[Team 08] Project C2: Leaf Wilting

Dip Patel
dpatel27@ncsu.edu

Kenil Shah
kshah9@ncsu.edu

Krupal Shah
khshah2@ncsu.edu

I. Methodology

The goal of this project is to use various machine learning based approaches in order to determine leaf wilting ratings. The dataset for this project consists of images of crop taken at various different time of the day under different illuminating conditions. We have a total of 5 classes for the prediction and the amount of wilting increases from 0 to 4 with 0 being no wilting and 4 being severe turgor loss.

Fig. 1: ***Left:*** No wilting. ***Right:*** Severe turgor.

For dealing with this classification task we propose the usage of Convolutional Neural Network. A Convolutional Neural Network (CNN) is a Deep Learning algorithm which can take in an input image, assign learn-able weights and biases to various aspects/objects in the image and be able to differentiate one from the other.

We developed a convolutional neural network model, for this image classification task, using a pre-trained model at the beginning. Our model implements a ResNet-50 [1] network pre-trained on the ImageNet database. The architecture of ResNet50 has 4 stages. The network can take the input image having height, width as multiples of 32 and 3 as channel width. The input image size of the image is 224 x 224 x 3. The ResNet architecture performs the initial convolution and max-pooling using 7×7 and 3×3 kernel sizes respectively. Then, in the Stage 1 of the network, the network has 3 Residual blocks containing 3 layers each. The size of kernels used to perform the convolution operation in all 3 layers of the block of stage 1 are 64, 64 and 128 respectively. The curved arrows refer to the identity connection. The dashed connected arrow represents that the convolution operation in the Residual Block is performed with stride 2, hence, the size of input will be reduced to half in terms of height and width but the channel width will be doubled. As we progress from one stage to another, the channel width is doubled and the size of the input is reduced to half. For each residual function F, 3 layers are stacked one over the other. The three layers are 1×1, 3×3, 1×1 convolutions. The 1×1 convolution layers are responsible for reducing and then restoring the dimensions. The 3×3 layer is left as a bottleneck with smaller input/output dimensions.

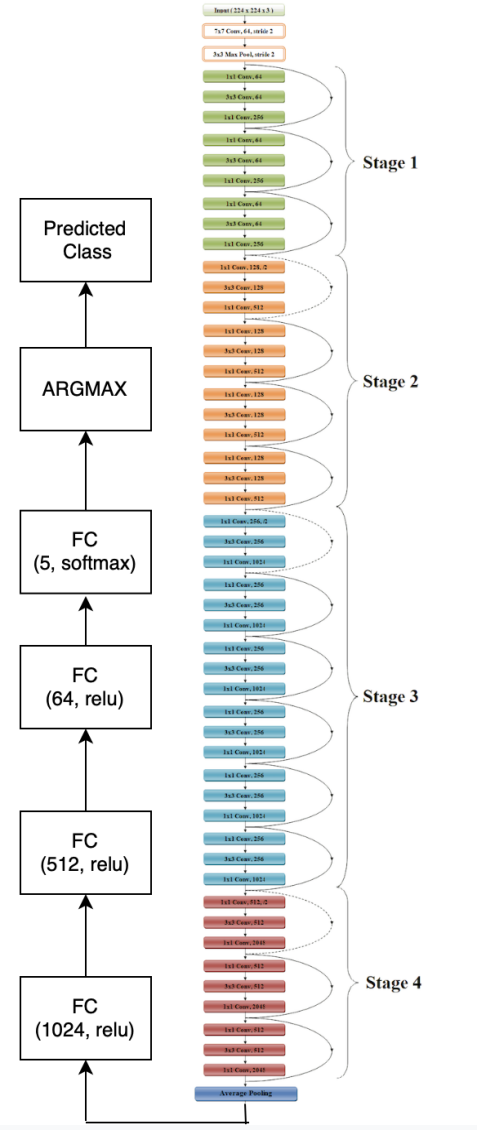

Fig. 2: Model Architecture

We have removed the last softmax layer (which previously outputted 1000 class names), and added a series of fully connected layers with neurons, 1024, 512, and 64, with relu activation function. Finally, we added a fully connected layer with 5 neurons (one for each class) with softmax activation

function. The model is shown in Fig 2. For determining the final class from the classification model we took the class with maximum probabilty score from the softmax layer.

The toolbox and libraries that we have used for this project is as followed:-

Libraries	Functions
sklearn	For generating report, accuracy measure, predicting probabilites, confusion matrix etc
numpy	Image to vector, dumping array to npy, other basic array operations
opencv	Reading the Images
imgaug	For offline Augmentation
csv	For reading the training files from csv
tensorflow/keras	Load pretrained model, tweak hyperparamters, predict the test data using the model, Data generator for training.

TABLE I: Toolboxes used

II. Model Training and Selection

A. *Model Training*

*1) **Training-Validation Split**:* In order to tweak the parameters of the model and decide which one performs better than other one it was inevitable to split the training data into training and validation sets. We used 80% of the training images for training our model and used the remaining 20% as validation to evaluate the performance of our model. Our validation set was comprised of 255 images with 51 images from each class. Due to the balanced validation split, we were able to identify the performance of our model without any bias in the dataset. The split in the dataset and the number of images in each class has been discussed in the below table II:-

Dataset	0	1	2	3	4
Training Split	437	278	79	80	146
Validation Split	51	51	51	51	51

TABLE II: Dataset (Without Augmentation)

*2) **Data Augmentation**:* By analysing the table II, we can identify the imbalance present in the number of images of various classes. The number of images present in the class 0 and 1 are way more than that of class 2,3 and 4. Our Image Augmentation technique was divided into 4 steps in order to deal with this imbalance of classes:-

1) Augment [2] only the images of class 1,2,3 and 4 to remove the imbalance in the dataset.
2) Sample only 437 images from every class as the maximum images is in class 0 with 437 images.
3) Augment the images of every class in order to increase the amount of images present in the dataset, to avoid overfitting.
4) Use runtime augmentation

For increasing the number of images in class 1,2,3 and 4 we used augmentation strategies like horizontal flipping, adding sigmoid contrast, adding log contrast, perspective transforming and shearX. After this set of augmentation the images of class with less images increased and we ended up with more images in class 1,2,3 and 4. So after performing this set of augmentations we used under-sampling technique and selected 437 images from each class as the least number of images was 437 present in class 0.

This amount of images would still be not enough in-order to train a deep neural network and so we ended up using various other augmentation strategies for increasing the dataset size by 5 fold and we ended up with 10925 images in the training dataset. Moreover, we also added runtime augmentation for training our model. The main benefit of using runtime augmentation was that the same image would not appear in every epoch as the inbuilt function would randomly perform augmentation strategies to the images. In this way, we can avoid over-fitting on the dataset as every epoch would be comprised of images with different rotated angles, zoom range, width shift and height shift. The dataset distribution after performing augmentation is discussed in the table III :-

Libraries	Applying augmentation to class 1,2,3 and 4	After under-sampling and offline augmentation
0	437	2185
1	556	2185
2	474	2185
3	480	2185
4	584	2185

TABLE III: Dataset (With Under-sampling and Aug.)

Thus, we ended up training our model on **10925 training images** and **255 validation images**. Moreover, in order to avoid data leaks and redundancy we split the dataset before applying any augmentation strategy.

B. *Model Selection*

*1) **Selecting Model Architecture**:* Primary step for selecting the model would be finalizing the architecture for model. We used different architectures such as VGG16, Resnet50 and Xception. In order to figure out the best model for our dataset, we used the augmented dataset to train each of the above mentioned architecture. Besides the base model, we added 4 fully connected layers to the base model. For the purpose of comparing various architectures we used the following sets of hyper-parameters and the validation accuracy for these set of hyper-parameters are mentioned in table IV :-

1) Batch Size :- 64
2) Epochs :- 20
3) Learning Rate :- 0.001
4) Optimizer :- S.G.D.

Architecture	Validation Accuracy	Validation Loss
VGG16	69.3	0.88
ResNet50	***71.1***	***0.79***
Xception	68.5	0.96

TABLE IV: Validation Accuracy (Architecture Selection)

After this experiment we decided to move forward with the Resnet50 as our primary base model.

*2) **Selecting Hyper-Parameters**:* For exploring the strategy of transfer learning [3], it was necessary for us to identify the number of layers to unfreeze from the pre-trained model. We decided to unfreeze only the last convolution layer from the base model as more trainable parameters would led to our model to over-fit on the dataset. The graph for fully trainable Resnet50 is shown in Fig. III. It can be noted that the training accuracy has hit the 100% mark and the loss has been converged to 0. Thus, we can observe that the model is overfitted on the training dataset and it might end up not performing well on the test dataset. While experimenting with

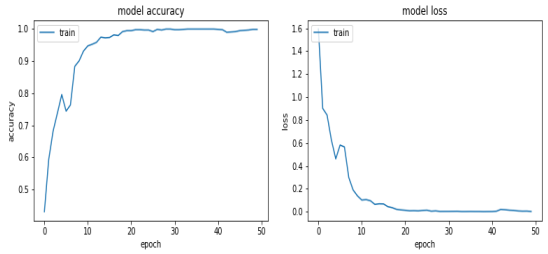

Fig. 3: Over-fitting (Training from scratch)

the dataset, we found that offline augmented dataset + run-time augmentation led to lower validation loss and so we decided to use both the augmentation strategies while evaluating our model. We also trained the model with different batch sizes while keeping other hyper-parameters such as epochs = 50, Learning Rate= 0.001, Optimizer = Adam constant. The results are as noted in the table:-

Batch Size	Validation Accuracy
32	75.4
64	74.5
128	76.8
255	72.2

TABLE V: Batch size experimentation

III. Evaluation

We finalized the following hyper-parameters to train our model:-

1) Batch Size :- 128
2) Epochs :- 50
3) Learning Rate :- 0.001
4) Optimizer :- Adam
5) Base model + 4 F.C. layers with 1024, 512, 64 and 5 neurons. respectively.

We were able to achieve an accuracy score of 76.86% on our validation split. But our model was able to achieve the accuracy of 58% on the testing dataset with a RMSE error rate of 0.72567.

The confusion matrix and the classification report the above stated model has been mentioned in the table VI and VII respectively.

class	0	1	2	3	4
0	45	5	1	0	0
1	18	31	2	0	0
2	4	2	36	8	1
3	1	0	9	39	2
4	0	0	0	6	45

TABLE VI: Confusion Matrix

	precision	recall	f1-score	support
0	0.66	0.88	0.76	51
1	0.82	0.61	0.70	51
2	0.75	0.71	0.73	51
3	0.74	0.76	0.75	51
4	0.94	0.88	0.91	51
accuracy			0.77	255
macro avg	0.78	0.77	0.77	266
weighted avg	0.78	0.77	0.77	255

TABLE VII: Classification Report :- Final model

The training/validation accuracy and loss graph for the classifier is as displayed in Fig. 4

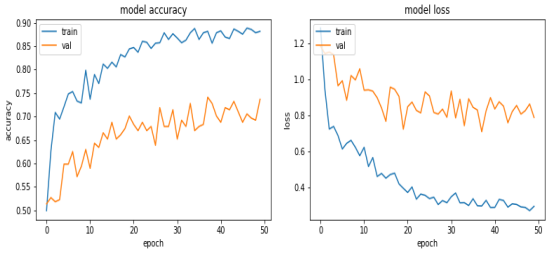

Fig. 4: Final model graphs

IV. Extra Credit: Using Weather Data

A. *Model Specification*

In order to accommodate the weather data as well as the image data we used an ensemble approach. We created two separate models: a base model for the images and another model for the weather data. We tried various machine learning models for the weather data like Support Vector Machines with Linear and R.B.F kernels, K Nearest Neighbours, Decision Tree and Random Forest. We used the testing data in order to identify which traditional machine learning algorithm would be a better choice for weather data, Based on the below stated

Model	Accuracy
SVC with Linear kernel	0.43
SVC with RBF kernel	0.60
K Nearest Neighbours	0.61
Decision Tree	0.65
Random Forest	0.63

results, we decided to move tweak the paramters of namely two approached, Random forest and Decision Tree. We used various methods for the feature selection like PCA, univariate feature selection and a tree based estimator. The tree based estimator can be used to compute feature importance, which

in turn can be used to discard irrelevant features when used with a meta-transformer.

Feature Selection	Model	Accuracy
PCA with top 5	Decision Tree	0.63
	Random Forest	0.65
Univariate	Decision Tree	0.65
	Random Forest	0.66
Tree Based	Decision Tree	0.66
	Random Forest	0.68

As stated in the above table, the accuracy obtained by using Random forest was comparatively higher than that if Decision tree and so we decided to use Random forest as our algorithm. The final model was trained with a balanced dataset, that was generated with re sampling of 338 samples in each class. The accuracy of the final model improved to 0.70.

The classifier was tuned with 3 hyper-parameters: the max-depth of the trees which was set to 5, the number of trees in the forest which was set to 10 and the number of features to split on which was set to 1, rest of parameters were set to default values.

B. *Evaluation*

*1) **Using only Weather data**:* The testing accuracy for the Random Forest Classifier on the weather dataset is 70%. The precision, recall, f1-score and support can be seen below.

	precision	recall	f1-score	support
0	0.54	0.65	0.59	20
1	0.75	0.75	0.75	20
2	0.84	0.80	0.82	20
3	0.67	0.60	0.63	20
4	0.74	0.70	0.72	20
accuracy			0.70	100
macro avg	0.71	0.70	0.70	100
weighted avg	0.71	0.70	0.70	100

TABLE VIII: Classification Report :- Using Weather data

*2) **Using only Image Data**:* The progress in the training can be seen in the accuracy vs epoch and the accuracy vs loss graph. We can see that after the training with 25 epochs the accuracy is 75% and the loss is below 0.75.

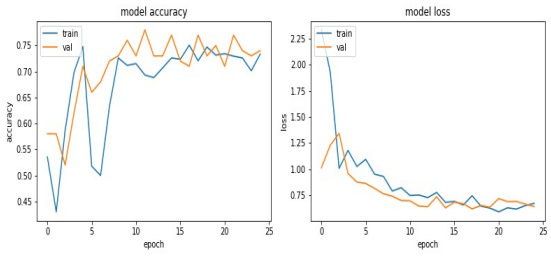

Fig. 5: Model Accuracy and loss graph

The testing accuracy for the Resnet50 Image classification model is 75%. The confusion matrix and the precision, recall, f1-score and support can be seen below.

	precision	recall	f1-score	support
0	0.86	0.90	0.88	20
1	0.90	0.95	0.93	20
2	0.70	0.35	0.47	20
3	0.52	0.85	0.64	20
4	0.93	0.70	0.80	20
accuracy			0.75	100
macro avg	0.78	0.75	0.74	100
weighted avg	0.78	0.75	0.74	100

TABLE IX: Classification Report :- Using Image data

*3) **Fusing Both models**:* For the ensemble approach we assigned weights to the probability outputs of both models and used that sum of weighted probability in order to decide the final output. After trial and error we found that assigning equal weights to each model yields an accuracy of 85% which outperforms the each individual models. The confusion matrix and the precision, recall, f1-score and support can be seen below.

	0	1	2	3	4
0	19	1	0	0	0
1	0	20	0	0	0
2	3	0	14	3	0
3	0	0	2	18	0
4	0	0	0	6	14

TABLE X: Confusion Matrix :- Fusing both Models

	precision	recall	f1-score	support
0	0.86	0.95	0.90	20
1	0.95	1.00	0.98	20
2	0.88	0.70	0.78	20
3	0.67	0.90	0.77	20
4	1.00	0.70	0.82	20
accuracy			0.85	100
macro avg	0.87	0.85	0.85	100
weighted avg	0.87	0.85	0.85	100

TABLE XI: Classification Report :- Fusing both Models

REFERENCES

[1] He, Kaiming, et al. "Deep residual learning for image recognition." Proceedings of the IEEE conference on computer vision and pattern recognition. 2016.

[2] Perez, Luis, and Jason Wang. "The effectiveness of data augmentation in image classification using deep learning." arXiv preprint arXiv:1712.04621 (2017).

[3] Bengio, Yoshua. "Deep learning of representations for unsupervised and transfer learning." Proceedings of ICML workshop on unsupervised and transfer learning. 2012.