[Team 23]Project 1.2:Leaf Wilting Detection in Soybean

Rajshree Jain
rjain27@ncsu.edu

Sai Shruthi Madhuri Kara
skara2@ncsu.edu

Sruthi Talluri
stallur2@ncsu.edu

1 METHODOLOGY

The data set we worked on comprises of soybean plant images through at different times of the day. The aim of the project is to identify the level of wilting that is observed in all the images. The level of wilting has been classified into 5 groups. The data set consists of 1,275 total images in the training data set. The test data consists of 200 images. The distribution of the images in training data is imbalanced. In second phase of the project we have tried to use various pretrained models for feature extraction. This would help in identifying the most important features from the given images and then form a data set on which neural network model is trained. As a part of of our project the various convolutional neural networks that we tried are VGG16, VGG19 and Resnet for feature extraction from our images. Further, we have used techniques like SMOTE and hyper parameter tuning to balance the data and get best parameters for training models respectively. The features that we extract by using the pre trained models are fed into a convolutional neural network for actual image classification. After testing all different types of pre-trained architecture we found that VGG19 performed the best in our case and hence has been used to make the final prediction for the test dataset.

The VGG16 architecture was trained on the ImageNet data set to classify around 1000 classes. It has 16 layers with trainable parameters out of which 13 are convolutional layers and 3 are fully connected layers. After every couple of convolutional layers the size of the image decreases taking the more important features. The starting convolutional layers till the last max pooling layer is used for feature extraction in the VGG-16 architecture in our case.

The VGG-19 architecture is another variant of the VGG-16 architecture. In this architecture as the name suggests there are a total of 19 layers with trainable parameters. So, basically there are 16 convolutional layers and 3 fully connected layers. In the same way as VGG-16 this architecture also has been trained on ImageNet data set and is used to classify around 1000 classes. In our case since these convolutional neural network models are used for feature extraction, the VGG-19 architecture having more convolutional layers to serve the task might lead to a better feature extraction. Hence, giving better results.

We also tried to use the Resnet architecture for feature extraction and then train a CNN on the extracted features. Resnet has a Convolutional Neural Network (CNN) architecture which was designed to enable hundreds or thousands of convolutional layers.

After trying different convolutional architectures we could understand that VGG19 made the best classification with a greater F1 score as compared to other models. Also, the validation accuracy increased and validation loss decreased. We have utilized the pre-trained VGG19 architecture on Image Net for feature extraction by resizing the leaves of the Soybean plant image to 240 by 320 by 3 pixels 3 denoting the RGB color channels of the Soybean plant images.
The output from the model has been squeezed into the shape of (2440, 2, 2, 512) and has been fed into the first layer of fully connected layers for classification. In the end we have n number of dense and dropout layers for making the classification. The number of these layers is decided by hyper parameter tuning. Since, we have 5 classes to classify the images into, we need 5 neurons at the output layer which has been taken care of. The layers and details of the fully connected model can be seen in Figure 2.

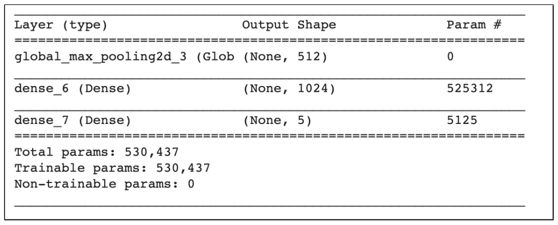
```
Layer (type)                 Output Shape              Param #
=================================================================
global_max_pooling2d_3 (Glob (None, 512)               0
_________________________________________________________________
dense_6 (Dense)              (None, 1024)              525312
_________________________________________________________________
dense_7 (Dense)              (None, 5)                 5125
=================================================================
Total params: 530,437
Trainable params: 530,437
Non-trainable params: 0
_________________________________________________________________
```

Figure 1: Model Structure before Hyper-parameter tuning

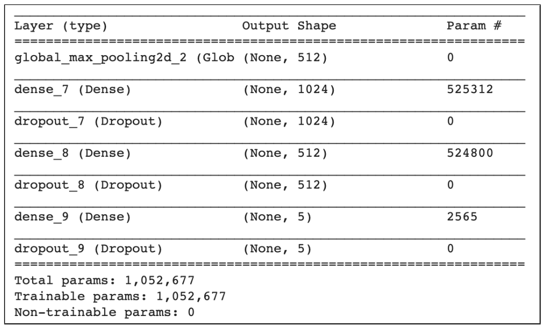
```
Layer (type)                 Output Shape              Param #
=================================================================
global_max_pooling2d_2 (Glob (None, 512)               0
_________________________________________________________________
dense_7 (Dense)              (None, 1024)              525312
_________________________________________________________________
dropout_7 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_8 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_8 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_9 (Dense)              (None, 5)                 2565
_________________________________________________________________
dropout_9 (Dropout)          (None, 5)                 0
=================================================================
Total params: 1,052,677
Trainable params: 1,052,677
Non-trainable params: 0
```

Figure 2: Model Structure after Hyper-parameter tuning

2 MODEL TRAINING AND SELECTION

2.1 Model Training

(1) Training and Validation Split
We tried to split the data in 2 ways, first having 80 percent training and 20 percent validation data and then 70 percent training data and 30 percent validation data. We tried all the feature extraction models (Resnet152, Vgg-16, Vgg-19) on both the splits. We could observe that the validation accuracy was more and validation loss was less in all cases for 70-30 split. Hence we choose the 0.3 as the split point for our training data.

(2) Oversampling Data
In order to prevent the problem of class imbalance, we oversampled the data using SMOTE technique (Near miss undersampling in case of Resnet 152). This technique involves duplicating examples in the minority class, although these examples don't add any new information to the model.

(3) Tensors in Image Classification:
We have converted all our input leaves images of the Soybean plant into the tensors as it can be used like a matrix to perform operations like manipulating the image - data augmentation.

(4) Image re-sizing
We tried to use various image sizes like (80, 80, 3), (200, 300, 3) and (240, 320, 3) and could observe that we got best accuracy and F1 results using the size (240, 320, 3).

2.2 Model Selection

Before, moving forward with the VGG19 architecture, we wanted to try a different architecture, that is, ResNet152, to see if we get better results. Unlike the VGG19 architecture which take images that are converted to tensors, we have to feed the ResNet architecture with actual images. In this scenario, we had to do the undersampling(NearMiss) to balance the class distribution of the images, as performing the oversampling gave memory outage error. Even though the validation accuracy[1][2] was significant as in the Figure 3 and Figure 4, the F1-score has not been very impressive on the test data dropping to 0.17. This is due to the only few images(650) used for training of the architecture. Hence, we decide to improve the VGG19 architecture with rigorous hyper-paramter tuning[3][4]. It is important to consider choosing an appropriate Optimiser that can help in understanding the loss accurately.Initially, we have tuned the model with hyper-parameters like dropout rate(0.1) and optimizer(RMS-prop) yielding the validation accuracy of 77% which improved out F1-score from 0.38 to 0.448 on test data.We have also taken care that we won't overfit our model by using the early stopping callbacks by monitoring the validation accuracy to 0.75 to analyse the epoch at which we get the best accuracy.

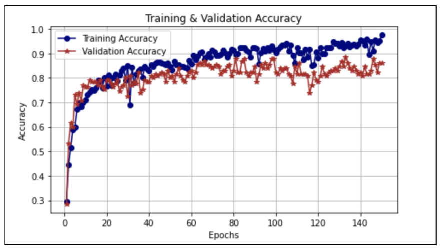

Figure 3: Validation Accuracy for Resnet152

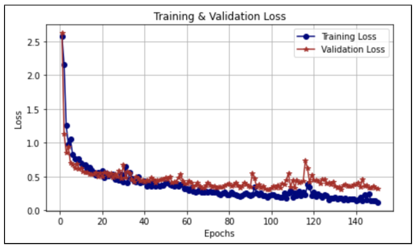

Figure 4: Validation Loss for Resnet152

Hyper Parameters	Values
Batch Size	40
Number of Epochs	15
Number of Hidden Layers	2
Dropout Rate	0.1
Learning rate	0.001
Optimizer	ADAM
Activation function for Hidden Layer	RELU
Activation function for output Layer	SOFTMAX
Loss	categorical_crossentropy

Figure 5: Hyper-Parameters

The Batch size and number of epochs is also a good way to determine which parameters and models perform better. Number of epochs, determines the number of times training data set will be propagated through the network. The batch_size defines, number of instances that will be propagated through the model at a point of time. Also, parameters like the learning rate,number of hidden layers will also help the model to learn better and improve the performance. Figure 1 and Figure 2 gives a clear picture of how the model structure has been changed to give better performance before and after hyper parameter tuning.

The table below lists all the final hyper-parameters that are tuned and used onto the sequential model built for the classification part. We can refer the Figure 5 to analyse how the parameters are selected for the hyper-parameter tuning that has been performed graphically.
Hence, considering all these parameters as mentioned in the table, the model could yield a better validation accuracy of 85% increasing our F1 score to 0.455 on test data.

3 EVALUATION

The metric that has been used to compare different architectures and understand the correctness of models classification on the test data is F1 score. A measurement that considers both precision and recall to compute the score. The F1 score can be interpreted as a weighted average of the precision and recall values, where an F1 score reaches its best value at 1 and worst value at 0.

In the approach we implemented different architectures for feature extraction and compared the F1 scores, for determining which one performs better.

We can refer to Table 1 for different metrics calculated for different architectures.

For VGG-16 we can see in Figure 7 that the validation accuracy for the validation data is less than the training accuracy. And the validation loss also varies a lot over different epochs as seen in Figure 8. Hence, considering all these factors we moved ahead with the VGG-19 model even the validation F1 score was considerable.

In VGG-19 with hyper parameter tuning that involves epochs, batch size, learning rate, dropout rate, optimizer, number of hidden layers performs the best having a validation F1 score of 0.86 on validation data.

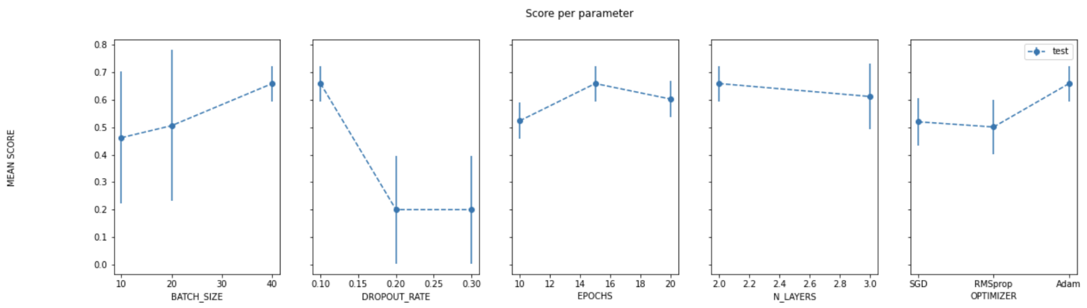

Figure 6: Graph Plots for Parameters

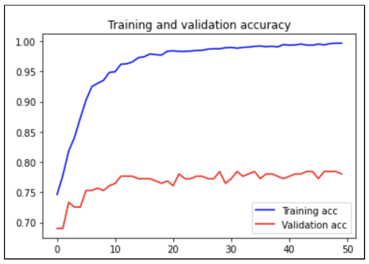

Figure 7: Validation Accuracy for VGG16

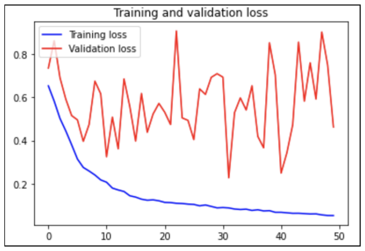

Figure 8: Validation Loss for VGG16

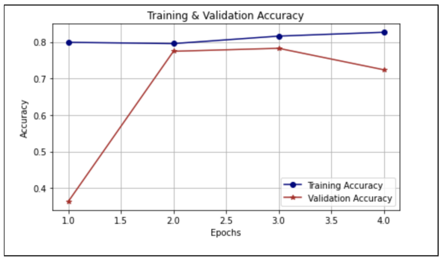

Figure 9: Validation Accuracy for VGG19

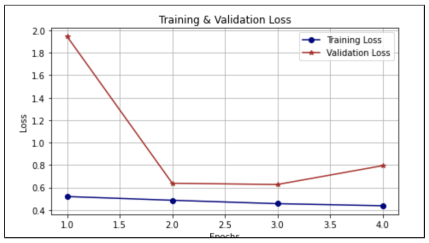

Figure 10: Validation Loss for VGG19

Model	Accuracy	Loss	Recall	Precision	F1Score
RF(Sampling)	0.93	-	0.93	0.93	0.93
VGG16	0.78	0.46	0.78	0.78	0.78
Resnet152	0.86	0.32	-	-	-
VGG19	0.78	0.62	0.80	0.74	0.76
VGG19 - HP tuning 1	0.77	0.72	0.76	0.78	0.77
VGG19 - HP tuning 2	0.85	0.51	0.83	0.86	0.86

Table 1: Accuracy, Loss, Precision, Recall Values for Validation Data, F1 Score for Validation Data

We can see that for VGG-19 the validation accuracy is almost equal to the training accuracy as seen in Figure 9. Also, the F1 score on test data came out to be the best as compared to all other models. The validation loss is also less for VGG-19 and is almost like training loss.

However, when we compare the results of VGG-19 without hyperparameter tuning with VGG-19 having hyper parameter tuning we can see that after hyper parameter tuning the fully connected layers helps in getting better results.

In case of Resnet also we can see that validation accuracy is 0.86 but the F1 score was less for testing data. Also, Figure 3 and Figure 4 show that Resnet model performs well on the training and validation data. But since we under sampled the data for Resnet, the F1 result was not comparable to VGG-19.

Further, regarding testing various approaches and the F1 scores on testing data we could observe that we got the best F1 score value with VGG-19 with hyper parameter tuning that is 0.455.

As observed from the Table 1, the F1 score(Our evaluation metric) of VGG19 architecture with feature extraction, oversampling is the best upon hyperparameter training.

REFERENCES

[1] https://towardsdatascience.com/multi-class-metrics-made-simple-part-i-precision-and-recall-9250280bddc2
[2] https://keras.io/api/metrics/
[3] https://towardsdatascience.com/simple-guide-to-hyperparameter-tuning-in-neural-networks-3fe03dad8594
[4] https://machinelearningmastery.com/grid-search-hyperparameters-deep-learning-models-python-keras/