[Team 3] Project C2: Leaf Wilting Detection Using Convolutional and Residual Neural Networks

Anh Nguyen
Department of Biological and Agricultural
North Carolina State University
Raleigh, North Carolina, USA
anguyen9@ncsu.edu

Shreeya Singh Dhakal
Department of Computer Science
North Carolina State University
Raleigh, North Carolina, USA
ssdhakal@ncsu.edu

Sneha Sudhir Shetiya
Department of Electrical and Computer Eng.
North Carolina State University
Raleigh, North Carolina, USA
sshetiy@ncsu.edu

I. Dataset Description

The given dataset contains images of soybean crops at various times of the day. The dataset contains 1275 images and annotations corresponding to the level of wilting. The level of wilting is defined from 0-4, 0 means there is no/minimum wilting and 4 means the maximum level of wilting. The distribution of each label in the dataset is given in Figure 1.

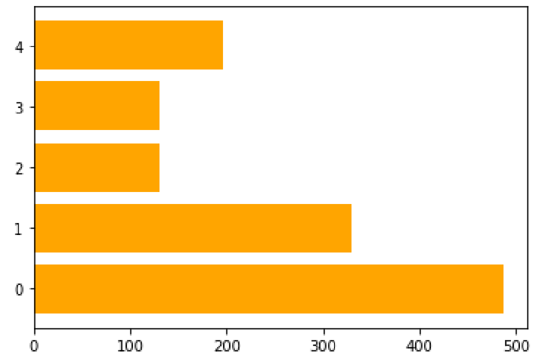

Fig. 1. Distribution of Labels in Training Dataset

II. Methodologies

The objective of this project is to automate leaf wilting ratings of soybean crops, given images of the crop at various times of the day. In this section, we discuss the key components of our model including data pre-processing steps, network structure and model selection.

We experimented with three representations of image: HSV, hue, and RGB. Our models performed better for RGB feature space on the validation data, so we fine-tuned our models on RGB space.

A. Pre-Processing

We resized all the images to 224×224 for faster computation. Given the unbalanced nature of the dataset we performed Image augmentation using ImageGenerator to generate more samples. However, our models did not perform very well with ImageGenerator. So, we used SMOTE (Synthetic Minority Oversampling TEchnique) for oversampling of the training data and Cluster Centroid based majority undersampling on the validation data. We split the data into train-validation sets(70%/30%) before oversampling and undersampling. After sampling, the number of samples total 1675 and 190 for training and validation sets, respectively. Incorporation of oversampling and undersampling gave us an improvement in the model's performance.

1) SMOTE: Oversampling techniques involve synthesizing new examples from existing examples. Although these techniques do not add any new information to the model, they are a good way to create a balanced dataset. Synthetic Minority Oversampling TEchnique (SMOTE) [1] is particular synthesizes new examples from minority class labels.

2) Cluster Centroid Undersampling: Undersampling methods remove instances of majority classes from dataspace to create a balanced dataset. In Cluster Centroid method of undersampling, clusters are created from datapoints belonging to majority classes. The datapoints that is further away from the centroid are considered to be less representitive of the class and hence dropped. The remaining data points make up the dataset, in our case the validation set.

We experimented with three different models: ResNet50, VGG16 and CNN with 4 convolutional layers. All models were trained for 15 epochs with batch size of 32 (unless indicated otherwise in their respective sections).

B. Convolutional Neural Network

Convolutional Neural Network (CNN) is a common artificial neural network structure for the image classification problem. The intuition behind CNN stems from the fact that information in digital images is often repetitive and a local understanding of an image yields adequate inference. Local understanding requires significantly less parameters (compare to fully-connected network), making training faster and more feasible. Due to its versatile and ease of implementation, we constructed a network with 4 convolutional layers as our initial classification model.

General CNN model has two main components: convolutional and pooling layers for features extraction, and output layer for classification. Each convolutional layer in our network uses a small filter of size 3x3 with ReLU as activation function followed by batch normalization. The last three convolutional layers includes pooling of size 2x2 and drop-out to avoid over-fitting and further reducing number of training parameters. Between the feature extractor and the output layer are two dense layers to interpret the features. Figure 1 shows the full detail of our CNN architecture. The model is constructed using Keras built-in library.

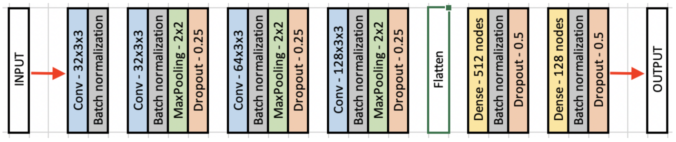

Fig. 2. Architecture of CNN Model with 4 convolutional layers

C. *Transfer Learning with RESNET50*

ResNet or Residual Networks are Artificial Neural Networks (ANN) that allows to train deep neural networks with high performance. ResNet50 is one of the commonly used Residual Neural networks [2] for transfer learning. Due to its high performances in various image classification tasks, we chose this implement this for our task.

ResNet50 consists of five stages and has over 23 million trainable parameters. Each of the stage contains a convolution and an identity block. The convolution block has three convolution layers and the identity block has three convolution layers. Figure 3 shows an architecture of ResNet50 Model[1].

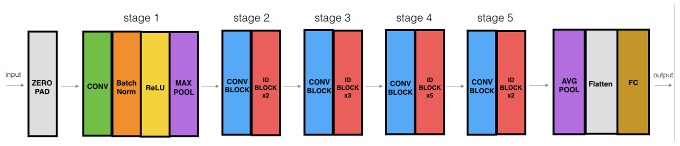

Fig. 3. Architecture of ResNet50 Model

We used a pretrained ResNet50 in Keras. We experimented with Adam and Stochastic Gradient Descent as optimizers. Table 1 summarizes the hyperparameters and layers of ResNet50 we played around with.

D. *Transfer Learning with VGG16*

VGG16 is a popular very deep Convolutional Neural Network [3] architecture for ImageNets [4]. It is sixteen layers deep and has over 14 million trainable hyperparameters. While the number of trainable parameters is less than that of ResNet50, we assumed that VGG16 would perform well given our dataset is small.

VGG16 has only 3*3 convolutions. It contains thirteen convolution layers and 5 max pooling layers. It contains

[1]Image take from: https://towardsdatascience.com/understanding-and-coding-a-resnet-in-keras-446d7ff84d33

TABLE I
HYPERPARAMETER RANGES

Hyperparameters	Ranges
Layers Freezed	None, All except: 2, 4, 6, 10, 12, 14
Learning rates	0.0001, 0.001, 0.01, 0.1
Initial weights	None, Imagenet
Batch size	16, 32, 64, 128
Epochs	15, 20, 50

two fully connected layers and an output layer with softmax activation. Figure 4 shows an architecture of VGG16 Model[2].

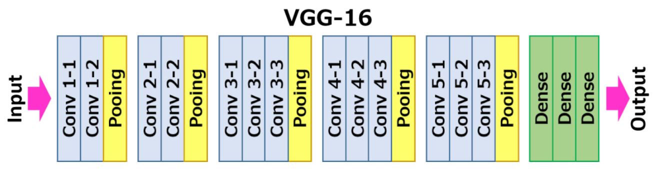

Fig. 4. Architecture of VGG16 Model

We used a pretrained VGG16 in Keras. We experimented with Adam and Stochastic Gradient Descent as optimizers. We used the same ranges of hyperparameters from Table 1.

III. EXPERIMENT AND RESULTS

We submitted the predictions using the CNN model for our first scoreboard and the accuracy and RMSE on the testset was 0.255 and 1.98, respectively. After the poor performance on the first scoreboard we decided to switch to transfer learning and leverage the existing deep learning models that are pretrained on imagenet. We experimented with VGG16 and ResNet50. VGG16 performed worse than our original CNN, while ResNet50 performed better. So we fine-tuned ResNet50 on our dataset.

Figure 5 shows learning curves (in term of both accuracy and loss) of the final ReseNet50 model training on RGB images. Table 2 shows the evaluation metrics for different models for RGB, HSV and Hue features. As can be seen from the figure HSV features were more representative of the image and using HSV features improved the accuracy of the SVM models.

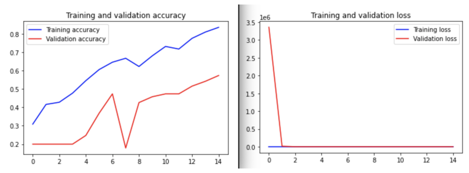

Fig. 5. Learning curves of ResNet50 model training on RGB images

[2]Image take from: https://neurohive.io/en/popular-networks/vgg16/

Overall the CNN model training on HSV images had the best evaluation results. However, when it comes to making prediction on the test dataset whose ground truth is unknown (however, it is known that the class distribution in the test set is much more balanced than that of the given training data), ResNet50 model training on RGB made the most balanced set of prediction with the highest test accuracy (provided by the Professor and the grading Teaching Assistant). For all three network structures, models trained on Hue data performed the worst. The precision, recall, F1-score and accuracy for each model with each type of features are reported in Table 2. We also experimented with ResNet by freezing some of its earlier layers and updated the weights on last few layers only. While the model became more robust and the performance on the validation dataset was not compromised, the results on the scoreboard was less than what we got when we fine-tuned all of its layers. So, for our final predictions, we did not freeze any of the layers in ResNet50.

TABLE II
RESULTS FOR DIFFERENT MODELS FOR RGB, HSV AND HUE FEATURES

Model	**Feature**	**Precision**	**Recall**	**F1-score**	**Accuracy**
ResNet50	RGB	0.65	0.64	0.63	0.64
ResNet50	HSV	0.59	0.56	0.56	0.56
ResNet50	HUE	0.48	0.35	0.31	0.35
CNN	RGB	0.70	0.68	0.68	0.68
CNN	HSV	0.73	0.69	0.69	0.69
CNN	HUE	0.27	0.30	0.23	0.30
VGG16	RGB	0.04	0.20	0.07	0.20
VGG16	HSV	0.38	0.31	0.26	0.32
VGG16	HUE	0.04	0.20	0.07	0.20

IV. CONCLUSION AND FUTURE WORK

In our experiment, we observed that RGB features were more representative of the image data than HSV and HUE features. ResNet50 model training on RGB images did not have the best evaluation results. However, the model was able to capture characteristics for all classes and was more robust than other network structures when applied to test data. A reason for this is that the number of samples in the data given was limited, unbalanced and was not robust enough to represent images seen in the test set. ResNet50 RGB model was able to capture some of the semantic in the images without overfitting. The two other network structures either overfit the training data or fail to learn the data altogether.

Using our final ResNet50 RGB model, we were able to achieve a test accuracy of 0.40, but it is the best that we can attain given time limit and resource limit of the project. This is not a high accuracy score. Potential work that can be done to improve this includes combining oversampling and subsampling with data augmentation to create more robust training data, tuning dropout values, and incorporating regularization to avoid overfitting. In addition, we can experiment with more sophisticated network architectures.

REFERENCES

[1] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[2] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[3] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.