Machine Learning Engineer Nanodegree

Capstone Project

Hesham Abdallah
December 10, 2019

I. Definition

Project Overview

Food losses due to crop infections and pests are persistent issues in agriculture for centuries across the globe. In order to minimize the disease induced damage in crops during growth, harvest and postharvest processing, as well as to maximize productivity and ensure agricultural sustainability. Globally, over one billion people are suffering from different situations of malnutrition due to lack of food supply and approximately twice that population do not have access to sufficient nutrients or vitamins to meet their daily nutrition needs.

For instance, in my homeland Egypt agriculture is one of the main pillars of the Egyptian economy; in 2012, agriculture contributed with 13.4% of the national gross domestic production and employed 27.1% of the total Egyptian workforce. Such an application will have a major impact on both of food security side and the economic side.

To cut down the wasted agriculture crops due to pests and disease, a plant disease detection application can help with this task. The main goal is to develop a model for early detection of the common plant disease for contamination and reduce the waste. The model will harness the Machine Learning technology for detection of the infected leaves of multiple crops, this is done by comparing the attributes of the input images to the attributes of the images that are feeded into the model during the training process. The dataset consists of 18335 images of 256*256 pixels each. The total size of the dataset is 326 MB. It is available at Kaggle in the following link:

https://www.kaggle.com/emmarex/plantdisease

Problem Statement

It is estimated that the agriculture waste due to plant diseases and pests is ranging between 20 and 40% of global agricultural productivity. Cutting down this waste will be helpful as the demand for food will continue to increase as the population increasing rapidly. The projections indicate that an additional 70% of food production is required by 2050 to meet the needs.

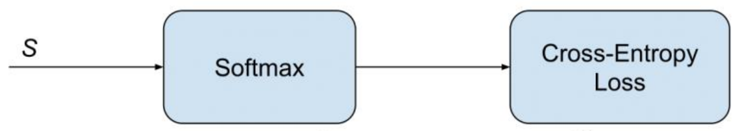
The challenge is to predict whether the captured pictures of the plants is healthy or infected, this is done by detecting the infection and bacterial patterns that the model will learn. This is a classification problem with supervised learning technique.

In order to obtain the model, the following process should be done:

1- Preprocess the dataset images (resizing the images, normalization, etc...)
2- Splitting the dataset into training and testing sets.
3- Creating a benchmark CNN model.
4- Use Transfer Learning technique and see it’s impact on the results.
5- Plot the model’s metrics to judge it’s performance.

Metrics

Categorical Cross Entropy will be used here as a loss function due to the multi-class classification problem. Categorical Cross Entropy will compare the distribution of the predictions (the activations in the output layer, one for each class) with the true distribution, where the probability of the true class is set to 1 and 0 for the other classes. To put it in a different way, the true class is represented as a one-hot encoded vector, and the closer the model’s outputs are to that vector, the lower the loss.



$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \qquad CE = -\sum_i^C t_i log(f(s)_i)$$

Accuracy will be the evaluation metric. By plotting the accuracy through each epoch we will have a clear view of the model’s performance.

II. Analysis

Data Exploration

“Plant Village Dataset “on Kaggle. It contains thousands of samples of healthy plants and infected plants. The dataset contains accurate disease classification for each plant species as shown in the below samples. The available species in the dataset is the widely cultivated and consumed around the world.

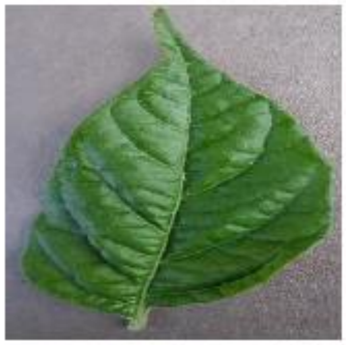


Figure 1 Healthy Plant

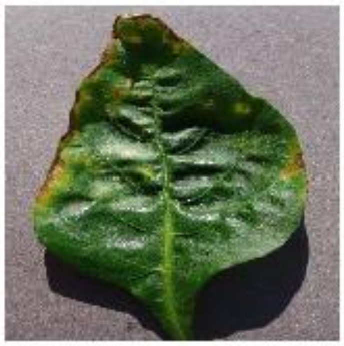
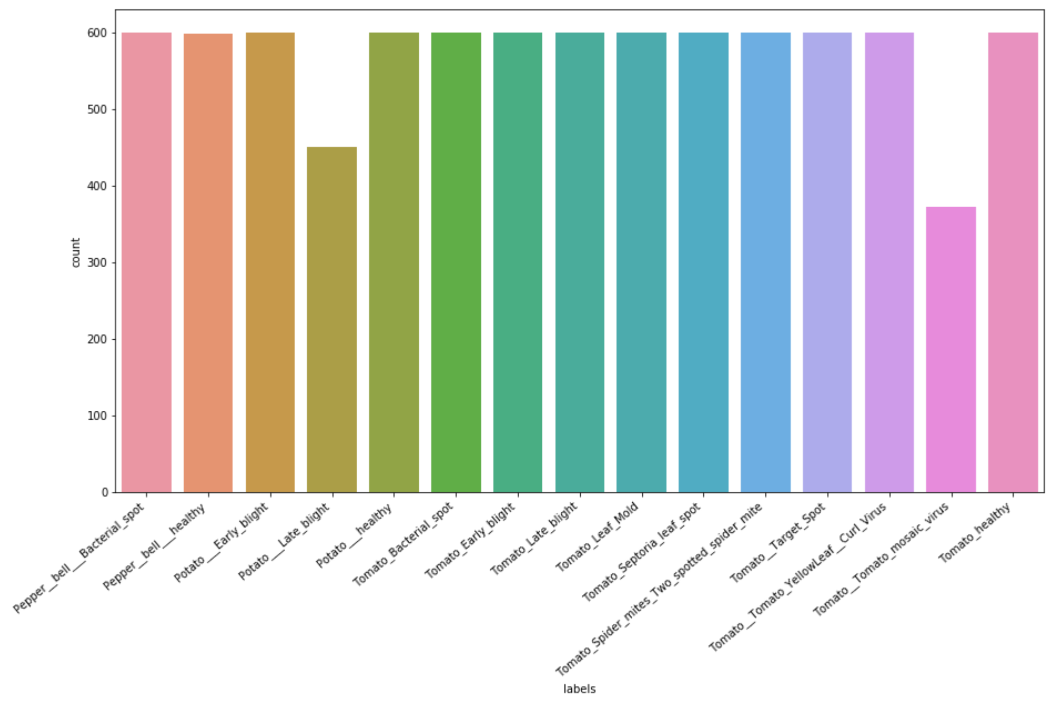


Figure 2 Inffected plant

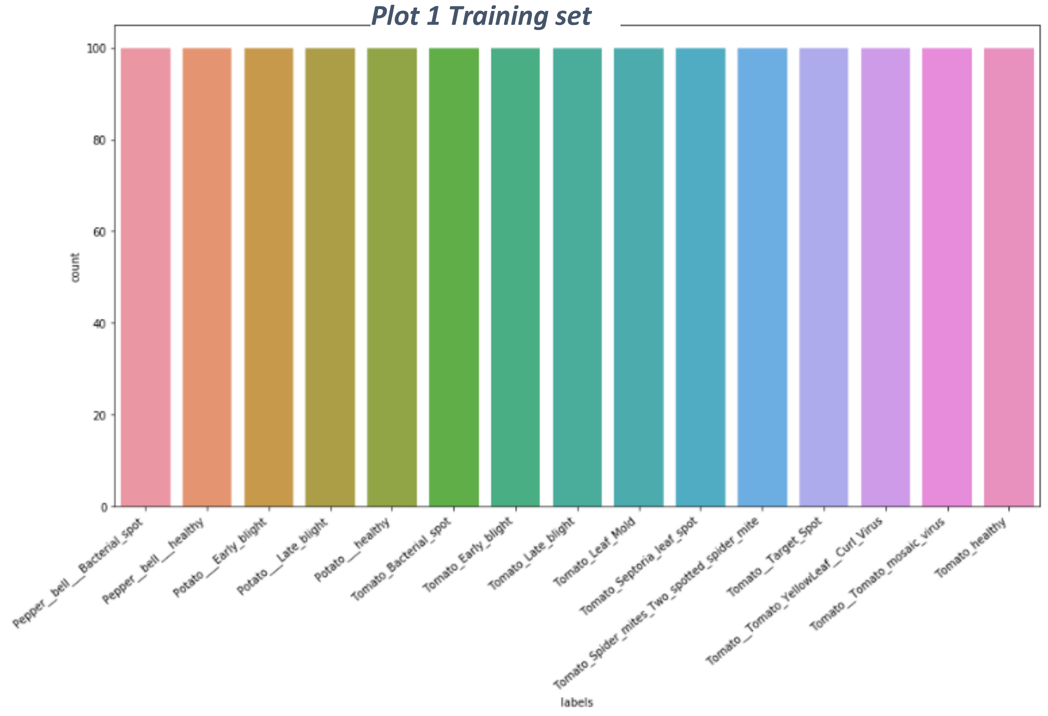
The dataset consists of 18335 images of 256*256 pixels each. due to the high difference of the number of samples of each class and to reduce the class imbalance, I will choose a fixed number of images from each class. The total size of the dataset is 326 MB. Below is the List of the total 15 classes of the dataset.

Exploratory Visualization

The images contained in the dataset is uniformly distributed in the validation set(plot2). The training set is almost uniform except for two classes (plot 1).



Plot 1 Training set



Plot 2 Validation set

Algorithms and Techniques

1- **Deep Learning**: A Machine Learning technique is best used to obtain and learning data representation

2- **Convolution Neural Network**: (CNN or ConvNet) is a class of deep, feed-forward artificial neural networks that has successfully been applied to analyzing visual imagery. A CNN consists of an input and an output layer, as well as multiple hidden layers. The hidden layers are either convolutional, pooling or fully connected. We give CNN an input and it learns by itself that what features it must detect. We won't specify the initial values of features or what kind of patterns it must detect. The following are the various layers a CNN can have:

- **Convolutional** - Also referred to as Conv. layer, it forms the basis of the CNN and performs the core operations of training and consequently firing the of the network. It performs the convolutional operation over the input.

- **Pooling layers** -Pooling layers reduce the spatial dimensions (Width x Height) of the input Volume for the next Convolutional Layer. It does not affect the depth dimension of the Volume.

- **Fully connected layer** - The fully connected or Dense layer is configured exactly the way its name implies. It is fully connected with the output of the previous layer. Fully connected layers are typically used in the last stages of the CNN to be connected to the output layer and construct the desired number of outputs.

- **Dropout layer** - Dropout is a regularization technique for reducing overfitting in neural networks by preventing complex co-adaptations on training data. It is a very efficient way of performing model averaging with neural networks. The term "dropout" refers to dropping out units (both hidden and visible) in a neural network.

- **Flatten** - Flattens the output of the convolutional layers to feed into the Dense layers.

- **Batch Normalization** - enables the use of higher learning rates, greatly accelerating the learning process. It also enabled the training of deep neural networks with sigmoid activations that were previously deemed too difficult to train due to the vanishing gradient problem

- **Global Average Pooling** (GAP) - layers to minimize overfitting by reducing the total number of parameters in the model. Similar to max pooling layers, GAP layers are used to reduce the spatial dimensions of a three-dimensional tensor. However, GAP layers perform a more extreme type of dimensionality reduction.

3. **Activation Functions** - In CNN, the activation function of a node defines the output of that node given an input or set of inputs. Some activation functions are:

- **SoftMax** - The SoftMax function squashes the output of each unit to be between 0 and 1, just like a sigmoid function. It also divides each output such that the total sum of the outputs is equal to 1.

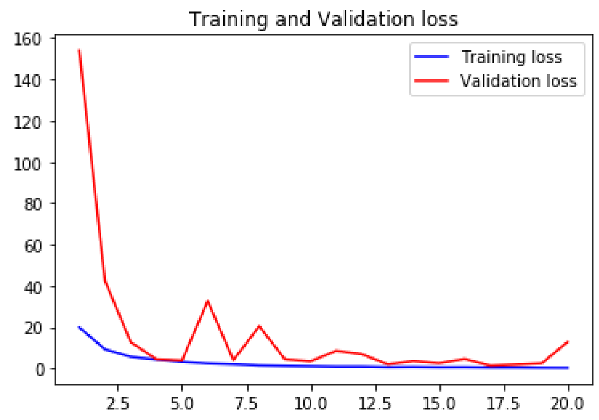
- **ReLu** - A ReLu (or rectified linear unit) has output 0 if the input is less than 0, and raw output otherwise. i.e., if the input is greater than 0, the output is equal to the input.

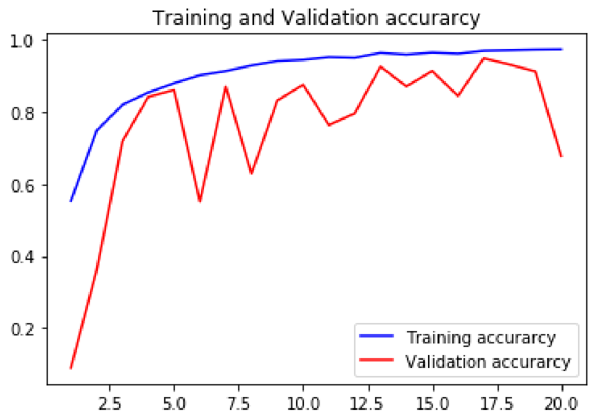
4. **Transfer Learning** - In transfer learning, we take the learned understanding and pass it to a new deep learning model. We take a pre-trained neural network and adapt it to a new neural network with different dataset. For this problem we will use multiple transfer learning models to see which solves our problem the best, such as:

- **VGG-16**
- **VGG-19**
- **RESNET**
- **InceptionV3**

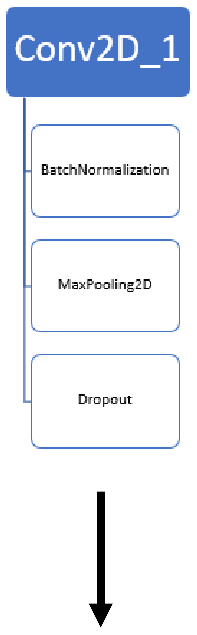
Benchmark

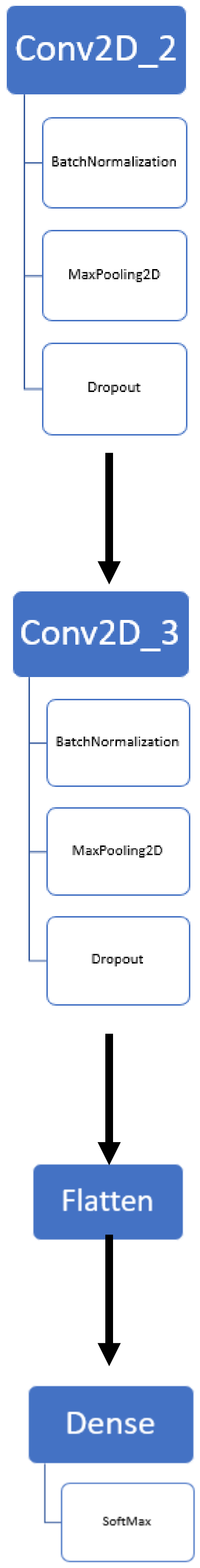
A custom CNN architecture was built for the benchmark model from scratch. The model was trained on the dataset and with the accuracy as an evaluation metric, after loading the best training weights the model yielded a performance of 82% accuracy.





The following graph represent the model's architecture:




Conv2D_2
BatchNormalization
MaxPooling2D
Dropout
Conv2D_3
BatchNormalization
MaxPooling2D
Dropout
Flatten
Dense
SoftMax

III. Methodology

Data Preprocessing

The data processing steps will be as follows:

- Loading The images: the images will be loaded from each class represented by a directory
- Resizing the images: each image will be resized to 224*224 pixels
- Splitting the dataset into training set, validation set and testing set.
- Label Binarizing the Labels
- Normalizing the RGB values of the images to reduce distortions caused by different lightening conditions in the image

Implementation

The project implementation steps were as follows:

1- Understanding and preprocessing the data, as mentioned above in the data preprocessing section.

2- Creating the benchmark model which was a simple CNN model:

  - The architecture was 3 convolution layers with kernel size of (2, 2).
  - MaxPooling2D after each convolution layer with pool size of (2, 2).
  - BatchNormalization and Dropout layers in between each convolution layer.
  - The activation function was “relu”.
  - The final dense layer with “SoftMax” activation function.

- A check pointer that keeps track with the training procedure and saves the best models weights constantly within each epoch in an H5 file.

- The next step was to train the model was batch size of 32 and 25 epochs.

- The benchmark model yielded about 82% accuracy as described above, which is considerably a good result for a benchmark model.

3- Using Transfer Learning to improve the model's performance:

- I tried many pretrained models available in keras application package such as:

  i. VGG16
  ii. ResNet50
  iii. InceptionV3

- The best accuracy came from the InceptionV3 model.

- After importing the model form keras, the final fully connected layer was removed and replaced by a dense layer, with a GAP layer and a Dropout layer.

- It was trained on the preprocessed dataset images with size of 112*112 pixels.

- The obtained validation accuracy was about 94%, which is a good improvement from the benchmark model.

The complications I faced during the implementation process were mainly tuning the parameters for every single model, as each model responded differently for every different set of parameters. The trouble was the time consumed for every trial

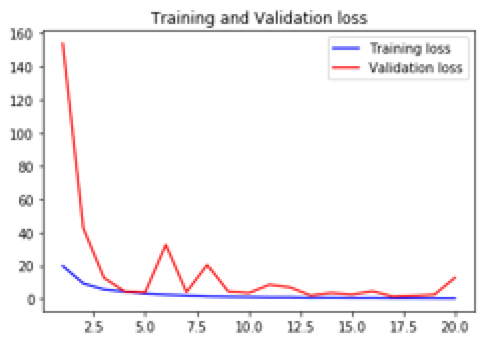
Refinement

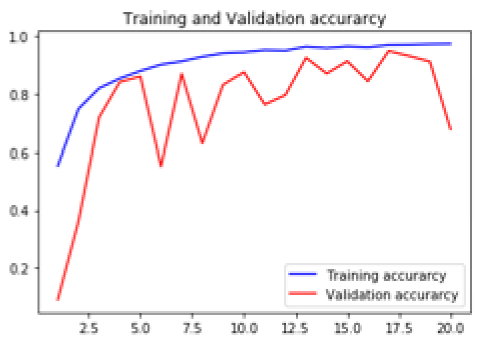
The model development cycle went through different phases that will be discussed:

1. Training the model using a custom CNN architecture designed from scratch, the details of the architecture is drawn above.

    - The model yielded about 82% using “rmsprop” optimizer and the “CategoricalCrossEntropy” loss function, it was trained for 25 epochs.

    - Considering the number of layers and parameters in the model and the accuracy it obtained, the model performed well.

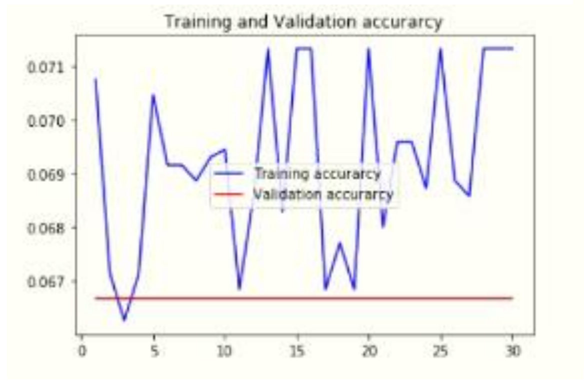


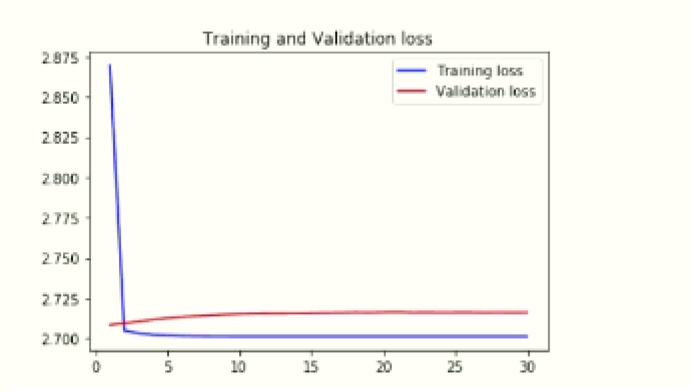


2. The second trial was to harness the transfer learning power to the model. I began with the “VGG16” model with the same tuning parameters of the benchmark model. The model’s performance was surprisingly worse than the simple benchmark model as it’s shown below

    - after tweaking the parameters, the model landed the best possible accuracy out of many trials.

    - The model performance is described by the following accuracy and loss graphs.



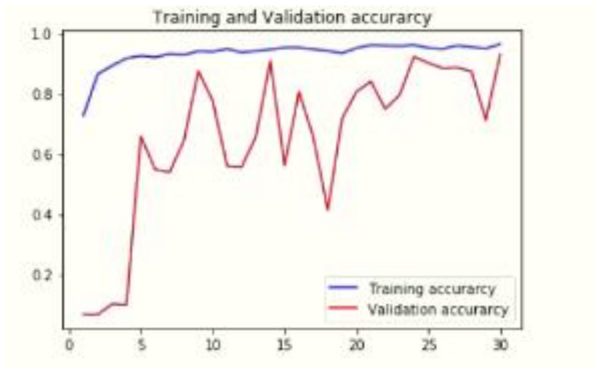


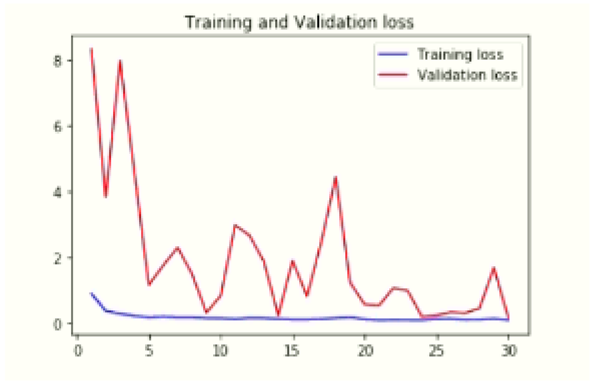
3. The third move was to try different pretrained models, ResNet50 model that was trained in ImageNet dataset was chosen.

    - I tried to change the image size to (224,224) to see it's impact on the accuracy, but it did not change much., except for the longer training time

    - The model was a large improvement from the VGG16 model with a reasonable accuracy of 90% and 0.35 loss

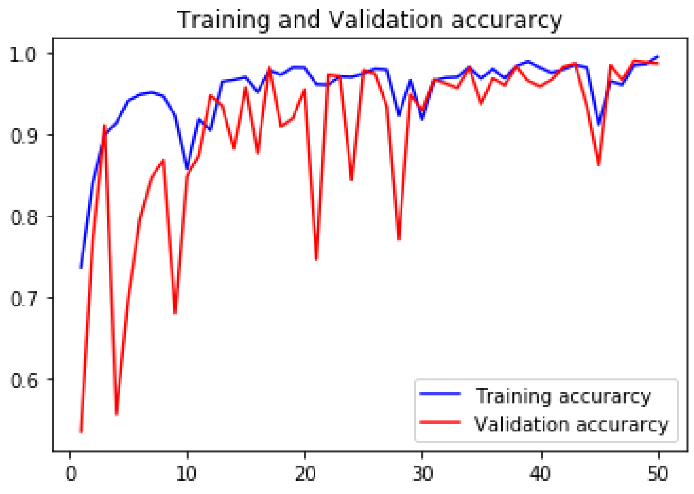
    - The model performance is described by the following accuracy and loss graphs.

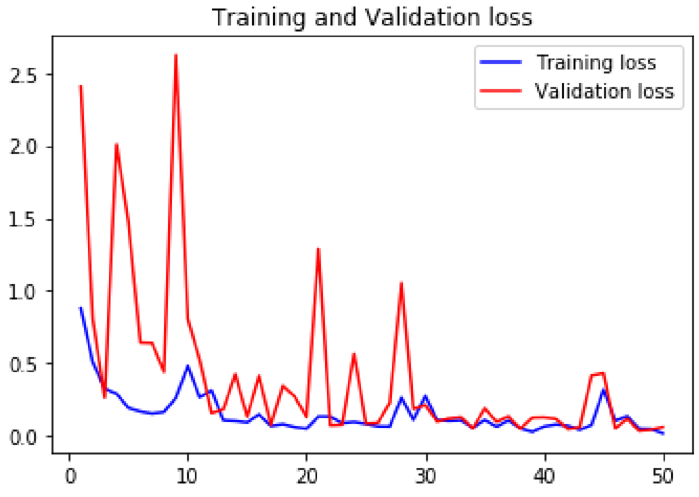




4. At each of the previous trials, I tried hyper-parameters tuning for example I tried multiple optimizers such as ("rmsprop"," sgd") with different learning rates, But the highest accuracy came along with the "adam" optimizer with the default parameters.

5. The final model that gained the best training and validation accuracy was a model that uses transfer learning InceptionV3 model, with the "adam" optimizer and "categoricalcorssentropy" as a loss function, it was trained for about 50 epochs





IV. Results

Model Evaluation and Validation

For evaluating the final model’s performance, the main evaluation metric was the accuracy. The performance was compared with the initial CNN benchmark performance which was about 82% accuracy. The improvement was significant as the model’s accuracy reached about 96.5%. That accuracy was reached after trying and tuning many hyper-parameters and finally settling on the “adam” as an optimizer with the default parameters and “categoricalcorssentropy” loss function.

The models architecture was simply the InceptionV3 model without the final layers of the model that are very specific to dataset that the model was originally trained on, then connect it to a GAP, Dropout and Dense Layer with a specific number of categories equal to the number of the classes in the dataset and train these final layers only

For testing the model’s performance, it was given a set of unseen test images and it performed well predicting them. the following visualization of the prediction of a set of test images demonstrate the model high accuracy as it managed to correctly predict all of the images as we can see in the next section.

Justification

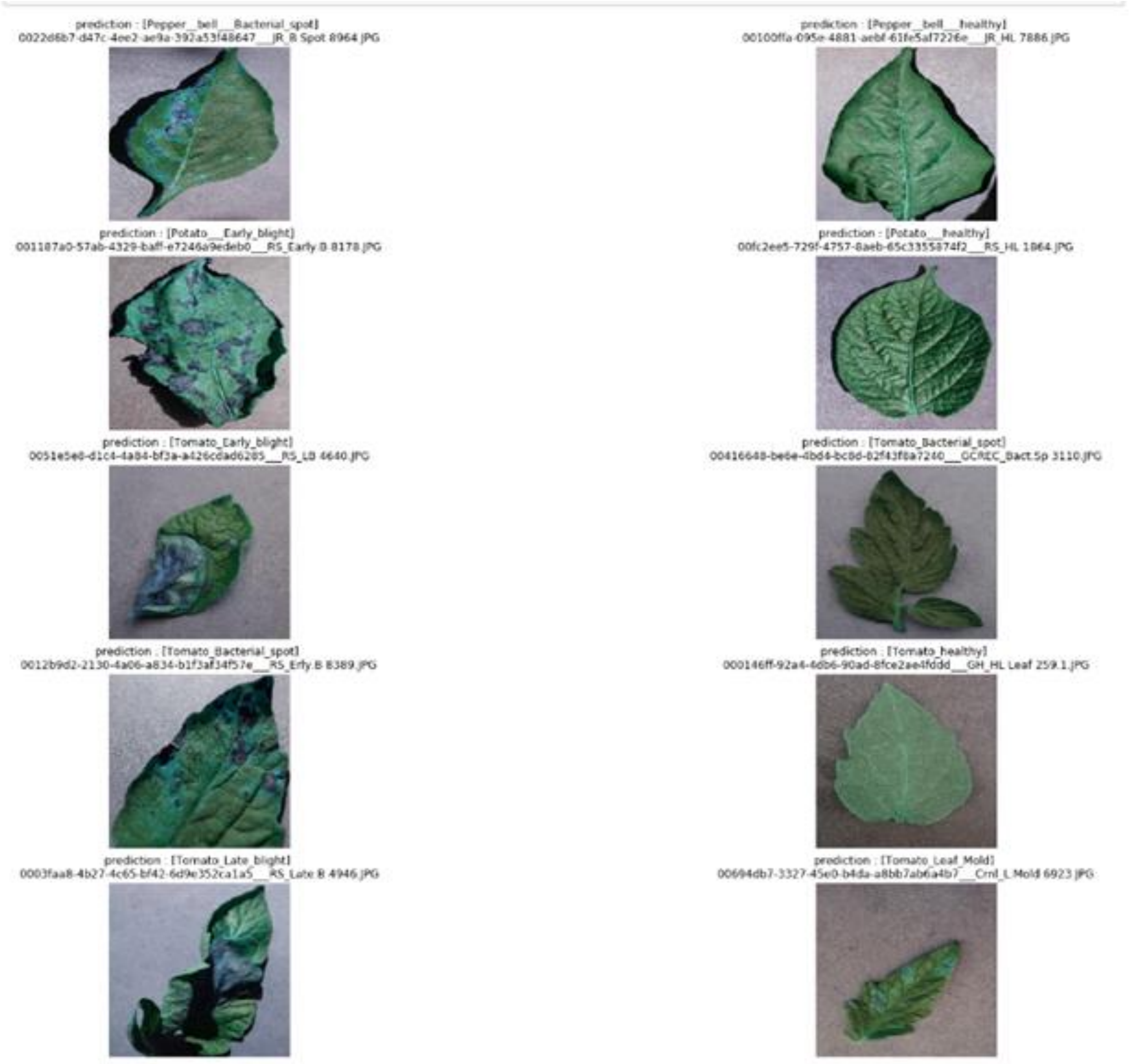
When looking at the development process from the benchmark model to the final model, we can name it a success. The final model landed an accuracy of 96.5%, which is 14% more than the benchmark model’s accuracy. This improvement is remarkable as the benchmark itself was a good model with 82% accuracy.

So, in conclusion, the final model is significant enough to solve the problem

V. Conclusion

Free-Form Visualization

To measure the robustness of the model and it’s performance, the model was tested to predict some images was not used in the original dataset, and the model could identify each image class correctly. Based on the model’s predictions on the images, the model can be trusted as a robust model in comparison with the previous models.


prediction : [Pepper__bell___Bacterial_spot]
0022d6b7-d47c-4ee2-ae9a-392a53f48647___JR_B Spot 8964.JPG
prediction : [Pepper__bell___healthy]
00100ffa-095e-4881-aebf-61fe5af7226e___JR_HL 7886.JPG
prediction : [Potato___Early_blight]
001187a0-57ab-4329-baff-e7246a9edeb0___RS_Early.B 8178.JPG
prediction : [Potato___healthy]
00fc2ee5-729f-4757-8aeb-65c3355874f2___RS_HL 1864.JPG
prediction : [Tomato_Early_blight]
0051e5e8-d1c4-4a84-bf3a-a426cdad6285___RS_LB 4640.JPG
prediction : [Tomato_Bacterial_spot]
00416648-be6e-4bd4-bc8d-82f43f8a7240___GCREC_Bact.Sp 3110.JPG
prediction : [Tomato_Bacterial_spot]
0012b9d2-2130-4a06-a834-b1f3af34f57e___RS_Erly.B 8389.JPG
prediction : [Tomato_healthy]
000146ff-92a4-4db6-90ad-8fce2ae4fddd___GH_HL Leaf 259.1.JPG
prediction : [Tomato_Late_blight]
0003faa8-4b27-4c65-bf42-6d9e352ca1a5___RS_Late.B 4946.JPG
prediction : [Tomato_Leaf_Mold]
00694db7-3327-45e0-b4da-a8bb7ab6a4b7___Crnl_L.Mold 6923.JPG

Reflection

The mindset for this project was to use Machine Learning technology, Deep Learning specifically to solve a real-world problem. The problem was chosen based on it's impact on many countries economically such as my own country, also for many countries that suffer from different situations of malnutrition due to lack of food supply. The development process in conclusion can be the following:

- Focusing on a specific problem as stated above.
- Collecting a related dataset.
- Preprocessing the data to be read for training the model.
- Creating a benchmark model as a reference.
- Training multiple models on the data
- Based on the results and the accuracy we select a robust model as our final model.
- Saving the to be ready for direct deployment and predictions.

The most interesting aspect is the potential of developing an application that can change a lot and have a direct impact on many people, even if it is not the first of it's kind. When it comes to the difficult aspects, it was certainly choosing the right architecture and model to be the final model and handling that amount of data. I believe that the model *fitted my expectations for the problem as it did perform well. It can be used in general settings to solve this problem.*

Improvement

One of the interesting most interesting ideas of this project that it can be used in many ways such as:

- Personal usage:
    - Like a mobile application whether the model is deployed using TensorFlow lite or the model process is run on a server. This can achieve a very wide spread

- For more efficient result it can deployed on a drone:
    - This is an efficient use in my opinion to scan larger areas of the fields, but it will require the model to be trained on more dynamic images and data augmentation will be a great tool in this case.

- Finally, the model can be trained on more data to be able to detect more classes and disease for more efficient usage, as there always room for improvements