[Team-16] Project C2 Leaf Wilting

Karthika Vadivel
Department of ECE
North Carolina State University
Raleigh,NC
kvadive@ncsu.edu

Sanjana Banerjee
Department of ECE
North Carolina State University
Raleigh, NC
sbaner24@ncsu.edu

Muthu Kumar Venkatesh
Department of ECE
North Carolina State University
Raleigh, NC
mvenkat3@ncsu.edu

I. METHODOLOGY

For the given task of automate classification of leaf wilting ratings under five categories for the soybean plant which represent two different soybean genotypes responding to a wide range of atmospheric and soil moisture conditions, we have decided to use Neural Network learned based on Transfer Learning technique[1]. The model we have chosen is the Densely Connected Convolutional Network[2], the DenseNet121 model. We will be using the Keras[3] library to construct our neural network with pretrained weights obtained by training on the ImageNet database. The various functions used in our code to construct the neural network were taken from the Keras API toolbox. The training data is augmented and stored in five sub folders classified according to their classes. The annotations from 0 to 4 provide the level of wilting observed in the plant.

• class 0- No Wilting
•class 1 - Leaflets folding inward at secondary pulvinus, no turgor loss in leaflets or petioles
• class 2 - Slight leaflet or petiole turgor loss in upper canopy.
• class 3 – Moderate turgor loss in upper canopy
• class 4 - Severe turgor loss throughout canopy

The model has all it's layer frozen by setting layer.trainable=False. The DenseNet model has MLP with two Dense Layers of 1024 and 512 neurons and the ReLU activation function. The last Dense Layer that classifies the images uses Softmax as its activation function. we have evaluated the model on categorical cross-entropy and used accuracy as the metrics. The optimizer used was an Adam optimizer. The batch size was kept at 32 and the network was trained on 50 epochs. The input image dataset was augmented and normalized to have their pixel values in range [0, 1] and resized to a shape of (224,224,3)

DenseNets are divided into DenseBlocks, where the dimensions of the feature maps remain constant within a block, but the number of filters changes between them. These layers between them are called Transition Layers and take care of the down sampling applying a batch normalization, a 1x1 convolution and a 2x2 pooling layers. DenseNet architecture overcomes the vanishing gradients problem and is thus efficient for CNN based problems since the information from the original input is carried forward until the last layer. DenseNets do not sum the output feature maps of the layer with the incoming feature maps but concatenate them. DenseNets layers are very narrow, and they just add a small set of new feature-maps.

II. MODEL TRAINING AND SELECTION

A. Model Training:

Data Preprocessing: The dataset that has been initially provided consists of 488 images in the first class, 329 images in the second class, 130 images in the third class, 131 images in the fourth class and 488 images in the fifth class. This dataset was first classified into 5 folders based on their labels. The classified dataset was then subjected to a 80/20 split into training and validation folders respectively. The splitting created the following distribution of the dataset.

CLASS	TRAINING	VALIDATION
0	400	88
1	252	77
2	107	23
3	109	22
4	165	32

Data Augmentation*:* The training and validation dataset was further augmented to deal with 2 issues: i) The unbiased dataset was augmented in a way that each label would consist of more or less equal number of images. This was done by applying more number of transformations to classes that had less number of images compared to others. ii) The augmentation also increased the total number of input images thus dealing with the problem of training a model with a small input dataset. The augmentation created the following distribution of the split dataset.

CLASS	TRAINING	VALIDATION
0	800	176
1	756	231
2	749	161
3	763	154
4	825	160

Different combinations of transformations such as zoom in, zoom out, increase contrast by using the function cv2.convertScaleAbs(image, alpha = 1, beta = 50) , decrease brightness using the function cv2.convertScaleAbs(image, alpha = 0.7, beta = 1) ,image blurring, rotate and zoom, crop and sharpening were used for augmenting the dataset . By comparing the 2 tables we can observe that the dataset is now more evenly distributed after the augmentation. The final input dataset consists of 3893 training images and 882 validation images.

B. Model Selection:

We tried 3 main model types for this particular problem.

a) Simple CNN: The first approach with no data augmentation and a simple Convolutional Neural Network gave us the least accuracy. The model consisted of 3 Convolutional Layers with 32,32 and 64 filters respectively followed by a Dense layer with 64 Neurons. A dropout of 0.5 was added and activation function Relu was used. Adam optimizer with learning rate 0.0001 was also used. Augmenting the data with this model only slightly increased the test accuracy.

b) Resnet50: This model gave good validation accuracy but did poorly on the test dataset. ResNets have proven that many layers are barely contributing and can be dropped. In fact, the number of parameters of ResNets are big because every layer has its weights to learn.

c)DensetNet121: This model with pretrained ImageNet weights and Data Augmentation gave the best values of RMSE and Accuracy on the given dataset and hence was chosen as the final model.

Following table shows Validation loss, Validation accuracy and Test accuracy of different models.

Model	Val_acc	Val_loss	Test_Acc
Simple CNN	0.5236	1.0108	0.15
CNN+dataAu	0.7566	0.9956	0.24
Resnet50 TL	0.829	0.5976	0.48
DenseNet121 TL	0.8751	0.4039	0.575

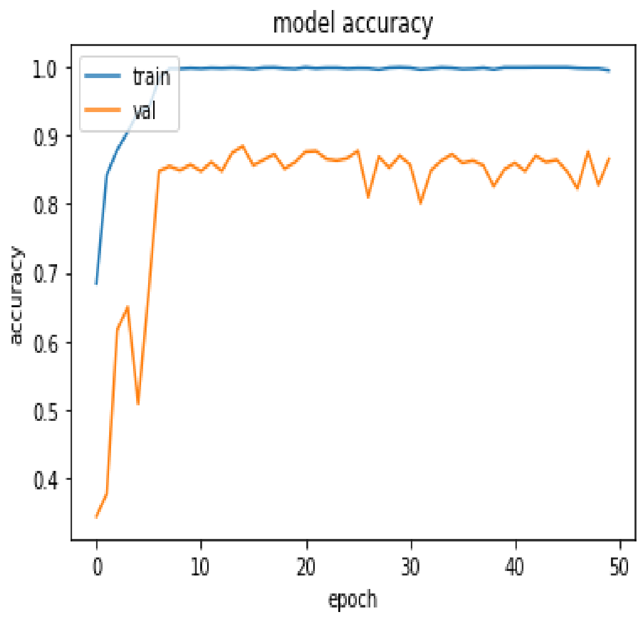
Hyperparameter Tuning: The DenseNet121 was hypertuned to get better performance of the model. The parameters epoch, number of neurons, pooling ,number of trainable layers and weights of the layers were hypertuned.

Keeping all the layers of the Model trainable and randomly initializing the weights did not yield a validation accuracy of more than 60%. Freezing the DensetNet121 layers and keeping the last 2 CNN layers trainable did not improve validation accuracy either. Finally freezing all the layers of the entire model with weights set to ImageNet weights gave the best accuracy. Hence Transfer Learning with DensetNet121 Model with layer.trainable=False was chosen as our final model.

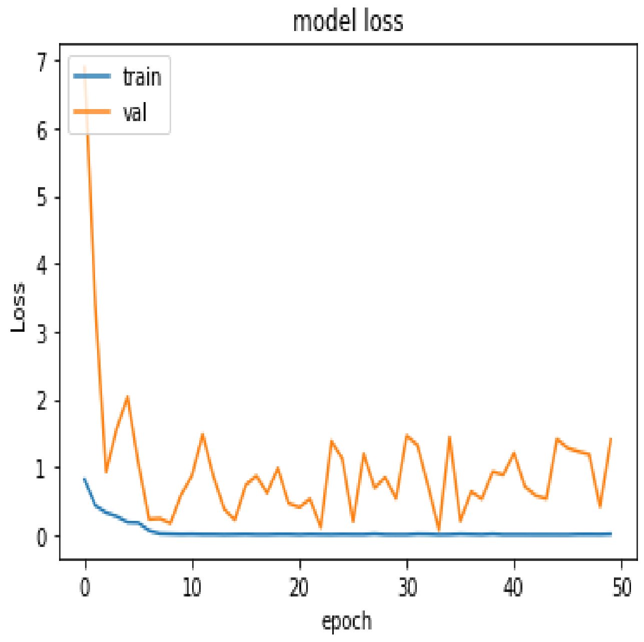
Epoch	Neurons	Pooling	V loss	V acc
10	512,256	Average	0.6599	82.13
40	512,256	Global	0.9867	77.28
50	512.256	Global	1.5432	76.44
50	1024,512	Global	0.2020	84.13
50	1024,512	Average	0.1956	87.76

III. EVALUATION

A. Accuracy Plot



B. Loss Plot



C. Performance Metric Table

Class	Accuracy	Precision	Recall	F1
0	**0.6573**	**0.6344**	**0.6786**	**0.6558**
1	**0.5957**	**0.5786**	**0.6131**	**0.5954**
2	**0.4802**	**0.4646**	**0.5038**	**0.4834**
3	**0.4954**	**0.4799**	**0.5105**	**0.4947**
4	**0.6281**	**0.6032**	**0.6379**	**0.6200**
Average	**0.5702**	**0.5521**	**0.5889**	**0.5610**

REFERENCES

[1] Weiss, K., Khoshgoftaar, T.M. & Wang, D. A survey of transfer learning. J Big Data 3, 9 (2016) doi:10.1186/s40537-016-0043-6.
[2] G. Huang, Z. Liu, L. Van Der Maaten and K.Q. Weinberger, "Densely Connected Convolutional Networks" *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Honolulu, HI, 2017, pp. 2261-2269
[3] https://keras.rstudio.com/
[4]https://medium.com/the-advantages-of-densenet/the-advantages-of-densenet-98de3019cdac
[5] K. He, X. Zhang, S. Ren and J. Sun, "Deep Residual Learning for Image Recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas, NV, 2016, pp. 770-778.