AWS Machine Learning Engineer Nano Degree Capstone Project Report

- By Manish Panjwani

Plants Disease Detection Using Deep Learning

1. Definition

1.1. Project Overview

Plant diseases are one of the major factors responsible for substantial losses in yield of plants, leading to huge economic losses[1]. According to a study by the Associated Chambers of Commerce and Industry of India, annual crop losses due to diseases and pest's amount to Rs.50,000 crore ($500 billion) in India alone, which is significant in a country where the farmers are responsible for feeding a population of close to 1.3 billion people [2]. The value of plant science is therefore huge.

Accurate identification and diagnosis of plant diseases are very important in the era of climate change and globalization for food security. Accurate and early identification of plant diseases could help in the prevention of spread of invasive pests/pathogens. In addition, for an efficient and economical management of plant diseases accurate, sensitive and specific diagnosis is necessary.

The growth of GPU's (Graphical Processing Units) has aided academics and business in the advancement of Deep Learning methods, allowing them to explore deeper and more sophisticated Neural Networks[3]. Using concepts of Image Classification and Transfer Learning we could train a Deep Learning model to categorize Plant leaf's images to predict whether the plant is healthy or has any diseases. This could help in the early detection of any diseases in plants and could help take preventive measures to prevent huge crop losses

1.2. Problem Statement

We will be using Deep Learning to categorize plant leaf images to predict whether the plant is healthy or has any diseases. This project will use a plant leaf's images dataset[4] that is based on the original "PlantVillage Dataset"[5]. The original plant disease detection dataset[5] has been used in multiple Deep Learning Contests[7][8]. The task may be stated simply as categorizing the plant disease from the provided photograph of the plant's leaves.

Given the challenge is to categorize whether the plant is diseased or healthy from the plant's images, it's reasonable to conclude that we should utilize a computer to accomplish this operation. As identification of such diseases by human's naked eye is challenging and time consuming as well. Even if someone is competent at this task of plant disease categorization by looking at plant images, doing so for thousands of plant images in a matter of minutes by humans is almost impossible, or at the very least will require a large number of highly competent people for this task, to work in parallel. As a result, automating this type of activity could help in identification of the plant diseases on a large scale. This could further help build other applications like API's or mobile applications that could integrate the Machine Learning model build in this project and leverage it to predict plant disease from the plant photographs.

With the recent increase in interest in home gardening, people could also leverage the model to check the health of their home plants.

We would be focusing only on building a Machine Learning model that will be able to identify whether the plant is healthy or has any diseases based on the input plant's leaf's image, by leveraging Deep Learning concepts and Machine learning tools. The method suggested is to build a classification model that uses a pre-trained Convolutional Neural Network to extract features from photographs, and then leveraging the concepts of Transfer Learning to build a final classification model that has been trained specifically for the task of plant disease detection.

1.3. Evaluation Metrics

Because the dataset that we would be using is more or less balanced, we will be using **Accuracy** as the final criterion to assess the outcomes of our model. For the given problem we have to classify the images into **9 categories** so we will be using **"categorical cross-entropy"** as **loss function** while training our models. We will also be using other evaluation metrics like **Precision**, **Recall** and **F-1 score** for evaluation of the performance of our model. Selection of these metrics was done as these are one of the crucial metrics to be analysed when performing analysis of multi-class classification models.

We have utilized the ResNet50 model as a pre-trained model for our project using the concepts of transfer learning.

Description of selected evaluation metrics will help in further clarify why they have been selected as the evaluation metrics. All the metrics selected for model evaluation are derived from confusion matrix. Below shown is a sample representation of a confusion matrix.

In the below image, TP = True Positives, TN = True Negatives, FP = False Positives & FN =False Negatives.

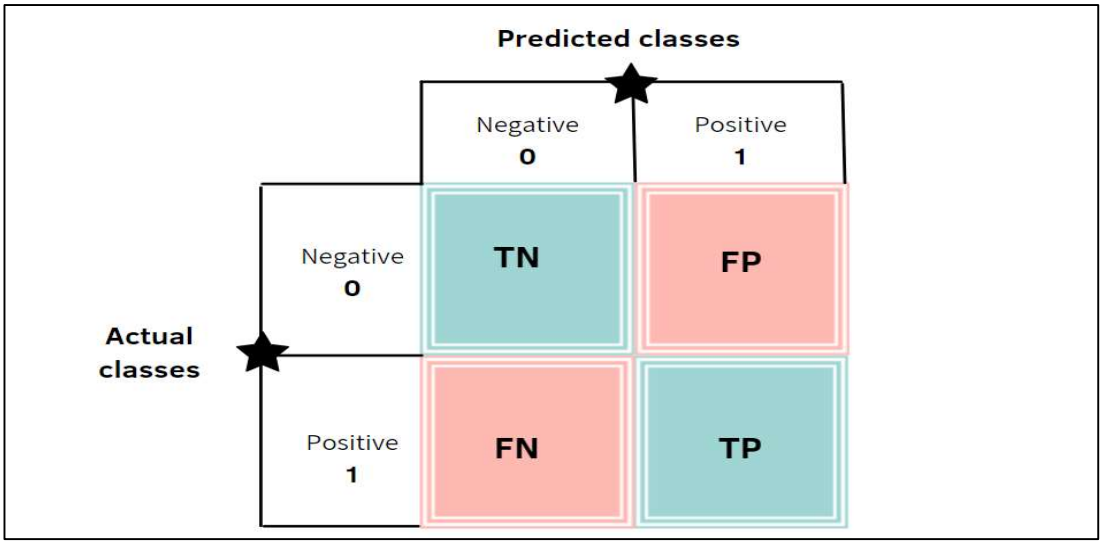

Figure 1 Sample Visual Representation of Confusion Matrix

Precision: Precision is used to evaluate the correctness of the positive identifications made by the model. It can also be defined as a ratio between true positives and total predicted positives. It's also called as Positive Predicted Value.

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall: Recall can be defined as the ratio of actual positives captured by the model to total actual positives. It's also called as True Positive Rate or Sensitivity.

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

F-1 Score: F-1 score is a measure to attain balance between Precision and Recall

$$F1 = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Accuracy: It's an ratio between total correct predictions to total predictions made by the model, Given by

$$Accuracy = \frac{True\ Positives + True\ Negative}{True\ Positives + False\ Positives + True\ Positives + True\ Negative}$$

Categorical cross-entropy: This loss functions is used for multi class identification tasks where the model has to identify a single category out of many. The function calculates the difference between two probability distributions.

$$Loss = \sum_{i=1}^{Output\ Size} y_i . log\ \hat{y}_i$$

In the above formula $\hat{y}_i$ is the i-th scaler value of the model output and yi is the corresponding target value.

2. Analysis

2.1. Data Exploration

We will be using the plant disease detection dataset[4] that is based on the original "PlantVillage Dataset"[5]. The dataset consists of 39 different classes, where 38 classes are of plant leaf categorised according to whether they are diseased or healthy and remaining one class consists of images with Background without plant leaves. The provided modified dataset[4] that we will be using has been created using six different augmentation techniques on the original dataset[5] for increasing the data-set size. The techniques are image flipping,

Gamma correction, noise injection, PCA colour augmentation, rotation, and scaling. The dataset that we will be using contains 61,486 images.

All the 39 classes are :

1. Apple_scab, 2. Apple_black_rot, 3. Apple_cedar_apple_rust, 4. Apple_healthy, 5. Background_without_leaves, 6. Blueberry_healthy, 7. Cherry_powdery_mildew, 8. Cherry_healthy, 9. Corn_gray_leaf_spot, 10. Corn_common_rust, 11. Corn_northern_leaf_blight, 12. Corn_healthy, 13. Grape_black_rot, 14. Grape_black_measles, 15. Grape_leaf_blight, 16. Grape_healthy, 17. Orange_haunglongbing, 18. Peach_bacterial_spot, 19. Peach_healthy, 20. Pepper_bacterial_spot, 21. Pepper_healthy, 22. Potato_early_blight, 23. Potato_healthy, 24. Potato_late_blight, 25. Raspberry_healthy, 26. Soybean_healthy, 27. Squash_powdery_mildew, 28. Strawberry_healthy, 29. Strawberry_leaf_scorch, 30. Tomato_bacterial_spot, 31. Tomato_early_blight, 32. Tomato_healthy, 33. Tomato_late_blight, 34. Tomato_leaf_mold, 35. Tomato_septoria_leaf_spot, 36. Tomato_spider_mites_two-spotted_spider_mite, 37. Tomato_target_spot, 38. Tomato_mosaic_virus, 39. Tomato_yellow_leaf_curl_virus

Given the dataset is huge and has 39 categories, for the purpose of this project we will be using only a subset of the plant classes for our project. We will only be using the classes:

Sr No.	Plant Image Class Name
1.	Cherry_healthy
2.	Cherry_powdery_mildew
3.	Pepper_bacterial_spot
4.	Pepper_healthy
5.	Potato_early_blight
6.	Potato_healthy
7.	Potato_late_blight
8.	Strawberry_healthy
9.	Strawberry_leaf_scorch

The provided modified dataset[4] from which we have selected this subset dataset of 9 classes has already been created using six different augmentation techniques on the original dataset[5] for increasing the data-set size. The techniques are image flipping, Gamma correction, noise injection, PCA colour augmentation, rotation, and scaling. Thus no major data pre-processing steps would be required for this dataset that we have utilized for this project. Further discussion on the data pre-processing has been done the Data pre-processing section.

2.2. Exploratory Visualization

Sample images of the 38 categories of the plant leaves from the complete dataset for exploratory visualization are shown below.

Figure 2 Visual Representation of samples of Plant Village Dataset. Image credits[7]

Given we will only be focusing on a subset of the plant classes for our project. Below is Tabular representation of the dataset with dataset size statistics.

Sr No.	Plant Image Class Name	Class Image dataset size
1.	Cherry_healthy	1001 images
2.	Cherry_powdery_mildew	1053 images
3.	Pepper_bacterial_spot	1000 images
4.	Pepper_healthy	1478 images
5.	Potato_early_blight	1000 images
6.	Potato_healthy	1001 images
7.	Potato_late_blight	1000 images
8.	Strawberry_healthy	1001 images
9.	Strawberry_leaf_scorch	1110 images

Let's look at a bar graph representation of the above dataset, to further understand the dataset distribution.

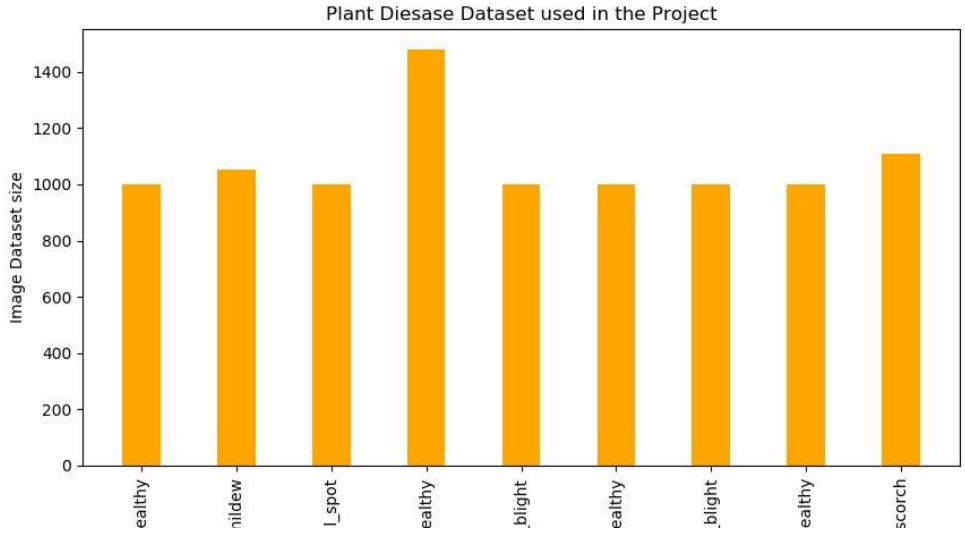

Figure 3 Plant Classes Data Distribution Bar graph

As could be seen from the above tabular representation, all the 9 classes that we will be using have images in more or less equal proportions. None of the classes seems to be under-sampled. However, we do see that "pepper_healthy" class has more images as compared to the other classes.

Let's look at a pie chart representation of the dataset for one more visual representation of the dataset distribution.

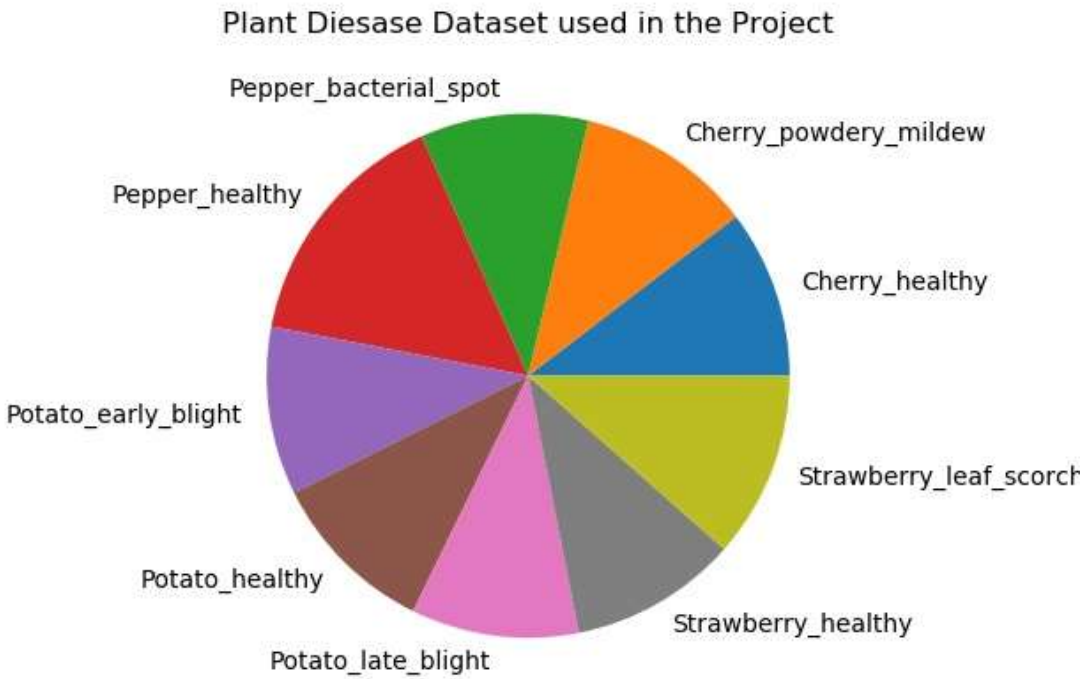

Figure 4 Plant Classes Data Distribution Pie chart

As could be seen from the above pie chart as well, the distribution of the dataset is proportionate for each of the classes. Thus we would say that the dataset is balanced.

The total dataset of that will be used for this project will be **9644 images**. All the images vary in dimensions, they are not standardized, and they are all coloured images. The model will be trained on the above 9 plant image classes, for our use-case. Given we will be using Resnet50 model that is trained on ImageNet dataset, we will be using its mean and standard deviation for normalizing our images.

2.3. Algorithms And Techniques

This research employed a Convolutional Neural Network (CNN), which is a cutting-edge architecture for solving problems like pattern recognition in images. CNN is a type of Neural Network that is typically used to solve visual imaging difficulties. CNN excels at this task in part because of its spatial invariant ability, which allows it to detect patterns that may be translated, moved, or distorted. This spatial invariant ability, which enables CNN to perform so well for pictures, can be attributed to the convolution layer's capacity to recognize high-level features of an image, such as edges, curves, straight edges, and colour. To do this, the convolution layer employs filters (also known as convolution kernels), which slide across the whole picture from top to bottom, delivering the input to the activation layer and resulting in a 2-dimensional activation map.

One of the methodologies used in this research is called Transfer Learning, which is approximately equivalent to taking a portion of a previously learned (weights) Neural Network and utilizing it in a new one while training the remainder for a new task. The Convolutional layers were used as a feature extractor to precisely identify the high-level features of an image, such as edges, curves, straight edges, and colour, which were already described about CNNs.

The Transfer Learning technique has the advantage of re-purposing a prior and costly effort that was utilized to create these Convolutional layers. They are costly to construct from the ground up. Even on strong and specialized hardware, the best known CNNs taught to detect a wide range of objects from photos take several hours, days, or even weeks to train. So, the model developed for this project that employs the Transfer Learning technique was far less expensive to train, yet it still works admirably if it is fed a decent pre-trained model.

Success of the model trained in this project can be quantified based on the Accuracy of the model achieved during its validation/testing phase. Along with Accuracy we could calculate other evaluation metrics such as Precision, Recall and F1-score to determine the performance of the model and compare it with the known benchmark model's performance. We would also consider and analyse the trend seen in the loss functions used in the model to give us a sense of the deviation of the model predictions from reality/expected outcomes [10]. Another measure of success of the model will be to identify how well the model performs on real world test images taken and given as input to the model.

2.4. Benchmark

A lot of research has already been done on implementing Deep learning models on the Plant Village dataset[5]. So we will be using outcomes of models from two specific research papers as a benchmark for our project.

First paper titled "Using deep learning for image-based plant disease detection."[11] uses a couple of variation of two models "AlexNet" and "GoogLeNet". Here the original dataset[5] of size 54,306 images without any data augmentation was used for training and testing on all the 38 plant classes. The training was performed on colour, greyscale and Segmented plant images. As mentioned in the research paper the GoogLeNet model was able to achieve an accuracy of 99.35% on the test set for the coloured images dataset after 30 epochs.

Second paper titled "Plant Disease Detection from Images"[12] uses a small subset of the same "PlantVillage Dataset" with data augmentation. It only used 4000 images belonging to "Tomato", "Potato" and "Pepper" plant classes for training and testing purposes. ResNet-34 and ResNet-50 pre-trained models were employed for training the model. ResNet-50 was able to outperform ResNet-34 and achieve an accuracy of 99.4% on the test set after 4-5 epochs only. But the dataset used in the research paper is similar but not identical to the dataset that we will be using for our training and testing purposes.

Thus a good outcome in our case would be in achieving an Accuracy of more than **80**% on the train and validation sets and getting a similar F-1 score of around **0.80-0.99** for the training and validation sets.

Given, for the current project we will be using a subset of coloured plant images dataset of only 9 classes (9644 images) from the version of the Plant Village Dataset with data augmentation[4] for our project's training and testing purposes. We will also be using a ResNet-50 pre-trained model for our project. One of the goal of the project will be to check how well the model trained only on the 9 classes and limited resources performs as compared to other known model's outcomes.

Thus, we will be using the outcomes of second paper[12] as benchmark for our project's model. We will also be comparing the outcome of our model with the outcomes of the first paper[11] and mentioning all our observations in the final project report.

3. Methodology

3.1. Data Pre-processing

The provided modified dataset[4] from which we have selected this subset dataset of **9 classes** has **already been created** using **six different augmentation techniques** on the **original dataset**[5] for increasing the data-set size. The techniques are image flipping, Gamma correction, noise injection, PCA colour augmentation, rotation, and scaling. Thus, we will not be performing any data augmentation techniques on this dataset as those have already been performed on the dataset that we will be using.

One important point to be noted is that the dataset that we have used is already a **balanced dataset**, as we have more or less equal number of images available for each of the 9 classes. Thus we do not need to perform any of the over-sampling techniques. This is a huge benefit that we get from a balanced dataset.

However, we will be performing a few transformations on the image dataset while training our model. Transformations that we will be utilizing will include Image Normalization, Resizing, Random Flips to make sure the trained model is not biased on the training dataset.

The subset dataset for these 9 classes has been downloaded from the website [4], and it will be provided (as part of this project) either as a zipped version or uploaded to the S3 and made public, where the link of S3 url will be mentioned in the Readme of the git repository. The dataset was split into trained set, validation set and test set. The proportion of the split into train, validation and test set is 80% for training, 10% for validations and 10% for testing.

3.2. Implementation

In this section, we will be discussing the implementation approach for fine-tuning a pretrained CNN model for using it for image classification using the plant disease image dataset.

3.2.1. Transfer learning – Using pre-trained model

For the purpose of this project, we used a pre-trained ResNet-50 model to extract the bottleneck features from the images. ResNet-50[9] model is 50 layers deep and is trained on a million images of 1000 categories from the ImageNet database. Further-more the model has a

lot of trainable parameters, which indicates a deep architecture that makes it better for image recognition. The input images was be resized to 244x244 and bottleneck features were extracted. We applied some transformations like crop images, image rotations and flips and normalization on the images. The classification model was be created and prepared to consume as an input, the features extracted using the pre-trained ResNet-50 CNN model. A fully-connected layer with Soft-max as activation function of the output layer was added. Once we have a model trained, a typical and simplified structure design of an algorithm to perform the classification is shown below:

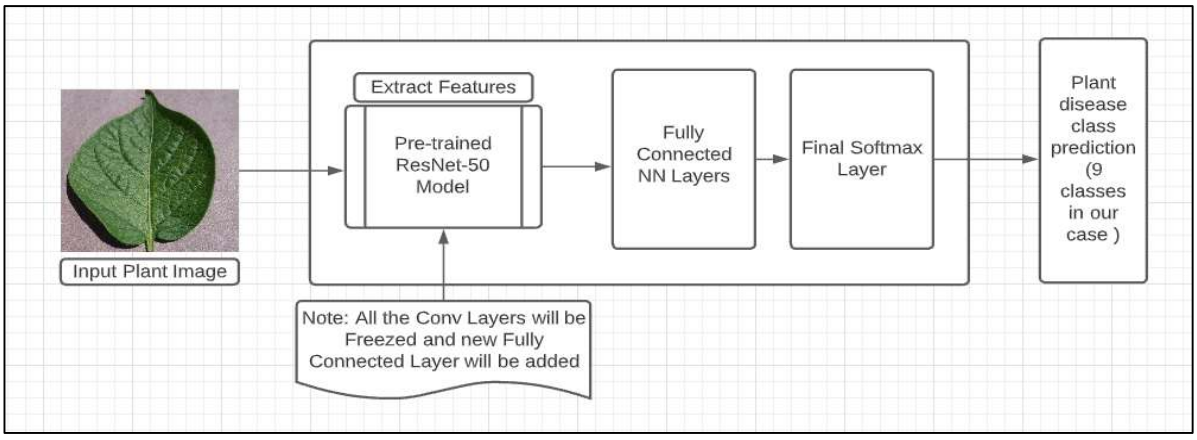

Figure 5 Simplified Design of Finetuning Pre-trained Model

3.3. Refinement

We started by trying to utilized the pre-trained Resnet50 model and use it with default parameter values for training the model on the plant dataset that we have selected. However , the initial model performed very poorly.

3.3.1. Training the ML Model with default params

Training the model on the Default Hyperparameter configurations gave us an accuracy of 11% on the test dataset, which is very low.

Default Hyperparameter Configs were:

* Epochs = 2

* Learning Rate: 0.01

* Batch Size: 64

These are pretty low values, so fine tuning the ML model is an essential step. Thus, we would definitely need to find some different values of the hyperparameter to increase the model's accuracy and performance. Let's look below if using the hyperparameter tuning helps increasing

3.3.2. Refinement - Fine-tuning ML Model

For the pre-trained ML model, we could perform a couple of steps to further fine-tune the models.

One way to fine-tune the model will be the selection of the Optimizer used while training the models. For example, we could try switching between different optimizers and check the model performance, like switching from Adam to AdamW, SGD or RMSprop.

Other way would be to perform tuning of the hyper-parameters used in the optimizer and training algorithm, by checking the model performance by using different permutations and combinations of values of hyper-parameters like learning rate, batch size and momentum. We could also leverage the AWS Sagemaker's Tuner for performing hyperparameter tuning for our both the models while training phase.

We selected a few hyperparameters for Hyperparameter tuning, the hyperparameters were, learning rate, batch size and epochs. Post the Hyperparameter tuning, we were able to find a configuration that yielded an better accuracy on the test dataset. Best values for the hyperparameters were: **{'batch_size': 128, 'lr': '0.0003352272978256416', 'epochs': '10'}**

Further discussion of the final model has been done in the Results section below.

4. Results

4.1. Model Evaluation and Validation

The final model was trained using the best hyperparameters that we got from the hyperparameter tuning step, as mentioned earlier. Using these parameters we trained our model and logged in a couple of important metrics of the model like the loss criterion and other evaluation metrics.

Below is a graph of the Cross Entropy Loss criterion calculated while the model was in the training and validation phases. As could be seen in the below graph, the blue line signifies the loss at each step while the mode was in the training phase and the orange line signifies the loss at each step while the model was in the validation phase.

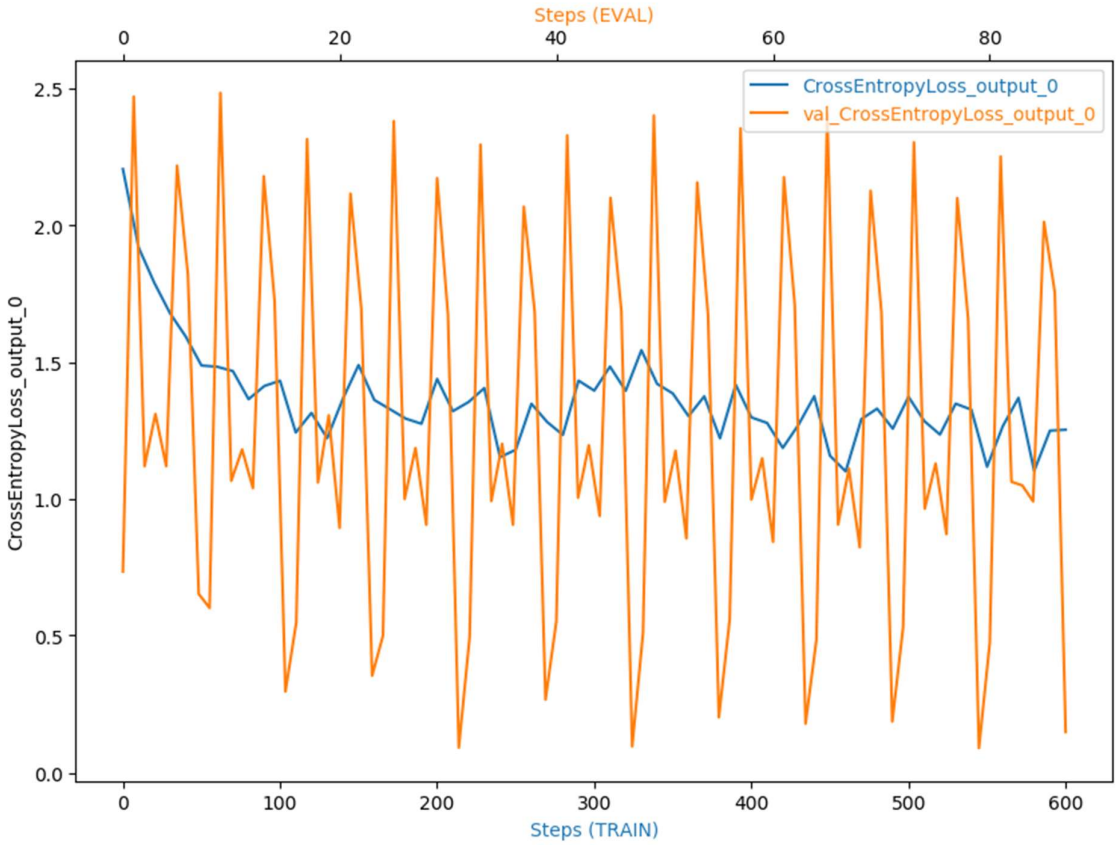

Figure 6 CrossEntrophy Loss Graph of Final Training Job's Train and Validation steps

As could be seen from the above graph the loss for the training phase could be seen decreasing as the number of steps are increasing which is a very good sign and would mean that the model is indeed learning with more and more iterations.

However, we could also see a huge spike in the line-graph of the loss for the validation steps, which again seems periodic in nature, as it quickly declines as well. The final loss at the validation phase could be seen to be much lesser then even the loss that was incurred in the training phase, which again is a good sign. Let's look at the Confusion Matrix for final validation and testing phases of the model. We will be looking at two set of Confusion matrix for each phase: 1. Normal Confusion Matrix, 2. Confusion Matrix with Normalized values.

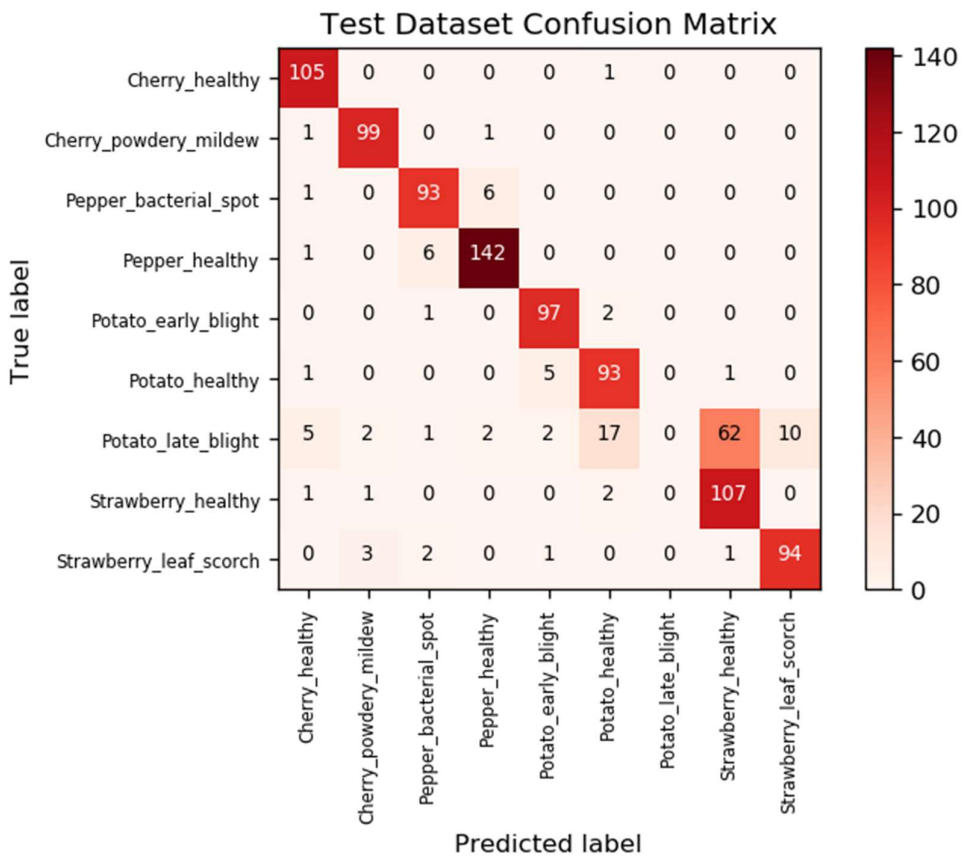

Figure 7

Confusion Matrix of Test Dataset

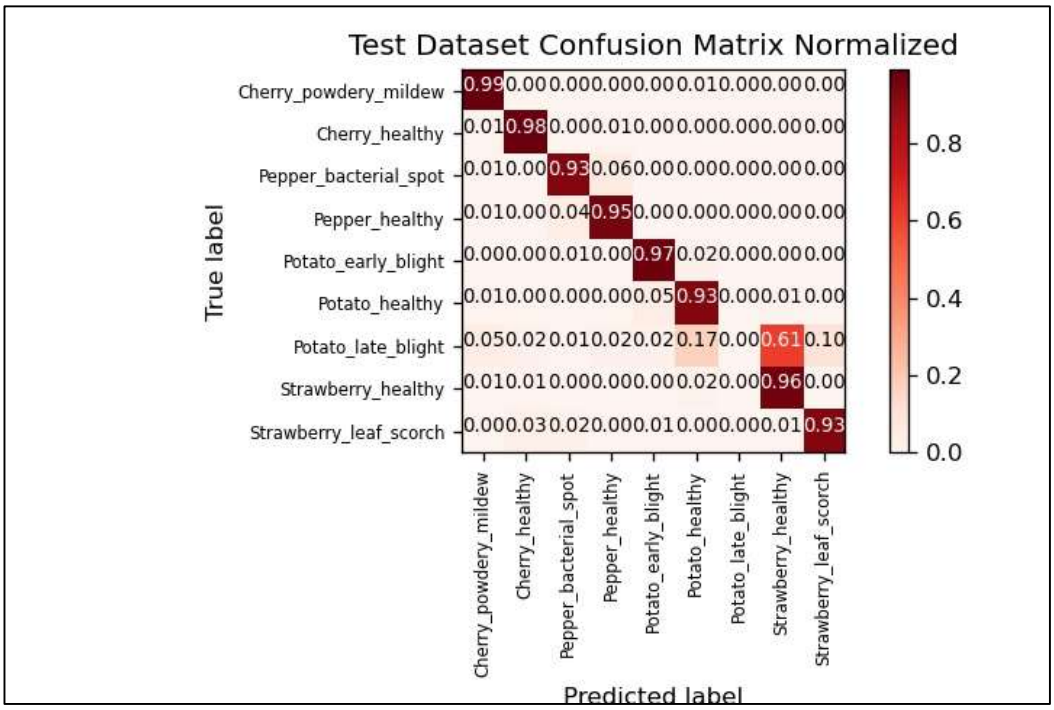

Figure 8 Confusion Matrix of Test Dataset Normalized

Looking at the Confusion Matrix of the testing phase, seems like the model is performing exceptionally well for most of the classes from the overall 9 classes. We see a few incorrect predictions for each class, but overall the model is performing sufficiently well for most classes. However the model seem's to be performing poorly for one of the classes named "Potato_late_blight" where most of it's values are getting incorrectly predicted to some other class. This would be one of the reason for a lower accuracy for our model.

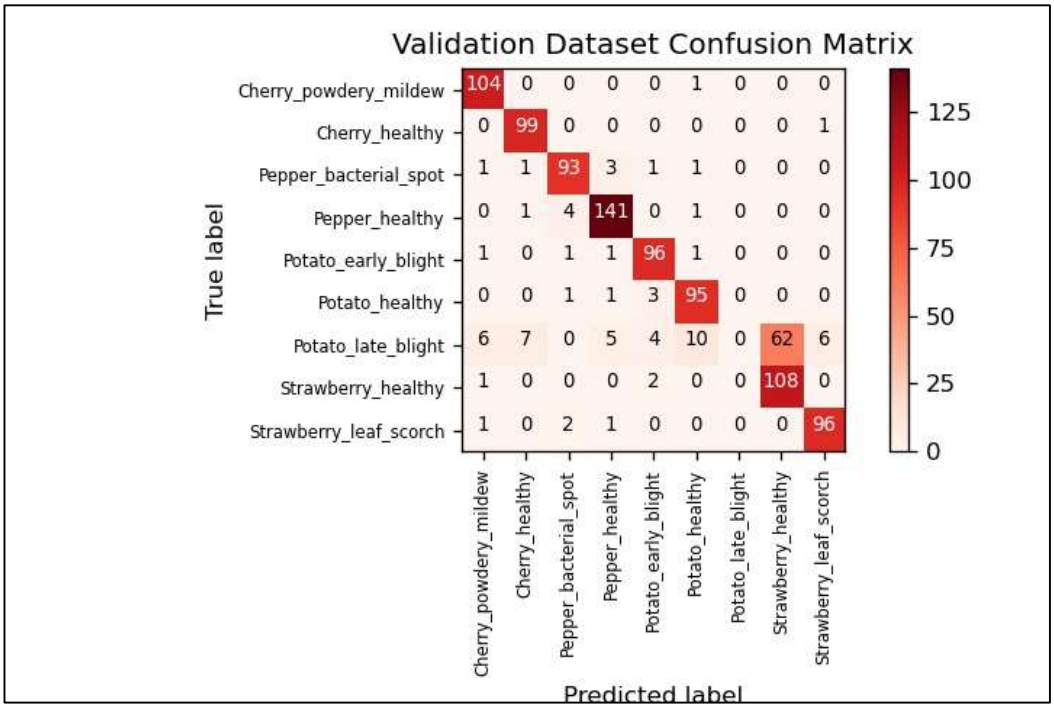

Figure 9 Confusion Matrix of Validation Dataset

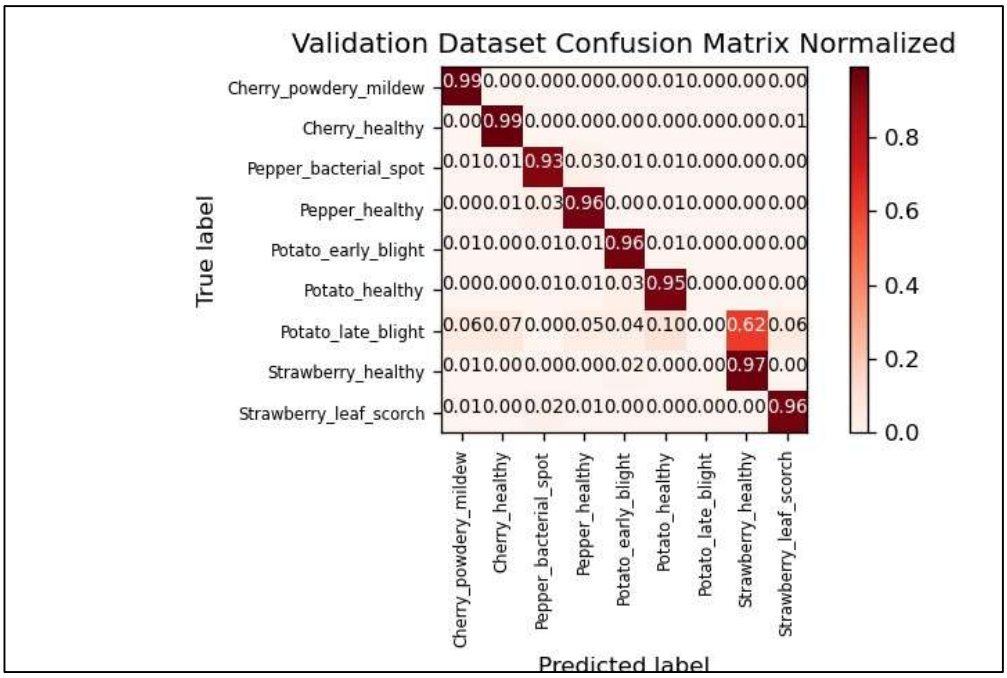

Figure 10 Confusion Matrix of Validation Dataset Normalized

Now, Looking at the Confusion Matrix of the validation phase, we see a similar trend as was seen in the confusion matrix of the testing phase, where seems like the model is performing exceptionally well for most of the classes from the overall 9 classes. We see a few incorrect predictions for each class, but over-all the model is performing sufficiently well for most classes. However here as well, the model seems to be performing poorly for one of the classes named “Potato_late_blight”

where most of the values are getting incorrectly predicted to some other class. This would be one of the reason for a lower accuracy for our model.

Let's look at other metrics of our model and compare it with similar metrics from the benchmark papers.

It could be seen from the above metrics that the model was able to achieve an overall accuracy on the training and validation datasets, which seems to be a pretty reasonable accuracy.

4.2. Justification

Comparing these metrics with the benchmark models shows that the model seems to be performing good, but not as much as the models in the research papers have performed.

First benchmark paper titled "Using deep learning for image-based plant disease detection."[11] uses a couple of variation of two models "AlexNet" and "GoogLeNet". Here the original dataset[5] of size 54,306 images without any data augmentation was used for training and testing on all the 38 plant classes. The training was performed on colour, greyscale and Segmented plant images. As mentioned in the research paper the GoogLeNet model was able to achieve an accuracy of 99.35% on the test set for the coloured images dataset after 30 epochs. So comparing the test results of our model, which was trained using a ResNet-50 model on a dataset of 9644 images of 9 classes and only for 5 epochs , our model as performed sufficiently well.

Second benchmark paper titled "Plant Disease Detection from Images"[12] uses a small subset of the same "PlantVillage Dataset" with data augmentation. It only used 4000 images belonging to "Tomato", "Potato" and "Pepper" plant classes for training and testing purposes. It does not use the dataset that was used in this project. ResNet-34 and ResNet-50 pre-trained models were employed for training the model. ResNet-50 was able to outperform ResNet-34 and achieve an accuracy of 99.4% on the test set after 4-5 epochs only.

Comparing the output of our model with the outcomes of the second benchmark does seems like our model did perform well, however there would be still some room for improvement. However, while focusing on increasing the accuracy the most important thing would be to avoid over-fitting on the training data.

However, one thing to be noted was that our model performed significantly well for most of the classes in the dataset, but was not able to predict correctly one of the class named "Potato_late_blight" where the model performed poorly. One of the reason for this could be that the model was not able to extract significant features to distinguish between this class and the other classes. Other reason would be to get more distinct images belonging to this category and increase the dataset size of the "potato_late_blight" category. However, this is not an easy task to achieve. Given the model did perform significantly well for the other 8 classes , we should be good.

5. Conclusion

5.1. Free-Form Visualization

The final model, which used transfer learning techniques to detect plant category/disease in photographs from the internet, was able to do so. As seen in the below images, it successfully classified the 3 out of the 4 random images that were tested.

```
Test image no: 1
Below is the image that we will be testing:
```

```
Expected plant category no : 4
Response: [[1.3045936822891235, 0.0, 0.0, 1.6936994791030884, 0.0, 0.0, 0.0, 0.0, 0.0]]
Response/Inference for the above image is : [4]
Test image no: 2
```

Figure 11 Test images sample num 1

As could be seen from the above image, we tried to test our model with the above test image of a plant belonging to category no : 1, that is “cherry_healthy” and the mode was able to successfully predict the same. Response from the model consists of an array of length 9 (9 classes/categories) , and the values at each position in the array signifies the confidence level for that category from the model. Position with max confidence value will be the output class/category from the model, which in this case was correctly class 1 (cherry_healthy).

Similarly, three more samples images were tested and their outputs could be seen below.

```
Below is the image that we will be testing:
```

```
Expected plant category no : 4
Response: [[2.73303484916687, 0.0, 0.0, 5.952653408050537, 0.0, 4.163468360900879, 0.0, 0.0, 0.0]]
Response/Inference for the above image is : [4]
Test image no: 3
```

Figure 12 Test images sample num 2

Figure 13 Test images sample num 3

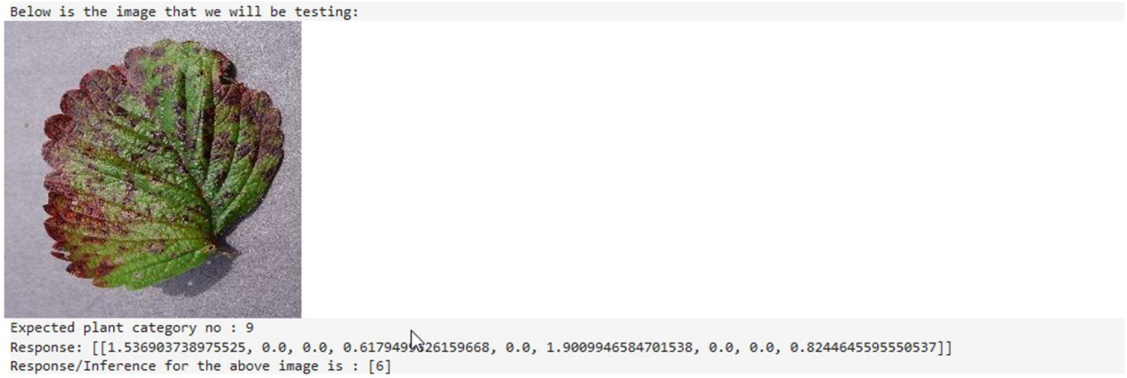

Figure 14 Test images sample num 4

5.2.Reflection

The entire process of building this end-to-end solution using AWS resources was:

• Find an interesting problem to solve and a publicly available data.

• Prepare the dataset, which in our case involved of selecting a sub class of the dataset

• Explore the dataset and check for any imbalance in dataset across classes and check if any data augmentation is required

• Create an initial ml model using a pre-trained resnet50 model by applying transfer learning concepts and default hyper parameter values

• Perform refinement of the model by finetuning the hyperparameters. Select the hyperparameter to finetune and using Hyperparameter Tuning find the best set of hyperparameters.

• Create and train the final model using the best hyperparameters

• Perform logging of Accuracy and also other Evaluation metrics like Precision, Recall, F1-Score and Confusion Matrix in the training script for post training analysis.

• Prepare to use the trained models to classify unseen images from internet.

The most difficult component was creating the models and doing adequate training. Initially, this was due to the large amount of resources required, as well as the fact that the model was often too sophisticated or too basic to discover patterns in data. Training, regulating the overfit, and convergence were equally difficult at initially, and it is a never-ending process. However, once a model begins to converge, boosting accuracy and minimizing loss, it is only a question of time and fine tuning to improve it. However, you could always make improvements to the model. But given there isn't a clear end, you'll have to find out when it's good enough for your needs. For my case getting an accuracy close to the benchmark paper models without overfitting on the training data would be the most important thing.

With an accuracy of even 86% the trained model would be sufficient for getting deployed into integrated applications for disease detection of plants using photographs of the plant leaves.

5.3. Improvements

There are a few things that might be done to better the project as a whole. One, in my opinion, is to utilize a more powerful machine, which would help in more memory utilization and therefore allow for the inclusion of more photos for training. Also a high powerful machine would mean that one could train the model for more number of epochs, which would help further increase the accuracy of the model. Unfortunately, I have used up most of my AWS credits attributed for this project and so was refraining from adding in more epochs for training my final model and sticked with the 5 epoch value that had given in the best accuracy while hyperparameter tuning. Another thing that I would do is dive deep into why the model was not able to predict the “Potato_late_blight” category efficiently. Another thing would be to include some more visualizations for my project if given more time and AWS credits.

I wanted to try out other pre-trained model’s like ResNet-34, InceptionV3 models, etc but could not find enough time to work on the same. Also due to the fact that changing the pre-trained model to InceptionV3, resulted in several unanticipated changes, as well as the appearance of faults, as the version of some of the libraries that were used in the project were incompatible. It's difficult to live on the ledge. To prevent sabotaging the entire effort, I refrained from training a model on the InceptionV3 model and kept working as per my Project proposal to train the model using a pre-trained ResNet-50 model.

6. References

[1] Balodi, Rekha & Bisht, Sunaina & Ghatak, Abhijeet & Rao, K.H.. (2017). Plant Disease Diagnosis: Technological Advancements and Challenges. Indian Phytopathology. 70. 275-281. 10.24838/ip.2017.v70.i3.72487.
[2] https://croplife.org/news/keeping-indias-pests-in-line/

[3] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet classification with deep convolutional neural networks." Advances in neural information processing systems.
[4] J, ARUN PANDIAN; GOPAL, GEETHARAMANI (2019), “Data for: Identification of Plant Leaf Diseases Using a 9-layer Deep Convolutional Neural Network”, Mendeley Data, V1, doi: 10.17632/tywbtsjrjv.1 (https://data.mendeley.com/datasets/tywbtsjrjv/1)
[5] https://paperswithcode.com/dataset/plantvillage
[6] https://github.com/spMohanty/PlantVillage-Dataset
[7] https://www.crowdai.org/challenges/1
[8] https://www.kaggle.com/c/plant-pathology-2020-fgvc7/overview
[9] He, Kaiming & Zhang, Xiangyu & Ren, Shaoqing & Sun, Jian. (2016). Deep Residual Learning for Image Recognition. 770-778. 10.1109/CVPR.2016.90.
[10] https://dzone.com/articles/what-kpis-measure-the-success-of-an-ai-project
[11] Mohanty, Sharada P., David P. Hughes, and Marcel Salathé. "Using deep learning for image-based plant disease detection." Frontiers in plant science 7 (2016): 1419.
[12] Anjaneya Teja Sarma Kalvakolanu, “Plant Disease Detection from Images” (https://arxiv.org/abs/2003.05379)