Image-based Plant Diseases Detection using Deep Learning

1. **Introduction**

Image based plant disease identification is the recently explored

area by many researchers. As the crop waste is increasing due to

diseases, it is becoming crucial to identify the diseases accurately

and timely. In developing countries, especially in South Asia most

of the population is dependent on agriculture directly or indirectly,

in such countries it is becoming important to use the application

based plant disease identification which can help farmers to know

the cause of diseases and get the precaution to treat them. Initial

identification of the plant diseases on the basis of size of leaf, color

of leaf, and growth of the pattern etc can be helpful to the farmers.

With the boom in the usage of smart phones all over the world it is

easy to get the picture image of the leaves, also many people

around the globe have access to basic internet facility available

to them. More than 300 million people access the internet for them

convenience and use various applications. Governments have

come across different facilities like 24\*7 helpline numbers

dedicated to farmers in order to solve their query but people residing in rural areas find it difficult to get the proper facilities and therefore struggle to get the solution to their problems. A basic

application where farmers can simply work on self-paced image

based disease identification would be helpful.

1. **Data Collection**

We are using Plant Village dataset that is an open

access dataset available at Kaggle. Plant Village dataset has

about 20,000 images of fully healthy and unhealthy crops, having

a range of 15 class already labelled. Every input image is kept into appropriate class by domain expert i.e. botanists: early healthy stage, middle healthy stage, healthy stage or in fully healthy stage. In fully

healthy stage there are no spots in the leaves. The early stage

healthy leaves have circular shape small spots with radius about

2.5 mm. The middle stage healthy leaves having more spots growing to random or shallow shape. The end stage healthy leaves

infected by the tree in greater amount and cannot manage it to

remain in the tree. All the input images are studied by the domain

experts and classified with labelling into an appropriate disease.

180 images that were difficult to classify by experts were deleted

from further consideration. In figure the example of every stages

have been shown. At the end, we have 1650 input images of fully

healthy leaves, 130 early stage healthy, 175 middle stage healthy,

and 130 end stage disease healthy images.

As we have a greater number of healthy leaves than the diseased

leaves, we have much variation in the number of samples for each

class. To reduce the potential bias in our network in order to have

more number of healthy stage class, Number of samples per class

should be balanced. To make a good balance we have following

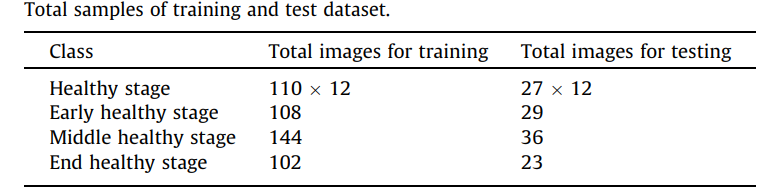
strategy: In early stage healthy, middle stage healthy, and end

stage healthy, we have 80–20 train test split strategy that means

about 80% of the input images are kept for the training set and

the reimagining’s are the kept for test set. To healthy stage leaves,

the images are classified into 15 classes.



1. **Data Preprocessing**

The samples of Plant Village datasets are different sized Red Green Blue images. By using basic deep learning, We need only 4 images in the first stage that is preprocessing stage This is done in different stages as follows: Initially, resizing of input image to 256 x 256 pixels for naive networks, 224 x 224 with all models we have used so far, in the Inception-V3 299 \* 299 pixels.

In the process of rescaling the images, suitable prediction and optimization have been applied. Then, division of 255 is performed to

each pixel values that is computable to starting value of our network. Then, normalization performed to each of the samples. That

actually increases the performance of the training notably. The

process of normalization follows followings steps: for all training

data, After calculating the mean and S.D., conversion of the training

data to X’ = (X–Mx)/Sx, have performed so that each features nearly

look a normalized data distributed to 0 mean and 1 variance. At the

end, some operations like rotation with varying angles, shearing,

zooming, and flipping are used to the given inputs. This ensures

over fitting and builds the model suitable to use.

1. **Feature Extraction**

The basic model of CNN (convolution neural

network) consists of convolution layers, pooling layers, and at the

end fully connected layers.In convolution layer for each input image, Here convolution operation have been denoted by ⁄ and Wi stands for the kernels of the convolution layer. Wi = [Wi1, Wi2, Wi3, . . .. . ..,Wik] and K denotes the total number of convolution kernels.M \* M\* N is a weight matrix with window size as M and number of input channels as N for each kernel.As we have non linear saturation in our image best selection of activation function should be ReLu as it is much time faster than

the other existing activation functions. ReLU is a rectified linear

activation function So in our model, We will be using ReLU.

As in the max pooling layer we have to calculate the largest value of the outputs of all convolution layers that do not intersect to

each other. This technique sets varying rank on the existing surroundings and optimizes the size of our output.

The role of fully connected layer is to keep final layer on the top.

Then all such c layers which are on the top compute the ReLU(WfcX), where X denotes the image to be given and fc denotes the

weight matrix of the final layer.

The loss function calculates the variation of the result with the

input, that is given by the domain expert, which is defined as the

aggregation of cross entropy.

If our networks starts training with very high speed and gives

very high accuracy in the beginning itself then we have to stop it

as this is over fitting problem. For this purpose we use early stopping technique. In this after every epoch performance is evaluated

with existing train data when loss of the test set data does not

improve itself Early stopping just stop the training of the network.

This way the over fitting is solved; we are conducting the transfer

learning in following manner:

All final layers are swapped by the help of forming one more

layer and keeping in mind that only well formed outer convolution

layer of VGG19 with VGG16 with the go for finally connected layer

. Delete the weights trained already to ensure the new gradient,

now the existing network must start with appropriate values

instead of setting any parameter. Now except the recent network,

put all layers in the network. At the end training of fully connected

layer is done by help of features produced in the final convolution

layer. After the parameter tuning weight learning or adjustment is

done. Then final convolution layer for all architectures used in our

network are resumed with parallel training and slightly lower

learning rate.

1. **Our Model**

While the Transformer architecture has become the de-facto standard for natural language processing tasks, its applications to computer vision remain limited. In vision, attention is either applied in conjunction with convolutional networks, or used to replace certain components of convolutional networks while keeping their overall structure in place. We show that this reliance on CNNs is not necessary and a pure transformer applied directly to sequences of image patches can perform very well on image classification tasks. When pre-trained on large amounts of data and transferred to multiple mid-sized or small image recognition benchmarks (ImageNet, CIFAR-100, VTAB, etc.), Vision Transformer (ViT) attains excellent results compared to state-of-the-art convolutional networks while requiring substantially fewer computational resources to train.

To feed images to the Transformer encoder, each image is split into a sequence of fixed-size non-overlapping patches, which are then linearly embedded. A [CLS] token is added to serve as representation of an entire image, which can be used for classification. The authors also add absolute position embeddings, and feed the resulting sequence of vectors to a standard Transformer encoder.

As the Vision Transformer expects each image to be of the same size (resolution), one can use Image Processor to resize (or rescale) and normalize images for the model.

Both the patch resolution and image resolution used during pre-training or fine-tuning are reflected in the name of each checkpoint. with patch resolution of 16x16 and fine-tuning resolution of 224x224

1. **Final Prediction**

The final accuracy of existing validation set

varies from 84.0% to 93.5%. The accuracy achieved after parameter

tuning is better than the existing model which is already trained.

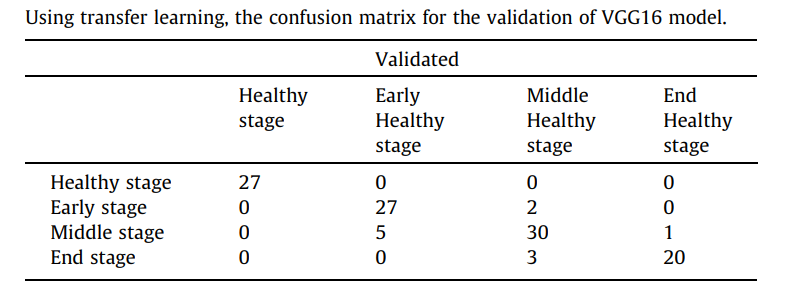
The most relevant outcome is celebrated in the VGG16 model, having accuracy 93.5%. Our result conclude that even we have not sufficient data, the transfer learning can give better result.

In a comparative study, our artificial neural network model uses

Stochastic gradient descent optimizer in the existing training data

for all layers. Using random guessing 37 % accuracy reached in validation. As if we do not feature extractor in the convolution, the

ANN is not able to locate correlations locally and also unable discriminate the features by utilizing available

****