Verdure: Plant Health Classification

Kunjan Khatri, Rohit Patwa
DS5220: Supervised Machine Learning

Abstract

Crop diseases are a noteworthy risk to sustenance security, however their quick distinguishing proof stays troublesome in numerous parts of the world because of the non-attendance of the important foundation. Emergence of accurate techniques in the field of leaf-based image classification has shown impressive results. Here we make use of Deep Convolutional Neural Networks (CNN) in identifying between healthy and diseased leaves from the data set chosen. Our proposed plan includes various phases of implementation namely feature extraction, training the classifier and classification. The dataset of diseased and healthy leaves is collectively trained under CNN to classify the diseased and healthy images. Overall, using Machine Learning to train the data sets available publicly gives us a clear way to detect the disease present in plants on a colossal scale.

1 Introduction

Crop diseases are a major threat to food security, but their rapid identification remains difficult in many parts of the world due to the lack of the necessary infrastructure.
Misdiagnosis of the many diseases impacting agricultural crops can lead to misuse of chemicals leading to the emergence of resistant pathogen strains, increased input costs, and more outbreaks with significant economic loss and environmental impacts. Current disease diagnosis based on human scouting is time-consuming and expensive, and although computer-vision based models have the promise to increase efficiency, the great variance in symptoms due to age of infected tissues, genetic variations, and light conditions within trees decreases the accuracy of detection.

Plant diseases affect the growth of their respective species; therefore, their early identification is very important. Many developed/modified Deep Learning architectures are implemented along with several visualization techniques to detect and classify the symptoms of plant diseases. Moreover, several performance metrics are used for the evaluation of these architectures/techniques. This project proposes a comprehensive usage of Deep Learning models used to visualize and classify various plant diseases.

2 Technical Approach

2.1 Logistic Regression

First method we used was multi class classification using logistic regression.
A logistic regression algorithm takes as its input a feature vector $\boldsymbol{x}$ and outputs a probability, $\boldsymbol{p(y = i \mid x)}$ that the feature vector represents an object belonging to the class. For images, the feature vector might be just the values of the red, green and blue (RGB) channels for each pixel in the image: a one-dimensional array of

$$n_x = n_{height} \times n_{width} \times 3$$

real numbers formed by flattening the three-dimensional array of pixel RGB values. A logistic regression model is so named because it calculates

$$\hat{y} = \sigma(z)$$

Where ***σ(z) = 1/1+exp(-z)*** is the logistic function and ***z = w'x + b*** for a set of parameters, ***w*** and ***b***. ***w*** is a n_x-dimensional vector (one component for each component of the feature vector) and b is a constant "bias".

2.2 Support Vector Machine(SVM)

Our second approach was to use SVM to classify given plant diseases. SVM is an exciting algorithm and the concepts are relatively simple. The classifier separates data points using a hyperplane with the largest amount of margin. That's why an SVM classifier is also known as a discriminative classifier. SVM finds an optimal hyperplane which helps in classifying new data points.

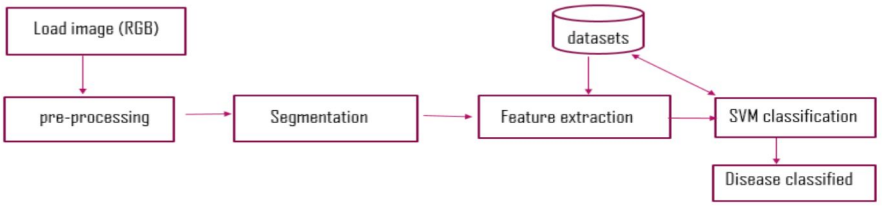


Figure 1: Flowchart for SVM implementation

The extracted feature dataset is used as a training data and given as input to the training classifier SVM so that it can classify the healthy/diseased plant images.

2.3 Convolution Neural Net - ResNet18

Every major ImageNet model has a different architecture, but each one has the common building blocks: **Conv2D, MaxPool, ReLU**.

Convolution: Convolution is a rather simple algorithm which involves a kernel (a 2D matrix) which moves over the entire image, calculating dot products with each window along the way.

$$Conv(f,h) = \sum_j \sum_k h_{jk} \cdot f_{(m-j)(n-k)}$$

In the above equation, the kernel *h* is moving across the length and breadth of the image. The dot product of *h* with a sub-matrix or window of matrix *f* is taken at each step, hence the double summation (rows and columns).

MaxPool: Max pooling is very similar to convolution, except it involves finding the maximum value in a window instead of finding the dot product of the window with a kernel. Max pooling does not require a kernel and it is very useful in reducing the dimensionality of convolutional feature maps in CNNs. This process can be represented with the equation below:

$$Maxpool(f,h)_{mn} = max(m+j-1, n+k-1 \ \ \forall \ \ 1 \leq m \leq l, \ 1 \leq n \leq l)$$

In the above equation, the window moves across the image and the maximum value in each window is calculated. Once again, this process is very important in reducing the complexity of CNNs while retaining features.

ReLU: ReLU is an activation function commonly used in neural network architectures. *ReLU(x)* returns 0 for *x < 0* and *x* otherwise. This function helps introduce non-linearity in the neural network, thus increasing its capacity to model the image data.

$$G_{mn} = ReLU(x)_{mn} = \begin{cases} 0 & \text{if } x_{mn} < 0 \\ x_{mn} & \text{if } x_{mn} \geq 0 \end{cases}$$

As mentioned earlier, this function is nonlinear and helps increase the modeling capacity of the CNN models.

ResNet18: ResNet18 is a pre-trained Deep Learning model for image classification of the Convolutional Neural Network(CNN) which is a class of deep neural networks, most commonly applied to analyzing visual imagery. ResNet18 is 18 layers deep and is trained on a million images of 1000 categories from the ImageNet database.The ResNet18 model consists of 5 stages each with a residual block. Each residual block has 3 layers with both 1*1 and 3*3 convolutions. The concept of residual blocks is quite simple.

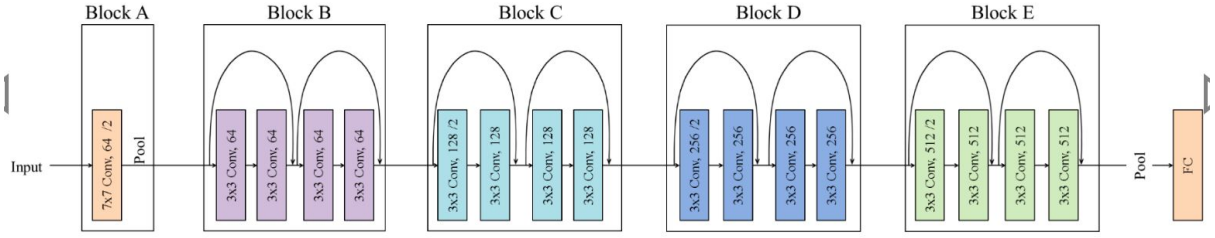


Figure 2: Architecture of ResNet

2.4 Convolutional Neural Network - EfficientNet-B5

EfficientNet is another popular (more recent) CNN-based ImageNet model which achieved the SOTA on several image-based tasks in 2019. EfficientNet performs model scaling in an innovative way to achieve excellent accuracy with significantly fewer parameters. It achieves the same if not greater accuracy than the previous model with a much shallower architecture.

There are three scaling dimensions of a CNN: *depth, width,* and *resolution.* **Depth** simply means how deep the network is which is equivalent to the number of layers in it. **Width** simply means how wide the network is. One measure of width, for example, is the number of channels in a Conv layer whereas **Resolution** is simply the image resolution that is being passed to a CNN. The figure below will give you a clear idea of what scaling means across different dimensions.

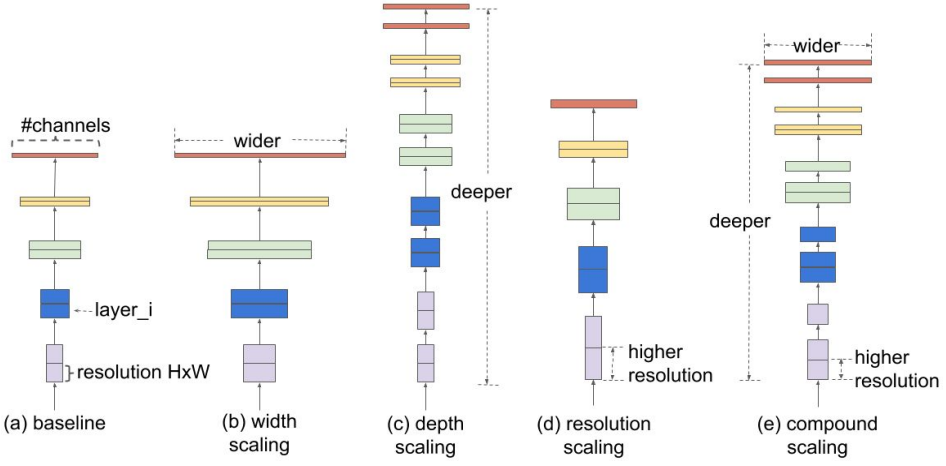


Figure 3: Scaling across different dimension of a CNN

The model consists of the EfficientNet head (without the top), followed by global average pooling and a dense layer (with softmax) to generate probabilities.

3 Experimental Results

3.1 Dataset

We used the dataset given in the Plant Pathology 2020 - FGVC7 contest in Kaggle. It is an image classification problem where each image can be classified into one of 4 categories (healthy, multiple_diseases, rust, scab). The training data size is 785MB and has 1821 images of plant leaves. Each image is an RGB image of size 2048*1365 containing one or more leaves**.**

3.2 EDA

Channel Distributions: By plotting some of the images and its distribution, we came to know that the green parts of the image have very low blue values, but by contrast, the brown parts have high blue values. This suggests that green (healthy) parts of the image have low blue values, whereas unhealthy parts are more likely to have high blue values. **This might suggest that the blue channel may be the key to detecting diseases in plants.**

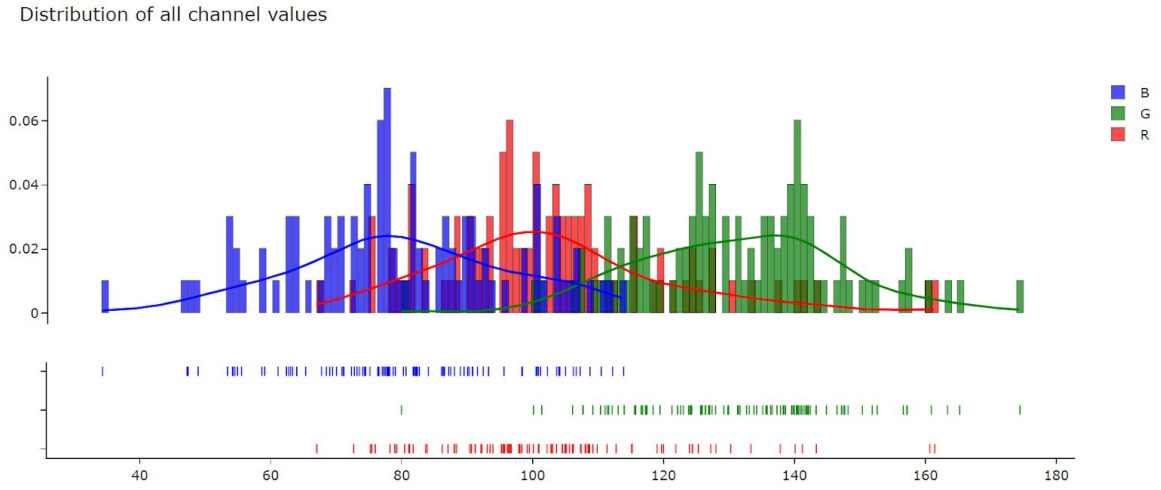
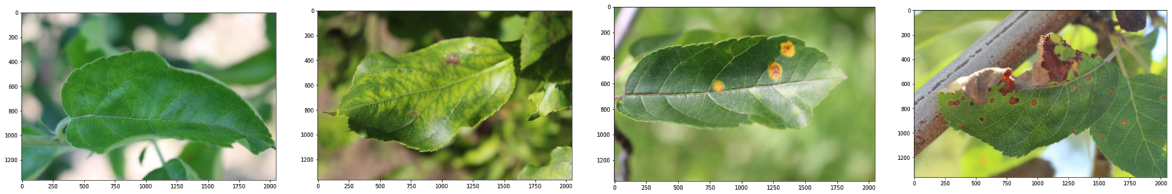


Figure 4: Distribution of all channel values for training set

From the above plot, we can clearly see which colors are more common and which ones less common in the leaf images. Green is the most pronounced color, followed by red and blue respectively. The distributions, when plotted together, appear to have a similar shape, but shifted horizontally.



Healthy Leaf | Leaf with Scab | Leaf with rust | Leaf with multiple diseases

Figure 5: Sample leaf from each class

In the above images, we can see that the healthy leaf is completely green, and does not have any brown/yellow spots or scars. Healthy leaves do not have scab or rust.We can see

that a leaf with "scab" has large brown marks and stains across the leaf. The brown marks across the leaf are a sign of these bacterial/fungal infections. A leaf with "rust" has several brownish-yellow spots across the leaf. The yellow spots are a sign of infection by a special type of fungi called "rust fungi".The last leaf shows symptoms for several diseases, including brown marks and yellow spots. These plants have more than one of the above-described diseases.

3.3 Pre-processing

Before passing data to train on the model, we performed some preprocessing steps.The RGB images are converted into the grey images using color conversion. Then we perform the haralick texture feature vector extraction. This helps in identifying different textures of the leaf. Then the histogram equalization which distributes the intensities of the images is applied on the image to enhance the plant disease images. The cumulative distribution function is used to distribute intensity values. These features extracted from different functions are then stacked in one long vector using numpy's hstack function. The feature vector is then normalized using MinMaxScaler from sklearn.

Transformations applied for CNNs are quite different from SVM. Flipping is a simple transformation that involves index-switching on the image channels. In vertical flipping, the order of rows is exchanged, whereas in vertical flipping, the order of rows is exchanged. All major features in the image remain the same, but to a computer algorithm, the flipped images look completely different. These transformations can be used for data augmentation, making models more robust and accurate. Another transformation applied is ShiftScaleRotate and Blurring which helps detect features more accurately.

3.4 Implementation

The SVM model was run by finding the best parameters using Grid Search. A list of parameters of C and gamma were provided. The model takes in an array containing image feature values and outputs the probabilities related to each class. Same approach for Logistic Regression and the model with highest accuracy on the validation set was used for making predictions. We used sklearn's implementation of SVC and LogisticRegression.

For implementing the deep neural networks, Resnet18 and EfficientNet-B5, we used pytorch framework on Google Colab. We store the dataset on google drive in .npy format for quicker access. The data is split into training and test with a 80-20 ratio. We create a dataset class which contains image transformations like random vertical/horizontal flips and random rotations. Then we normalize the input image with the imagenet preset values of mean and variance. Having implemented the data loader, we pass it to the pretrained Resnet18 and Efficient Net-B5 for fine tuning the model according to our dataset. This process is also called Transfer Learning. The predictions are then compared with the ground truth values to calculate the loss using cross entropy. We use a decaying learning rate for the training starting with a value of 8e-4.

The models are trained for 30 epochs each on GPU. Epoch vs loss and epoch vs accuracy curves are attached below. Then we predict the classes for our test dataset. Using these predicted values and the ground truth values, we create a confusion matrix (also attached in the results below).

3.5 Results

3.5.1 Loss vs Epoch

EfficientNet ResNet

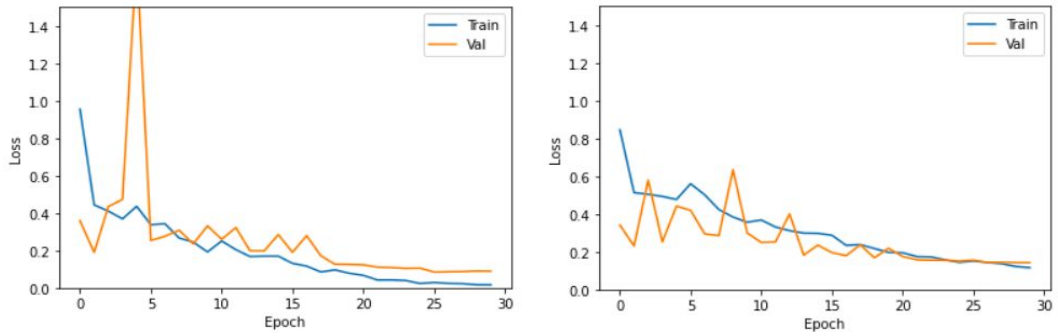


Figure 6: Performance comparison via Loss

From the above plot, we can see that EfficientNet starts achieving much lower validation loss around 5 epochs.

3.5.2 Accuracy vs Epoch

EfficientNet ResNet

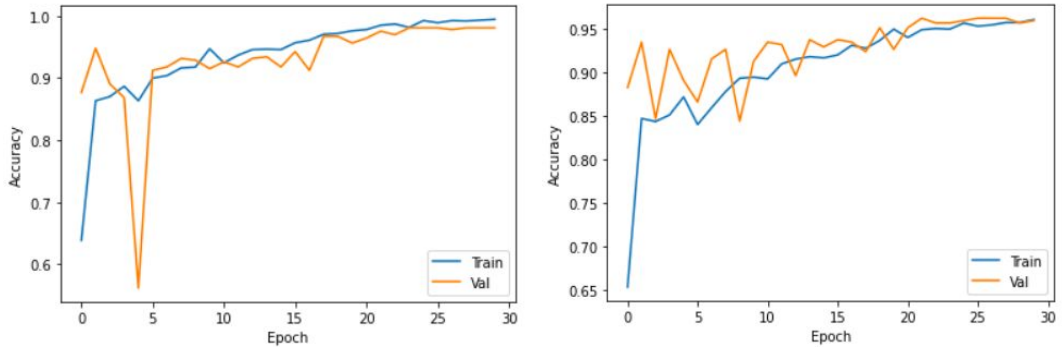
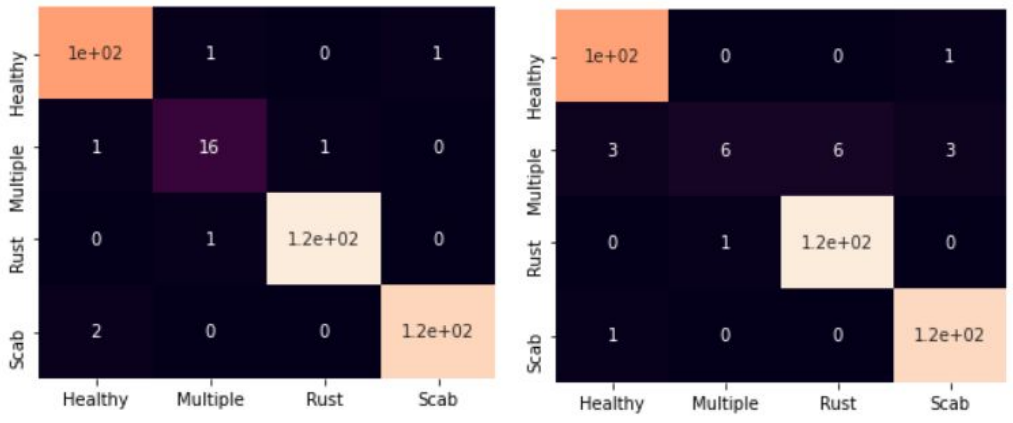


Figure 7: Performance comparison via Accuracy

To support our observation from Figure 11 supports our observation from Figure 10. ResNet achieves max accuracy of 94.57% while EifficientNet performing better achieves max accuracy of 98.03%

Confusion Matrix - All 4 models

EfficientNet ResNet



SVM Logistic Regression

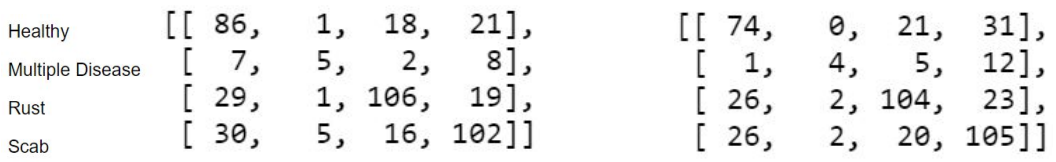


Figure 8: Performance comparison via Confusion Matrix

Accuracy

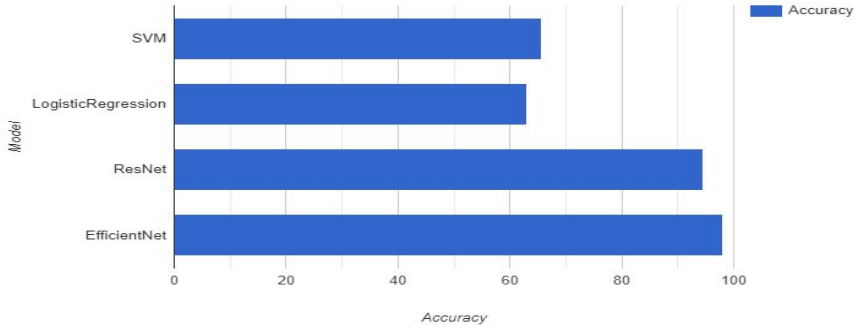


Figure 9: Performance Comparison

4 Conclusion and future work

We are able to classify the plant images with an accuracy of about 98%.EfficientNet is a clear winner over ResNet18 for this problem. Now, it would be worth investigating if we can improve the Network performance without exponentially increasing the computational complexity.Our idea is to extract the second last layers of both the trained networks, stack them together and connect with fully connected layers followed by a classifier. Expectation: the new network should work equally good if not better than the EfficientNet alone.

5 Contributions

The work was distributed evenly between both team members.Kunjan performed preliminary analysis and EDA along with implementation of Resnet model and Logistic Regression. Rohit performed extraction of features for SVM along with its implementation and EfficientNet implementation. Additionally, jointly contributed to the analysis of the final results, general interpretations and producing the final report.

References

https://www.ripublication.com/ijcir17/ijcirv13n7_23.pdf
https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview
https://www.mdpi.com/2223-7747/8/11/468/htm
https://www.frontiersin.org/articles/10.3389/fpls.2016.01419/full
https://arxiv.org/abs/2004.11958
https://github.com/lukemelas/EfficientNet-PyTorch
https://www.cs.princeton.edu/courses/archive/spring16/cos495/slides/ML_basics_lecture7_multiclass.pdf
https://medium.com/@nainaakash012/efficientnet-rethinking-model-scaling-for-convolutional-neural-networks-92941c5bfb95