

BTP Project – B22RS01

CROP DISEASE DETECTION

**Under The Guidance of
Dr R. Shathanaa**



OUTLINE

- Introduction
- Motivation
- Problem Statement
- Literature review
- Present work
- CNN, Inception-V3, VGG16 Architectures
- Model Results Comparison
- All Results
- RoadMap
- References

INTRODUCTION

- Agricultural production rate plays a vital role in the economic development of a country.
- The identification of plant diseases at an early stage is crucial for global health and wellbeing. So, controlling on the diseased leaves during the growing stages of crops is a crucial step.
- Moreover, increasing crop production is essential to areas where food is scarce.

INTRODUCTION

- Loss of crops from plant diseases would result in reduction of income for crop producers, higher prices for consumers and significant economic impact.
- The access to disease-control methods is limited which results in annual losses of 30 to 50 percent for major crops in various countries. [1]
- Hence, detection of crop diseases is very crucial for economic development.





MOTIVATION

- Agriculture is a very important sector of the Indian economy.
- The share of agriculture in GDP increased to 20.2 per cent in 2020-21 from 18.4 per cent in 2019-20. [2]
- So, for the identification and detection of plant diseases, human raters are employed.

MOTIVATION

- This process is very time consuming and expensive and sometimes may lead to poorly identify the crop disease.
- Thus, continuous monitoring must be done which is a repetitive process which involves large group of experts costing very high when dealing with large farms.
- Therefore, this motivated us to automate the process and perform evaluation metrics, based on a deep learning classifier.

PROBLEM STATEMENT

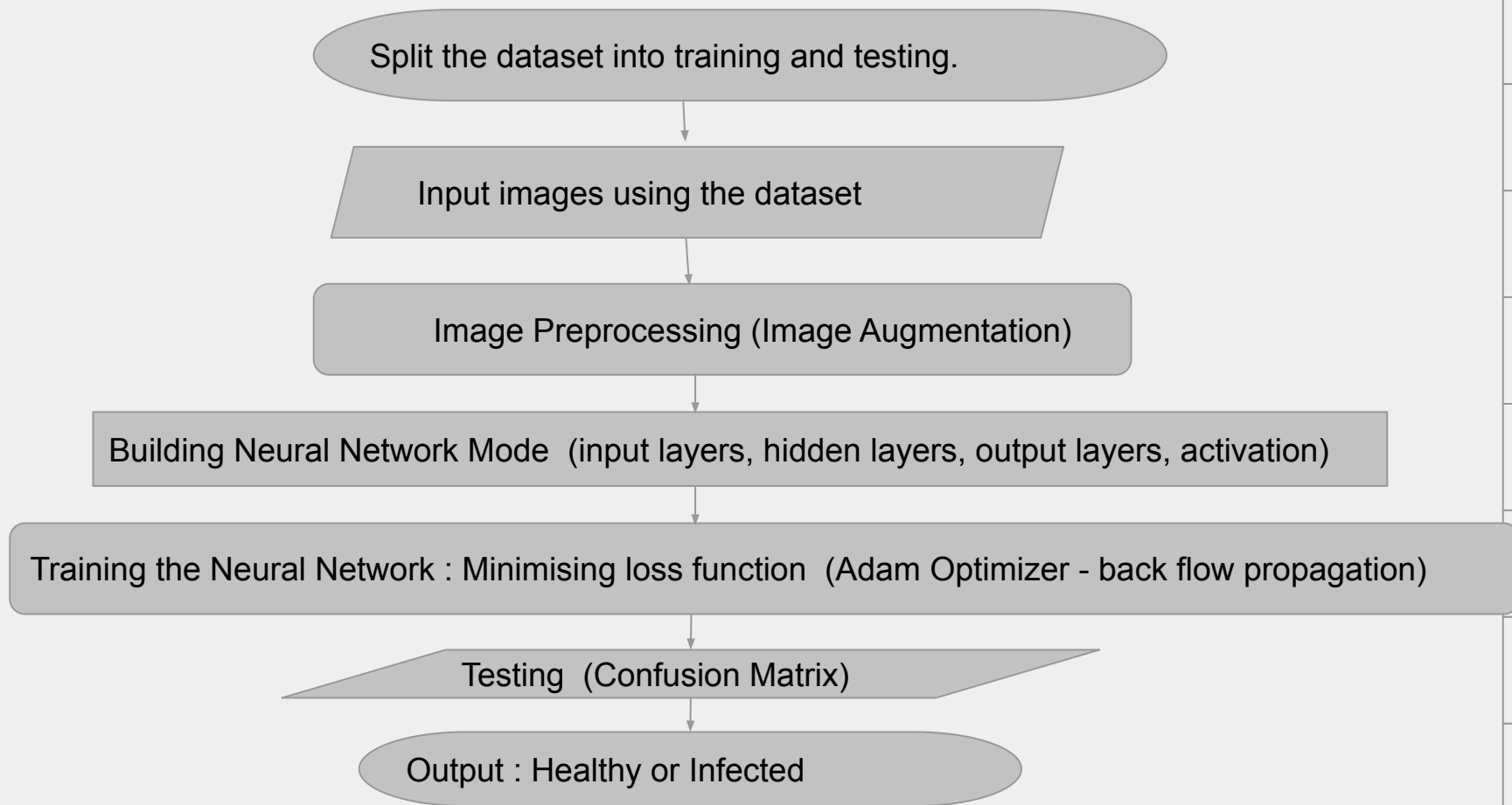
- Detect the crop diseases.
- Differentiate between various crop diseases for a particular plant and then for many various plants.
- Using object detection algorithms to find the diseased area in the crop on the basis of the features such as color, wilt, leaf spots, unusual size of the leaf.
- Choosing appropriate neural network for classification.



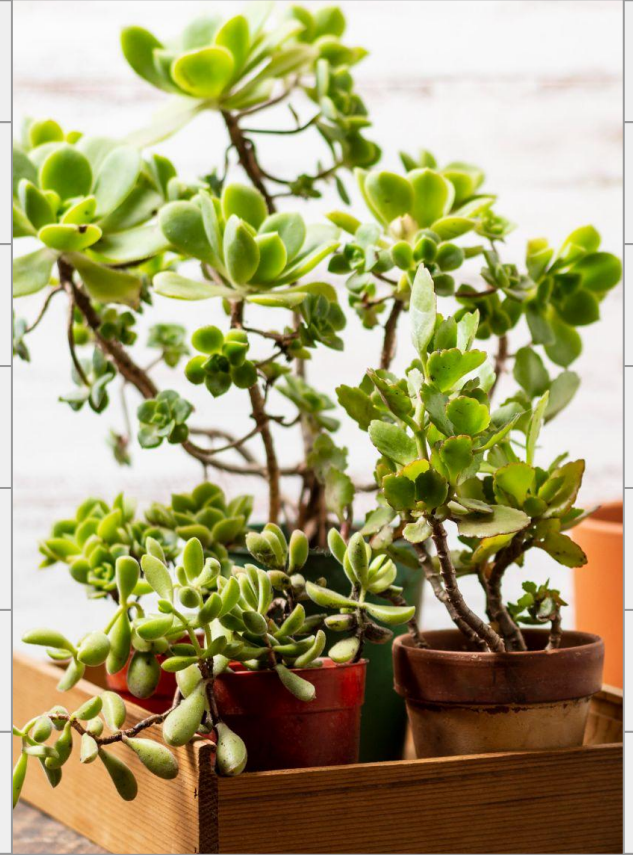
OBJECTIVE

1. Our Main Objective is that given the images of a specific crop, it should classify it as a healthy crop or the disease it may be infected with.
2. We intend to use deep convolutional neural networks (D-CNN) and some concepts of image processing to reach our goal of identifying the disease of a plant using color, leaf spots, etc.
3. We developed a web based application so that it can be used widely by large number of people to determine crop diseases.

PROCEDURE



LITERATURE REVIEW



Literature Review – Base Paper-1 [1]

AUTHOR AND YEAR	TITLE	METHODOLOGY	EVALUATION PARAMETERS	DRAWBACKS
Paymode, Ananda S., and Vandana B. Malode (2022) - ScienceDirect- ISSN 2589-7217	Transfer Learning for Multi-Crop Leaf Disease Image Classification using Convolutional Neural Network VGG [4]	1. Dataset (PlantVillage) 2. Image-processing using Picture filtering, grey transformation, picture sharpening, and scaling ..) 3. Image augmentation (flipping, cropping, rotation, colour transformation) 4. Transfer learning VGG	Accuracy $(TP+TN)/(TP+FP+FN+TN)$ Accuracy for grapes crop : 1. Proposed VGG16 : 98.40% 2. Inception-ResNet-V2: 81.11% Accuracy for tomato crop: 1. Proposed VGG16 : 95.71% 2. Inception-ResNet-V2: 86.10%	1. Preparation of genuine datasets and applying to the deep learning models with multiple crops leaves images more than two is not done. 2. The use of Inception V3 and ResNet-based CNN models for much deeper analysis of crop images is an anticipation which is not done.

MODELS

We have implemented 3 models successfully on tomato plant.

1. CNN (with 3 different variants)

2. INCEPTION-v3

3. VGG-16

STATE OF ART - EXISTING ARCHITECTURE

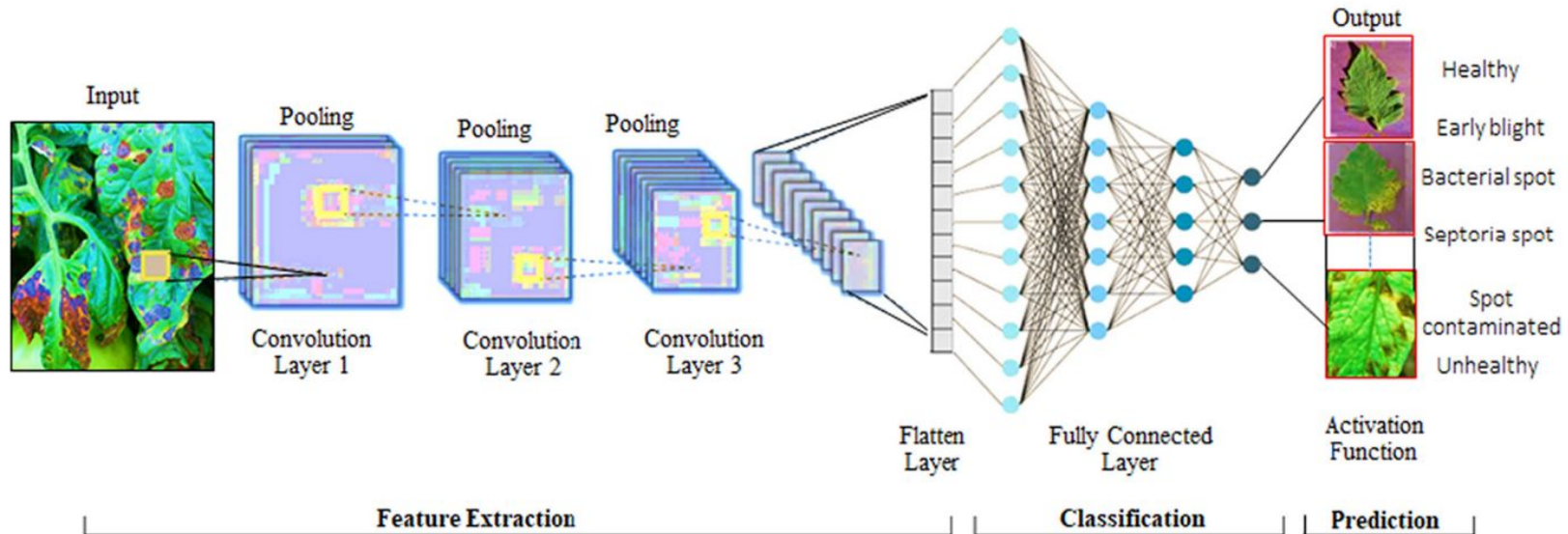


Fig 6. Proposed convolutional neural network CNN architecture.

OLD DATASET

Classes(10)	Tomato__Bacterial_spot, Tomato__Early_blight, Tomato__healthy, Tomato__Late_blight, Tomato__Leaf_Mold, Tomato__Septoria_leaf_spot, Tomato__Spider_mites Two-spotted_spider_mite, Tomato__Target_Spot, Tomato__Tomato_mosaic_virus, Tomato__Tomato_Yellow_Leaf_Curl_Virus.
Total images	10000 images for training data 1000 images for testing data
Image Size	255 X 255
Other Information	<ul style="list-style-type: none">• The data set contains the top 10 classes of diseases which are highly occurred on tomato plant.• The images had taken with different angles, with different backgrounds, and in different lighting conditions.

Dataset Link: <https://www.kaggle.com/datasets/kaustubhb999/tomatoleaf>

PRESENT WORK



NEW AUGMENTED DATASET

Classes(10)	Tomato___Bacterial_spot, Tomato___Early_blight, Tomato___healthy, Tomato___Late_blight, Tomato___Leaf_Mold, Tomato___Septoria_leaf_spot, Tomato___Spider_mites Two-spotted_spider_mite, Tomato___Target_Spot, Tomato___Tomato_mosaic_virus, Tomato___Tomato_Yellow_Leaf_Curl_Virus.
Total images	18,345 images for training data 4,585 images for testing data
Image Size	255 X 255
Other Information	<ul style="list-style-type: none">• A combination of picture flipping, rotation, blur, relighting, and random cropping, image augmentation artificially builds training images. In this, we scale, shear, zoom, and horizontally flip the photos as part of the image augmentation process.• The images were stored in the RGB image by default and were preserved in the unprocessed JPG file type.

Dataset Link: <https://www.kaggle.com/datasets/noulam/tomato>

CHANGES MADE FROM PAST REVIEW

The `flow_from_directory()` method takes a path of a directory and generates batches of augmented data. I have specified the size as 32.

1. We performed the K-fold cross-validation and hold-out method for the CNN1 model.

2. Firstly, in k-fold cross-validation, we split the dataset into `X_train`, `Y_train`, `X_test`, and `Y_test`. Now, we performed K-fold cross-validation with `k=10` and then passed the model into the for loop. Each time the accuracies of the model were evaluated and noted down. To keep track of all the validation accuracies, we appended them to a list.

3) One thing to further note is that we used epochs as 50 now to train the model. An epoch is a single iteration through the training data. Each and every sample from our training dataset will be used once per epoch, whether it is for training or validation. Therefore, the more epochs, the more the model is trained.

PERFORMING VALIDATION

4) We now plot the k-fold vs Val accuracy and observe it and then finally evaluate on the test set to observe the accuracy and loss over the epochs.

5) Now, for the holdout method, we used the ImageDataGenerator method of Keras pre-processing. From ImageDataGenerator, we used the validation_split attribute with subsets such as training and validation categories to split the data as per the choice.

Suppose, if the parameter is 0.2, we get the train data of 80% and the validation data of 20% of the original one.

6) Now, created the validation data with the name validation_generator with rescaling it for pre-processing using the flow_from_directory method which would now be used for calculating validation accuracies and loss.

ITERATING THROUGH 50 EPOCHS

7) Finally, built the model using the same above steps with `validation_data` input to the `fit` method of the model and obtained the curves for training and validation accuracies.

8) Performed the same steps using the hold-out method for each and every model with 50 epochs and new augmented dataset and observed the results.

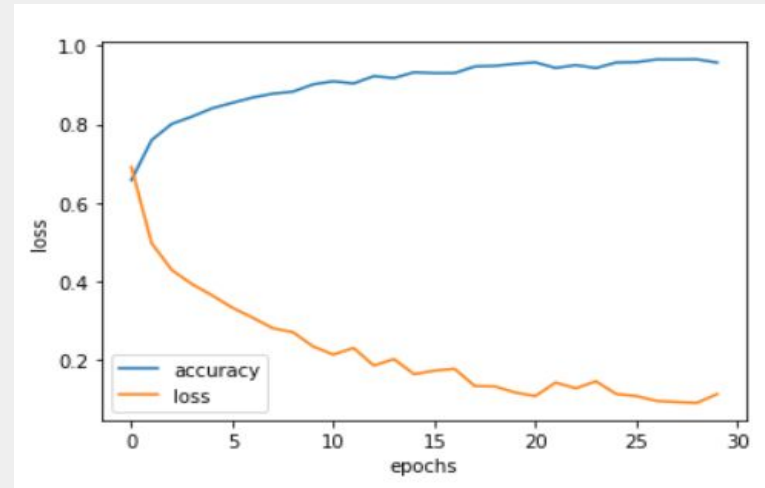
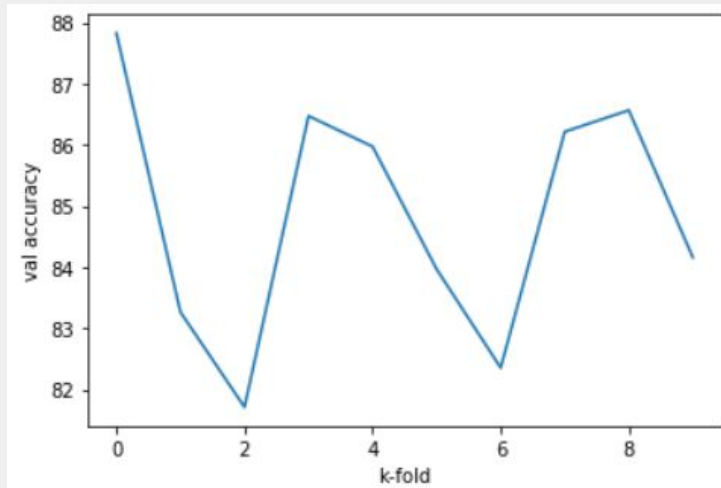
CROSS-VALIDATION TECHNIQUES

1) K-FOLD CROSS-VALIDATION

- K-fold Cross-Validation is when the dataset is split into a K number of folds and is used to evaluate the model's ability when given new data. K refers to the number of groups the data sample is split into.
- For example, if you take the k-value is 10, we can call this a 10-fold cross-validation.
- For this purpose of cross validation, a 10 fold stratified K-fold cross validation is performed. Hence, the training data is picked from a strata randomly with ten different hold-outs per fold.
- Finally, we plot the cross validation accuracy across 10 folds and find the mean to get a validation accuracy score. From my observation, going for this approach instead of a typical train-val split helped increase test accuracy by more than 10%.

K-FOLD CROSS-VALIDATION

The following are the plots for k-fold vs validation accuracies over 10 folds and the other plot signifies the validation accuracies and loss of the data for 30 epochs



HOLD-OUT METHOD

- The holdout method is the simplest kind of cross validation. Hold-out is when you split up your dataset into a 'train' and 'test' set. The training set is what the model is trained on, and the test set is used to see how well that model performs on unseen data.
- Holdout data is important in supervised machine learning to verify that the model that was trained and validated on historical data which will produce similar performance when using new data while in operation.
- A common split that is generally used and that we have used when using the hold-out method is using 80% of data for training and the remaining 20% of the data for testing.

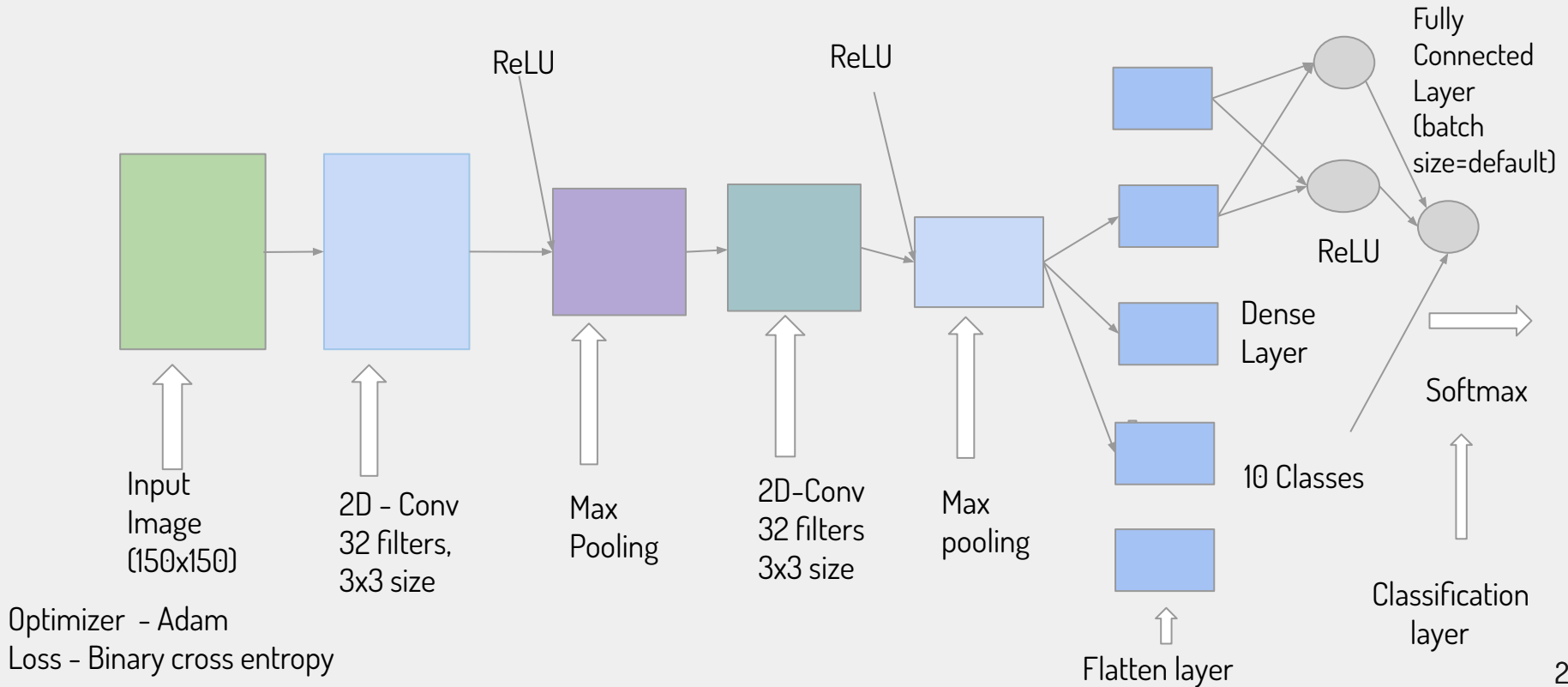
RE-ITERATING THE MODELS

- Moreover, to split the given data into training and validation data set, we used the help of ImageDataGenerator with validation_split and subsets parameters to split the data into two sets. Now, created the validation data with the name validation_generator with rescaling.
- Suppose, if the parameter is 0.2, we get the train data of 80% and the validation data of 20% of the original one.
- Now, we performed this validation over each and every model of ours by passing the validation data as an additional argument and re-iterated all the models that we have built with 50 epochs.
- For example,
`temp = cnn.fit(x = train_generator, validation_data=validation_generator,epochs=50)`

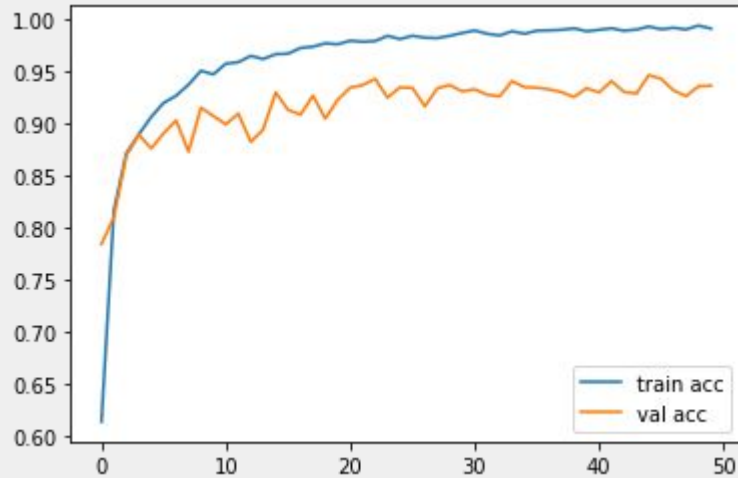
MODELS



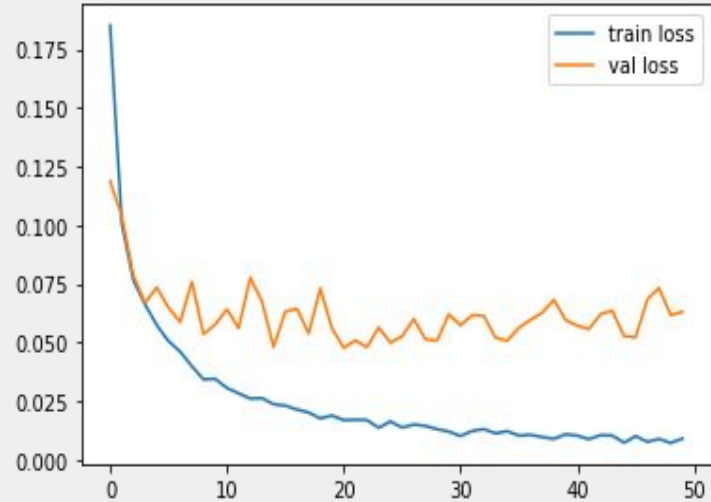
CNN ARCHITECTURE -1



CNN1 RESULTS



Training and Validation Accuracy



Training and Validation Loss

CNN1 RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL TESTING IMAGES
Tomato leaf disease Detection (Total 18,345)	14,678 (80%)	4,585

ACCURACY

Training accuracy - 99.37

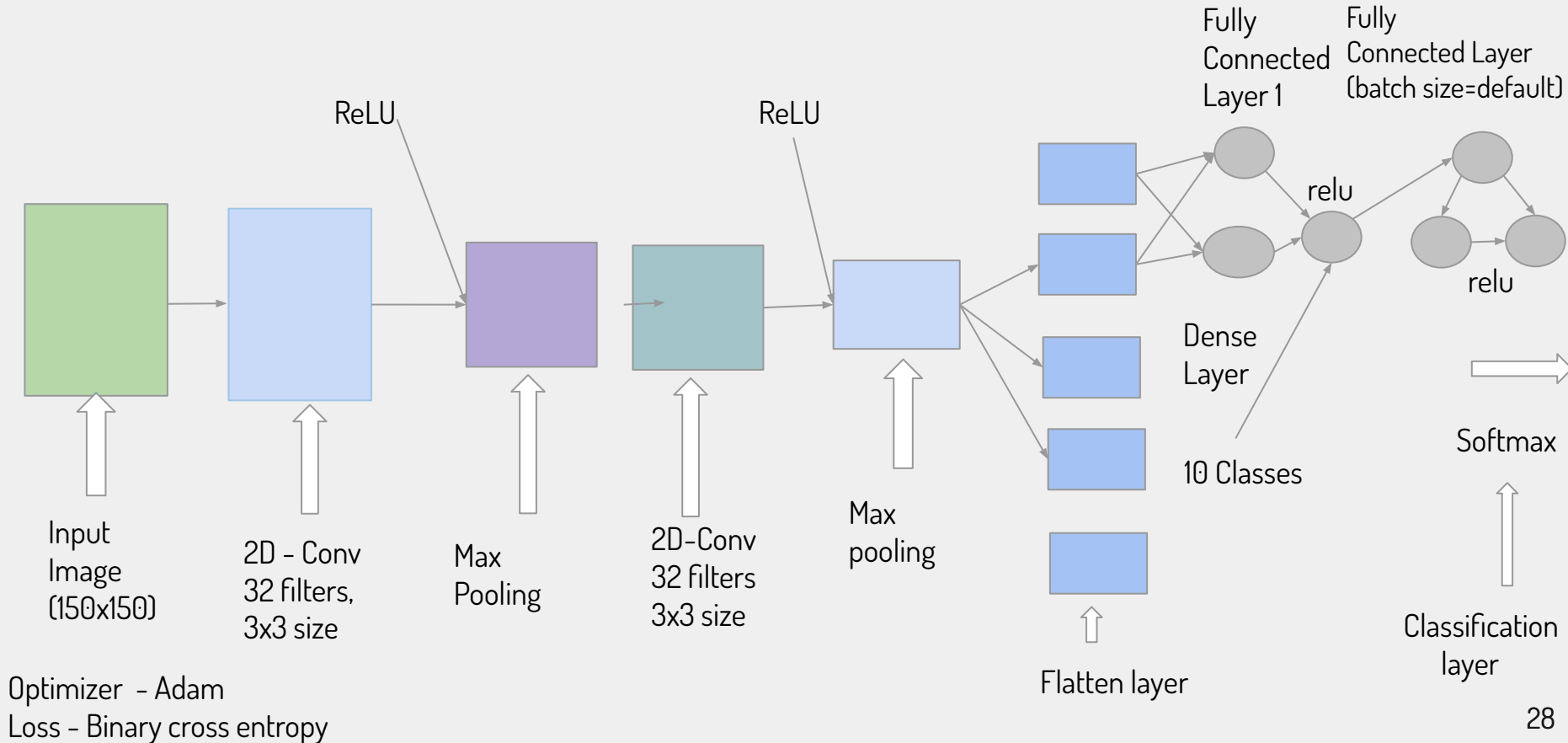
Testing accuracy - 93.54

LOSS

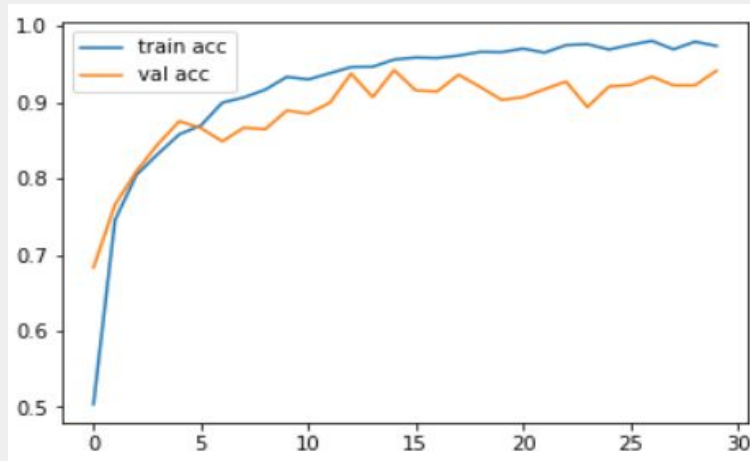
Training loss - 0.0057

Testing loss - 0.0738

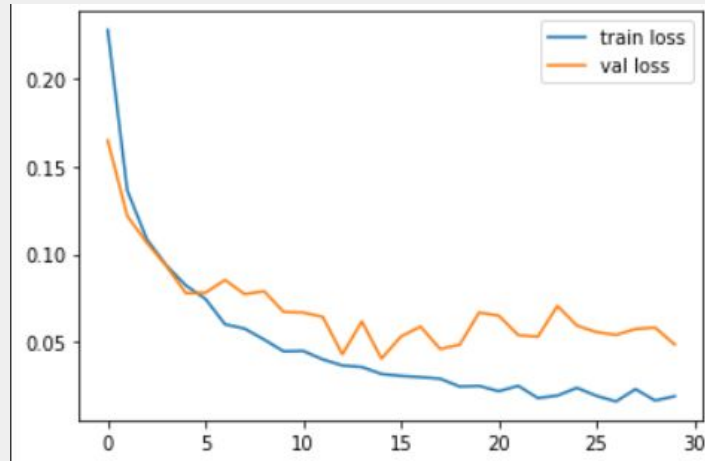
CNN ARCHITECTURE -2



CNN2 RESULTS



Training and Validation Accuracy



Training and Validation loss

CNN2 RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL TESTING IMAGES
Tomato leaf disease Detection (Total 18,345)	14,678 (80%)	4,585

ACCURACY

Training accuracy - 98.58

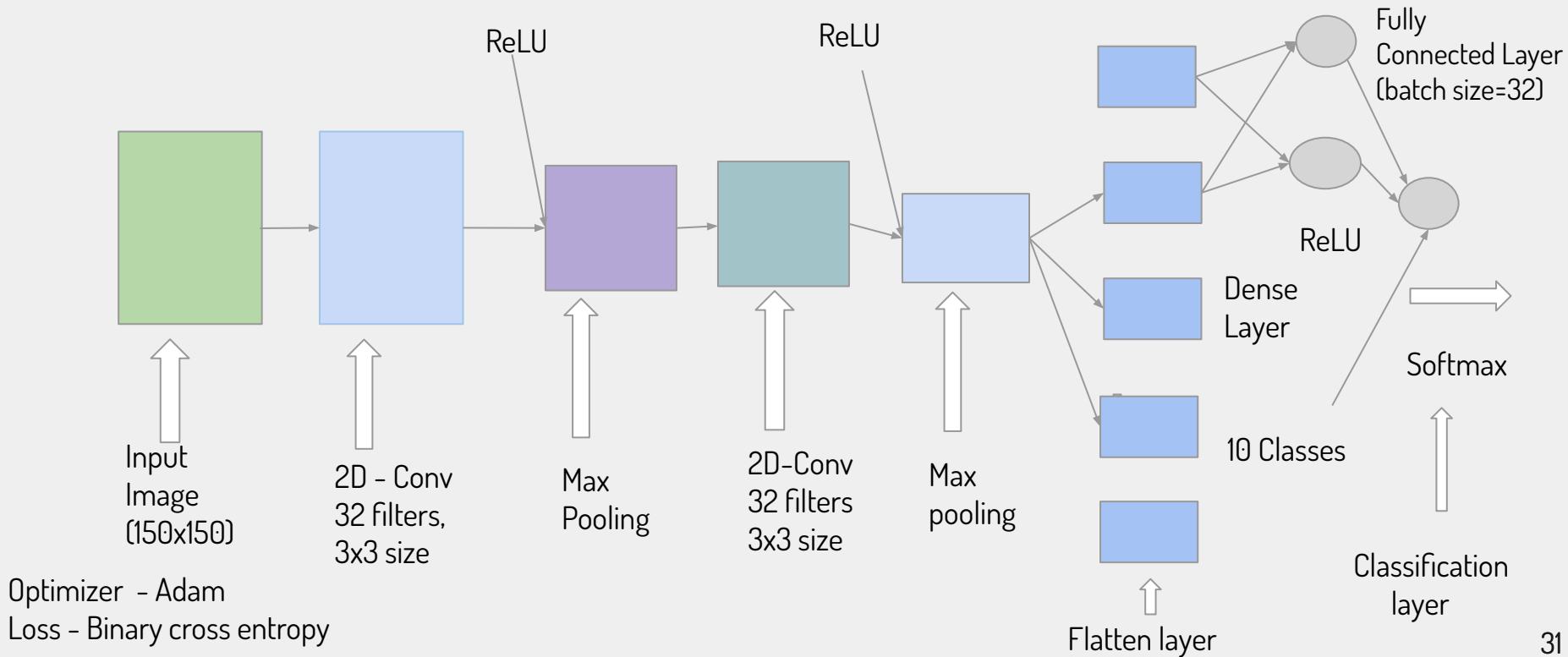
Testing accuracy - 88.80

LOSS

Training loss - 0.0126

Testing loss - 0.1029

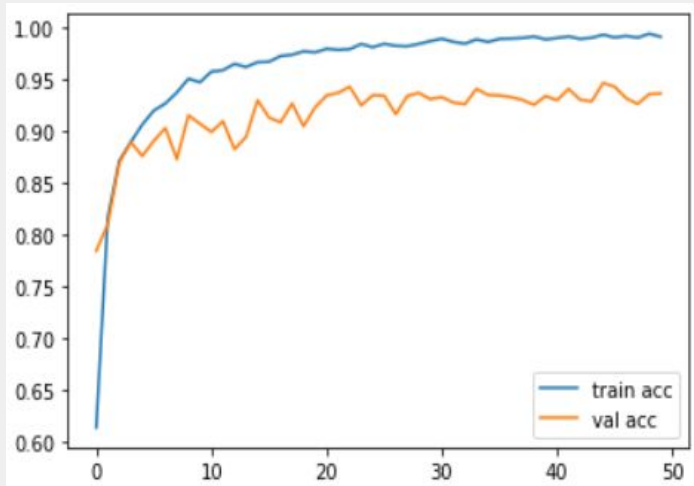
CNN ARCHITECTURE -3



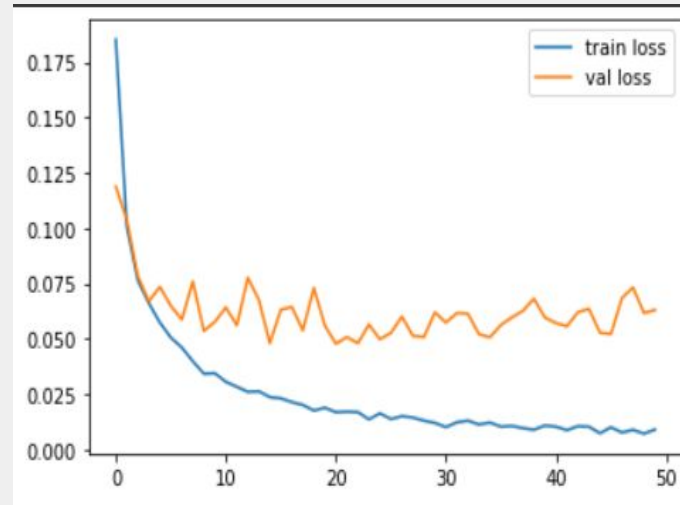
CNN ARCHITECTURE -3

- A convolutional neural network (CNN) is a type of artificial neural network used in image recognition and processing that is specifically designed to process pixel data.
- CNNs use image recognition and classification in order to detect objects, recognize faces, etc. They are made up of neurons with learnable weights and biases.
- Convolution is a mathematical operation on two functions that produces a third function that expresses how the shape of one is modified by the other. A CNN typically has three layers: a convolutional layer, a pooling layer, and a fully connected layer
- The Architecture contains the same layers as CNN1 but where fully connected layer of batch size 64 is taken (2 Conv Layers, 2 Max pooling, 1 Flatten Layer, 1 Dense Layer, 1 Fully Connected Layer with batch size 64)

CNN3 RESULTS



Training and Validation Accuracy



Training and Validation Loss

CNN3 RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL TESTING IMAGES
Tomato leaf disease Detection (Total 18,345)	14,678 (80%)	4,585

ACCURACY

Training accuracy - 99.35

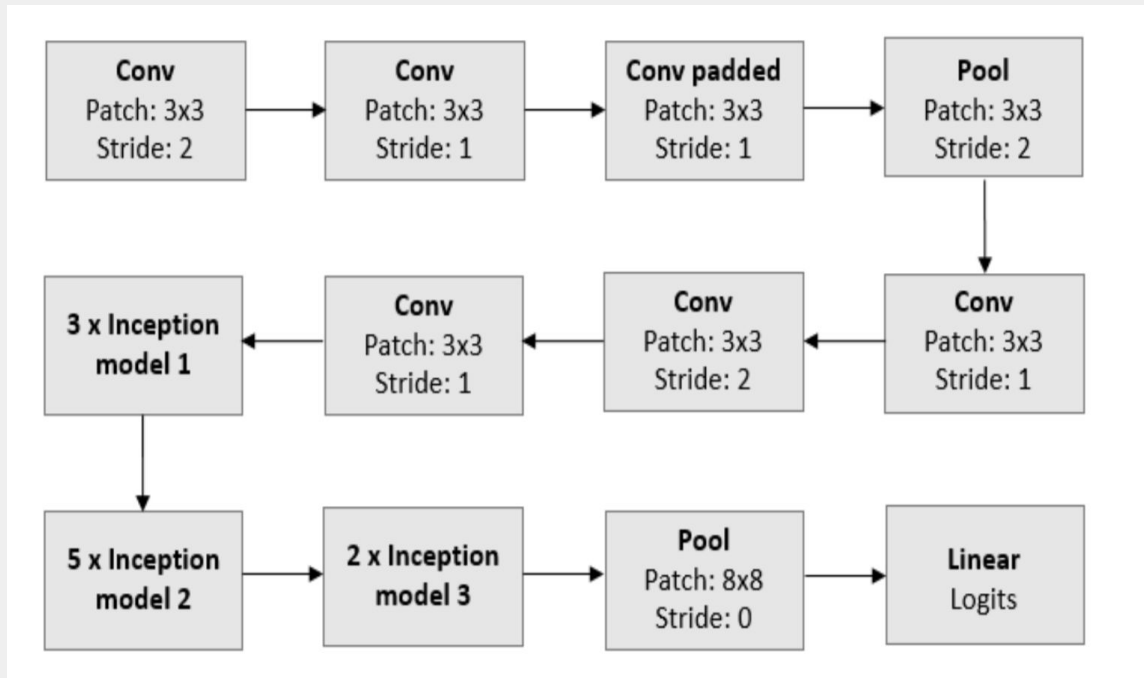
Testing accuracy - 91.77

LOSS

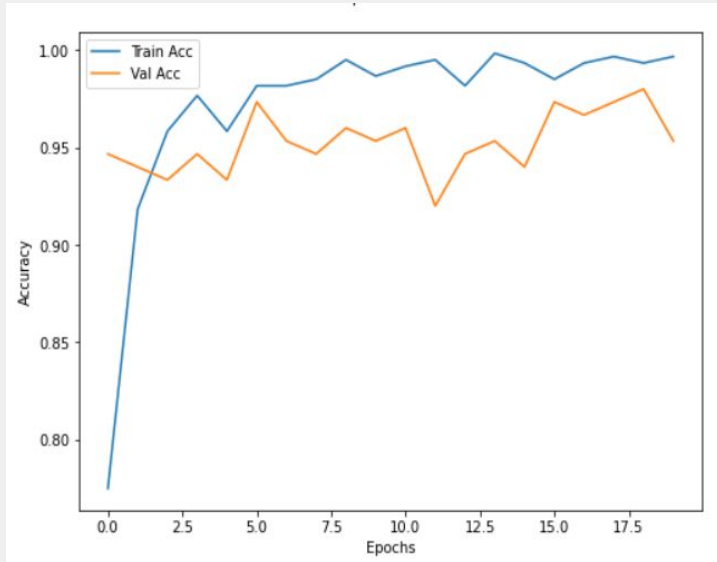
Training loss - 0.0059

Testing loss - 0.0859

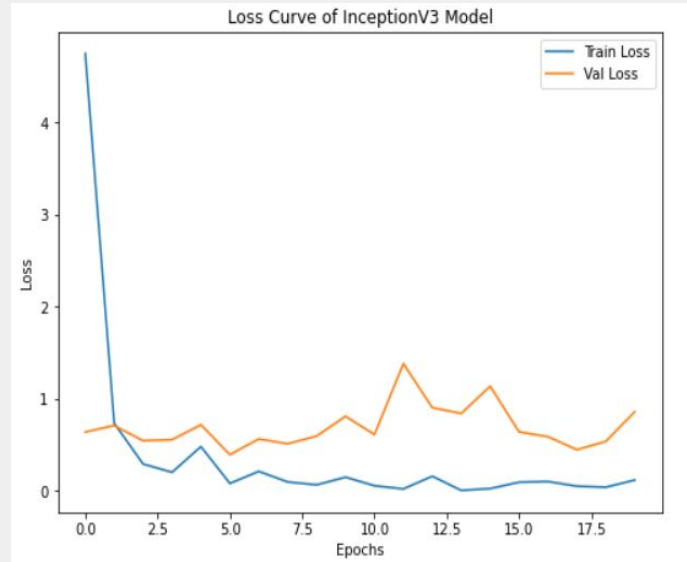
INCEPTION -V3 ARCHITECTURE



INCEPTION-V3 RESULTS



Training and Validation Accuracy



Training and Validation Loss

INCEPTION-V3 RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL TESTING IMAGES
Tomato leaf disease Detection (Total 18,345)	14,678 (80%)	4,585

ACCURACY

Training accuracy - 99.37

Testing accuracy - 93.54

LOSS

Training loss - 0.0057

Testing loss - 0.0738

VGG-16 ARCHITECTURE

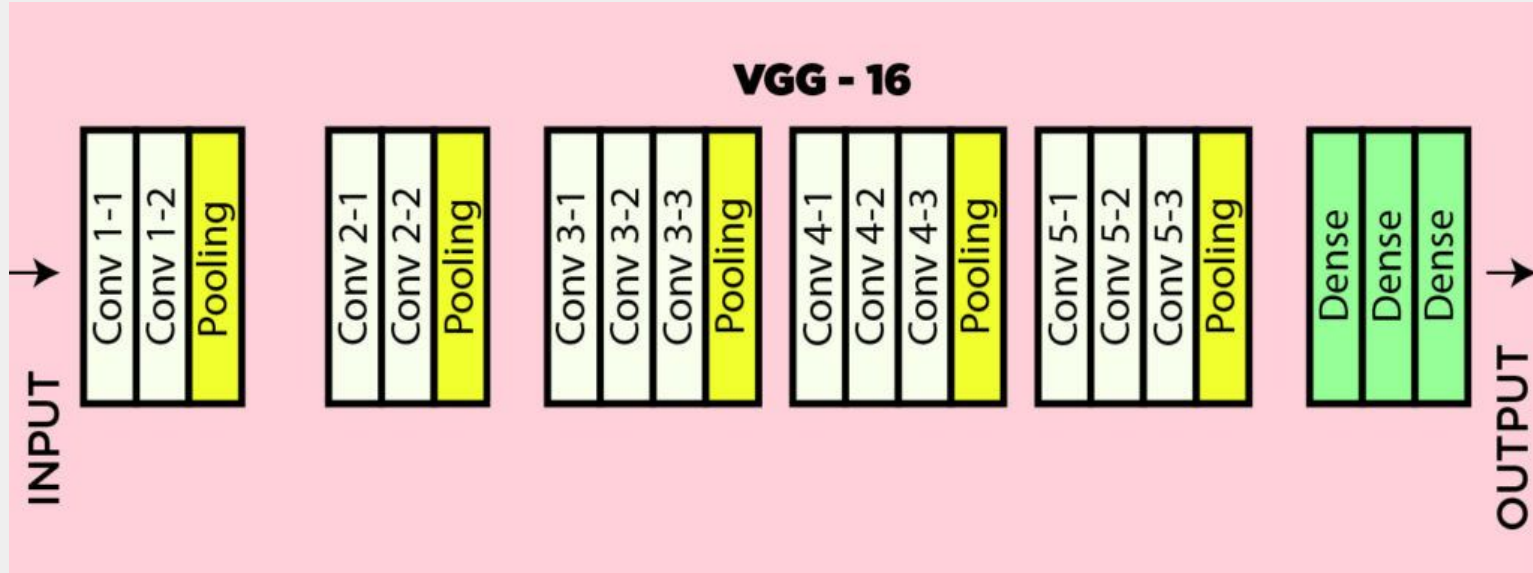
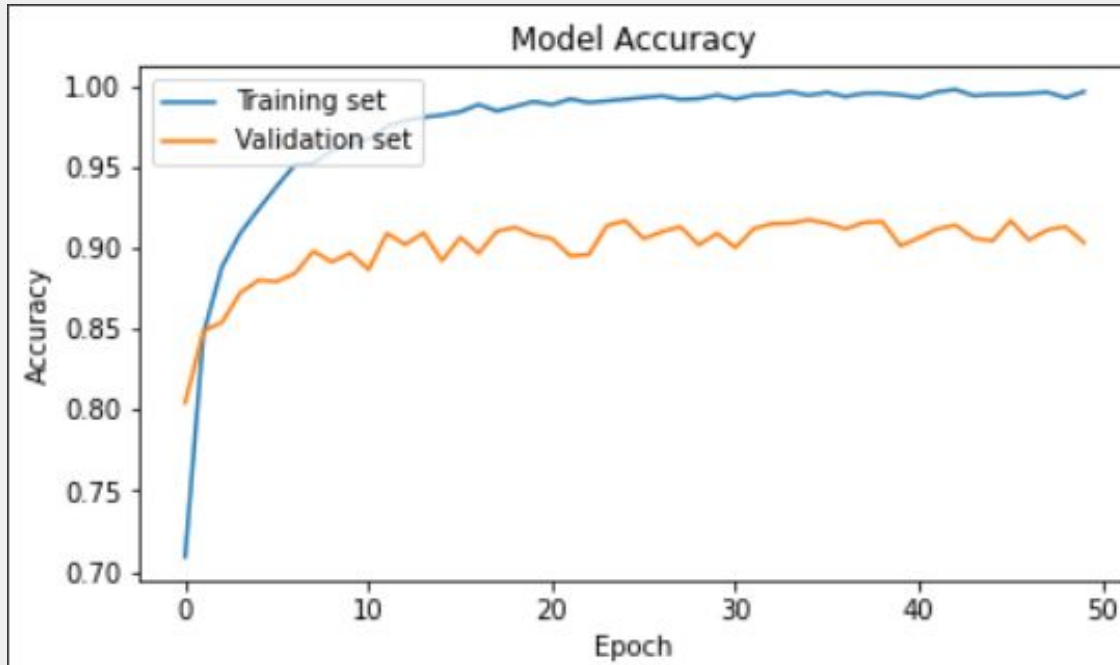
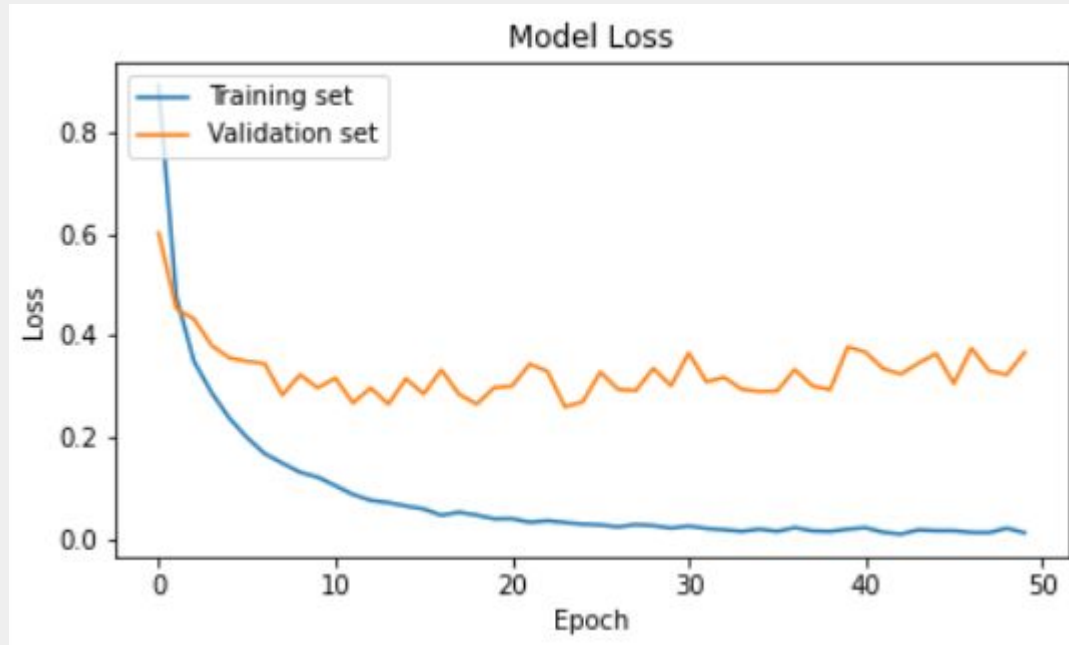


Image Credits: [9]

VGG16 RESULTS - TRAINING AND VALIDATION ACCURACY



VGG16 RESULTS - TRAINING AND VALIDATION LOSS



VGG16-RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL TESTING IMAGES
Tomato leaf disease Detection (Total 18,345)	14,678 (80%)	4,585

ACCURACY

Training accuracy - 99.37

Testing accuracy - 94.17

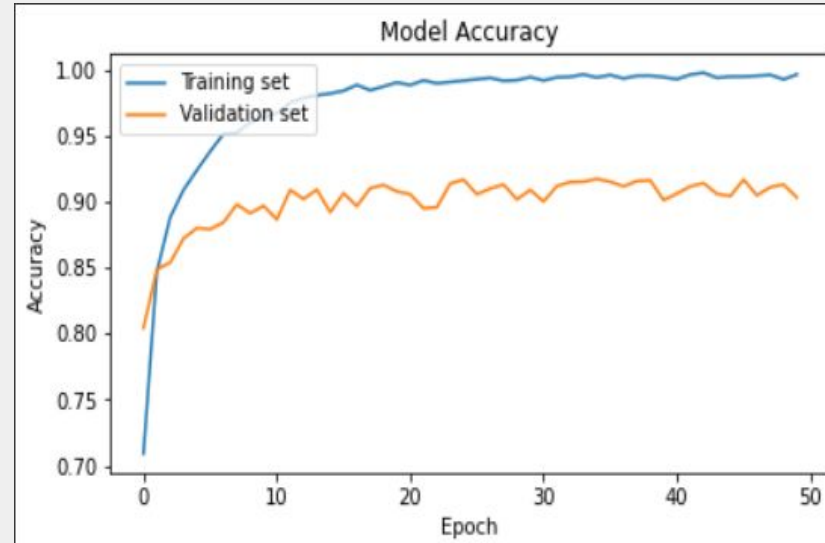
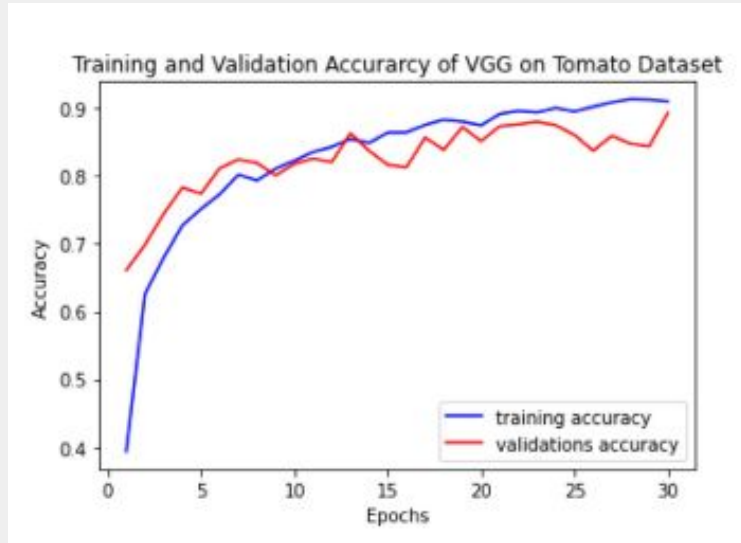
LOSS

Training loss - 0.0020

Testing loss - 0.2110

COMPARISON OF MODELS

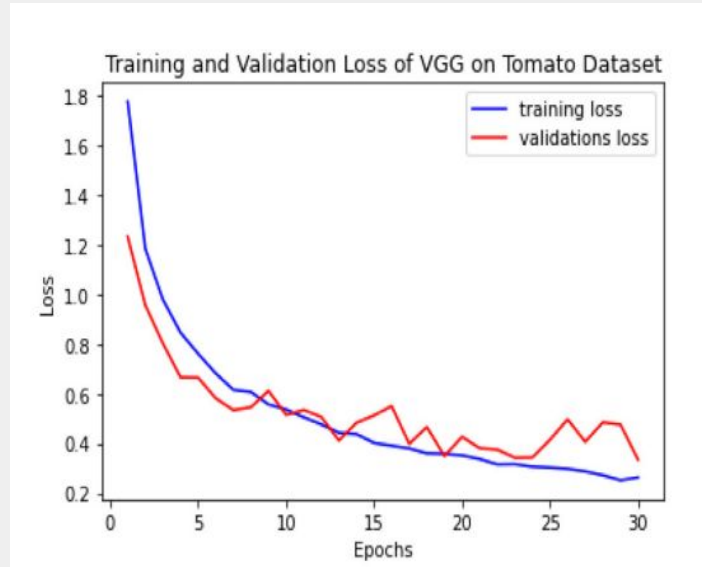
OUR VGG16 VS BASE PAPER-1 (ACCURACY)



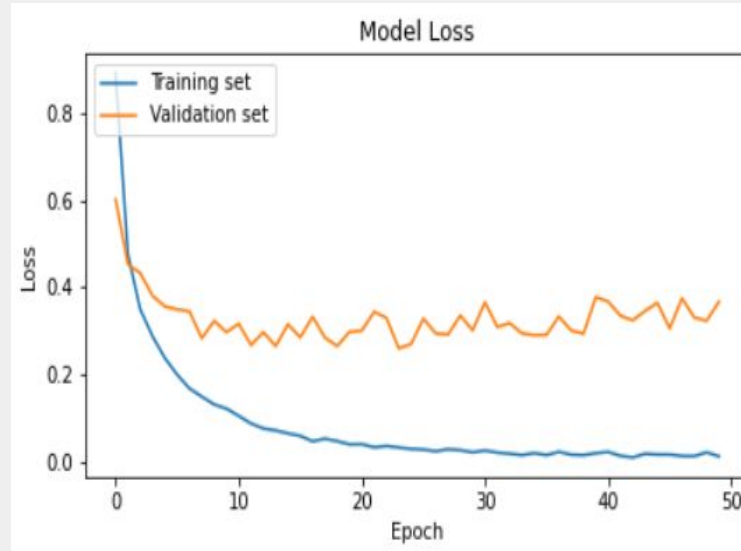
Base Paper 1 - VGG16
Testing Accuracy - 95.71%

Our VGG16
Testing Accuracy - 94.17%

OUR VGG16 VS BASE PAPER-1 (LOSS)



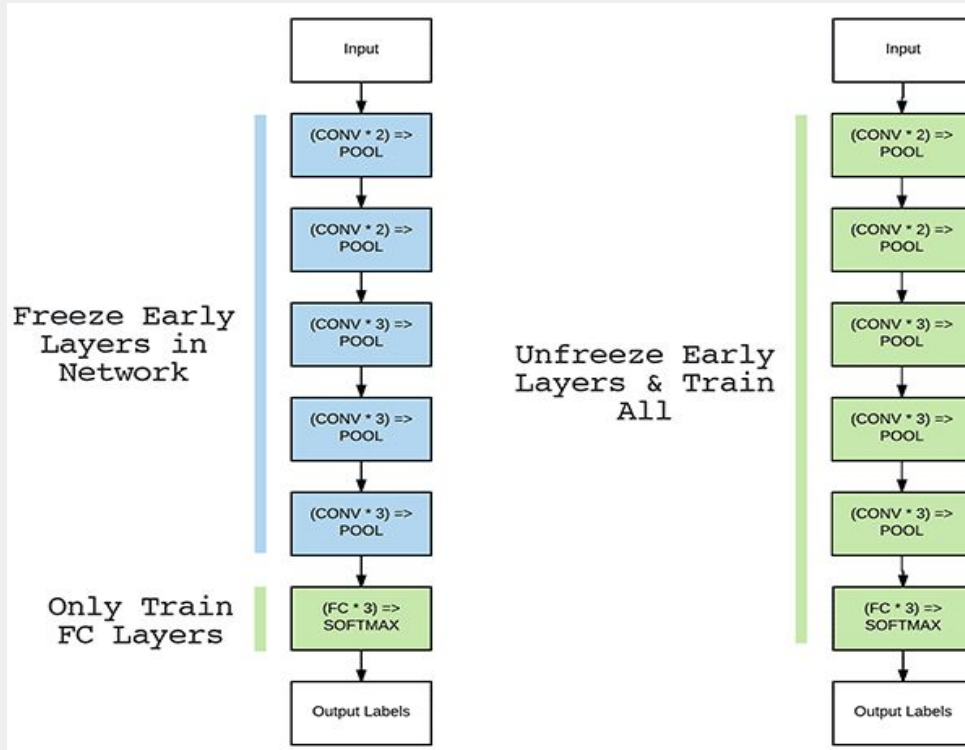
Base Paper 1 - VGG16
Testing Loss - 0.3042



Our Model - VGG16
Testing Loss - 0.2110

VGG-16 FINE TUNING

VGG16-FINE-TUNING-ARCHITECTURE



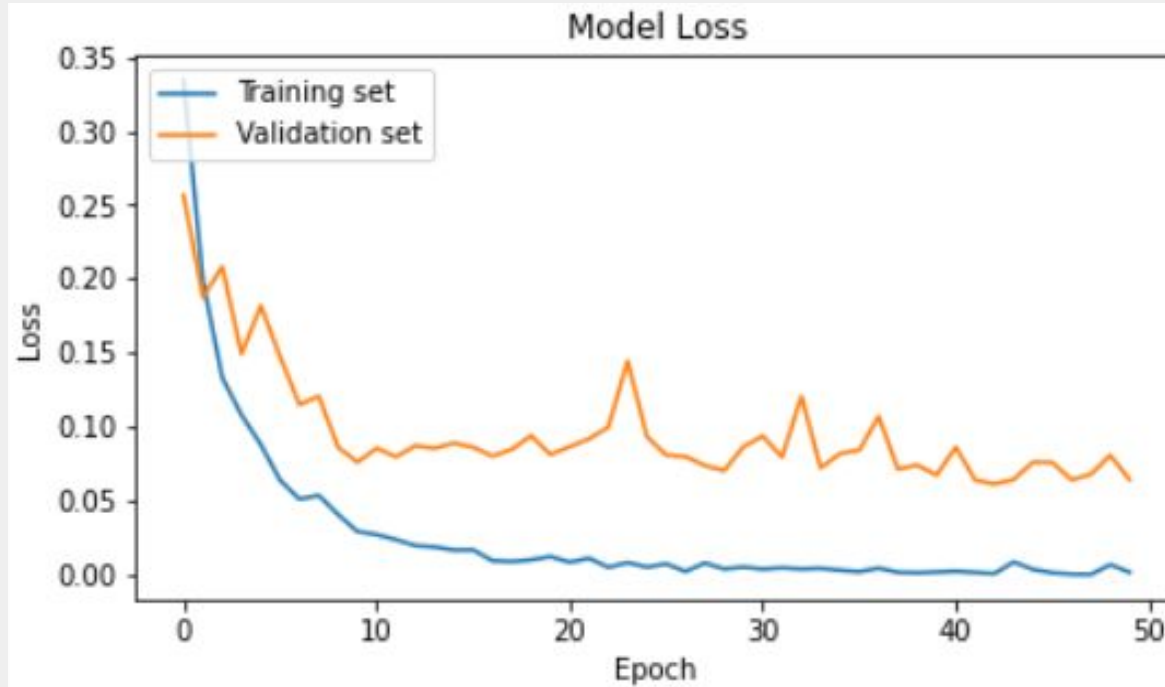
VGG16-ARCHITECTURE-FINE TUNING

- ❑ Fine-tuning is simple method. It uses already trained network and re-trains the part of that by the new data set.
- ❑ High learning rates increase the risk of losing previous knowledge so it is best to set a low learning rate.
- ❑ The neural network architecture is already trained by huge amount of data. You can add one or some layers just after the architecture and train just those part(sometimes including some layers before).

VGG16-FINE TUNING RESULTS - TRAINING AND VALIDATION ACCURACY



VGG16-FINE TUNING RESULTS - TRAINING AND VALIDATION LOSS



VGG16-FINE TUNING RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL VALIDATION IMAGES
Tomato leaf disease detection	18,345	4,585

ACCURACY

Training accuracy - 100

Testing accuracy - 98.82

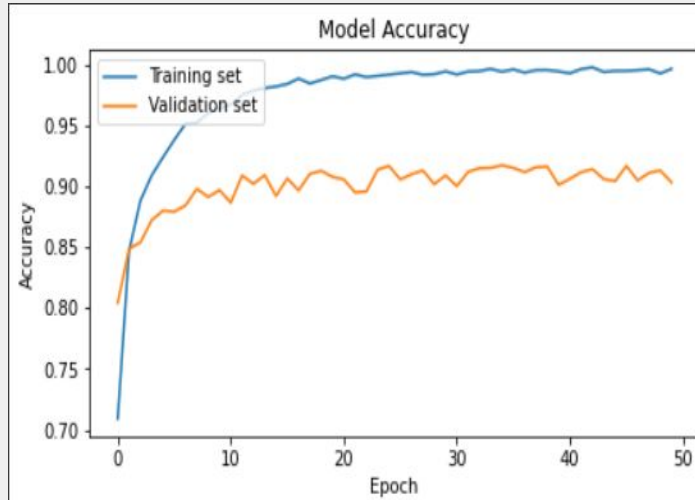
LOSS

Training loss - 0.0001

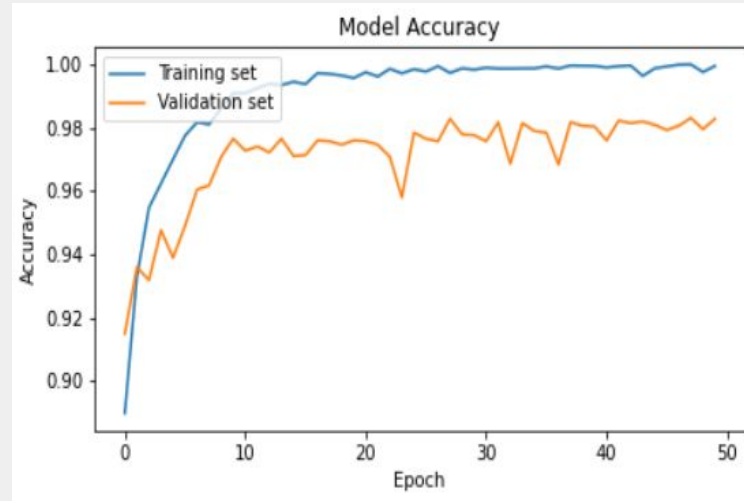
Testing loss - 0.0483

COMPARISON OF MODELS

OUR VGG16 VS VGG16-FINE TUNING (ACCURACY)

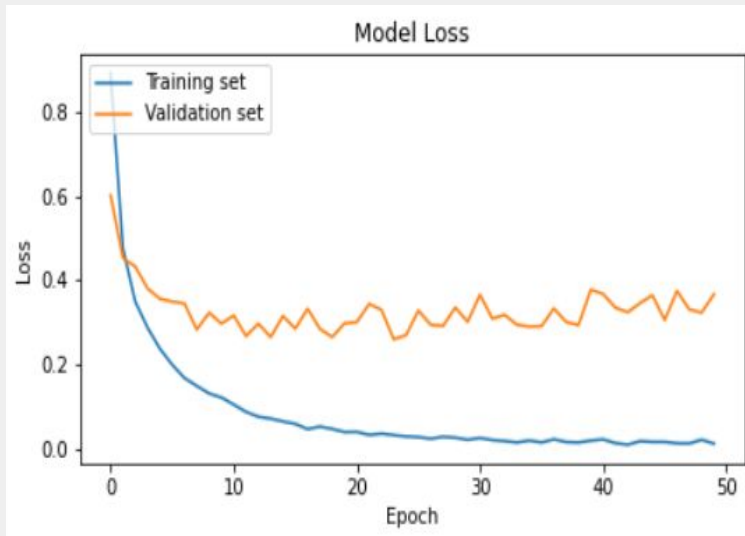


VGG16 Initial Architecture
Testing Accuracy - 94.17%

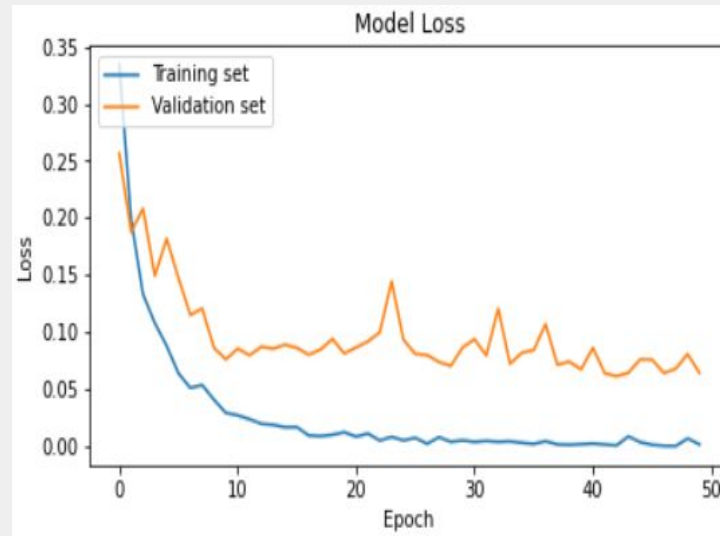


VGG16 Fine-Tuned Model
Testing Accuracy - 98.82%

OUR VGG16 VS VGG16-FINE TUNING (LOSS)



VGG16 Initial Architecture
Testing Loss - 0.2110



VGG16 Fine-Tuned Model
Testing Loss - 0.0483

VGG19-ARCHITECTURE

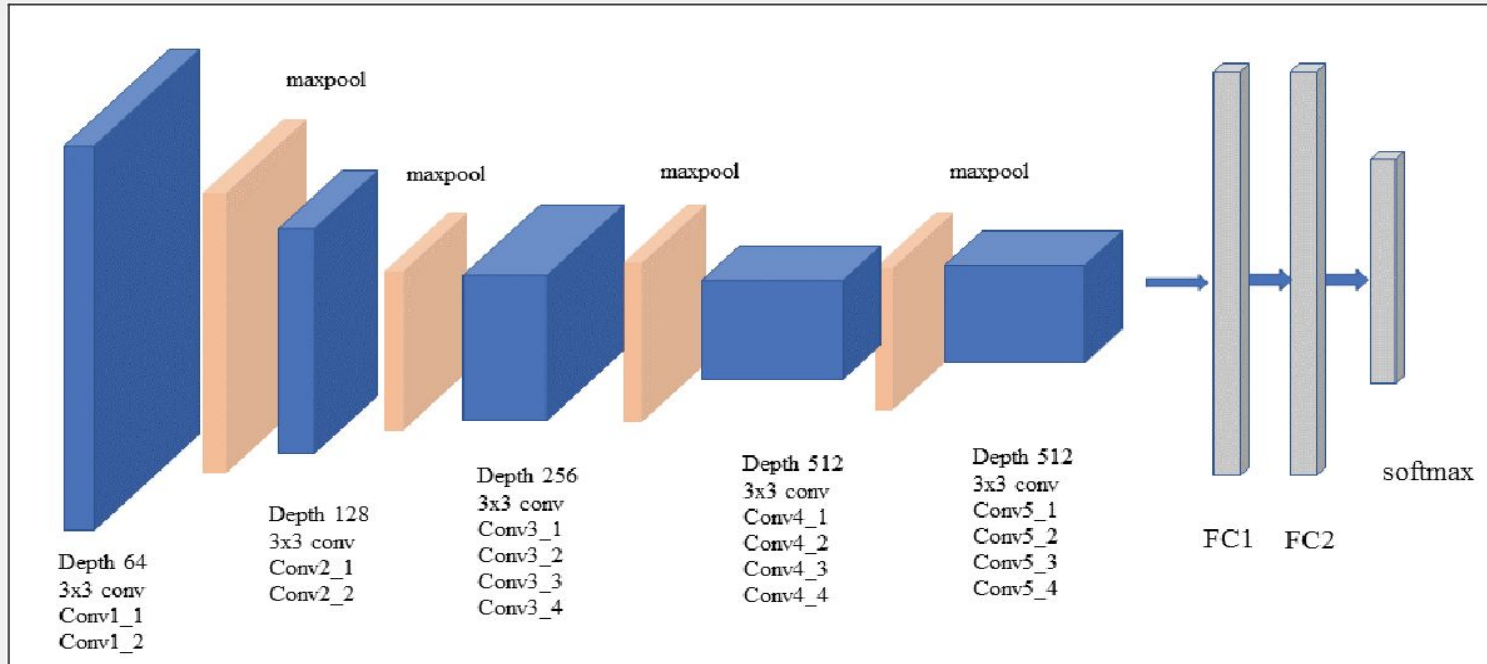
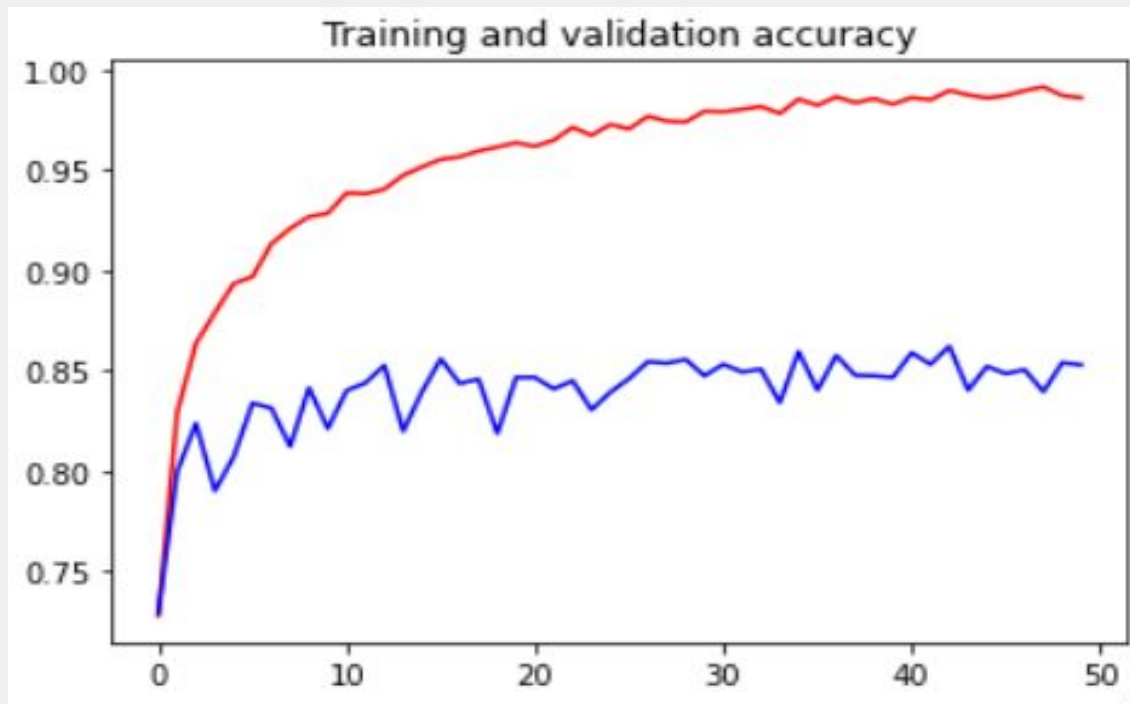
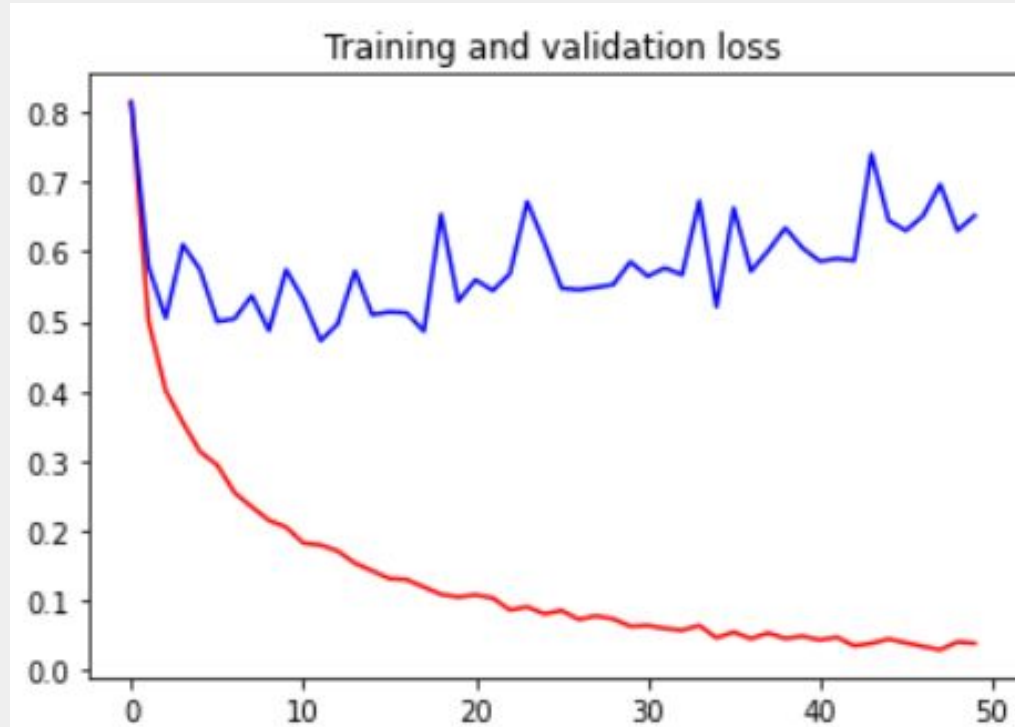


Image Credits: [10]

VGG19 - TRAINING AND VALIDATION ACCURACY



VGG19 - TRAINING AND VALIDATION LOSS



VGG19 RESULTS- ACCURACY

DATASET NAME	TOTAL TRAINING IMAGES	TOTAL VALIDATION IMAGES
Tomato leaf disease detection	18,345	4,585

ACCURACY

Training accuracy - 98.79

Testing accuracy - 88.37

LOSS

Training loss - 0.0372

Testing loss - 0.4795

ALL RESULTS-1

Model	Dataset	Network	Architecture	Results
CNN1	kaggle	CNN	2 Conv Layers, 2 Max pooling, 1 Flatten Layer,1 Dense Layer, 1 Fully Connected Layer with batch default size	Testing accuracy - 93.54%
CNN 2	kaggle	CNN	2 Conv Layers, 2 Max pooling, 1 Flatten Layer,1 Dense Layer, 1 Batch Normalization, 2 Fully Connected Layer with batch default size	Testing accuracy - 88.80%
CNN 3	kaggle	CNN	2 Conv Layers, 2 Max pooling, 1 Flatten Layer,1 Dense Layer, 1 Fully Connected Layer with batch size 64	Testing accuracy - 91.77%
INCEPTION-V3	kaggle	D-CNN	48 layered deep architecture	Testing accuracy - 93.54%
VGG-16	kaggle	D-CNN	16 layered deep architecture	Testing Accuracy - 94.17%

ALL RESULTS-2

Model	Dataset	Network	Architecture	Results
VGG16 - Fine Tuning	kaggle	D-CNN	16 layered deep architecture	Testing accuracy - 97.96%
VGG19	kaggle	D-CNN	19 layered deep architecture	Testing accuracy - 88.37%


WEB APPLICATION

This is the User Interface of the web-application, we have built using streamlit api which is an open source app framework in Python language. It helps us create web apps for data science and machine learning in a short time.



WEB APPLICATION


durgance-tomato-disease-pred-app-nu8xh2.streamlitapp.com



Tomato Disease Prediction

Please upload the Tomato leaf image :

Upload Image

 Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG


Browse files

Process

Here, we can upload the sample images from the testing dataset and click process to classify a plant leaf as diseased with the disease category or as an healthy leaf.


WEB APPLICATION

Here, we can see that the website classified the healthy leaf correctly as healthy.



 **Tomato Disease Prediction**

Please upload the Tomato leaf image :


Upload Image

 Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG

Browse files

 0a334ae6-bea3-4453-b200-85e082794d56___GH_HL Leaf 310.1_flipTB.JPG 13.9KB 

Process



Uploaded Image

Tomato healthy


WEB APPLICATION

Here, diseased leaf is classified with the disease name correctly.



 **Tomato Disease Prediction**

Please upload the Tomato leaf image :


Upload Image

 Drag and drop file here
Limit 200MB per file • PNG, JPG, JPEG

Browse files

 0e244a32-08cd-471c-90dd-141c354c20b3___GHLB2_Leaf_8634_flipLR.JPG 8.5KB 

Process



Uploaded Image

Tomato Late blight

AVAILABLE DATASETS

- **Tomato Plant Dataset (Used) : [Link](#)**

There are 10000 images belonging to 10 classes for training and 1000 images belonging to 10 classes for testing.

- **Augmented Tomato Plant Dataset (Used) : [Link](#)**

There are 18345 images belonging to 10 classes for training and found 4585 images belonging to 10 classes for testing.

- **Wheat Plant Dataset : [Link](#)**

There are 876 images to train the model on and 610 images in the test set for evaluation

ADDITIONAL DATASETS

For Multi-class:

- **Plant Village Dataset: [Link](#)**

This includes 152 crop solutions, 38 crop classes, and 19 crop categories, for 54,303 crop leaves images. In the dataset, high quality JPEG image format with 5471 width and 3648 height pixels are available.

- **Grape Plant Dataset : [Link](#)**

The dataset provide a collection of images of grapevine leaves, related to two classes: unhealthy leaves acquired from plants affected by Esca disease and healthy leaves.

- **Multiple Crops, Disease wise Dataset : [Link](#)**

This dataset includes various types of crops each one with different no of disease classes.

FUTURE WORK

- Performing the multi-crop classification which would be extended to different crops like Potato and Corn which would be scraped out from the PlantVillage Dataset.
- Creating a new model or fine-tuning the existing models such that it works efficiently in multi-crop classification.
- Finally, updating the existing web application for multiple crops and deploying it.

ROAD MAP (TENTATIVE)

Phase 1. Data Gathering - Done

Phase 2. Data Preprocessing - Done

Phase 3. Building our Initial Model -Done

Phase 4. Training and Detection - Done

Phase 5. Testing (2 weeks) - Done

Phase 6. Optimising (2 weeks) - Done

Phase 7. Building an web application (2 weeks)
-Done

Phase 8. Multi-class Classification (4 weeks) -
Yet to complete



REFERENCES

- [1] Plant Disease and Plant Pathology | Britannica. [Link](#)
- [2] Ministry of Agriculture & Farmers Welfare | PIB Delhi. [Link](#)
- [3] Pantazi, X. E., Moshou, D., & Tamouridou, A. A. (2019). Automated leaf disease detection in different crop species through image features analysis and One Class Classifiers. Computers and electronics in agriculture, 156, 96-104. [Link](#)
- [4] Paymode, A. S., & Malode, V. B. (2022). Transfer learning for multi-crop leaf disease image classification using convolutional neural networks VGG. Artificial Intelligence in Agriculture. [Link](#)

REFERENCES

[5] Zhang, Y., Song, C., & Zhang, D. (2020). Deep learning-based object detection improvement for tomato disease. IEEE Access, 8, 56607–56614.

[Link](#)

[6] Kulkarni, O. (2018, August). Crop disease detection using deep learning. In 2018 Fourth international conference on computing communication control and automation (ICCUBE) (pp. 1–4). IEEE.

[Link](#)

[7] Li, L., Zhang, S., & Wang, B. (2021). Plant disease detection and classification by deep learning—a review. IEEE Access, 9, 56683–56698.

[Link](#)

REFERENCES

[8] Nguyen, L. D., Lin, D., Lin, Z., & Cao, J. (2018, May). Deep CNNs for microscopic image classification by exploiting transfer learning and feature concatenation. In 2018 IEEE. International Symposium on Circuits and Systems (ISCAS) (pp. 1-5). IEEE. [Link](#)

[9] VGG16 | CNN Model. [Link](#)

[10] Extract Features, Visualize Filters and Feature Maps in VGG16 and VGG19 CNN Models [Link](#)

GROUP MEMBERS

Anirudh Jakhotia (S20190010007)

Khushi Pathak (S20190010091)

V.Naveen kumar (S20190010192)





THANKS!



Do you have any questions?