ALOK SINGH

Agricultural Image Captioning for Smart Farming

Problem Statement:

We need to create a model that can look at pictures of crops and describe what it sees. This includes identifying the type of crop, its growth stage, its health, and any problems like pests or diseases.

Business Use Case:

Farmers and crop experts can use this tool to keep an eye on their crops. It helps them find issues early, manage their fields better, and ultimately grow more and lose less.

Dataset:

We're using the Plant Pathology 2020 - FGVC7 from Kaggle. This dataset has images of many different plants and notes on various diseases. It will help train our model to recognize and describe different aspects of the crops.

Expected Outcome:

With this model, farmers will be able to get detailed descriptions of their crops from images, helping them make better decisions and improve their harvests.

Introduction

Diagnosing plant diseases early using just images of leaves is crucial. The main categories of diagnosis are "healthy," "scab," "rust," and "multiple diseases." Early detection can save a lot of crops each year. This helps reduce hunger and ensures farmers get the harvest they deserve.

Contents

Exploratory Data Analysis (EDA)

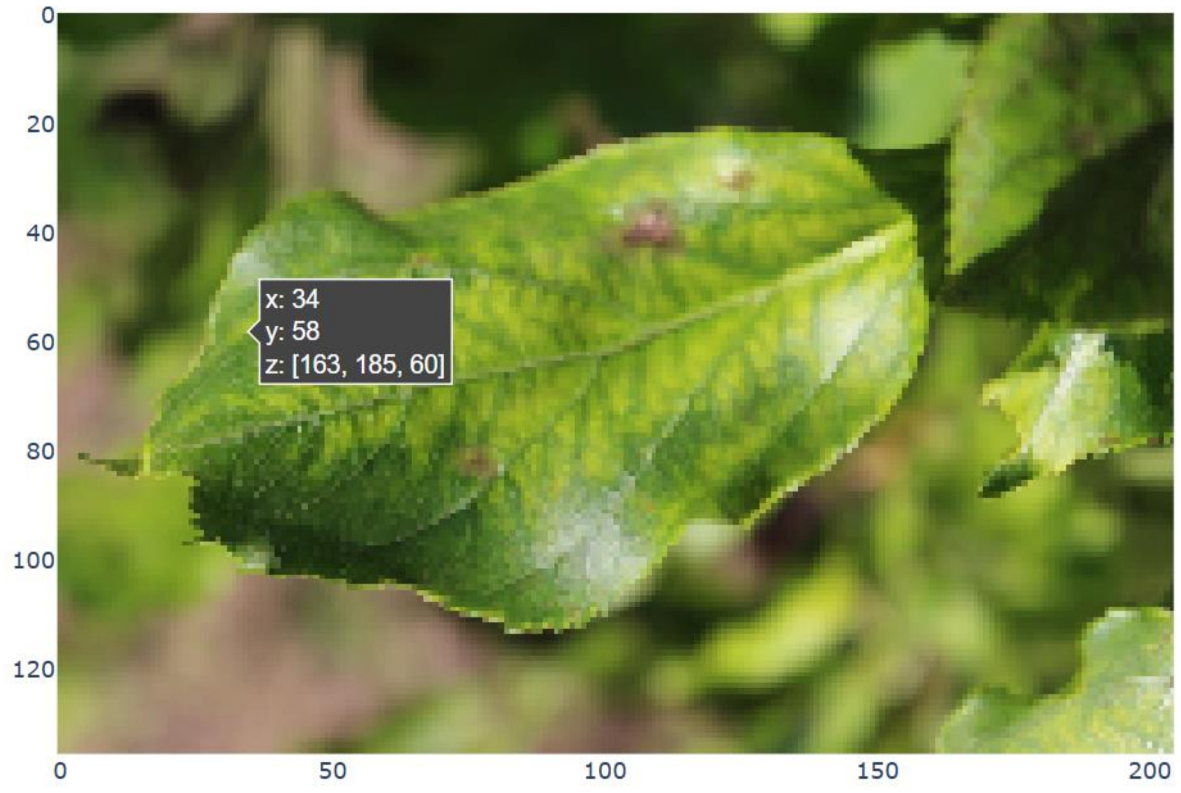
1 - Install and import necessary libraries

2 - Load the data and define hyperparameters

3 - Load sample images

4- Visualize one leaf

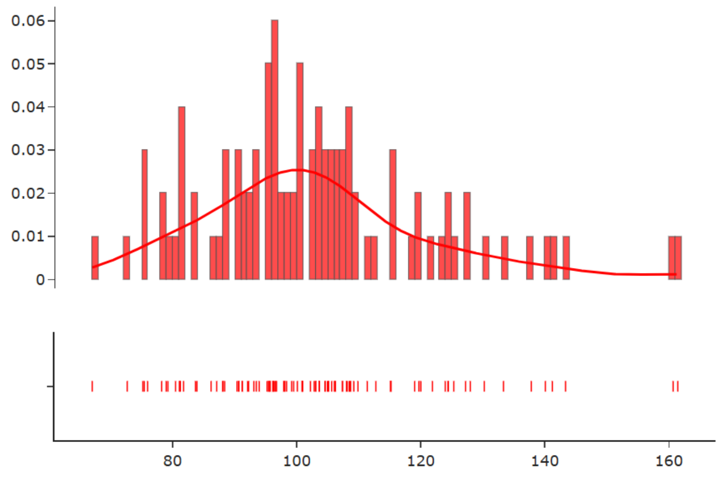
I plotted the first training image and noticed that green areas have low blue values, while brown areas have high blue values. This suggests that healthy (green) parts have low blue values, and diseased parts have high blue values. Therefore, the blue channel could be important for detecting plant diseases.



5 - Channel distributions

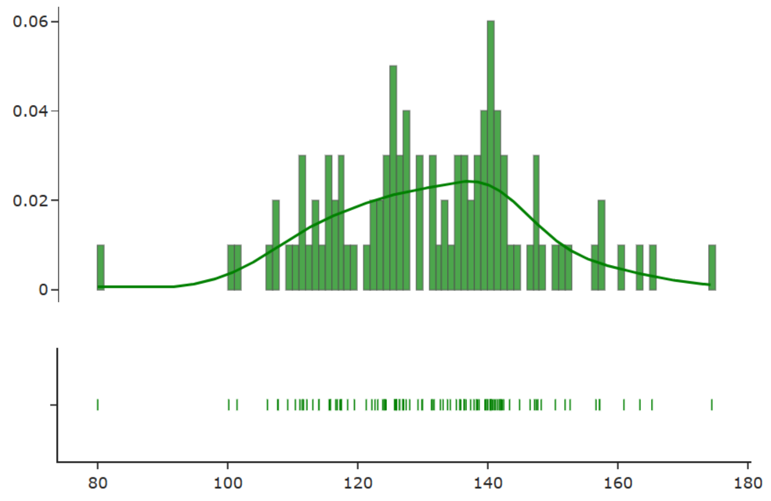
Red channel values

The red channel values seem to roughly normal distribution, but with a slight rightward (positive skew). This indicates that the red channel tends to be more concentrated at lower values, at around 100. There is large variation in average red values across images.



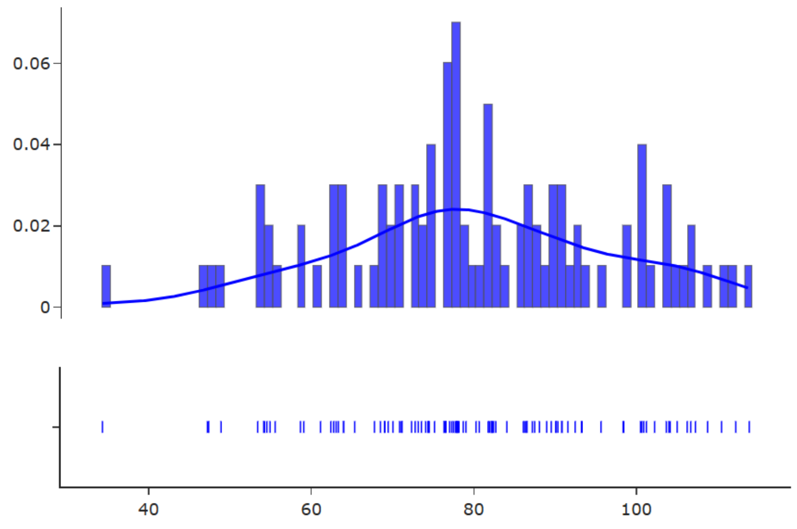
Green channel values

The green channel values have a more uniform distribution than the red channel values, with a smaller peak. The distribution also has a leftward skew (in contrast to red) and a larger mode of around 140. This indicates that green is more pronounced in these images than red, which makes sense, because these are images of leaves!



Blue channel values

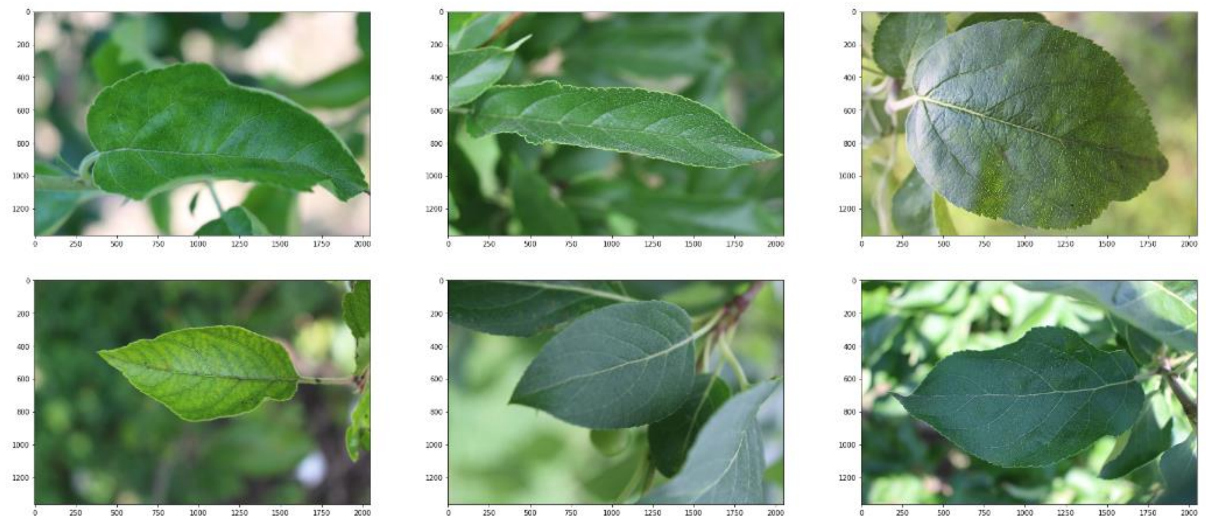
The blue channel has the most uniform distribution out of the three color channels, with minimal skew (slight leftward skew). The blue channel shows great variation across images in the dataset.



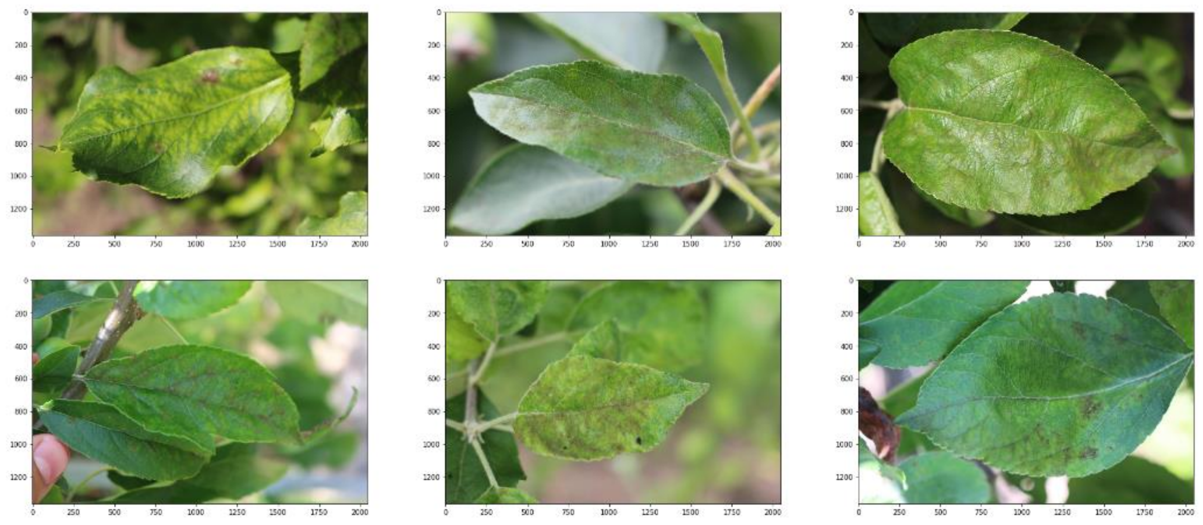
6 - Visualize sample leaves

Now, I will visualize sample leaves beloning to different categories in the dataset.

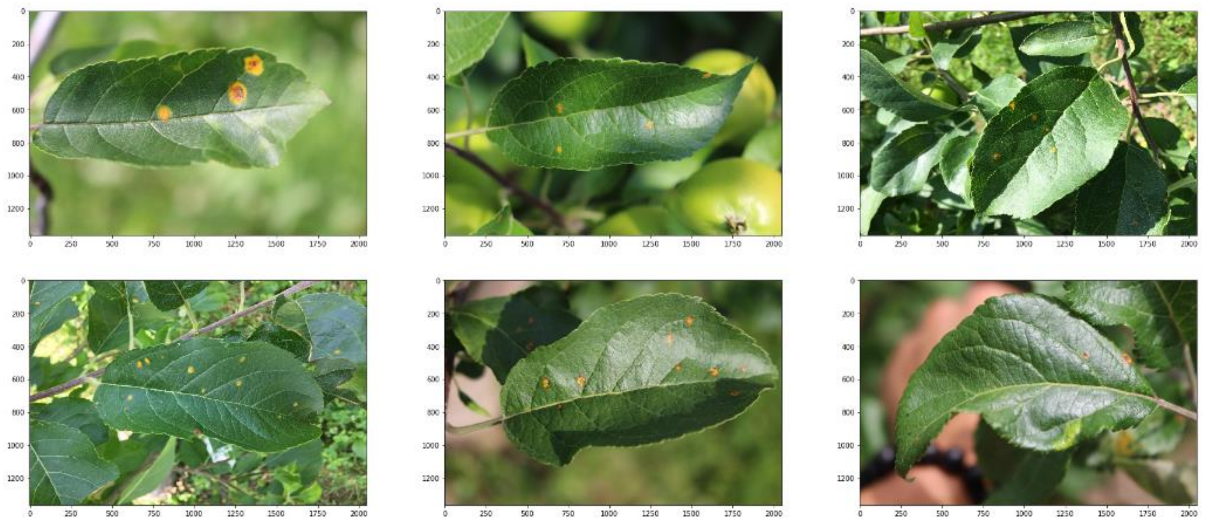
Healthy leaves



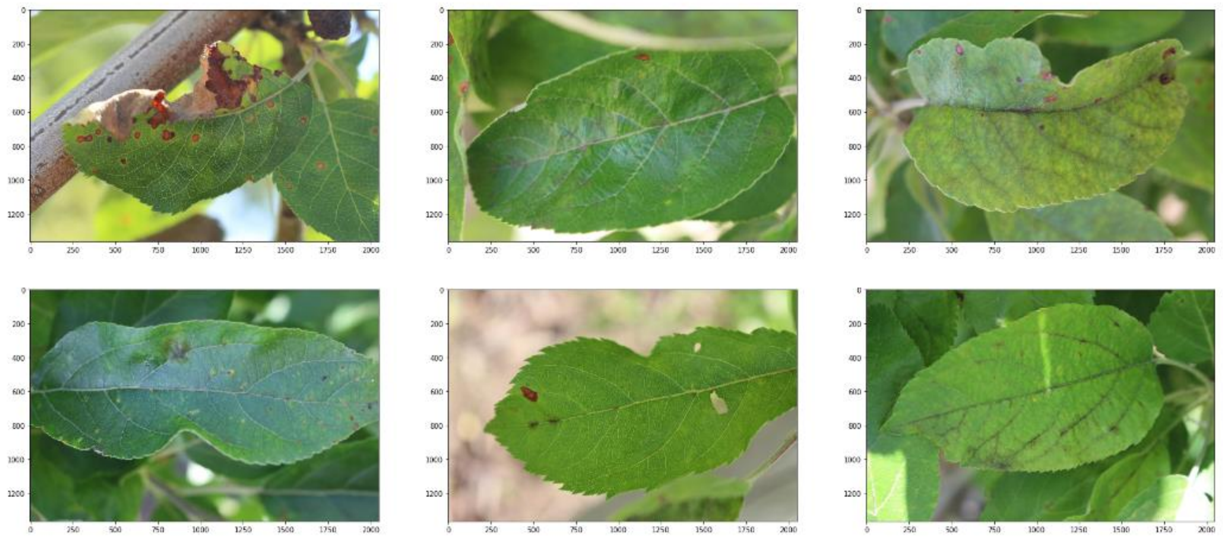
Leaves with scab



Leaves with rust



Leaves with multiple diseases



Visualize targets

Now, I will visualize the labels and target data. In all the below plots, blue represents the "desired" or "healthy" condition, and red represents the "undesired" or "unhealthy" condition.

Pie chart

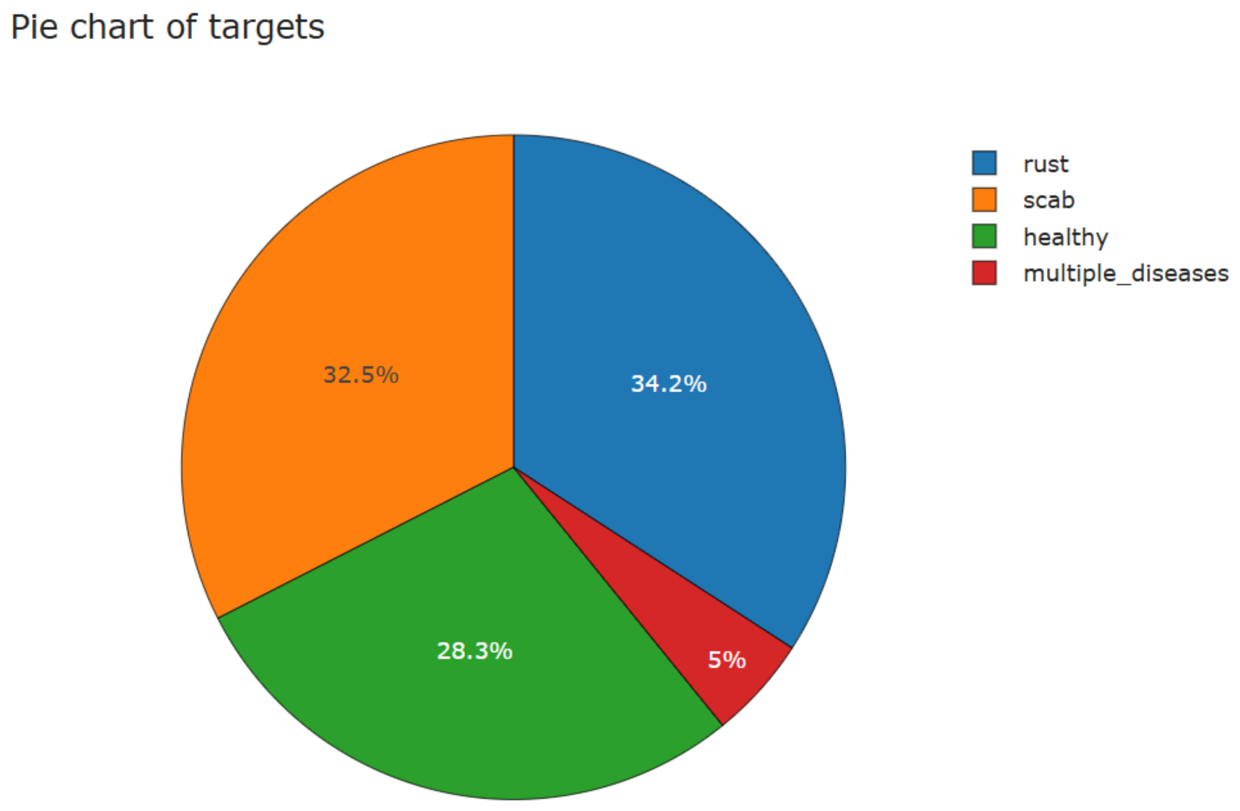


Image processing and augmentation

Canny Edge Detection

Canny is a popular algorithm that detects edges in images, created by John F. Canny in 1986. It works in several steps:

Noise Reduction: Removes noise using a 5x5 Gaussian filter.

Finding Intensity Gradient: Filters the image to get the first derivative in horizontal (Gx) and vertical (Gy) directions, finding the edge gradient and direction.

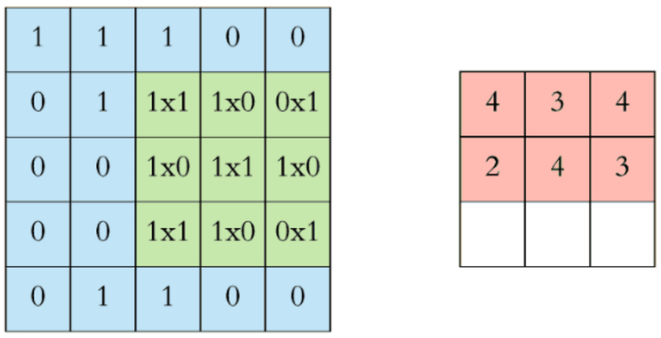
Rounding: Rounds the gradient direction to vertical, horizontal, or diagonal.

Non-maximum Suppression: Removes unwanted pixels by checking if each pixel is a local maximum in its gradient direction.

Hysteresis Thresholding: Uses two thresholds (minVal and maxVal) to decide if pixels are edges or not. Pixels between thresholds are edges if near "sure-edge" pixels.

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. The GIF below demonstrates convolution in action.



The above process can be summarized with an equation, where *f* is the image and *h* is the kernel. The dimensions of *f* are *(m, n)* and the kernel is a square matrix with dimensions smaller than *f*.

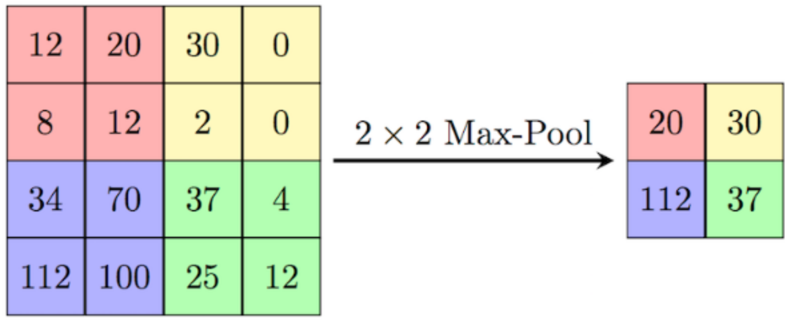
Modeling

Preparing the ground

Before we move on to building the models, I will explain the major building blocks in pretrained CV models. Every major ImageNet model has a different architecture, but each one has the common building blocks: Conv2D, MaxPool, ReLU. I have already explained the mechanism behind convolution in the previous section, so I will now explain MaxPool and ReLU.

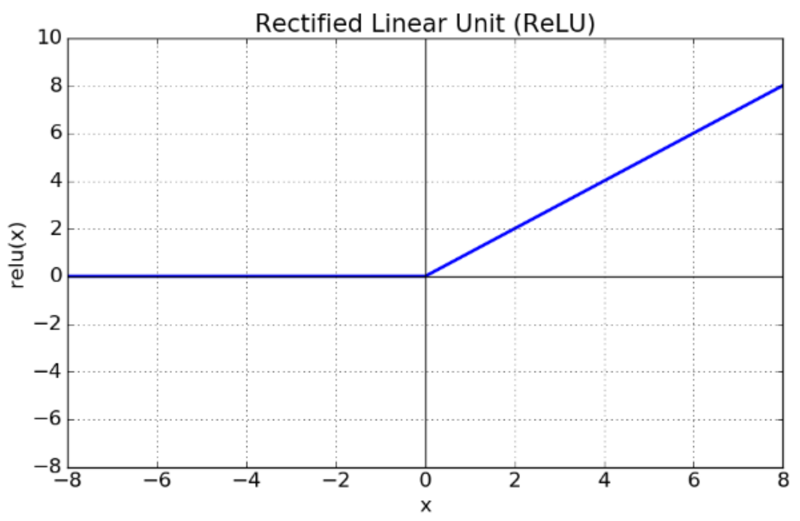
MaxPool

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. The image below demonstrates the working of MaxPool.



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 introducenon-linearity in the neural network, thus increasing its capacity ot model the image data. The graph and equation of ReLU are:

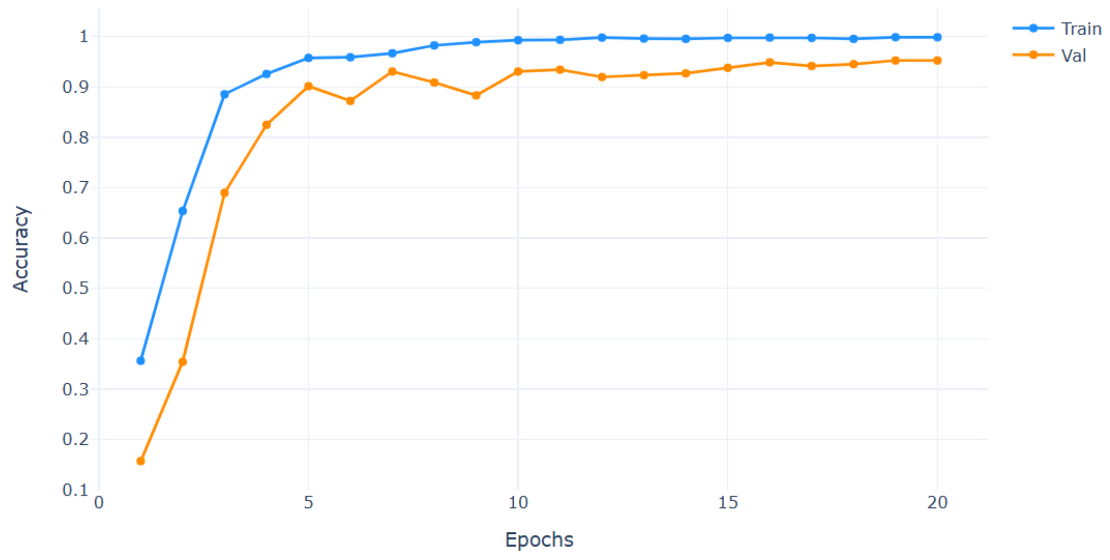


$$G_{mn} = ReLU(x)_{mn} = \begin{cases} 0 & \text{if } x_{mn} < 0 \\ x_{mn} & \text{if } x_{mn} \geq 0 \end{cases}$$

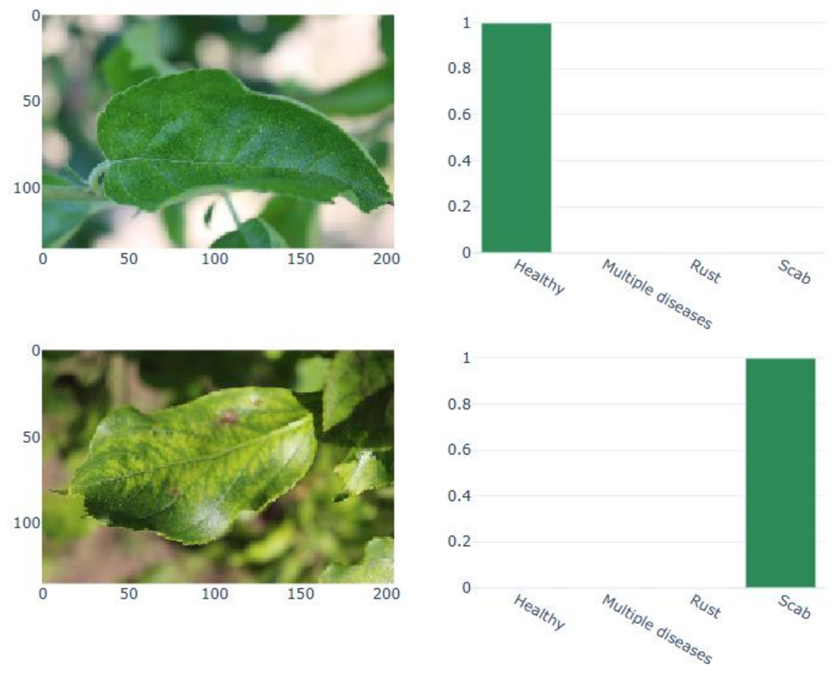
<u>DenseNet</u>

Densely Connected Convolutional Networks (DenseNets), are a popular CNN-based ImageNet used for a variety of applications, inclusing classification, segmentation, localization, etc. Most models before DenseNet relied solely on network depth for representational power. Instead of drawing representational power from extremely deep or wide architectures, DenseNets exploit the potential of the network through feature reuse. This was the main motivation behind the DenseNet architecture. Now let us train DenseNet on leaf images and evaluate its performance.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
densenet121 (Model)          (None, 16, 16, 1024)      7037504
_________________________________________________________________
global_average_pooling2d (Gl (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 4)                 4100
=================================================================
Total params: 7,041,604
Trainable params: 6,957,956
Non-trainable params: 83,648
_________________________________________________________________
```



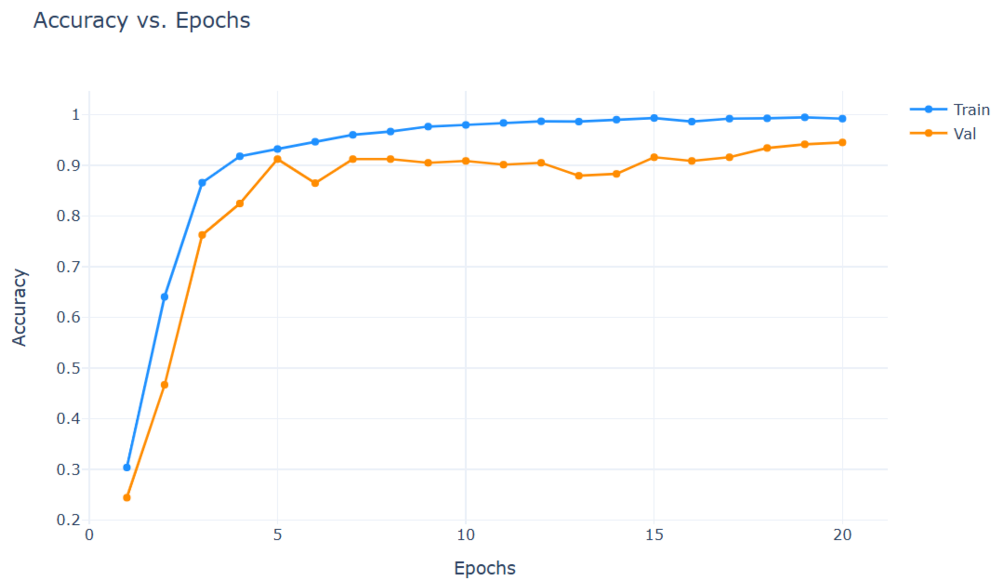
DenseNet Predictions

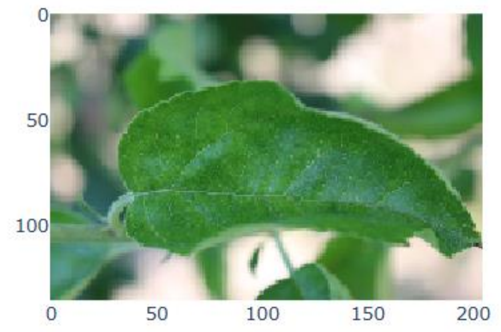


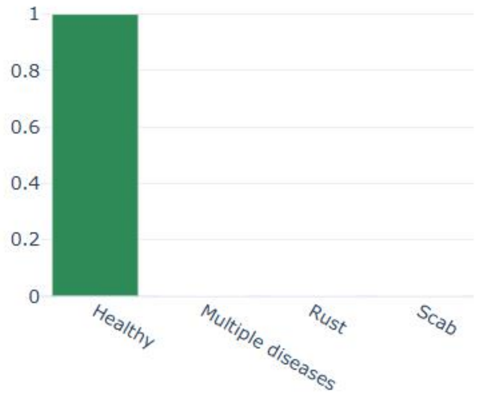
EfficientNet

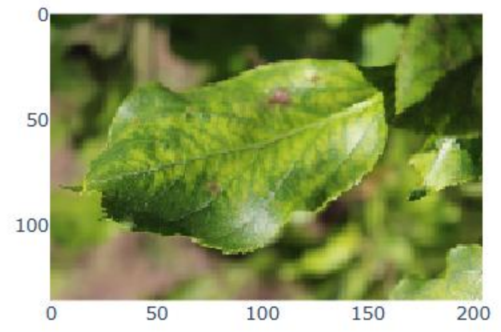
EfficientNet is another popular (more recent) CNN-based ImageNet model which achieved the SOTA on several image-based tasks in 2019. EfficientNet performs model scaling in an innovative way to achieve excellent accuracy with significantly fewer parameters. It achieves the same if not greater accuracy than ResNet and DenseNet with a mcuh shallower architecture. Now let us train EfficientNet on leaf images and evaluate its performance.

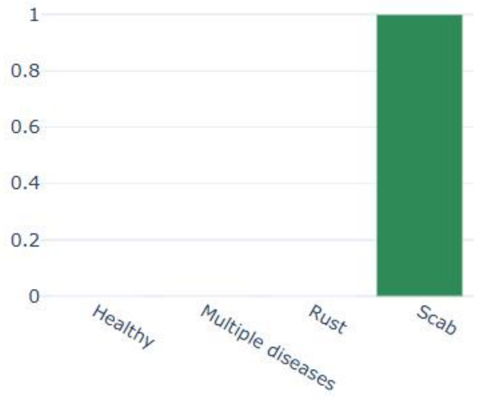
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
efficientnet-b7 (Model)      (None, 16, 16, 2560)      64097680
_________________________________________________________________
global_average_pooling2d_1 ( (None, 2560)              0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 10244
=================================================================
Total params: 64,107,924
Trainable params: 63,797,204
Non-trainable params: 310,720
_________________________________________________________________
```











Conclusion

Considering these factors, EfficientNet might be slightly better if you value efficiency and scalability alongside accuracy. EfficientNet is known for achieving comparable or better performance with fewer parameters and lower computational cost. If the choice is purely based on the provided accuracy and epochs, both models are very similar, but EfficientNet's design for efficiency gives it a slight edge.