Plant Disease Detection Using Deep Learning

Hariharan K[1], Kamai D[2], Nandhini S[3], Dr. S. Saravanan[4]
[1,2,3,4]*Computer Science and Engineering, Agni College of Technology*

***Abstract* - Humans need to increase food production by an estimated 70% by 2050. Currently, infectious diseases reduce the potential yield by an average of 40% with many farmers in the developing world experiencing yield losses as high as 100%. Identification of the plant diseases is the key to preventing the losses in the yield and quantity of the agricultural product. Fast and accurate plant disease detection is critical to increasing agricultural productivity in a sustainable way. It is very difficult to monitor the plant diseases manually. The widespread distribution of smartphones among crop growers around the world offers the potential of turning the smartphone into a valuable tool for diverse communities growing food. One potential application is the development of mobile disease diagnostics through machine learning and crowdsourcing.**

In this proposed system, we provide a plant disease recognition using image processing and machine learning techniques. In particular, we concentrate on the use of RGB images owing to the low cost and high availability of smartphone cameras. We are planning to focus on the use of deep. Disease detection involves the steps like image acquisition, image pre-processing, image segmentation, feature extraction and classification. We will be using Transfer learning technique using MobileNet pretrained on ImageNet.

I. INTRODUCTION

It is very important to get an accurate diagnosis of plant diseases for global health and wellbeing. In this ever-changing environment, identifying the disease including early prevention is important to avoid problems that we might face otherwise. Some of these problems could have devastating impacts on humanity including global shortage of food. It is crucial to prevent unnecessary waste of financial resources achieving a healthier lifestyle, by addressing climate change from an ecological perspective. It is difficult for the naked eye of a human being to catch all sorts of problems with plant diseases. Also doing this time and time again is also laborious and unproductive work. In order to achieve accurate plant disease detection, a plant pathologist should possess good observation skills so that one can identify characteristic symptoms. An automated system designed to help identify plant diseases by the plant's appearance and visual symptoms could be of great help. This can be deployed in agricultural fields so that the whole pipeline can be automated. This would not only lead to better efficiency as machines could perform better than humans in these redundant tasks but also improve the productivity of the farm. Our work builds on the above-mentioned problem of automating plant disease classification using deep learning and computer vision techniques.

We shall be using MobileNet as it is lightweight in its architecture. It uses depth wise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it. This has the effect of filtering the input channels. Or as the authors of the paper explain clearly: "For MobileNet the depth wise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs the depth wise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depth wise separable convolution splits this into two layers, a separate layer for filtering and a separate layer for combining. This factorization has the effect of drastically reducing computation and model size."

II.RELATED WORK

In this Section we have studied few papers which shows how the plant disease can be identified using deep learning algorithm,

a. Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand and Marco Andreetto proposed "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications" where they present a class of efficient models called

MobileNets for mobile and embedded vision applications.

b. Michael T. Rosenstein, Zvika Marx and Leslie Pack Kaelbling proposed “To Transfer or Not To Transfer” where with transfer learning, one set of tasks is used to bias learning and improve performance on another task.

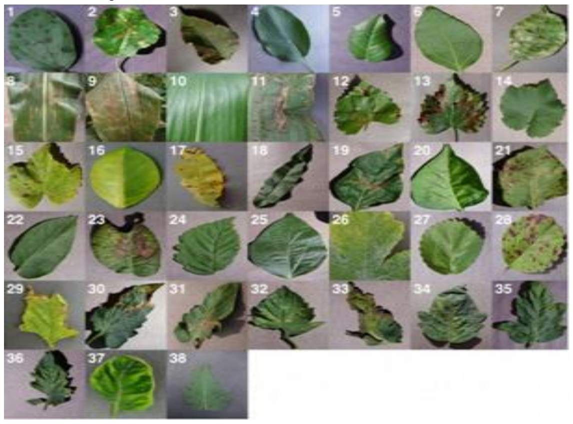
c. Gao Huang Zhuang Liu, Laurens van der Maaten and Kilian Q. Weinberger proposed “Densely Connected Convolutional Networks” where d introduce the Dense Convolutional Network (DenseNet), which connects each layer to every other layer in a feed-forward fashion

d. Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov proposed “Dropout: A Simple

Way to Prevent Neural Networks from Overfitting” where they introduced a technique to randomly drop units (along with their connections) from the neural network during training.

III.DATASET

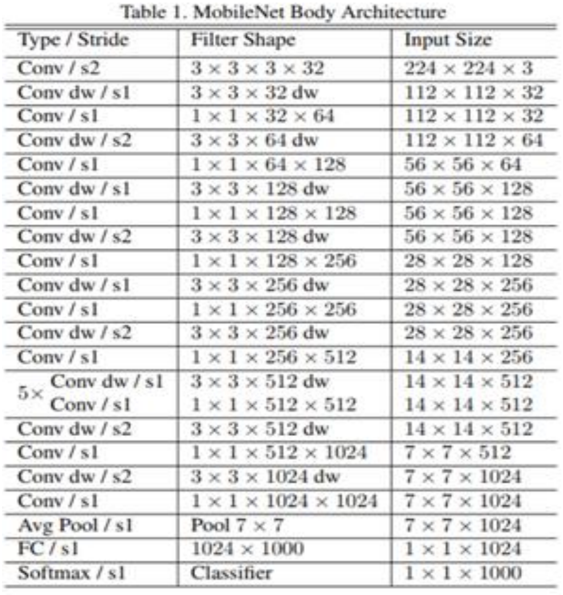
The dataset we have used for the model consists of 70295 images consisting 38 different categories of both healthy and diseased images. We have trained our model to detect diseases of the following plants:
Apple, Blueberry, Cherry, Corn, Grape, Orange, Peach, Pepper, Potato, Raspberry, Soybean, Squash, Strawberry, Tomato



IV. PROPOSED SYSTEM

We shall be using Mobilenet as it is lightweight in its architecture. It uses depth wise separable convolutions which basically means it performs a single convolution on each colour channel rather than combining all three and flattening it. This has the effect of filtering the input channels.

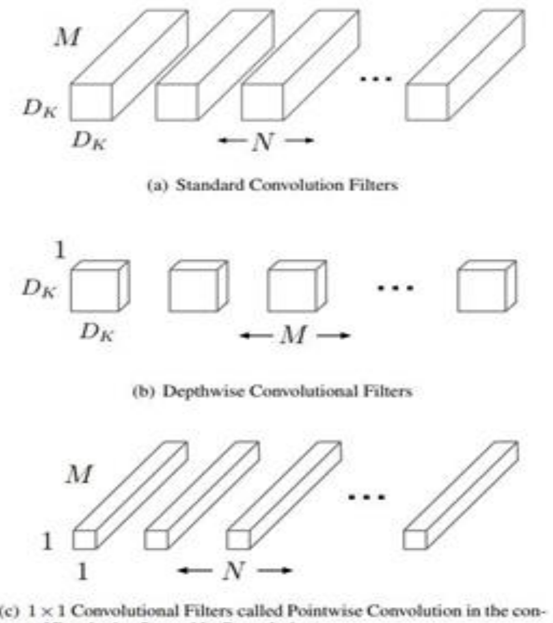
Table 1. MobileNet Body Architecture



Type / Stride	Filter Shape	Input Size
Conv / s2	3 × 3 × 3 × 32	224 × 224 × 3
Conv dw / s1	3 × 3 × 32 dw	112 × 112 × 32
Conv / s1	1 × 1 × 32 × 64	112 × 112 × 32
Conv dw / s2	3 × 3 × 64 dw	112 × 112 × 64
Conv / s1	1 × 1 × 64 × 128	56 × 56 × 64
Conv dw / s1	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 128	56 × 56 × 128
Conv dw / s2	3 × 3 × 128 dw	56 × 56 × 128
Conv / s1	1 × 1 × 128 × 256	28 × 28 × 128
Conv dw / s1	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 256	28 × 28 × 256
Conv dw / s2	3 × 3 × 256 dw	28 × 28 × 256
Conv / s1	1 × 1 × 256 × 512	14 × 14 × 256
5× Conv dw / s1	3 × 3 × 512 dw	14 × 14 × 512
5× Conv / s1	1 × 1 × 512 × 512	14 × 14 × 512
Conv dw / s2	3 × 3 × 512 dw	14 × 14 × 512
Conv / s1	1 × 1 × 512 × 1024	7 × 7 × 512
Conv dw / s2	3 × 3 × 1024 dw	7 × 7 × 1024
Conv / s1	1 × 1 × 1024 × 1024	7 × 7 × 1024
Avg Pool / s1	Pool 7 × 7	7 × 7 × 1024
FC / s1	1024 × 1000	1 × 1 × 1024
Softmax / s1	Classifier	1 × 1 × 1000

So, the overall architecture of the Mobilenet is as follows, having 30 layers with

A. convolutional layer with stride 2
B. depth wise layer
Selection: Highlight all author and affiliation lines.


M
D_K
D_K
$\leftarrow N \rightarrow$
1
D_K
D_K
$\leftarrow M \rightarrow$
M
1
1
$\leftarrow N \rightarrow$

(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

It is also very low maintenance thus performing quite well with high speed. There are also many flavours of pre-trained models with the size of the network in memory and on disk being proportional to the number of parameters being used.

V.MODULES

A. Image Preprocessing

The Images are resized by the factor of 1/255 with a shear range of 0.2, zoom range of 0.2 along with width shift range and height shift range of 0.2. For the filling mode we have used nearest method.

The images are generated into two sets train data and test data. The batch size of the datasets is 32. The size of image is rescaled to 255 x 255.

B. Base Model

The base model is the MobileNet Architecture where we freeze our weights with the value of weights from the ImageNet challenge. We do not include the top layer and set the input size to 225 x 225.

We set the MobileNet model as non-trainable i.e., we do not change the values of it.

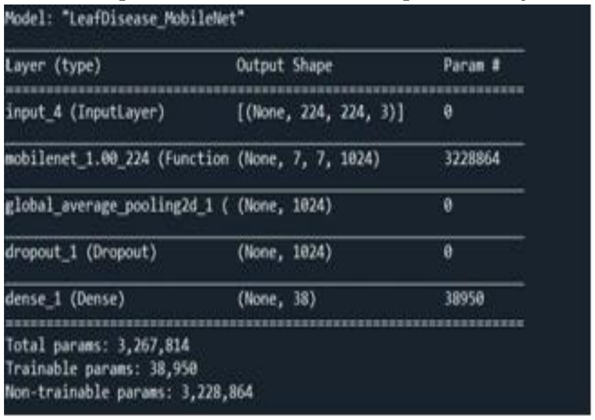
C. Transfer Learning

We manipulate the MobileNet architecture and retrain the top few layers and employ transfer learning. We add few new layers on top of the MobileNet model.

We add global average pooling 2d layer, it applies average pooling on the spatial dimensions until each spatial dimension is one and leaves other dimensions unchanged.

We then add dropout layer, the dropout layer randomly sets input units to 0 with a frequency of rate at each step during training time, which helps prevent overfitting.

We finally add dense network layer with SoftMax activation, Image result for dense layer. The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer.



```
Model: "LeafDisease_MobileNet"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
mobilenet_1.00_224 (Function (None, 7, 7, 1024)        3228864
_________________________________________________________________
global_average_pooling2d_1 ( (None, 1024)              0
_________________________________________________________________
dropout_1 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_1 (Dense)              (None, 38)                38950
=================================================================
Total params: 3,267,814
Trainable params: 38,950
Non-trainable params: 3,228,864
```

We use Adam optimizer for optimization of training.

D. Training

Here we train our model for 25 epochs and 150 steps per each epoch.

For validation, we use test data set and 100 steps.

After training, we store 38,950 parameters for later usage.

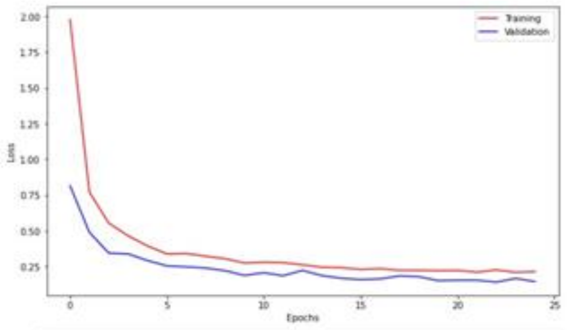
E. Prediction

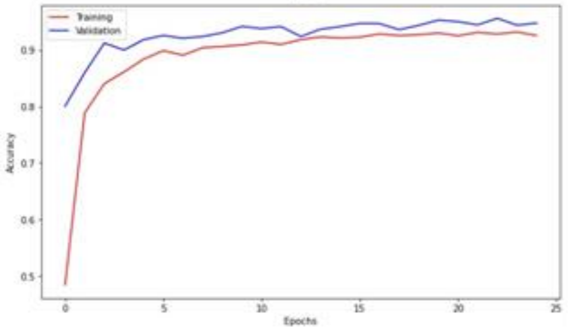
We load the image and convert it to the target size 225 x225. we preprocess the image by scaling it with a factor of 1/255.

We use the preprocessed image and predict the output using our model. We get the prediction with the maximum probability.

VI. RESULTS

After 25 epochs of training, we get an accuracy of 95.05% and a loss of 14.72%.





REFERENCES

[1] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto and Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv:1704.04861

[2] Michael T. Rosenstein, Zvika Marx, Leslie Pack Kaelbling and Thomas G. Dietterich, "To Transfer or Not to Transfer,"

[3] Gao Huang Zhuang Liu, Laurens van der Maaten and Kilian Q. Weinberger, "Densely Connected Convolutional Networks," 21-26 July 2017 17355312.

[4] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever and Ruslan Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting," unpublished.