Competition Project: Phase 1

Machine leaning approaches towards classifying  
 visual water-stress found in soybean images

Abhishek Ranjan Singh *Electrical and Computer Engineering*  
*North Carolina State University*Raleigh, NC, USA  
arsingh3@ncsu.edu

Adam Christopher Watts  
*Wilson College of Textiles  
North Carolina State University*Raleigh, NC, USA  
[acwatts4@ncsu.esu](mailto:acwatts4@ncsu.esu)

Sankalp Singh Gaharwar  
*Computer Science*   
*North Carolina State University*Raleigh, NC, USA  
ssgaharw@ncsu.edu

# Data

## Data

A training set consisting of 1025 agricultural soybean photos each with a resolution of 640 X 480 pixels was provided. The training set consisting of 5 classes listed in Table 1. Visual inspection of the photos revealed a wide range of exposure levels with some photos being underexposed and rather dark while others overexposed and rather bright.

##### Table 1. Soybean plant classifications and descriptions.

|  |  |
| --- | --- |
| **Categorical Number** | **Classification Description** |
| 0 | No Wilting |
| 1 | Leaflets folding inward at secondary pulvinus, no turgor loss in leaflets or petioles |
| 2 | Slight leaflet or petiole turgor loss in upper canopy |
| 3 | Moderate turgor loss in upper canopy |
| 4 | Severe turgor loss throughout canopy |

## Augmentation

Image augmentation was performed by using Keras ImageDataGenerator class to synthetically extend the training set to 3075. The first 1025 images were the original training set while the remaining 2050 consisted of 2 complete rounds of augmentation using the parameters found in Table 2. The parameters and values were chosen to emulate the variation found in the provided data set.

##### Table 2. Keras ImageDataGenerator parameters

|  |  |
| --- | --- |
| **Parameter** | **Value / Boolean** |
| Rescale | 1.0/255 |
| Shear Range | 0.1 |
| Zoom Range | 0.25 |
| Horizontal Flip | True |
| Rotation Range | 25 |
| Fill Mode | ‘Wrap’ |

## Image Pre-Processing

Sometimes to identify an object in a particular category all we need to know is the shape of the object and the information containing the color is redundant. In these cases, we can drop the color channel information and still obtain fairly good classification results. Doing so reduces both the time and space complexity of any given model. Having a classification task which is independent of color would have been ideal but the Visual inspection of the data concurs that we do indeed need the information containing the color of images. To make sure we tried training our network on the grayscale input and were only able to reach a stable accuracy of 78%.

Visual inspection of the photos revealed a wide range of exposure levels with some photos being underexposed and rather dark while others overexposed and rather bright. This issue can cause the network to learn some undesirable features. To address this problem, we are going to use histogram equalization on the input image space. Histogram equalization is a type of transformation which preserves the correlation of pixels in an image while taking care of the issue of overexposure and under-exposure.

# Machine Learning Models

## Mixture of Gaussian (MOG)

We first decided to approach problem using Gaussian

Mixture Model (GMM), with that we were able to reach the accuracy of 65% on the validation set. The idea behind using GMM was that with GMM we can learn the likelihood of the model and then we can also take use of the prior which might help us do better on the test set which can have a different distribution from the training set. But since we were not able to reach anything above 65% with GMM we decided to give other models a try and see if we can do any better with them.

## Convolution Neural Networks (CNN)

The first CNN was constricted based on to the LeNet-5 architecture using Tensorflow with the Keras API [1], [2]. The input dimensions in the original LeNet-5 were only 32 X 32 greyscale images compared to the 640 X 480 RGB images used here. Activation functions for this project were modified to Relu instead of the original tanh. The input layer was connected to three, 2-Dimensional convolution layers, followed by being flatten down to two fully dense multilayer perception (MLP) of 16 and 8 neurons respectively for a total of 5 layers. The final output layer was a Stride sizes for all convolution layers were set to 1 with the Kernels decreasing from 11, 5, and finally 3 for respectively for the 3 convolution layers. The number of filters increased for each subsequently deeper convolutional layer with 64, 128, and 256 filters respectively. A maximum pooling layer with a pool size of 2 were placed in between each convolutional layer. 50% dropout was performed between each dense MLP.

Stochastic gradient descent (SGD) with used as the optimization algorithm with a batch size of 16. Training was performed on the training images without any pre-processing or augmentation. A validation split of 15% was utilized with a batch size of 16 without shuffling. Total number of trainable params were 1.7 million.

Poor stability was noticed with the training leading to exploding gradients. The optimization algorithm was altered from SGD to ADAM which lead to stable learning [3]. Although, stability was achieved, the training and validation accuracy failed to improve with each epoch. Training accuracy and validation accuracy was stagnant at 46.84% and 51.95% respectively. The CNN architecture was modified to as shown in Table 3 to improve accuracy.

Accuracy didn’t improve until dropout was completely removed leading to overfitting. Although removing dropout led to overfitting of the training data, it demonstrated that high dropout on can severely impede model learning even when the architecture was drastically changed to almost 80 million parameters. Dropout was restored prior to the Softmax output layer with a 0.20 value, resulting in a training and validation accuracy of 96.10% and 73.38% respectively at 15 epochs.

##### Table 3. CNN architecture for second attempt. C, S, F, D, O stand for convolution, subsampling, flatten, dense MLP, and dropout layer respectively.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Layer** | **Output Size** | **Filters** | **Kernel / Pool** | **Padding** |
| 1 | 640 X 480 | - | - | - |
| C1 | 640 X 480 | 64 | 11 X 11 | SAME |
| S2 | 320 X 240 | - | 2 X 2 | - |
| C3 | 320 X 240 | 128 | 3 X 3 | SAME |
| C4 | 320 X 240 | 128 | 3 X 3 | SAME |
| S5 | 160 X 120 | - | 2 X 2 | - |
| C6 | 160 X 120 | 256 | 3 X 3 | SAME |
| **Layer** | **Output Size** | **Neurons** | **Dropout** | **Activation** |
| F7 | 1,228,800 | - | - | - |
| D8 | 64 | 64 | - | Relu |
| O9 | 64 | - | 0.50 | - |
| D10 | 32 | 32 | - | Relu |
| O11 | 32 | - | 0.50 | - |
| D12 | 5 | 5 | - | Softmax |

Augmented images with their histograms equalized were then added to the training and data. It was hypothesized that the larger data set would improve accuracy as the CNN would have a larger data set to learn from over the same number of epochs. However, both training and validation accuracy deteriorated and could not improve above 50-60%. Batch normalization was added between each dense layer which speed up convergence but didn’t improve accuracy [4]. Furthermore, shuffling of dataset was also performed to minimize the chances that the CNN would “memorize” the data. The augmented images were then removed from the training set to further diagnose the poor accuracy results.

To further optimize the validation accuracy, we modified the CNN architecture by implementing Max Pooling of 4 before reaching the Dense MLP Layers for our CNN. The Dense Layers themselves were fortified by incorporating 4000 units within the first dense MLP layer and 2000 units within the second dense MLP layer. Additionally, we sought to implement our CNN with the more computational power and hence we used a machine with greater GPU memory that allowed us to use increase the batch size to 21, prior it was 16. Other aspects of the CNN such as Batch Normalization between Dense layers and the Dropout were retained as well. By running this modified CNN architecture over 21 epochs, we obtained promising trends as the validation accuracy for our network shot up and started approaching the 80% mark while ensuring that our model was not overfitting on training data.

Our final task was to optimize the CNN parameters to enhance the accuracy of our model to the maximum extent possible. As a part of our experiments with the network, we altered the dropout between the dense layers iteratively to gauge the optimal value. Starting from a dropout value of around 25%, we scaled it up in a stepwise manner and progressively saw the accuracy of the model improve along the way. We finally settled on 50% dropout between the dense layers as our final chosen dropout value.

Additionally, we noticed that our CNN architecture generally trended towards overfitting after the 18th epoch and hence limited the number of epochs to 18. Hence, our final CNN model with histogram equalization on the input images gave performance metrics of 98 % for training accuracy and 85.7% for validation accuracy over the data set.

# Results

Our CNN model tends to be stable as it improves its performance with progressive epochs. We notice that the validation accuracy tends to improve along with the training accuracy, as our model learns the data set better with each passing epoch. Apart from some abrupt drops entailed by random sampling, our model never drops below 50% validation accuracy and tends to achieve a respectable validation accuracy in the range of 80%-85 % as our training accuracy approaches 90%. An important trend to note is that we can sustain this validation accuracy over 8-9 epochs which validates the good performance of the model.

##### Table 4. Final CNN architecture.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Layer** | **Output Size** | **Filters** | **Kernel / Pool** | **Padding** |
| 1 | 640 X 480 | - | - | - |
| C1 | 640 X 480 | 64 | 11 X 11 | SAME |
| S2 | 320 X 240 | - | 2 X 2 | - |
| C3 | 320 X 240 | 128 | 3 X 3 | SAME |
| C4 | 320 X 240 | 128 | 3 X 3 | SAME |
| S5 | 160 X 120 | - | 2 X 2 | - |
| C6 | 160 X 120 | 256 | 3 X 3 | SAME |
| S7 | 80 X 60 | - | 2 X 2 | - |
| C8 | 80 X 60 | 256 | 2 X 2 | SAME |
| S9 | 20 X 15 | - | 4 X 4 | = |
| **Layer** | **Output Size** | **Neurons** | **Dropout** | **Activation** |
| F10 | 76,8000 | 76,8000 | - | - |
| D11 | 4,000 | 4,000 | - | Relu |
| O9 | 4,000 | 4,000 | 0.50 | - |
| D10 | 2,000 | 2,000 | - | Relu |
| O11 | 2,000 | 2,000 | 0.50 | - |
| D12 | 5 | 5 | - | Softmax |

Similar trends are seen for the Validation Loss and Training Loss metrics which tend to decrease with each progressive epoch, and we are able to ensure that both the loss metrics are maintained at values less than 1 over the later epochs of our model.

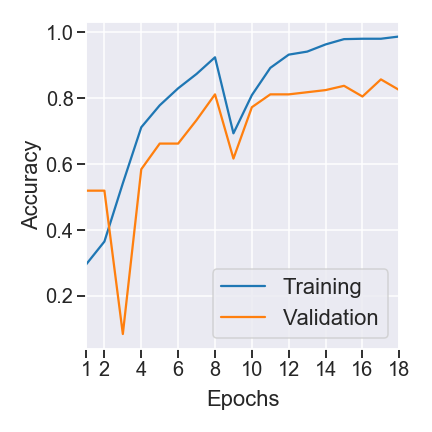


Fig.1. Training and Validation accuracy vs Epochs

Our initial training data set was heavily skewed towards the class-0 label from the data set with minimal representation for classes 1,2,3 and 4. This trend continues in the predicted labels assigned to our testing data where we see the maximum amount of images being labelled as class-0

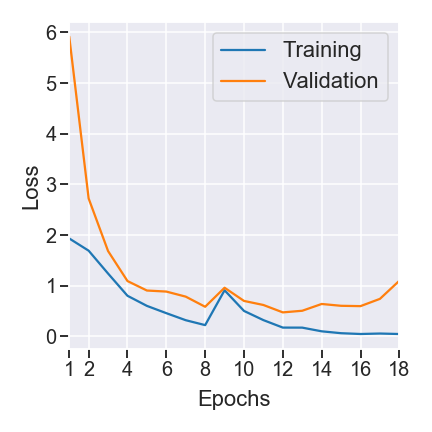


Fig.2. Training and Validation loss vs Epochs

An interesting trend is that a small ratio of testing data images is still labelled as Classes 1,2 or 3, no test data image is assigned the label for Class 4. Our prognosis for this trend is that this may be due to an inherent inequality in the distribution for test data images that mirrors the training data set or we may be limited in our prediction due to selection bias creeping in from the training data set that is evidently highly skewed towards the class-0 and gives minimum representation to class-4.

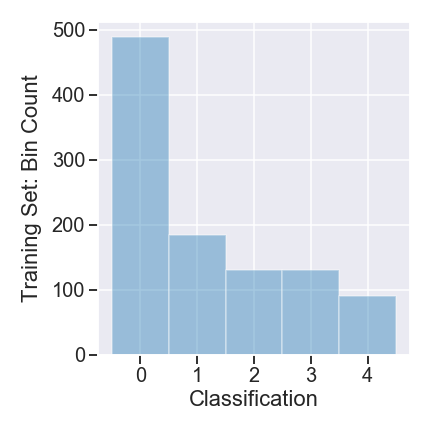


Fig.3. Distribution of training data set vs Class labels

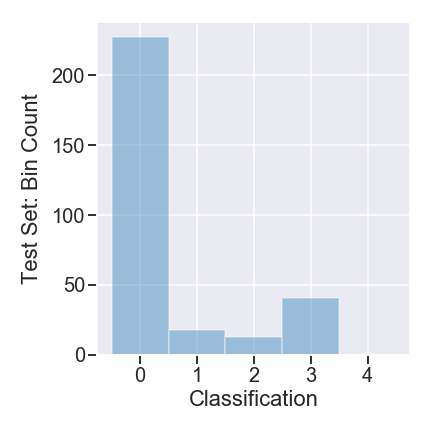


Fig.4. Distribution of test data set vs Predicted class labels

# Conclusion

Capitalize only the first word in a paper title, except for proper nouns and element symbols.

For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation [6].

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Ha, “Gradient-Based Learning Applied to Document Recognition,” p. 46, 1998.

[2] F. Chollet, *Keras*. 2015.

[3] D. P. Kingma and J. Ba, “Adam: A Method for Stochastic Optimization,” *ArXiv14126980 Cs*, Jan. 2017.

[4] A. Géron, *Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O’Reilly Media, Incorporated, 2019.