Leaf Wilting Detection in Soybean

1st Sriram Sundararaj
Unity id: ssunda24
North Carolina State University
Dept. of Electrical and Computer Engineering

2nd Hema Ramesh
Unity id: hsramesh
North Carolina State University
Dept. of Electrical and Computer Engineering

3rd Sai Krishna Wupadrashta
Unity id: swupadr
North Carolina State University
Dept. of Electrical and Computer Engineering

I. METHODOLOGY

Our Project aims to develop a model that indicates drought tolerance or sensitivity in Soybean plants, by studying the leaf wilting and classifying the degree of leaf wilting into different categories based on the extent of wilting. We are using 5 classes (from 0 to 4) in total for implementing this model, with 0 being the image depicting the least wilting to 4 depicting the most wilting by the soybean leaves in the images.

The data we are using to architect this novel classification system consists of images of soybean crops captured at various times of the day, and a csv file containing the respective annotations for the images. The csv file contains the information in one-hot encoding format which serve as the annotations for the training data.

So, the machine learning model we are employing for prediction uses certain crafted features of Computer Vision, mainly focusing on the implementation of a classic Convolutional Neural Network (CNN).

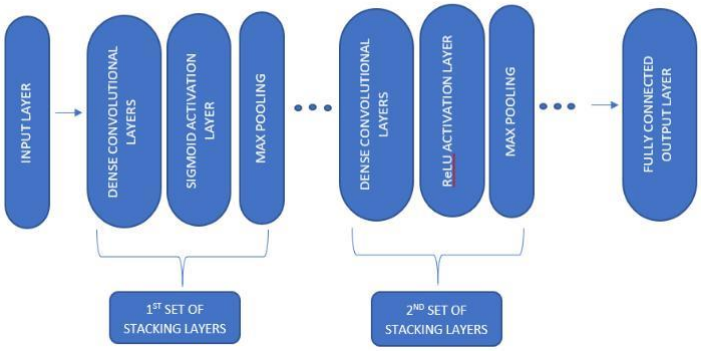

Figure 1 (FLOW OF THE CNN MODEL)

A rough sketch of the Convolutional Neural Network (CNN) we have implemented is shown in *Figure 1.*

Now, we have imported a folder containing 1025 images for training and validation purposes. These images represent all the categories of leaf wilting as discussed above. For input and training purposes, after using the cross-validation approach, the data has been split in a ratio of 4:1 for training and validation., ie., 80 % of the images are for training and 20% of the images have been chosen for validation. The training images are used to train the model and increase the training accuracy, and the validation images are used to tune the hyperparameters to make the model potentially more robust to newer images and improve the performance of the model in general.

The table below shows how we have divided the training dataset:

Total no. of input images	1275
No. of images for training	1020
No. of images for validation	255

Table 1

Now, apart from this we have some images solely for the purpose of testing. The images meant for testing aren't to be passed to the network before the training has been completed. These are fed as input to the network only after the training process has been completed, and the hyperparameters have been tuned. The total number of images for testing here are 300. The table below represents the summary of the distribution between the training data and the testing data sizes:

Total number of images for training	1275
Total number of images for testing	200

Table 2

Thus, this is the summary of the data we have used for this project. We resize the images into a resolution of 64x64 size before training ensues. The image data is then fed into the CNN model that has been designed in the python environment using Jupyter Notebook. We have used 6 primary libraries for implementing the model:

i) Numpy
ii) Scikit Learn (sklearn)
iii) Pandas
iv) Matplotlib
v) Keras
vi) TensorFlow

Numpy is used to handle numerical and array based calculations in python with ease. The Pandas library is used to import or export data between data frames. The Matplotlib is used for plotting and visualizing the graphs for accuracy and loss functions. The sklearn library is used for splitting the data for training and validation and for resizing the images for processing. Keras in TensorFlow contains the majority functions of CNNs -like 2D convolution, Max-Pooling, Flatten, Dense and dropout and the optimizer models.

The CNN model designed can be roughly split into 4 categories as follows:

a) Input Layer

b) 1st Set of stacking Layers

c) 2nd Set of Stacking Layers

d) Fully Connected Output Layer

A. INPUT LAYER:

The input layer is the layer from which the data is fed into the CNN. We have used One-Hot encoding to match the input images to their annotations- which are then fed into the network. We are using a batch size of 16- so we pass 16 inputs into the network at a time. So, the input layer consists of 16 neurons.

B. 1ST SET OF STACKING LAYERS:

The first set of stacking layers consist of convolutional layers followed by sigmoid activation function layers-each consisting of 512 neurons. These sets of layers are then stacked one on top of the other some *n* number of times. Here for the C-1 stage implementation of our project, we have chosen *n*=1.

C. 2nd SET OF STACKING LAYERS:

The second set of stacking layers consist of convolutional layers followed by ReLU activation function layers-each consisting of 256 neurons. These sets of layers are then stacked one on top of the other some *m* number of times. Here for the C-1 stage implementation of our project, we have chosen *m*=1.

D. FULLY CONNECTED OUPUT LAYER :

The final layer is a fully connected layer- as opposed to the prior layers of the CNN which aren't designed to be fully connected by weights owing to optimization and computation issues. The final layer consists of 5 neurons- which denotes the 5 classes available for classification. The image after passing through the model gets classified and comes out in the form of One-Hot encoding. The *predict.csv* file consists of all the output readings observed from the final layer after passing the images present in the testing data set.

II. MODEL TRAINING AND HYPERPARAMETER SELECTION

A. MODEL TRAINING:

The model is trained by using the 1025 images by employing the method of cross-validation thereby having 820 images for testing and 205 images for validation. We have designed the model to run for 20 epochs, by employing the methodology highlighted above.

B. HYPERPARAMETER SELECTION

Now, the validation dataset is used to make the model more robust to newer outputs by facilitating the tuning of hyperparameters.

We have chosen the following parameters and the values for each hyperparameter, as shown in the table:

Dimensions of the image for processing	64x64
Split-up ratio for training to validation	4:1 (80% training , 20% validation)
Batch size	16
Epochs	15
No. of neurons in the 1st set of stacked layers	512
Width of 1st set of stacked layers	*n=1*
No. of neurons in the 2nd set of stacked layers	256
Width of 2nd set of stacked layers	*m=1*
Dimensions of filter for Max-Pooling	5x5
Stride used for Max-Pooling in the x and y direction	(1,1)

Table 3

These are the hyperparameters we have used and on tuning, we have set these values for them to achieve good performance for the model built for stage C-1 of the project. Further improvement can be done by enhancing the depth of the network and tuning the hyperparameters appropriately.

III. EVALUATION METRICS AND PLOTS

We have used the following list of metrics to evaluate the performance of the model built:

i) Training Loss

ii) Training Accuracy

iii) Validation Loss

iv) Validation Accuracy

v) Precision

vi) Recall

vii) F-1 score

We have obtained the readings for the training loss, training accuracy, validation loss, validation accuracy for each epoch. The final testing accuracy has been recorded once after the processing has taken place over the entire testing dataset.

Final Training Loss	4.98778e-04
Final Training Accuracy	99 %
Final Validation Loss	0.8593
Final Validation Accuracy	84.88 %
Final Testing Accuracy	**50%**

Table 4

Graphs have been plotted to show how the training and validation loss and accuracy behave with increasing number of iterations.

Finally, a table of data is obtained that shows the precision, recall and the F-1 score obtained for each class from 0 to 4

after the classification using the testing dataset has been completed.

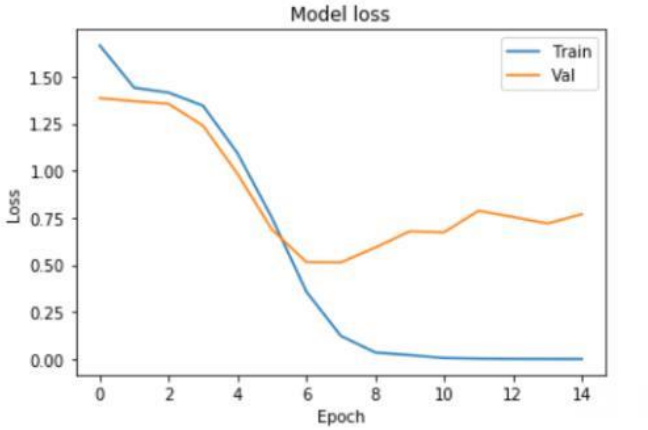

Figure 2 (Training and Validation loss graph)

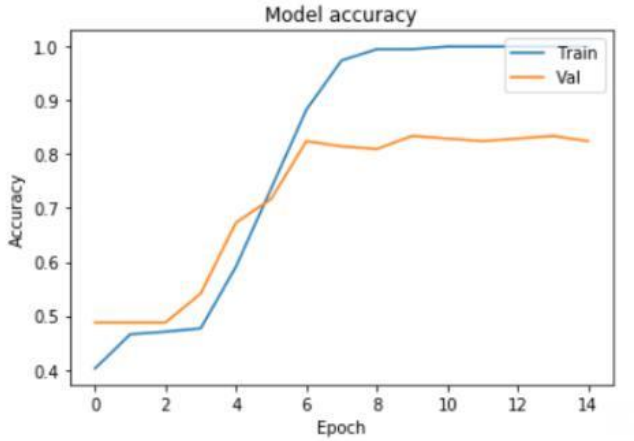

Figure 3 (Training and Validation accuracy graph)

	precision	recall	f1-score
0	0.83784	0.93000	0.88152
1	0.70968	0.61111	0.65672
2	0.86364	0.67857	0.76000
3	0.80645	0.92593	0.86207
4	1.00000	0.71429	0.83333
accuracy			0.82439

Figure 4 (Table for precision, recall, F-1 score & accuracy)

IV. REFERENCES

[1] [1] King AC, Purcell LC, and Brye KR (2009) Differential wilting among soybean genotypes in response to water deficit. Crop Science 49:290-298. doi: 10.2135/cropsci2008.04.0219

[2] [2] Pathan, SM et al. (2014) Two soybean plant introductions display slow leaf wilting and reduced yield loss under drought. Journal of Agronomy and Crop Science 200:231-236. doi: 10.1111/jac.12053

[3] B. Zhong, Q. Ge, B. Kanakiya, R. Mitra, T. Marchitto, E. Lobaton, "A Comparative Study of Image Classification Algorithms for Foraminifera Identification," IEEE Symp. Series on Computational Intelligence (SSCI), 2017.