Pepper Bell Leaf Disease Detection

Hamdan Ali Balouch - 1802037
Noor Nabi - 1802002
Qazi Arsalan Shah - 1802030

***Abstract*—We have trained two models i.e. ALEXnet and our custom CNN model for binary classification of RGB images to either healthy or unhealthy class. Data-set is taken from Kaggle with total images of more than 2500. We have trained both the models with different data splits and we achieved more than 95% training and acceptable testing accuracy. ALEXnet architecture is very complex and to be used for simple binary classification problems. Our custom model is relatively much more simpler with fewer layers and filters than ALEXnet. Test accuracy of each model is recorded and then compared. Different classification metrics are measured and compared. Both models are also tested on unseen raw data outside the data-set.**

I. Introduction

A CNN or covnet is a deep learning algorithm used mostly for image classification, detection etc. It takes images as an input and tries to extract useful features from the image and assigns weights to them. Good thing about CNN models is that preprocessing is very simple and less as compared to conventional neural network models. Different filters are applied on different layers to extract features from given input. CNN architecture is inspired by the human visual cortex. Alexnet is one of the CNN architectures. We have used Alexnet architecture to train our first classification model and then used our custom cnn architecture to train the second model. Both models are trained to more than 90 percent accuracy and then recorded their test accuracy. Then we compared both of them in some metrics i.e. accuracy, precision and recall. Second comparison is on the basis of data splits. We have also concluded that ALEXnet is a bit complex to be used for this simple classification task which can be done with our simple CNN model.

II. Problem Statement

Implement the classification model using ALEXNet architecture and model of our choice for binary classification on pepper bell health data. The data-set is available on Kaggle. After implementing both the models we have to perform comparisons of both based on some measuring metrics i.e. accuracy, recall and precision. Second comparison on the basis of the number of splits of the data-set. Both models should be tested on unseen data outside this data-set. All the results must be analyzed and all findings must be listed.

III. About Data-set

The data-set is taken from Kaggle. This data-set is a sub part of a comparatively large data-set on plant health. We selected only two classes i.e. “pepper bell healthy” and “pepper bell bacterial spot”.Class “pepper bell healthy” contains 1478 data objects i.e. images. And the second class “pepper bell bacterial spot” contains almost 1000 images. Below are a few sample images from both the classes with our comments on them.[**dataset**]

A. Bell Pepper Healthy

These are a few sample data objects of healthy class. Leaves are well contrasted from the background and it will

Fig. 1. Sample of healthy bell peppers

surely help our model to get a good insight of data. But few images are more dark than the others, few have some shadow behind them. These factors might add some bad impact while training.[**dataset**]

B. Pepper bell Bacterial

These are some sample data objects from an unhealthy class.

Fig. 2. Samples of unhealthy bell peppers.

As we can see, few images are very dark in color and this issue was with many of the images from this class as compared to healthy class data. And while testing we faced many times when the image is brighter and with less bacterial spots it is mostly classified as healthy or barely classified as unhealthy. [**dataset**]

IV. Data Preparation

There are two models we have worked with, one is a custom CNN and other is ALEXNet. Preparation of the data for both the models is defined below.

A. Data Preparation for ALEXNet

ALEXNet net requires an image size of 227*227px as an input so all the images were first resized to 227*227, to do so we used OpenCV cv2.resize() function. Then the data was normalized by deviding it by a factor of 255. We have used tensorflow.keras.utils's to-categorical function to categorize the data to know whether it belongs to class A(Bacterial Infection) or B(Healthy).

B. Data Preparation for Custom CNN

All the preparation for this model remains the same except image size which is 200px*200px, so all the images are converted to the size of 200px*200px before being fed to the model.

V. Model Architecture

A. AlexNet Architecture

The architecture of AlexNet is visually explained in figure. it contains 8 layers, 5 of which are a mix of convolutional layers with max pooling, then it has 3 fully connected dense layers whcih use Relu as activation function except the output layer that uses softmax as activation function. A dropout layer is added to avoid overfitting. As mention earlier each convolutional layer has a max pooling combination, every layer has a max pool size of (3,3). In the first convolutional layer 96 filters are applied with kernal size of (11,11) and the image input size is fixed to (227,227,3) and the image is RGB. In second convolutional layer 256 filters are applied with a kernal size of (5,5). In third convolutional layer 384 filters are applied with a kernal size of (3,3). In fourth convolutional layer 384 filters are applied with a kernal size of (3,3). In fifth convolutional layer 256 filters are applied with a kernal size of (3,3). After that the output is flattened, the first droupout is added at the rate of 0.5, after that we have the first fully connected layer which has a relu activation function and an output size of 4096 units. The second droupout id added at the rate of 0.5, ehich is then followed by second fully connected layer with 4096 units. finally we have the last layer with softmax activation and an output size of 2 units.[**alexnetarch**][**aleximple**]

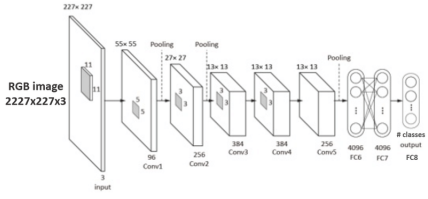

Fig. 3. AlexNet Architecture **alexnetarcimage**

B. Custom Model Architecture

The architecture of our model is explained visually in figure [1]. It contains 6 layers with four convolutional layers and 2 fully connected dense layers. Last layer is the output layer that contains 2 units and a softmax activation function is used on this last layer. In all layers except the output layer, the RELU activation function is used. Dropout layer is added between some layers to avoid overfitting. With each convolutional layer, max pooling layer is added with pool size of (2,2).

In first convolutional layer, 32 filters are applied with kernel size of (5,5) and input size of image is fixed to (200*200*3) The image is RGB. In the second convolutional layer, 64 filters are applied with kernel size of (3,3). In the third convolutional layer, 128 filters are applied with kernel size of (3,3). In the fourth convolutional layer, 256 filters are applied with kernel size of (3,3). Then the output is flattened with Flattened layer. After this, 2 fully connected dense layers are applied containing 100 units each. Dropout is also added to avoid overfitting.[**custommodelarch**]

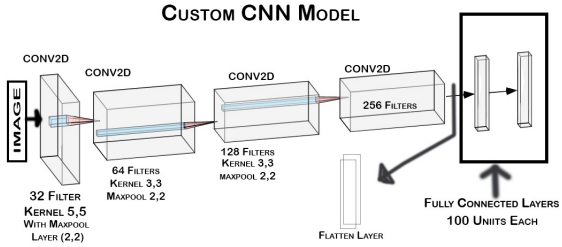

Fig. 4. Custom CNN Architecture

VI. Comparison of both models

We compared our both models in two ways. One, on the basics of a few measuring metrics i.e. accuracy, precision and recall. And second on different data splits i.e. 60-40 , 70-30 and 80-20 train test data splits and then measured test accuracy of each.

A. Comparison w.r.t accuracy, recall and precision

Model	Train Accuracy	Valid Accuracy	Test Accuracy
AlexNet	98.49%	92.77%	90.72%
Custom	96.7%	95%	97.17%

The below table show the classification reports of the both models

Model	Class	Precision	Recall
AlexNet	0	89%	100%
Custom	1	100%	91%

The next table show the confusion matrix for Alexnet.

	0	1
0	215	0
1	26	255

For the sake of explanation, 215 (index 0,0) in the above figure are True Negatives i.e. those examples which were actually

negative and classified as negative.Similarly, 255 index(1,1) presents true positives.

The table below shows the confusion matrix for custom model.

	0	1
0	174	2
1	3	317

These are the results after training the model and they show a really good accuracy with Custom model despite being really simple as compared to alexnet. These results have been been calculated using a part of the dataset that was kept for testing earlier. Both the models seem to be similarly precised with the given test results.

B. Comparision over split

What we have here is the results of the models with different splits.

C. Split 80:20

Model	Train Accuracy	Valid Accuracy	Test Accuracy
AlexNet	98.49%	92.77%	90.72%
Custom	89.53%	90.68%	93.13%

The split was made in a way that the data for training was 80 percent and data for test was 20 percent.

The table below shows the classification report of both the models.

Model	Class	Precision	Recall
AlexNet	0	89%	100%
Custom	1	100%	91%

The next table show the confusion matrix for Alexnet.

	0	1
0	215	0
1	26	255

The table below shows the confusion matrix for custom model.

	0	1
0	174	2
1	3	317

D. Split 30:70

Model	Train Accuracy	Valid Accuracy	Test Accuracy
AlexNet	98.02%	88.48%	90.72%
Custom	90.56%	69.74%	96.2%

The split was made in a way that the data for training was 70 percent and data for test was 30 percent.

The table below shows the classification report of both the models.

Model	Class	Precision	Recall
AlexNet	0	99%	90%
Custom	1	94%	99%

The next table show the confusion matrix for Alexnet.

	0	1
0	200	15
1	1	280

The table below shows the confusion matrix for custom model.

	0	1
0	191	10
1	4	291

E. Split 60:40

Model	Train Accuracy	Valid Accuracy	Test Accuracy
AlexNet	96.35%	94.85%	94.15%
Custom	85.70%	83.89%	84.491%

The split was made in a way that the data for training was 60 percent and data for test was 40 percent.

The table below shows the classification report of both the models.

Model	Class	Precision	Recall
AlexNet	0	99%	87%
Custom	1	91%	99%

The next table show the confusion matrix for Alexnet.

	1	0
0	361	54
1	4	573

The table below shows the confusion matrix for custom model.

	1	0
0	174	2
1	3	317

F. Analysis of Comparison

As all the data i.e. accuracy, precision and recall etc are presented in above tables. It is obvious that both models are trained to more than 90 percent of accuracy. Custom model being simple, trained relatively more early in terms of epochs. Test accuracy is almost the same but when we changed data splits, custom cnn model remain almost unchanged in terms of test accuracy but ALexnet dropped its accuracy. On the same test dataset i.e. from the kagle data, both models perform great but when they are fed with few unseen raw data they start miss classification. Custom CNN model was still better in predictions over unseen data as compared to Alexnet.

Alexnet was crafted to be used for more complex problem and for this particular problem, simple model can be trained well rather using that much relatively complex architecture.

VII. Results On Unseen Raw Data

On data outside the dataset, our custom cnn model performed a bit better as compared to Alexnet. There were few things we missed were that, test images must be processed according to the ones in the dataset i.e. lef well isolaetd from background but we just center cropped to remove extra sides only. Below are the results of two images tested with both models.

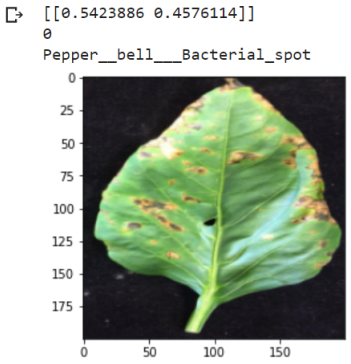

Fig. 5. Custom CNN Prediction

Above is the image of leaf with bacterial spot and our CNN model predicted it correctly but confidence is low.

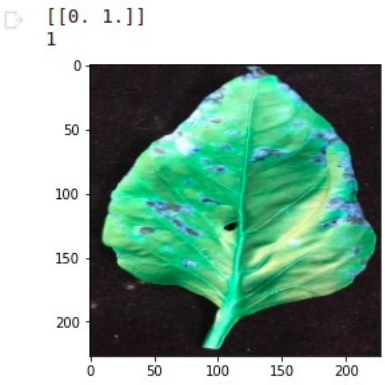

Fig. 6. Alexnet Predictions

As you can see, Alexnet predicted it as healthy but it was not healthy.

Similarly second test image was of healthy plant leaf, both models predicted correctly.

Above image is of healthy plant leaf and our model does correct predictions.

Alexnet also predicted it correctly.

Custom CNN model that was trained on different data splits i.e. 80/20 , 70/30 etc 80/20 split trained model was comparatively better than others when it comes to raw data from outside.

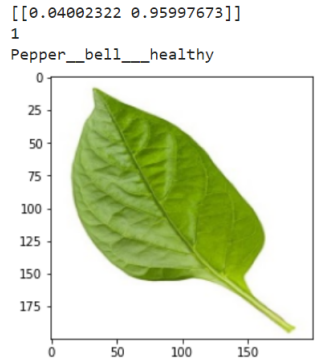

Fig. 7. Our Custom model prediction on healthy image

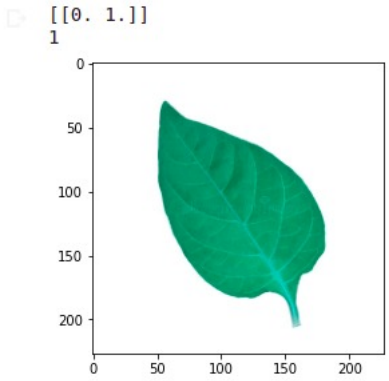

Fig. 8. Alexnet Prediction on Healthy Leaf

Below are few more raw examples that were tested with our custom cnn model.

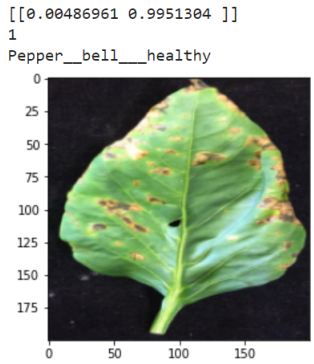

Fig. 9. Custom CNN (70/30 split) Prediction On Unhealthy leaf

Figure 9 is the predicted result of our Custom model that was trained with 70-30 data split. It was not able to classify this image as unhealthy but 80-20 splited model classified it correctly.

The image below [figure 10] i.e. unhealthy image is also miss classified. That was a common thing noticed when testing raw images, very few of them are correctly classified. We have mentioned about quality of data objects in unhealthy class as many images were dark that thing might contributed to a bit over fit on the data set.

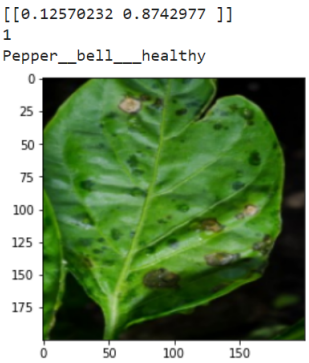

Fig. 10. Custom CNN (80/20 split) Prediction On Unhealthy leaf

REFERENCES

[1] Introduction to The Architecture of Alexnet *Shipra Saxena* https://www.analyticsvidhya.com/blog/2021/03/introduction-to-the-architecture-of-alexnet/

[2] AlexNet Architecture representation*Image Uploaded by: Alexander Khvostikov* https://www.researchgate.net/profile/Alexander-Khvostikov/publication/322592079/figure/fig3/AS:584350454263818@1516331413967/AlexNet-architecture-Includes-5-convolutional-layers-and-3-fullyconnected-layers.png

[3] A Gentle Introduction to Dropout for Regularizing Deep Neural Networks *by Jason Brownlee on December 3, 2018 in Deep Learning Performance*

[4] Implementing AlexNet CNN Architecture Using TensorFlow 2.0+ and Keras By Richmond Alake

[5] Dataset Used https://www.kaggle.com/emmarex/plantdisease