Plant Disease Classification Using MobileNetV2

Ahmed Aboutaleb, Ismael Elsharkawi, and Marwah Hisham

The American University in Cairo

Abstract - *Crop diseases are a major threat to the supply of food. As the smartphone technology became widespread worldwide, it has now become technically feasible to leverage image processing techniques to be able to identity types of plant diseases using just a simple photo. Identifying diseases can lead to quicker decisions being taken to intervene to reduce the effects of crop diseases on food supply. Using a public dataset of almost 54,303 images of diseased and healthy plants, a depthwise convolutional network and semi supervised methods are trained to classify crop species and disease status of 38 different classes. This research explores the possibility of using MobileNetV2 in the field of plant disease classification. While MobileNet V2 does not improve the accuracy over the best performing model used, which is ResNet50, it is still a good choice since it is more applicable for mobile devices. The best model using MobileNetV2 achieves testing accuracy of 96.91%.*

Key words - **MobileNetV2, CNN, Plant Diseases, PlantVillage, segmentation, depth wise separable.**

1. Introduction

Plant diseases and pests are very common and harmful, especially in the developing countries where smallholders are responsible for agricultural production. That results in huge crop losses that can have significant negative impacts economically. To minimize this impact and ensure the availability of more healthy plants and food safety, technology can play a pivotal role by using Deep Learning techniques in detecting such diseases at early stages [1]. Convolutional Neural Networks (CNN) can be used to analyze the photos of the plants and identify whether they are healthy or not, and if not, their type of disease will be identified. Using this technology will replace the human naked eye diagnosis, which will save time and can even give better accuracies. It can also be applied in the agricultural fields in the developed countries to make the whole process pipeline automated and give higher productivity [1]. Since almost everyone owns a smartphone nowadays, a mobile application that uses the same algorithm can be deployed to detect the plant diseases using input captures, and that application can be used by farmers. In this paper, we are presenting the results of a plant disease classifier we built that can identify 14 plant species and their 26 diseases using different CNN architectures that uses MobileNetV2 CNN-based model, which is more compatible with mobile devices.

2. Related Work

2.1 ResNet 50

ResNet is a short name for Residual Network. As indicated by its name, the innovation that this network brings to the field is residual learning. Generally, in a deep Convolutional Neural Network, many layers are stacked together and are trained to a specific task at hand. The network is taught many low to high level features by the end of its layers [2]. In residual learning, the network does not only try to learn some features, but it also tries to learn some residual as well. Residual can be very simply understood as subtraction of a feature learned from the input of that given layer. ResNet is able to do so using some shortcut connections that are directly connecting input of n^{th} layer to some $(n+x)^{th}$ layer. ResNet50 is a 50-layer Residual Network [2]. RetNet architecture is usually used on various computer vision tasks such as image classification, object localization, object detection, etc. It can also be applied to non-computer vision uses to give the tasks the benefit of depth as well as reduce their computational expense [2].

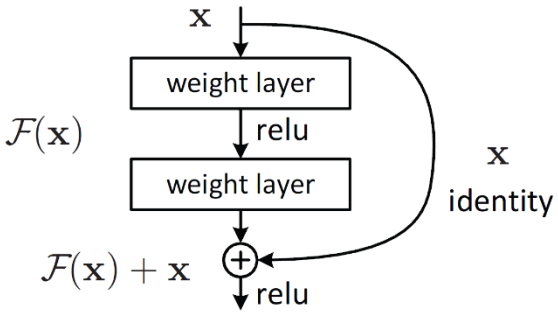

Figure 1: ResNet 50 [2]

In Kalvakolanu's paper, the CNN model he created from the ResNet50 was trained by using the discriminative learning. The initial training was done only for the newly added layers after performing the hyper parameter tuning for learning rate. The model he proposed with Discriminative learning rate achieved an accuracy of 99.44% which is the current-state-of-art for this dataset [3].

2.2 VGG 16

VGG16 is a CNN that has a total of 12 Convolutional layers. Those convolutional layers have very small receptive areas(3x3). Rangarajan and Purushothaman used VGG 16 to classify plant diseases specifically for Eggplant. Using VGG, they have achieved an accuracy of 99.4% on a dataset other than Plant Village because Plant Village did not contain EggPlant. In their experiments, they used Hue Saturation Value (HSV), YCbCr and grayscale in addition to the RGB images [4].

2.3 MobileNet V1

MobileNet V1 is a type of CNN created to be able to run on mobile devices with low processor power capabilities. They are built on a streamlined architecture that utilizes depthwise separable convolutions. This helps in building a lightweight deep neural network that has low latency for mobile and embedded devices [5]. MobileNet networks can be built for classification, detection, embeddings and segmentation in the same way that other well-known large-scale models, such as ResNet, are used. MobileNets can be ran effectively on mobile and embedded devices with TensorFlow Lite. MobileNet is designed in away that it depends heavily on pointwise separable convlution rather than full convolution [5]. As the table below highlights, only the first layer is a full convolution layer, after that all the layers that are between that full convolution layer and the Avg Pool layer are depthwise separable convolution layers [5].

Table 1: MobileNet V1 Architecture [5]

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

Depthwise Separable Convolution is a different type of convolution in which a single convolutional filter is applied to each input channel. In the normal 2D convolution, that is normally performed on multiple input channels, the filter should be as deep as the input which allows us to freely mix all that channel output to generate each element in the filter output [6]. However, the way depth-wise separable convolution is made makes it completely different. Depthwise Separable Convolution is divided into 2 main parts. The first is depthwise convolution and the second is point wise convolution. In Depthwise convolution, there is a kernel for each layer or each channel, that kernel is separate than kernels for other channels [6]. The result of the Depthwise convolution is N layers for an N-layer input. After Depthwise convolution, the next step is pointwise convolution. In pointwise convolution, as illustrated in figure 2, there is a kernel that has the same depth as the input matrix. The convolution operation is performed across the whole depth of

the matrix, not only a single layer [6]. The output will be a single layer. To have N layers as output, there must be N kernels. There are two main reasons why depth wise separable convolution is considered better than normal convolution. The first is that it produces the same output as the convolution, if the right filters are chosen. The second is that the number of multiplication operation is reduced by a factor of $\frac{1}{N} + \frac{1}{D_k^2}$ where N is the number of output channels and D_k is the size of the input matrix [6].

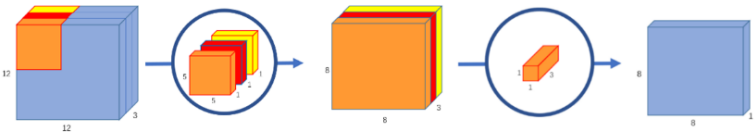

Figure 2: Depthwise convolution followed by pointwise convolution [6]

2.4 MobileNetV2

MobileNetV2 is similar to MobileNet V1. Both use depthwise separable convolution. However, in version 2 of MobileNet, inverted residual blocks are used. That takes place by having a pointwise convolution layer, then having a depthwise convolution layer, then another pointwise convolution [7]. There are 2 flavors of the inverted residual blocks, the one with a stride of 1 and the one with a stride of 2. In the stride 2 flavor, the input is added to the output of the last pointwise convolution [7].

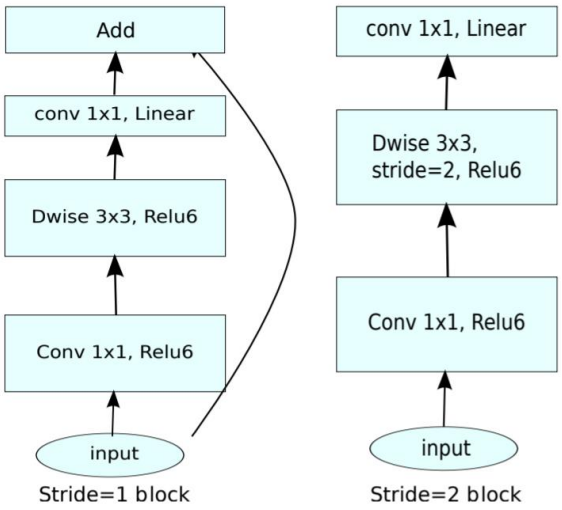

Figure 3MobileNet V2 inverted residual blocks [7]

3. Method

3.1 Architecture choice

The architecture that was used is MobileNet V2. After choosing MobileNetV2 as the basis for our network, we proceeded with trying different variations by adding more trainable layers to it, and by finetuning a number of layers at the end and observe the performance. After trials with different architectures, we chose the model that yielded the best results with relatively low number of parameters. In this model, we have added one convolutional layer of 32 3x3 filters and ELU activation function, one dropout layer of 0.5 dropout rate, one average pooling layer, and the final dense output layer of 38 neurons and softmax activation on top of MobileNetV2, and we will refer to this architecture as model 1 (Table 2). This architecture had a total of 2,627,910 parameters, of which 369,946 are trainable. The model was tried several times, and for 10 epochs, in order to achieve the best performance and to settle on the hyper parameters that can be used in the rest of the experiments. At the end, a learning rate of 0.001 was found to be the one that is to achieve the best results in terms of accuracy.

Table 2: MobileNetV2-based model 1

Layer (type)	Output Shape	Param #
mobilenetv2_1.00_224 (Functi	(None, 7, 7, 1280)	2257984
conv2d_5 (Conv2D)	(None, 5, 5, 32)	368672
dropout_5 (Dropout)	(None, 5, 5, 32)	0
global_average_pooling2d_5 (	(None, 32)	0
dense_5 (Dense)	(None, 38)	1254

Total params: 2,627,910
Trainable params: 369,926
Non-trainable params: 2,257,984

In order to check the possibility of getting better results, finetuning the last 60 layers of model 1 was experimented. It had 2,232,518 trainable parameters, which is definitely more than the previous method, but it yielded the best results.

3.2 Using Segmented Data

Model 1 has been tried on different variations of the datasets that were already existing. To understand the effect of noise in the background on the yielded results, the model has been tested on images in which the background is segmented in order see the true classification of the plant leaves.

4. Dataset and Features

4.1 General Dataset Information

We used the plant village dataset, which consisted of a total of 54,303 images of both diseased and healthy plants. The dataset is divided into 38 different classes which span 14 different species of plants [8]. The dataset was divided into training, validation and testing. The training set includes 34,757 images. The validation set includes 8,673 images, and the testing set includes 10,876 images. All images were resized to have a width of 224 pixels and a height of 224 pixels to be compatible of the MobileNetV2 model. Then this was repeated for images that were segmented.

Figure 4: Plant Village Dataset [8]

4.2 Dataset Variations

The plant village data set comes with 3 versions of the same sized. The first variation is the normal colored set of images. The second variation is the same image dataset but in grey scale. And finally, the final version of the dataset is a segmented dataset. The segmented dataset is a set in which the background of the images is removed, which allows the classification to be based entirely on the features that are in the leaves themselves. The figure below shows the differences between the colored, greyscale, and segmented versions of a given image.

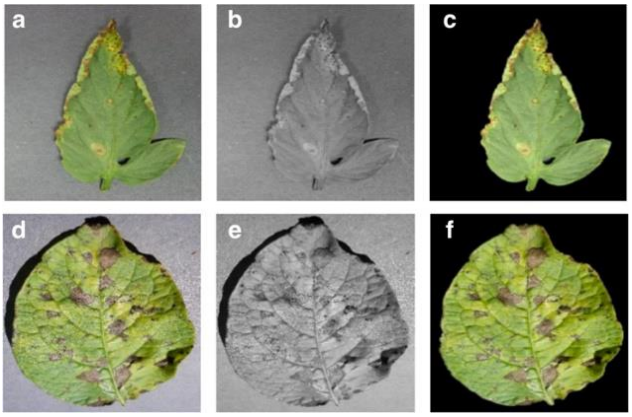

Figure 5: Classes of Datasets [8]

5. Experiments/Results

5.1 Model 1 - Unsegmented

According to figure 5, model 1 yielded testing accuracy of 92.34% and cross entropy loss of 0.231 for the non-segmented images, after being ran for 10 epochs, before it starts to overfit. The same network is tried at a lower and a higher learning rate, but this learning rate of 0.001 achieves the highest accuracy. The figures below show the change in accuracy and loss at different epochs.

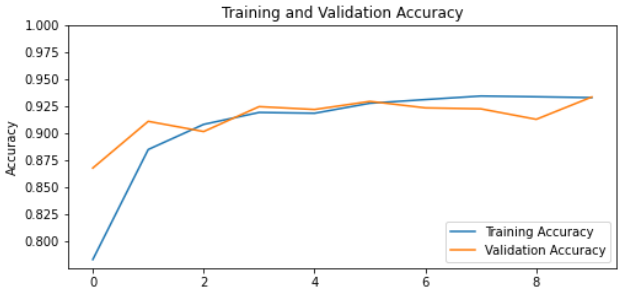

Figure 6: Model 1 Accuracy - unsegmented

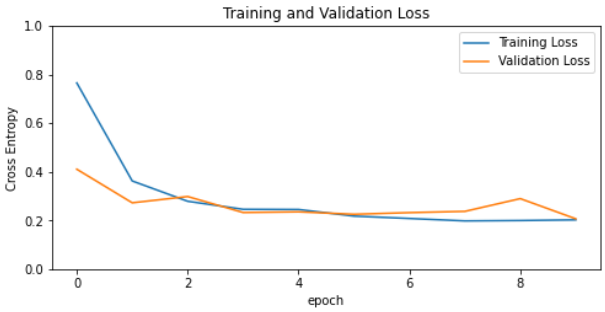

Figure 7: Model 1 loss – unsegmented

5.2 Model 1 – segmented images

The same model is experimented with background-segmented images, and it resulted in slightly higher accuracy and less loss. The accuracy of testing is 92.819% and the total loss is 0.223.

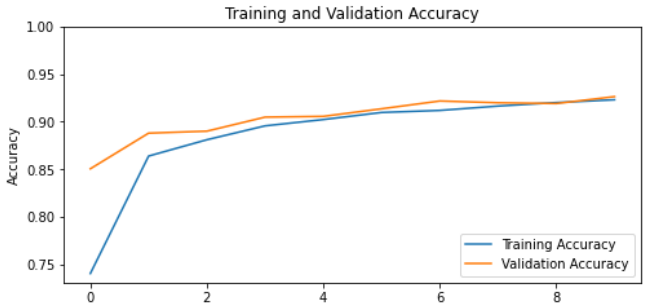

Figure 8: Model 1 Accuracy - segmented

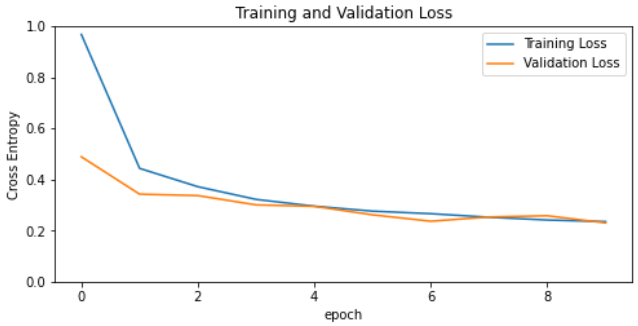

Figure 9: Model 1 Accuracy - segmented

5.3 Finetuning Model 1

In the previous experiment, we trained additional layers on top of the MobileNetV2 pre-trained model. We proceeded next with trying finetuning the last 60 layers of model 1 with the non-segmented images, and it yielded better results. It achieved an increase of 4.57% in the testing accuracy, which became 96.91% and it also achieved a lower loss of 0.1007.

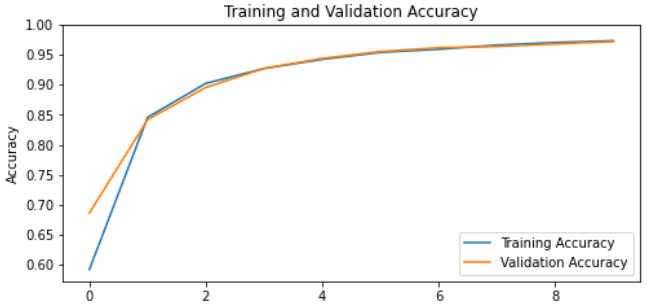

Figure 10: Training vs validation accuracy – finetuning non-segmented

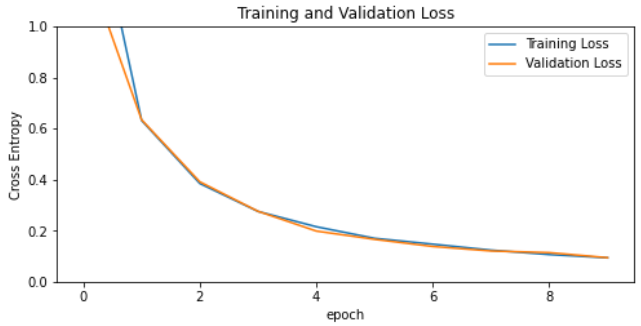

Figure 11: Training vs validation loss – non-segmented

Finetuning was also tried with background-segmented images, but it yeilded very close results: 96.6% test accuracy and 0.11 loss.

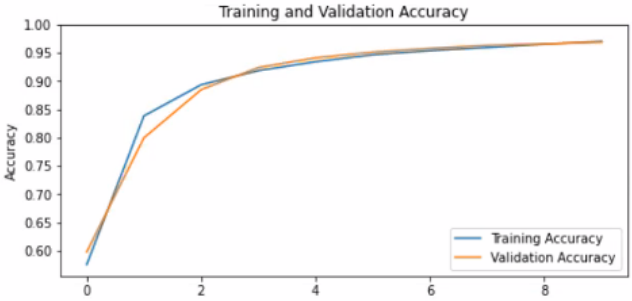

Figure 12: Training vs validation accuracy – finetuning segmented

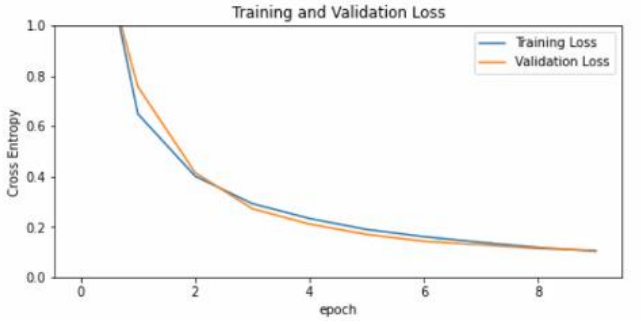

Figure 13: Training vs validation loss – finetuning segmented

5.4 Summary of Results

To create a network that yields the best results for the given dataset, several variations of the number of layers have been experimented with in order to be able to compare. It has been found that the variation of the network in which the additional one convolutional layer of 32 3x3 filters and ELU activation function, one dropout layer of 0.5 dropout rate, one average pooling layer, and the final fully connected output layer of 38 neurons and softmax activation on top of MobileNetV2 gives

the best results when finetuning the last 60 layers using the unsegmented images. The table below summarizes the final Testing loss and accuracies achieved in all the mentioned experiments.

Table 3: Results Summary

Experiments	Cross Entropy	Total Accuracy
Model 1 – non segmented imaged	0.231	92.34%
Model 1 – segmented images	0.223	92.819%
Model 1- finetuning + non segmented images	0.1007	**96.91%**
Model 1- finetuning + segmented images	0.11	96.6%

6. Conclusion

6.1 Concluding Remarks

The results that have been obtained in the paper show that MobileNet V2 are promising networks in the field of plant disease classification and can be actually relied on in real life application. Our best model achieves an accuracy of 96.91% on the testing set when classifying images into 38 different classes. This is a reliable value to be used by farmers especially that farmers would normally be using this application over a small set of plant classes. So, for a lower-class problem, higher accuracies would be expected even more. Even though out model does not exceed the 99.44% that was achieved by the state-of-the-art ResNet50 model, it is still better since it is usable in real life mobile applications.

6.2 Recommendations

It is recommended to try different lower-class problems based on the locations in which research is being conducted, to make sure that irrelevant classes don't overcomplicated the model. Also, further fine tuning of hyper parameters, changing the network architecture, and expanding the dataset can be done to try and achieve higher accuracies. Finally, the newly emerged MobileNet V3 should be tried since it is the new state-of-the-art model for mobile applications and embedded systems.

Work Cited

[1] S. P. Mohanty, D. Hughes, and M. Salathé, "Using Deep Learning for Image-Based Plant Disease Detection," *Frontiers in plant science*, vol. 7, 2016. Available: https://arxiv.org/ftp/arxiv/papers/1604/1604.03169.pdf

[2] V. Fung, "An Overview of ResNet and its Variants," 17-Jul-2017. [Online]. Available: https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035.

[3] A. T. Kalvakolanu, "Plant Disease Detection using Deep Learning," *International Journal of Recent Technology and Engineering Regular Issue*, 2020.

[4] A. K. Rangarajan and R. Purushothaman, "Disease Classification in Eggplant Using Pre-trained VGG16 and MSVM," Scientific Reports, vol. 10, no. 1, 2020.

[5] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," arXiv.org, 17-Apr-2017. [Online]. Available: https://arxiv.org/abs/1704.04861.

[6] C.-F. Wang, "A Basic Introduction to Separable Convolutions," Medium, 14-Aug-2018. [Online]. Available: https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728.

[7] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov and L. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks", *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018. Available: https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf.

[8] "spMohanty/PlantVillage-Dataset", *GitHub*, 2016. [Online]. Available: https://github.com/spMohanty/PlantVillage-Dataset/tree/master/raw.