Plant Disease Classification using Machine Learning techniques

Aesha Sandeep Shah
ashah116@asu.edu

***Abstract*—Pest and disease invasion are the two main causes of crops being partially or completely destroyed, which significantly affects crop productivity and ultimately results in food poverty. Additionally, the majority of underdeveloped nations have very little awareness of such diseases and pest management or control. One of the few important reasons that contributes to food shortage is the presence of toxic pollutants, inadequate disease control, and harsh climate change. Diseases of plants are physiological anomalies. When a plant becomes afflicted with a disease, it will start to show particular symptoms. These signs include physical characteristics that gradually change and become observable to the unaided eye. Some symptoms include wilt leaf spots, yellow patches, blight, and rot. Our main aim is to use machine learning and image augmentation techniques to identify the infected plant and the corresponding disease by observing its morphology.**

Keywords— Machine Learning, Convolutional Neural Network, ResNet50

I. Introduction

Thanks to modern technology, it is now possible to produce enough food to meet demand on a global scale. But a variety of problems—including plant diseases, climate change, etc.—remain a threat to food security. Accurately identifying plant diseases continues to be a challenge for farmers.A plant that is afflicted displays several obvious illness symptoms, such as form, size, dryness, and wilting. These are very useful for figuring out how healthy the plant is. Deep learning's most recent developments in computer vision have made it possible to diagnose plant illnesses.

In this project, we will apply image processing instead of human intervention to detect plant diseases, which will save time and effort. It will provide a low-cost alternative to disease control strategies that could otherwise waste resources and prevent further plant losses. The farmer will find it easier to monitor symptoms and identify ill or unhealthy plants on the farm using our method. As a result, plant disease will be diagnosed. Our major goal is to create a model that recognizes a leaf in a photograph and then classify it in accordance with the appropriate disease category.

The implemented baseline model relies on the idea of transfer learning. Different image processing algorithms are utilized for plant weight calculation and disease diagnosis. In order to change the weight of images that are kept in a learning database, back propagation is used. In order to enhance the plant disease identification methodology, we tested out two other experiments by adding image augmentation in one and further modifying the architecture of our model for the second experiment.

We observe and analyze the performance and results of each of these models and compare them and conclude which of these experiments were successful and which ones failed.

II. Description Of Solution

A. Dataset

For carrying out this experiment, we use the PlantVillage dataset [4]. It contains healthy and diseased plant images of Pepper Bell, Potato and Tomatoes. We have segregated healthy plants and also separate classes for different kinds of some common diseases for each plant.

We download this dataset from Kaggle. It is divided into 15 different classes each containing images of healthy and diseased leaves captured in a controlled environment. There are a total of around 20,600 images of resolution fixed at 224 x 224 pixels.

B. Baseline Model and Evaluation Metrics

The current baseline model uses the concept of Transfer Learning. Transfer learning (TL) concentrates on preserving knowledge obtained while solving one problem and applying it to another that is unrelated but nonetheless similar. [2] The baseline model makes use of a pretrained ResNet50 model with 48 Convolution Layers, 1 MaxPool Layer, and 1 AveragePool Layer. With an input size of 224 × 224 x 3, we feed the image into its array representation. In order to use ResNet-50's effective categorization in our model, we also retrained the model and added more layers at the end. The architecture of our base model is shown in the following diagram..-

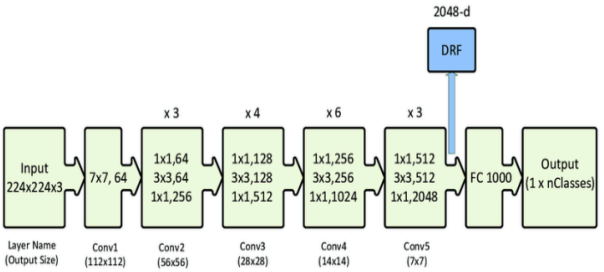

Fig.1. Baseline Model Architecture

The baseline neural network architecture consists of several layers and each of these layers play a significant role. We summarize each layer of the model as follows -

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 7, 7, 2048)        23587712

 flatten (Flatten)           (None, 100352)            0

 dense (Dense)               (None, 512)               51380736

 dropout (Dropout)           (None, 512)               0

 dense_1 (Dense)             (None, 15)                7695

=================================================================
Total params: 74,976,143
Trainable params: 52,443,151
Non-trainable params: 22,532,992
_________________________________________________________________
```

Table 2. Baseline Model Layer Summary

As indicated below, each of these layers and their significance can be understood -

1) ResNet50

The pretrained model ResNet is the short for Residual Network. It supports Residual Learning. The 50 in ResNet50 also denotes the 50 layers of the network. The ResNet50 network, one of the most widely used network architectures in computer vision, was incorporated into our baseline model primarily due to its superior performance when compared to other popular deep learning architectures for image classification, such as CNN, GoogLeNet, MobileNetV2, etc. Additionally, ResNet50 has been cited in numerous academic articles as being the top model for plant disease detection and one of the Top 4 Pre-trained models for image classification.

2) Flatten

For a typical neural network technique, the flatten layer is used to transform a multi-dimensional array into a 1-dimensional flatten array, or single-dimensional array. As seen, the flatten layer reduces the ResNet50 network's 3-dimensional output of 7,7,2048 values to a 1-dimensional array of 100352 values.

3) Dense

The Dense layer is also known as a fully connected layer since it is fully or deeply connected to its preceding layer. Typically, matrix vector multiplication is done in a dense layer of the neural network to change the output's dimension.

4) Dropout

To prevent overfitting, we add the dropout layer between the two thick layers. Typically, the dropout layer is put between the fully linked layers and after. This is due to the fact that fully connected layers typically contain more parameters, which forces them to overly self-adapt, leading to overfitting.

For the evaluation metrics, we have used confusion matrix and accuracy as metrics to evaluate our model and will use it further as a means to compare the other two improved models with the baseline model. We obtained a validation accuracy of 90.13% from our baseline model. Thus, it performs well enough. However, it can be improved further by the experiments that follow later.

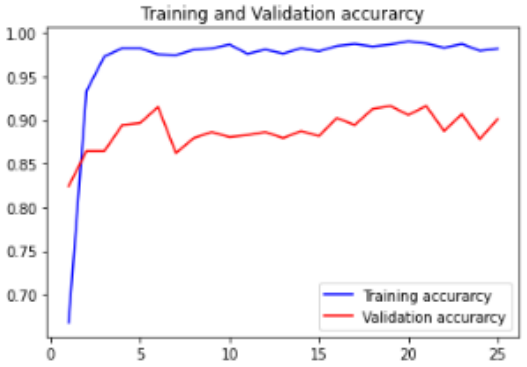

Fig.3. Accuracy graphs of baseline model

We can see from the curves above that when more data is introduced into the model, accuracy rises and validation gets better—up to a point. Additionally, the fact that the validation curve is uneven suggests that the confusion model has difficulty predicting specific labels because several disease classes have comparable symptoms.

Therefore, the baseline model's existing architecture proves to be a good model, but we can try to make it even better by adding more layers and doing image augmentation.

III. Working and Failed Experiments

We separate the data into training and testing parts. 80% of the dataset is made up of training data, and the remaining 20% is composed of testing data. In both the training and testing datasets, the images are categorized into 15 distinct folders, each of which corresponds to a class, precisely as they were in our original data. Due to the limited computing capabilities of our system, we will only be using a fraction of the original data. We have 300 pictures in each folder, which is denoted by N. This quantity may be altered to obtain better results depending on the machine's computational power.

The process of dimensionality reduction, which divides and reduces an initial collection of raw data into smaller, easier-to-manage groupings, includes feature extraction as another crucial step. Each of the experimental models' image sizes have also been changed. We used LabelBinarizer on the image labels to classify the classes.

We increased the dataset by incorporating images augmentation in order to strengthen the baseline model even further. We did this since we saw that when we give the model new data, both the validation and the model accuracy improve. We used a variety of data augmentation techniques, including rotation, shifting, zooming, and flipping, to diversity the image collection. It has been discovered that including augmented photos in a model lets it learn characteristics from multiple sections of an image more effectively, increasing performance accuracy on test data.

The subsequent layers received 25% dropout, maximal pooling, and batch normalization. We anticipated seeing improved accuracy outcomes by putting the aforementioned improvisations into practice, expanding the basic model with more layers, or creating a whole new model from scratch. We also plan to employ the Adam Optimizer for our

training because it is known to function more quickly and deliver better global minimum convergence than the other optimization algorithms.

To summarize, we implemented two different experimental models building on top of the above mentioned baseline model. The experimental models are:

1) Adding image augmentation to baseline model

To add variation to our training dataset, we introduced picture augmentation to our baseline model. The definition of image augmentation and the variety of augmentation techniques that can be used on the dataset will be covered first.

The technique of introducing minor modifications to the data in order to increase the initial dataset is known as **image augmentation**. Machine learning techniques are used to artificially create new data points. Numerous changes can be made to the image by using augmentation techniques.

For our model, we have applied the below mentioned changes to our original data images:

a) Rotation_range: 25
b) Horizontal_flip: True
c) Height_shift_range: 0.1
d) Width_shift_range: 0.1
e) Zoom_range: 0.2
f) Shear_range: 0.2
g) Fill_mode: "nearest"

The architecture for this model is kept the same as that of the baseline model.

We trained the model using the augmented data as mentioned above and were able to get the following results:

● *Best epoch*

```
Epoch 10/25
100/100 [==============================] - ETA: 0s - loss: 0.2516 - accuracy: 0.9188
Epoch 10: val_loss improved from 0.35974 to 0.29838, saving model to model.h5
100/100 [==============================] - 33s 308ms/step - loss: 0.2516 - accuracy: 0.9188 - val_loss: 0.2984 - val_accuracy: 0.9219
```

● *Final accuracy*

```
28/28 [==============================] - 3s 88ms/step - loss: 0.2712 - accuracy: 0.9311
Final Loss: 0.2711987793445587, Final Accuracy: 0.9311136603355408
```

We view this as a successful experiment given that the model outperforms the baseline model (93% accuracy).. We will compare and analyze the results further in the later section.

2) Adding image augmentation and Modifying network architecture of the baseline model.

Additionally, we explore by altering the baseline model's design and including new layers. Following is an explanation of the various layers we added throughout the experimenting phase -

a) Conv2D

A 2 dimensional convolutional layer is called Conv2D. The parameters of these layers contain a set of K learnable features, or kernels. These filters are rectangular in design because of their height and width. This layer's primary function is to generate a convolution kernel that is wound with the input supplied to the layer and aids in the production of a tensor of outputs. Using this layer, the images are convolutioned into several images with activation functions.

b) Activation

These layers are a fundamental component of the neural network architecture and often represent the activation functions. These layers are typically not explicitly mentioned in the network diagrams because it is expected that they are always there as part of the architecture. A nonlinear activation function, such as ELU, ReLU, or Leaky ReLU, is applied after each convolutional layer. Due to the element-wise application of the activation function, the output dimension of an activation layer is always the same as the input dimension.

c) BatchNormalization

Before sending an input volume to the following layer of the network, batch normalization layers are utilized to normalize the activations. [2] It enables the network's layers to each learn more effectively and independently. We utilize it to perform regularization as a safeguard against overfitting and to normalize the output of the layer before it.

d) MaxPooling

It is common practice to reduce input size by using the pooling layers. To reduce the inputs' spatial size, these layers are typically placed in between 2 successive convolutional layers. As a result, less parameters are needed in the network, which also cuts down on computation time and expense. Additionally, pooling layers aids in preventing overfitting. One of the most popular pooling layers is MaxPooling.

We experimented with the number of each type of layer we added to our neural network, the sequence in which we added these layers, and the number of layers overall. As a result, we tried a variety of various layer combinations. However, these combinations were ineffective, and their performance was worse than that of our baseline model. The following diagram illustrates one possible arrangement of layers in a neural network or neural network architecture. -

Model: "sequential"

_____

Layer (type) Output Shape Param #

==========

conv2d (Conv2D) (None, 225, 225, 32) 896

activation(Activation) (None, 225, 225, 32) 0

batch_normalization (BatchN (None, 225, 225, 32) 128

ormalization)

max_pooling2d (MaxPooling2D (None, 75, 75, 32) 0

dropout (Dropout) (None, 75, 75, 32) 0

conv2d_1 (Conv2D) (None, 75, 75, 64) 18496

activation_1 (Activation) (None, 75, 75, 64) 0

batch_normalization_1 (Batc (None, 75, 75, 64) 256

hNormalization)

conv2d_2 (Conv2D) (None, 75, 75, 64) 36928

activation_2 (Activation) (None, 75, 75, 64) 0

batch_normalization_2 (Batc (None, 75, 75, 64) 256

hNormalization)

max_pooling2d_1 (MaxPooling (None, 37, 37, 64) 0

2D)

dropout_1 (Dropout) (None, 37, 37, 64) 0

conv2d_3 (Conv2D) (None, 37, 37, 128) 73856

activation_3 (Activation) (None, 37, 37, 128) 0

batch_normalization_3 (Batc (None, 37, 37, 128) 512

hNormalization)

conv2d_4 (Conv2D) (None, 37, 37, 128) 147584

activation_4 (Activation) (None, 37, 37, 128) 0

batch_normalization_4 (Batc (None, 37, 37, 128) 512

hNormalization)

max_pooling2d_2 (MaxPooling (None, 18, 18, 128) 0

2D)

dropout_2 (Dropout) (None, 18, 18, 128) 0

flatten (Flatten) (None, 41472) 0

dense (Dense) (None, 1024) 42468352

activation_5 (Activation) (None, 1024) 0

batch_normalization_5 (Batc (None, 1024) 4096

hNormalization)

dropout_3 (Dropout) (None, 1024) 0

dense_1 (Dense) (None, 15) 15375

activation_6 (Activation) (None, 15) 0

=================================
Total params: 42,767,247
Trainable params: 42,764,367
Non-trainable params: 2,880

Table 4. Modified Network Model Layer Summary

We incorporate an Image Augmentation technique into our model in addition to these. The following findings are seen if we raise the batch size to 36 and adjust the image size from 200 x 200 to 225 x 225.

- *Best epoch*

```
Epoch 16/25
96/96 [==============================] - ETA: 0s - loss: 0.4373 - accuracy: 0.8571
Epoch 16: val_loss improved from 1.38274 to 0.74816, saving model to model.h5
96/96 [==============================] - 39s 407ms/step - loss: 0.4373 - accuracy: 0.8571 - val_loss: 0.7482 - val_accuracy: 0.7750
```

- *Final accuracy*

```
28/28 [==============================] - 1s 27ms/step - loss: 1.0402 - accuracy: 0.7382
Final Loss: 1.0402412414550781, Final Accuracy: 0.738231897354126
```

As the model doesn't perform with a higher accuracy than the baseline model, we consider this experiment a failure. The final accuracy of this experimental model is only 74%. We will compare and analyze the results further in the next section.

Shown below is a sample image picked out randomly to test and compare all three of our models.

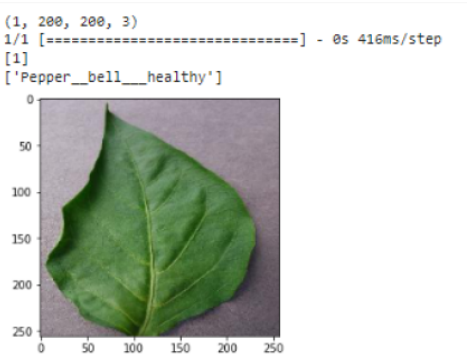

Fig.5. Testing 'bell_pepper_healthy' on the Baseline Model with Image Augmentation

IV. Observations and Analysis of Results

After examining the data to support our opinion about the best functioning model, we thus come to the conclusion that the model that was trained with Image Augmentation on top of the baseline model is the best working experiment. The results did not much improve when we changed the architecture, therefore we came to the conclusion that the experiment was unsuccessful.

Model	Training Accuracy	Final Validation Accuracy
1. Baseline	98.23%	90.12%
2. Baseline with Image Augmentation	95.33%	93.11%
3. Modified Network Architecture with Image Augmentation	89.69%	73.82%

Table 5. Accuracies of each model

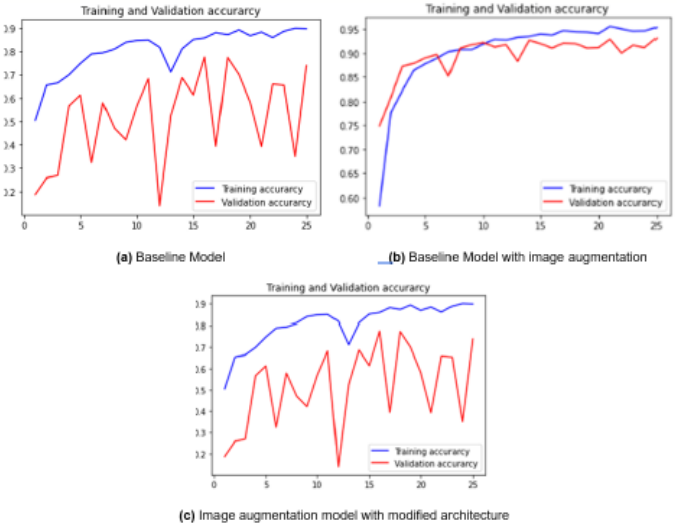

Fig. 6. Graphs for accuracies of each model

We can derive the following conclusions based on the observations from the graphs and performances of these models -

1. The baseline model performs well on the training data, with a training accuracy of 98.2%. It does not do as well on the validation data, where it only offers an accuracy of 90%. Although it's not horrible, it could do better.

2. This leads us to our working experiment where we show that adding image augmentation to our baseline model improves accuracy on the validation set. With a validation accuracy of 93.11% and a training accuracy of 95.3%, we observe a strong balance in which our model successfully executes on both training and test datasets. The graph also demonstrates the excellent accuracy and strong correlation between the training and validation accuracies. The matching loss values, which are similarly low, provide us our ideal model.

3. The training accuracy for the model with the changed architecture and image augmentation can reach up to around 90%, despite the fact that the final validation accuracy is only up to about 74%.

The graph displays the model's high level of instability on the validation data. This can be a sign that the model tries to fit the data too closely and will have trouble using data that isn't observed.

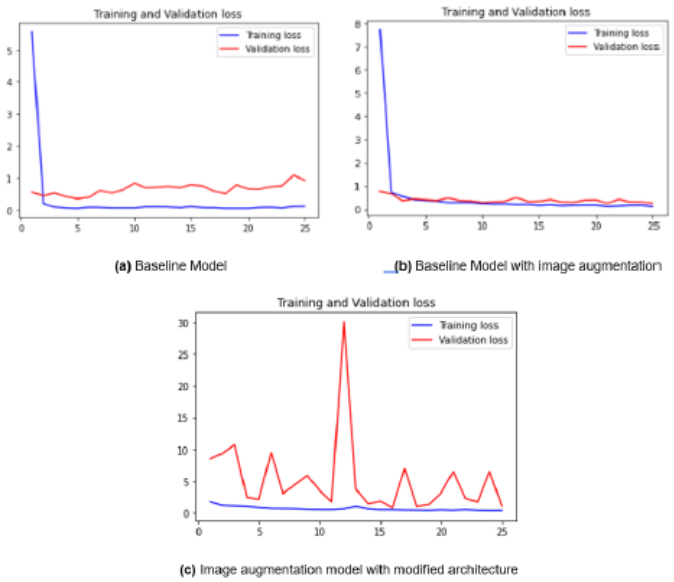

Fig. 7. Graphs for training and validation loss of each model

Furthermore, On testing our models on two randomly picked image from the dataset, we make the following observations that support our conclusion and analysis:

1. One of the two test photos is correctly classified by the baseline model, but the other is not.

2. Our conclusion that the baseline model with Image Augmentation is a better model is supported by the fact that it accurately classifies both test images.

3. These test images can also be successfully classified using the model with updated architecture and augmented images. Due to the low validation accuracy of this model, we cannot depend on it to constantly provide us with correct predictions.

V. Individual Contribution

In addition to selecting and researching about the topic and datasets, I mainly worked on the modified network architecture of the Convolutional Neural Network. I implemented this experimental model to check if performance is improved as against the baseline model.

I split the data into training and testing data: 80 and 20% respectively. I took N = 100 images per class due to computational complexity and performed image augmentation on the obtained data in order to introduce variety in the data with an objective of gaining better performance of this trained model. With an aim to optimize the performance of this model, I implemented the following two measures: 1) Image Augmentation and 2) Network Architecture Modification by introducing more powerful layers viz. MaxPooling and BatchNormalization.

These layers are introduced in various combinations in a recurring manner of the CNN. I recorded a stark contrast between the accuracies of this and the baseline model. This trained model recorded a mere accuracy of 89.69% as against the baseline's high accuracy of 98.23%. Moreover, the model fails to surpass the validation accuracy of the baseline model as well. In fact, it performs very poorly with a recorded accuracy of only 73.82%.

Furthermore, the graph of the validation loss seems to be unstable therefore, this might be a potential indicator of overfitting which could imply that the model is overly complicated. Thus, it can be concluded that this was a failed experiment. Potential steps to fix this could include testing out other possible combinations in the convolutional network.

This experiment helped me gain deeper insight and understanding of Convolutional Neural Networks and how to compare and analyze the factors that impact the performance patterns of multiple models.

VI. Team Members

The project was a team effort and involved the following team members:

1. Aesha Sandeep Shah
2. Priyal Jayvijaysinh Padheriya
3. Shrutwa Sandip Shah

VII. References

[1] Kansal, A., Arora, N., Maurya, T., Agarwal, V. K., & Tyagi, B. (1970, January 1). [PDF] detection of plant diseases using resnet50 v2: Semantic scholar. [PDF] Detection of Plant Diseases using ResNet50 V2 | Semantic Scholar. Retrieved December 2, 2022, from https://www.semanticscholar.org/paper/Detection-of-Plant-Diseases-using-ResNet50-V2/4e4fc02c2ce4b258b378d1946c36e3bbaf83e83b

[2] Rosebrock, A. (2021, June 24). Convolutional Neural Networks (CNNs) and layer types. PyImageSearch. Retrieved December 2, 2022, from https://pyimagesearch.com/2021/05/14/convolutional-neural-networks-cnns-and-layer-types/

[3] https://riteshajoodha.co.za/sitepad-data/uploads//2022/01/Charlotte-Savage-CAM-Final-Report.pdf

[4] Emmanuel, T. (2018, October). PlantVillage Dataset, Version 1

[5] Aesha Sandeep Shah, "CSE_598_Project_Group39" in CSE 598 – Introduction to Deep Learning 2022.