Advanced CNNs using color images.

Data: The Sugarcane Leaf Disease Dataset has 5 classes: Healthy, Mosaic Redrot, Rust and Yellow disease captured with smart phones of various configurations. There are a total of 2522 images. The dataset was collected in Maharashtra, India. Image sizes are not constant, and all images are in RGB format.

First attempt.

2522 images were used for training/testing. The file was split to 1765 training and 757 testing.

The model called "SugarCane.ipynb" (sugarcane.pdf) consisted of a simple model with 32 channels in the first layer, a kernel size of 2 x 2, and an image size of 244 x 244 units. Figure 1 (left and right) shows the training accuracy and loss for 10 epochs. Figure 1 (left) shows that the accuracy with this model plateaus at about 70% for the validation data. The overall accuracy for classify the data was 72.1 %.

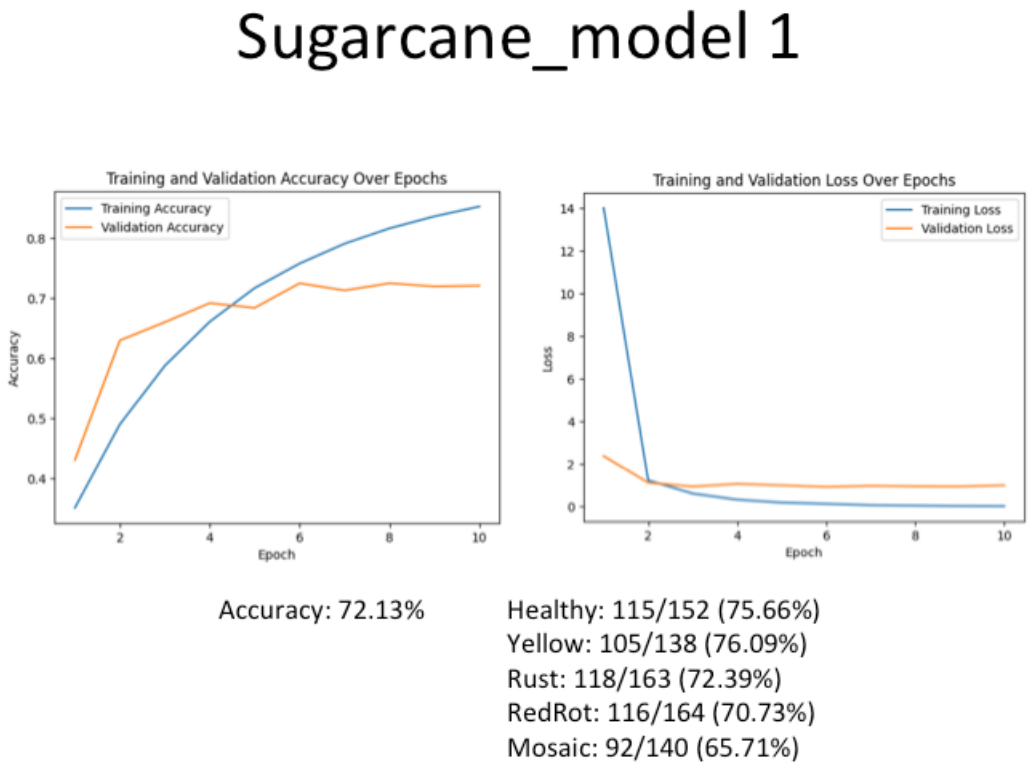

Figure 1. Training and testing results for Sugarcane model 1.

Second attempt.

To determine how to improve the predictive model, I investigated EfficientNet, which is a pre-trained model based on compound scaling. It tackles trade-offs between model size, accuracy and computational efficiency.

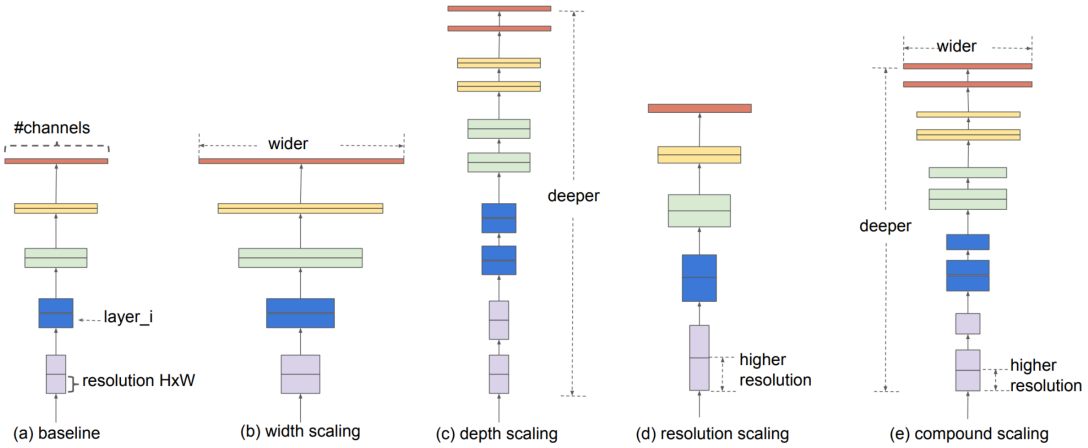

Figure 2. Model scaling and architecture. EfficientNet (far right) is a compromise between going deeper and wider. Model 1 was similar to baseline. The compound scaling is shown on the far right and based on the paper by Tan and Le (2019) entitled: "EfficientNet: Rethinking model scaling for convolutional neural networks (https://arxiv.org/abs/1905.11946).

What makes EfficientNet different from my model 1 is that it is composed of 9 layers while I only used 2. *Would increasing the number of layers improve classification accuracy?*

Results indicated better prediction accuracies with EfficientNet presumably because it uses 8 CNN layers (Figure 3 and 4). I improved upon model 1 by increasing the number of layers from 2 to 8. I also included Batch normalization. Batch Normalization is a technique used in deep neural networks to normalize the inputs of each layer in a mini-batch. It helps improve the training stability and speed by normalizing the input to a layer so that it has a mean close to zero and a standard deviation close to one. Prior to sending the images to the CNN, I transformed the images by resizing, center cropping, and rotating them (see Jupyter notebook script). I also split the files into training (n=1764), validation (n=504) and testing (n=253) files.

EfficientNet (Validation=92% accuracy)

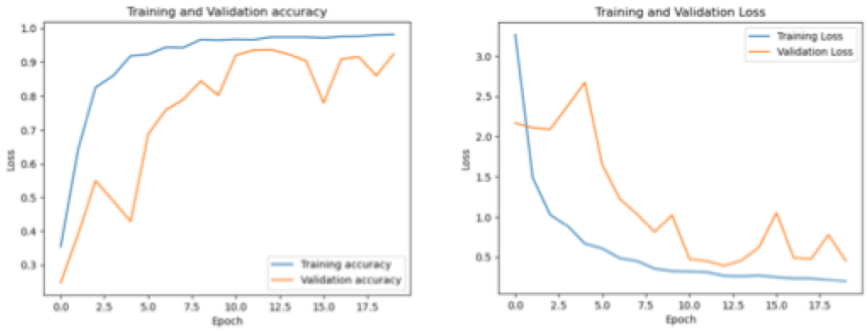

Figure 3. Training and validation accuracy and loss for EfficientNet using TensorFlow. Accuracy is much better using 8 layers!

EfficientNet Confusion matrix

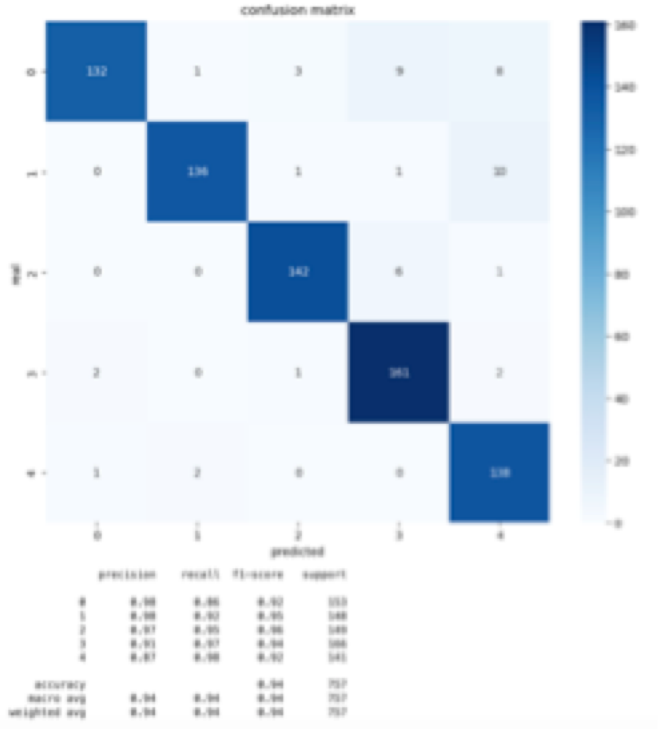

Figure 4. Confusion matrix for EfficientNet.

Third attempt. Sugarcane Model 2 with 8 layers.

Figure 5 shows that prediction accuracy of the validation data was much improved over model 1 and similar to the EfficientNet results. The confusion matrix (Figure 6) shows a breakdown in predictions by group label. Overall, the prediction accuracies were similar to the EfficientNet results. Figure 7 shows the confusion matrix for the validation data set (87%), which is somewhat lower than the EfficientNet. Figure 8 shows the activation statistics in terms of mean and variance. While there are elevated activations in the 5 to 12 layers, most of the variance in the activation occurs in the last layer. Figure 9 shows the configuration of the model based on layers. Recall that each color (RGB) has its own layer. Figure 10 shows the ordination plot of all layers. Apparently, the last layer plays the most important role in the predictions. The complete analyses can be seen in the Final_sugarcane jupyter notebook (here).

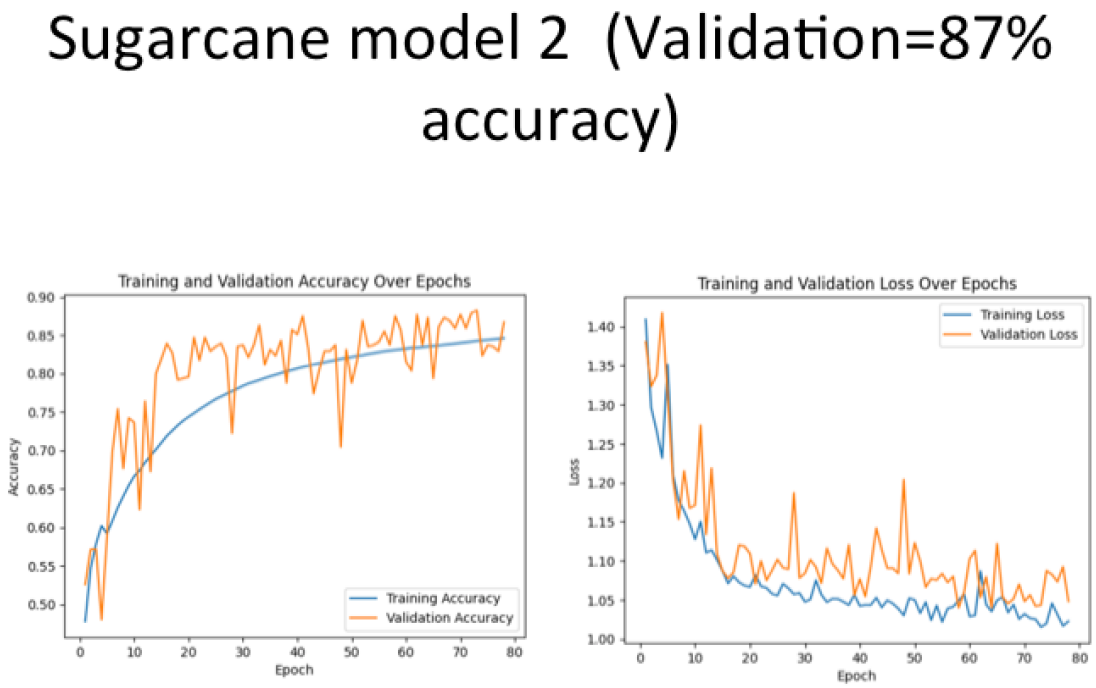

Figure 5. Model 2 yielded better predictions than Model 1 presumably due to 8 layers, resizing, rotation, cropping, and batch normalization.

Sugarcane Model 2 Test Confusion matrix

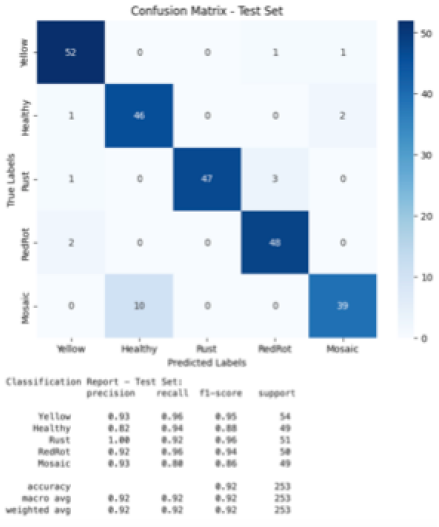

Figure 6. Confusion matrix for test data using Model 2. Test accuracy was 92%. The test data set was not used in training the model.

Sugarcane Model 2 Validation Confusion matrix

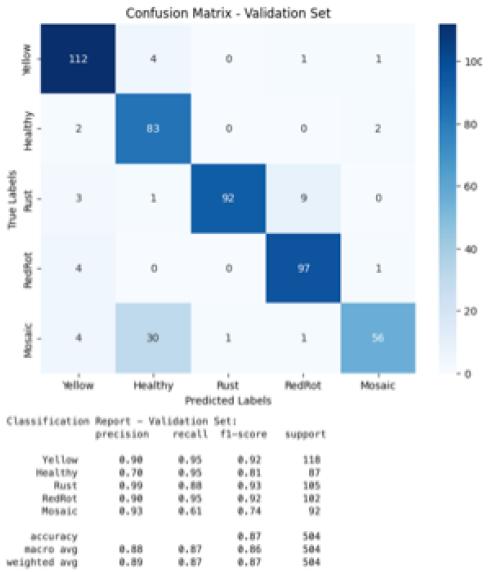

Figure 7. Confusion matrix for validation data using Model 2. Validation accuracy was 87%. The validation data set was used in training the model.

Activation Statistics

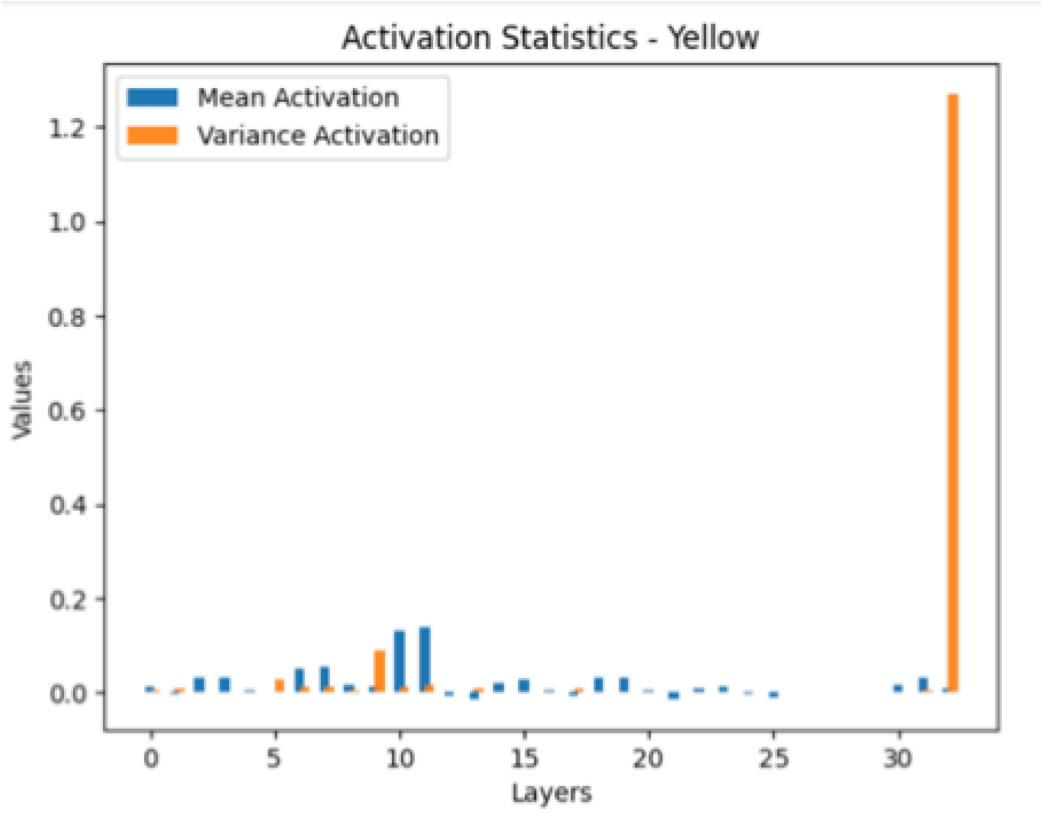

Figure 8. Activation layers for model 2. See figure 9 for layer configuration.

Layer 1 - Shape: torch.Size([1, 32, 489, 489])
Layer 2 - Shape: torch.Size([1, 32, 489, 489])
Layer 3 - Shape: torch.Size([1, 32, 489, 489])
Layer 4 - Shape: torch.Size([1, 32, 244, 244])
Layer 5 - Shape: torch.Size([1, 16, 245, 245])
Layer 6 - Shape: torch.Size([1, 16, 245, 245])
Layer 7 - Shape: torch.Size([1, 16, 245, 245])
Layer 8 - Shape: torch.Size([1, 16, 122, 122])
Layer 9 - Shape: torch.Size([1, 4, 123, 123])
Layer 10 - Shape: torch.Size([1, 4, 123, 123])
Layer 11 - Shape: torch.Size([1, 4, 123, 123])
Layer 12 - Shape: torch.Size([1, 4, 61, 61])
Layer 13 - Shape: torch.Size([1, 16, 62, 62])
Layer 14 - Shape: torch.Size([1, 16, 62, 62])
Layer 15 - Shape: torch.Size([1, 16, 62, 62])
Layer 16 - Shape: torch.Size([1, 16, 31, 31])
Layer 17 - Shape: torch.Size([1, 32, 32, 32])
Layer 18 - Shape: torch.Size([1, 32, 32, 32])
Layer 19 - Shape: torch.Size([1, 32, 32, 32])
Layer 20 - Shape: torch.Size([1, 32, 16, 16])
Layer 21 - Shape: torch.Size([1, 64, 17, 17])
Layer 22 - Shape: torch.Size([1, 64, 17, 17])
Layer 23 - Shape: torch.Size([1, 64, 17, 17])
Layer 24 - Shape: torch.Size([1, 64, 8, 8])
Layer 25 - Shape: torch.Size([1, 128, 9, 9])
Layer 26 - Shape: torch.Size([1, 128, 9, 9])
Layer 27 - Shape: torch.Size([1, 128, 9, 9])
Layer 28 - Shape: torch.Size([1, 128, 4, 4])
Layer 29 - Shape: torch.Size([1, 488, 5, 5])
Layer 30 - Shape: torch.Size([1, 488, 5, 5])
Layer 31 - Shape: torch.Size([1, 488, 5, 5])
Layer 32 - Shape: torch.Size([1, 488, 2, 2])
Layer 33 - Shape: torch.Size([1, 5])

Figure 9. Configuration of layers in model 2. First number is layer, second number corresponds to padding, third number corresponds to number of channels, fourth and fifth corresponds to kernel size.

PCA of model 2

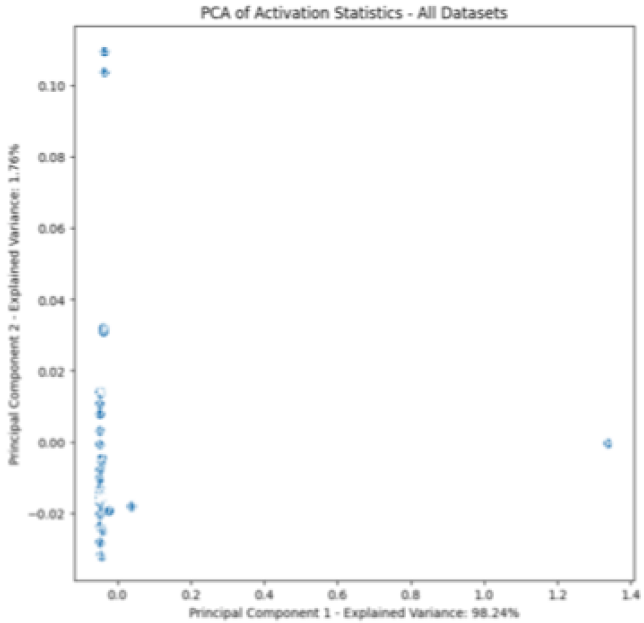

Figure 10. Ordination plot showing that most of the variance is in the first component and last layer (98%).