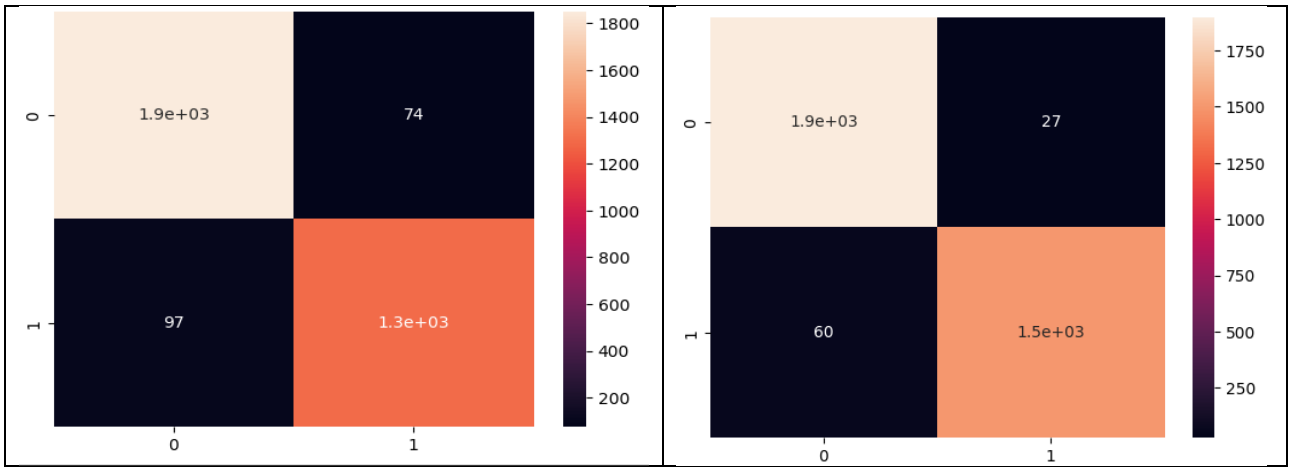


Figure 9. Confusion Matrix a) VGG19 and b) Inception V3

Table 5. Evaluation of confusion matrix for VGG19 and Inception V3

Classification Report	Numbers for VGG19	Numbers for Inception V3
True Positive (TP)	1852	1899
False Positive (FP)	74	27
False Negative (FN)	97	60
True Negatives(TN)	1303	1503

Table 6. Evaluation metrics of VGG19 and Inception V3

	VGG19		Inception V3	
Evaluation Metrics	Healthy	Unhealthy	Healthy	Unhealthy
Precision	95%	95%	97%	98%
Recall	96%	93%	99%	96%
F1-Score	96%	94%	98%	97%
Support	1926	1400	1926	1563
Accuracy	94%		98%	

A detailed numerical evaluation of the confusion matrix presented in Figure 9 (a) and Figure 9 (b) has been tabularized in Table 5. The VGG19 algorithm has shown a total of 1,852 true positive (TP) values, 74 false positive (FP) values, 97 false negative (FN) values and 1,303 True Negative (TN) values. Comparatively, the Inception V3 has shown a higher number of TP values i.e. 1,899. Consequently, it shows 27, a lesser number of FP numbers than VGG19. The number of FNs shown by Inception V3 are 60 while the TNs are 1,503.

Both algorithms under consideration, the VGG19 and Inception V3 have been simulated for healthy and unhealthy tomato leaves. Table 6 summarizes the evaluation results obtained after the simulation of the two models. VGG19 showed a 95% precision for both healthy and unhealthy leaves. A recall level of 96% for healthy leaves and 93% for unhealthy leaves was shown by VGG19 while the F1 Score was 96% and 94% for healthy and unhealthy leaves dataset respectively. Summarizing this data (Fig. 10), the VGG19 model showed an overall accuracy of 94%. Inception V3 has shown far better results for both healthy and unhealthy leaves dataset when compared to VGG19. The precision was found to be 97% and 98% for healthy and unhealthy parts respectively. The recall values were observed to be 99% for healthy and 96% for unhealthy leaves subparts. The F1 Score was found to be 98% and 97% for healthy and unhealthy counterparts which was much higher than what was seen from VGG19 simulation. Finally, an overall accuracy level of 98% was demonstrated by the Inception V3 model which is 4% higher than the accuracy shown by VGG19. Hence, it can be concluded at the end of this comparison that the Inception V3 model performed much better than the VGG19 model when compared for healthy and unhealthy leaves datasets.

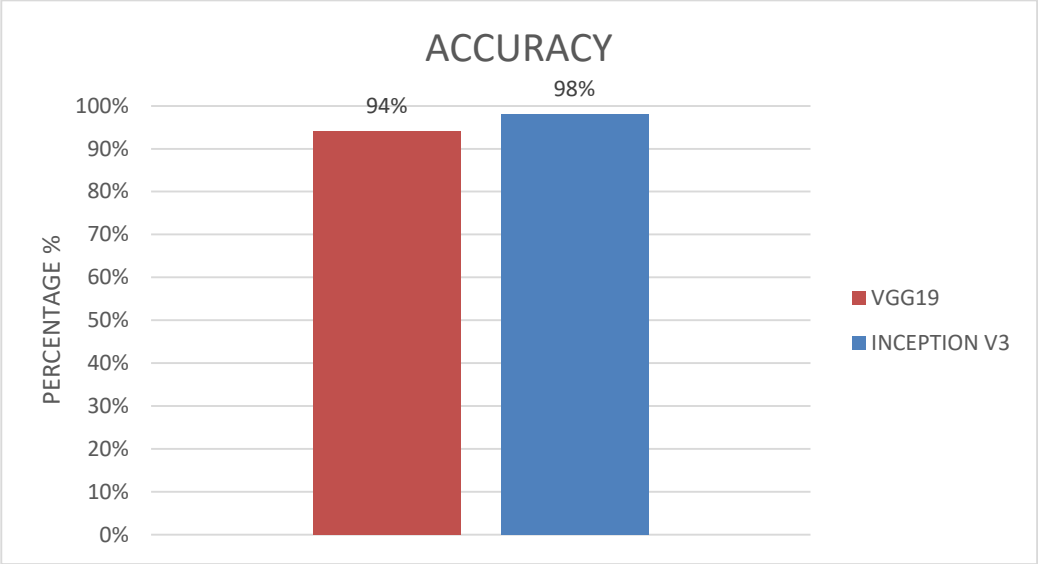


Fig. 10. Accuracy of VGG19 and Inception V3

6 Conclusion

This paper primarily focuses on identifying healthy and unhealthy tomato leaves of the plants, for the dataset taken from Kaggle. The proposed model used to classify whether uploaded tomato leaf image is healthy or not. From the result, farmer have to care only the unhealthy leaf of tomato plant with the required materials. This paper compared two algorithms, VGG19 and Inception V3. By analyzing both the algorithms, Inception V3 algorithm was able to obtain an accuracy of 98% and validation accuracy of 89%, but VGG19 algorithm was able to achieve only 94% accuracy. From the values obtained across the evaluation metrics, Inception V3 performed well. Therefore, Inception V3 was the algorithm used for training the model for identifying the health status of the leaves. The selected model (Inception V3) also underwent hyperparameter tuning with learning rates and various activation functions. The best values were taken from the hyperparameters (learning rate =0.0001 and activation function as sigmoid)Further all these processes can be implemented as a mobile app for the helpfulness of farmers to detect the healthiness of tomato leaf. This app imports images using the camera of phone and upload it to the model directly and classify the health of tomato leaf.

Several ways in which the system could be further improved in terms of effectiveness and usability can be considered in future research. Looking at other state-of-the-art deep learning architectures, like EfficientNet or transformer-based models, to achieve potentially better accuracy and efficiency is another direction that can be pursued. Development of real-time disease detection systems able to process video feeds or integrate with drone technology for large-scale monitoring would significantly increase this application's scope.

Funding: “This research received no external funding”

Conflicts of Interest: “The authors declare no conflict of interest.”

Acknowledgements

Authors would like to thank VIT Chennai for the computational facilities and for the motivation. The third author is thankful to ABCD Future Environmental Leaders Scholarship 2022 funded by the DAAD “Global Water and Climate Adaptation Centre-Aachen, Bangkok, Chennai, Dresden (ABCD-Centre)”, Indo-German Centre for Sustainability (IGCS) and RWTH-IIT Madras Junior Research Fellowship (JRF) for the financial support to stay in ITA, RWTH Aachen University at the time of manuscript writing.

Funding: This research received no external funding.

Conflict of Interests: The authors declare that they have no conflict of interest.

Data Availability Statement: Data are accessed through the following link https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf

References

[1] K. S. Kumar, S. Paswan, and S. Srivastava, "Tomato—a natural medicine and its health benefits," Journal of Pharmacognosy and Phytochemistry, vol. 1, no. 1, pp. 33-43, 2012.

[2] K. Satyagopal, S. N. Sushil, P. Jeyakumar, G. Shankar, O. P. Sharma, D. R. Boina, and S. Latha, AESA Based IPM Package for Tomato, 2014.

[3] Y. Deng, H. Xi, G. Zhou, A. Chen, Y. Wang, L. Li, and Y. Hu, "An effective image-based tomato leaf disease segmentation method using MC-UNet," Plant Phenomics, vol. 5, p. 0049, 2023.

[4] M. Agarwal, R. K. Kaliyar, G. Singal, and S. K. Gupta, "FCNN-LDA: A Faster Convolution Neural Network model for Leaf Disease identification on Apple's leaf dataset," in Proceedings of the 2019 12th International Conference on Information & Communication Technology and System (ICTS), 2019, pp. 246-251, IEEE.

[5] S. Huang, W. Liu, F. Qi, and K. Yang, "Development and validation of a deep learning algorithm for the recognition of plant disease," in Proceedings of the 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019, pp. 1951-1957, IEEE.

[6] Kaustubhb999, Tomato Leaf Disease Dataset. Available: https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf. Accessed: 2023.

[7] Y. Deng, H. Xi, G. Zhou, A. Chen, Y. Wang, L. Li, and Y. Hu, "An effective image-based tomato leaf disease segmentation method using MC-UNet," Plant Phenomics, vol. 5, p. 0049, 2023. DOI:10.34133/plantphenomics.0049.

[8] A. Aggarwal, "Biological tomato leaf disease classification using deep learning framework," International Journal of Biological and Biomedical Engineering, vol. 16, no. 1, pp. 241-244, 2022.

[9] J. Sujithra and M. F. Ukrit, "A review on crop disease identification and classification through leaf images," European Journal of Molecular & Clinical Medicine, vol. 7, no. 09, pp. 2020-2020, 2020.

[10] A. Malik, G. Vaidya, V. Jagota, S. Eswaran, A. Sirohi, I. Batra, and E. Asenso, "Design and evaluation of a hybrid technique for detecting sunflower leaf disease using deep learning approach," Journal of Food Quality, vol. 2022, pp. 1-12, 2022.

[11] L. D. Nguyen, D. Lin, Z. Lin, and J. Cao, "Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation," in Proceedings of the 2018 IEEE International Symposium on Circuits and Systems (ISCAS), 2018, pp. 1-5, IEEE.

[12] A. Guerrero-Ibañez and A. Reyes-Muñoz, "Monitoring tomato leaf disease through convolutional neural networks," Electronics, vol. 12, no. 1, p. 229, 2023. https://doi.org/10.3390/electronics12010229.

[13] H. Nagamani and S. Dr. Sarojadevi, "Tomato leaf disease detection using deep learning techniques," International Journal of Advanced Computer Science and Applications (IJACSA), vol. 13, no. 1, 2022.

[14] M. Bouni, B. Hssina, K. Douzi, and S. Douzi, "Impact of pretrained deep neural networks for tomato leaf disease prediction," Journal of Electrical and Computer Engineering, vol. 2023, 2023. https://doi.org/10.1155/2023/5051005.

[15] S. S. Harakannanavar, J. M. Rudagi, V. I. Puranikmath, A. Siddiqua, and R. Pramodhini, "Plant leaf disease detection using computer vision and machine learning algorithms," Global Transitions Proceedings, vol. 3, no. 1, pp. 305-310, 2022. https://doi.org/10.1016/j.gltp.2022.03.016.

[16] P. Kaur, S. Harnal, V. Gautam, M. P. Singh, and S. P. Singh, "An approach for characterization of infected area in tomato leaf disease based on deep learning and object detection technique," Engineering Applications of Artificial Intelligence, vol. 115, p. 105210, 2022. https://doi.org/10.1016/j.engappai.2022.105210.

[17] T. Vadivel and R. Suguna, "Automatic recognition of tomato leaf disease using fast enhanced learning with image processing," Acta Agriculturae Scandinavica, Section B — Soil & Plant Science, vol. 72, no. 1, pp. 312-324, 2022. DOI: 10.1080/09064710.2021.1976266.

[18] S. Jeong, S. Jeong, and J. Bong, “Detection of tomato leaf miner using deep neural network,” Sensors, vol. 22, no. 24, p. 9959, 2022. https://doi.org/10.3390/s22249959.

[19] Z. Tang, X. He, G. Zhou, A. Chen, Y. Wang, L. Li, and Y. Hu, “A precise image-based tomato leaf disease detection approach using PLPNet,” Plant Phenomics, vol. 5, p. 0042, 2023.

[20] C. Vengaiah and S. R. Konda, “Improving tomato leaf disease detection with DenseNet-121 architecture,” International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 3, pp. 442-448, 2023.

[21] M. Bhandari, T. B. Shahi, A. Neupane, and K. B. Walsh, “Botanicx-ai: Identification of tomato leaf diseases using an explanation-driven deep-learning model,” Journal of Imaging, vol. 9, no. 2, p. 53, 2023.

[22] O. Attallah, “Tomato leaf disease classification via compact convolutional neural networks with transfer learning and feature selection,” Horticulturae, vol. 9, no. 2, p. 149, 2023.

[23] X. Huang, A. Chen, G. Zhou, X. Zhang, J. Wang, N. Peng, and C. Jiang, “Tomato leaf disease detection system based on FC-SNDPN,” Multimedia Tools and Applications, vol. 82, no. 2, pp. 2121-2144, 2023.

[24] S. U. Rahman, F. Alam, N. Ahmad, and S. Arshad, “Image processing based system for the detection, identification and treatment of tomato leaf diseases,” Multimedia Tools and Applications, vol. 82, no. 6, pp. 9431-9445, 2023.

[25] W. Ahmad, S. M. Adnan, and A. Irtaza, “Local triangular-ternary pattern: a novel feature descriptor for plant leaf disease detection,” Multimedia Tools and Applications, pp. 1-27, 2023.

[26] Zhou, G., Zhang, W., Chen, C., "Deep Convolutional Neural Network for Leaf Disease Identification in Rice," *IEEE Access*, Vol. 7, pp. 115442-115454,

[27] Kamilaris, A., Prenafeta-Boldú, F.X., "Deep Learning in Agriculture: A Survey," *Computers and Electronics in Agriculture*, Vol. 147, pp. 70-90, 2018.

[28] Ferentinos, K.P., "Deep Learning Models for Plant Disease Detection and Diagnosis," *Computers and Electronics in Agriculture*, Vol. 145, pp. 311-318, 2018.

[29] Ramcharan, A., McCloskey, P., Baranowski, K., et al., "A Mobile-Based Deep Learning Model for Cassava Disease Diagnosis," *Frontiers in Plant Science*, Vol. 8, pp. 1852, 2017.

[30] Brahimi, M., Arsenovic, M., Laraba, S., et al., "Deep Learning for Plant Diseases: Detection and Saliency Map Visualization," *IEEE Access*, Vol. 7, pp. 102750-102763, 2019.

[31] Wang, G., Sun, Y., Wang, J., "Automatic Image-Based Plant Disease Severity Estimation Using Deep Learning," *Computational Intelligence and Neuroscience*, Vol. 2017, Article ID 2917536, 2017.