Hemp Disease Classification Using Machine Learning and Deep Learning

Ziwei Song	Kristie Nguyen	Ashima Malik	Jie Hu
014557231	010631179	014573195	014525602
ziwei.song@sjsu.edu	kristie.nguyen@sjsu.edu	ashima.malik@sjsu.edu	jie.hu@sjsu.edu

Abstract

Hemp crop is used in almost every industry from food to clothing and is prone to infectious diseases. It has been a major challenge to classify the healthy hemp image with the diseased hemp. The main problem we faced was the lack of hemp disease data. To solve this problem, augmentation using Keras image generator library was used on the images. In this research project, different machine learning techniques such as CNN, Logistic Regression, and Random Forest classifiers were applied and then compared to find the best model that can help distinguish between healthy and diseased hemp. We set the Logistic Regression model as a baseline with 70% accuracy for this project. Regarding the Random Forest Classifier, the final model outputs 92% accuracy rate. For deep learning approaches, convolutional neural networks (CNNs) techniques are commonly used to tackle the problem of image classification. Compared with the same training dataset, our project results show that CNN approach achieves the best accuracy with 95 % accuracy.

Keywords: Hemp, disease, Keras, image classification, convolutional neural network, Logistic Regression, Random Forest

1.Introduction

Hemp is a staple crop that is used to produce many hemp-based products in food, beverages, cosmetics, paper, clothing, and building materials. Recently, a provision of the 2018 Farm Bill federally legalized the growth of hemp. As of February 2019, 41 states allowed the cultivation of hemp for commercial, research, or pilot programs [1]. Growing hemp is a laborious task, so it is important for farmers to optimize their supplies. However, diseases prevent these plants from developing healthily and spread an outbreak to another nearby hemp, which causes farmers to lose profits. About 11% of the crop is lost to diseases, and this number does not include pests that also damage hemp [2]. Also, time, money, and supplies can also be lost if a disease is misdiagnosed. To prevent costly infections, machine learning algorithms are developed to detect and classify different types of hemp diseases.

In the field of plant disease detection and classification, the company Medical Genome created a solution called PCR® Plant Screening Platform. This solution requires DNA from the sample of the targeted plant. If the solution turns pink, then it means that it is positive. While this product detects pathogens (a bacterium, virus, or other microorganism that can cause disease), it doesn't classify what the pathogens are whereas the machine learning algorithms do.

In addition to the current business solutions, deep learning and artificial intelligence approaches are also applied. The examples that Artificial Intelligence methods have been used to identify plant disease in agricultural fields are increasing quickly. Compared with traditional manual methods, these systems are more efficient and effective. Sannakki et al. [1] proposed to use k-means based clustering performed on each image pixel to isolate the infected spot. They used the machine vision and fuzzy logic model to build a grading system. The results show that it is very useful to help classify the plant disease. But the shortage is also very obvious that it is very sensitive to noise. Cheng and Matson [2] used three methods to identify weed and rice. Among the Decision Tree, Support Vector Machine (SVM), and Neural Network, the Decision Tree helped them get the highest accuracy 98.2%. But this system only suits for identifying one obvious feature between different plants. Sankaran and Ehsani [3] adopted two ways to identify citrus leaves, containing healthy leaves and infected with disease canker and Huanglongbing (HLB). The first model is quadratic discriminant analysis (QDA), and the second one is k-nearest neighbor (KNN) which helped them get the highest overall accuracy of 99.9%. Both methods have their own merits, the shortage is also obvious which easily arises from the sample imbalance problems.

Machine deep learning could provide a higher accuracy and stable identification, which outperforms above systems. Ferreira et al. [4] proposed to use ConvNets to identify the plants. It proved to be able to find weed in soybean crop pictures and classify these weeds among grass and broadleaf. The best accuracy they achieved is 99.5%. Sladojevic et al. [5] built a deep convolutional neural network to automatically classify and detect 15 categories of plant leaf diseases. Meanwhile, their model was able to distinguish plants from their surroundings. They got an average accuracy of 96.3%. Mohanty et al. [6] successfully identified 14 crop species and 26 diseases by training a deep convolutional neural network based on the pretrained AlexNet and Google Net. The accuracy is 99.35% on a held-out test dataset. Sa et al. [7] utilized deep CNN to detect fruits. They adapted the Faster Region-based CNN (Faster R-CNN) model, through transfer learning. The F1 score is 0.83 in a field farm dataset. Mohanty et al. [8] compared two CNNs methods to identify 26 plant diseases. The dataset contains 14 kinds of plant leaves images. The highest success rates in the model is 99.35%.

The objective of the machine algorithms is to classify images of hemp diseases while achieving an accuracy of 80% or above. In our work, we have utilized two types of CNN model, Random Forest and Logistic Regression to focus on the identification of diseases in Hemp. The models will be trained

with a variety of healthy and infected hemp leaves pictures. The strength of our work is that we have provided model comparison with respect to the accuracy and explain each model under certain use cases. Besides, our models will cover the most common hemp diseases, Fungal Diseases and Bacterial Diseases. In addition, it describes the manipulation processes for the development of the necessary image database, and the techniques used for data augmentation and creation of the training and testing datasets of the developed models, leading to the selection of the final identification model. Finally, we conclude the paper with some relevant conclusions and the plans for future work.

2.Team Contribution

We are a team of four people and each of us has contributed in the data collection, exploratory data analysis, data preprocessing, data visualization and the project report. Apart from this we all have chosen different machine learning models to predict the diseased hemp leaves. Each one of us from the team has contributed to the collection of the data for diseases Grey Mold, Grey Powder, Mosaic Virus and at last the healthy hemp leaves as well. Then each one of us chose a model and trained the models. We chose two kinds of CNN model, Logistic Regression and Random Forest Classifier and together we compared the model results for all these models.

Table 1: Teamwork Distribution

Responsibility	Team-Member Contribution
Data Preparation	Everyone
Data Pre-Processing	Everyone
Data Visualization and Proof of Concept	Everyone
CNN	Ashima Malik
Deep CNN	Jie Hu
Multinomial Logistic Regression	Kristie Nguyen
Random Forest Classifier	Ziwei Song

3. Data Preparation

3.1 Training and Test Data

As part of the data preparation, first we tasked ourselves to gather the data from the web using Google Image Search and as well as, we used the web scraping scripts to collect the various disease images from the web. We couldn't find any handful of web resources/centralized addresses from where we could have grabbed the required images with respect to various disease images as well as healthy hemp images but used various web resources to collect as much data images as possible. The data images included Healthy Hemp and Hemp diseases images categorized by Grey Powder, Gray Mold and Mosaic Virus disease categories. After the data preprocessing, and augmentation, our final tally for the training dataset was around ~16000 images and for the test dataset, the count was ~4300 images.

Table 2: Hemp Class Image Distribution

Class Name	Type of Disease	Image Count	Label Number
Grey Mold	Fungal	874	0
Grey Powdery Mildew	Bacterial	6641	1
Mosaic Virus	Viral	2360	2
Healthy	N/A	10232	3


```
print()
print("X_train :", X.shape)
print("y_train :", np.array(y).shape)
print("X_test  :", X_test.shape)
print("y_test  :", np.array(y_test).shape)

print()
print("Number of training examples =", X.shape[0])
print("Number of testing examples =", np.array(y).shape[0])
print("Image data shape =", X[0].shape)
print("Number of classes =", [CATEGORIES[i] for i in np.unique(np.array(y_test))])
```

```
X_train : (16801, 224, 224, 3)
y_train : (16801,)
X_test  : (4683, 224, 224, 3)
y_test  : (4683,)

Number of training examples = 16801
Number of testing examples = 16801
Image data shape = (224, 224, 3)
Number of classes = ['GreyPowder', 'GrayMold', 'MosaicVirus', 'Healthy']
```

Figure 1. Training Images Examples and Dataset Stats

3.2 Data Preprocessing

As part of the data preprocessing as the images were mainly collected from the web, many images had unwanted information in there, so we manually parsed through the downloaded images and filtered out the images which could not be used to train the model.

Next, as the diseases were the small distortions or the spots on the leaf so, in order to correctly classify the images, we used the image annotation tool such as LabelMe and labeled the images and assigned the correct labels for the various diseases. The generated JSON file was used to generate the masked and labeled images for better feature segmentation.

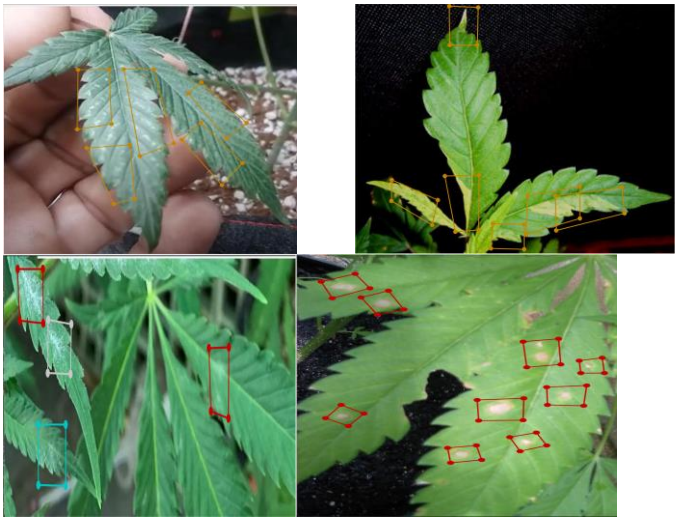


Figure 2. Labeling of the diseases in the Images

As shown in Figure 3, we have masked the background of the images and classified the various image features with respect to the different classification categories that we considered for this project.

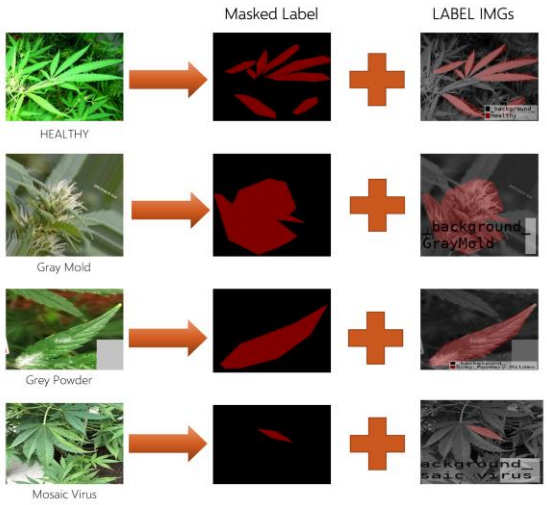


Figure 3. Original Image - Masked + Labeled

Due to limited number of the images availability for the hemp diseases and considering the unwanted features in the background of the images, we performed two levels of augmentation, where in first part, we segmented the hemp leaves which were affected with the disease/s and the healthy leaves then used those images to produce more images with the different rotation, zooming, horizontal/vertical flipping techniques. The mentioned can be referred to in Fig4 and 5.

Figure 4. Bisecting the Hemp Leaves

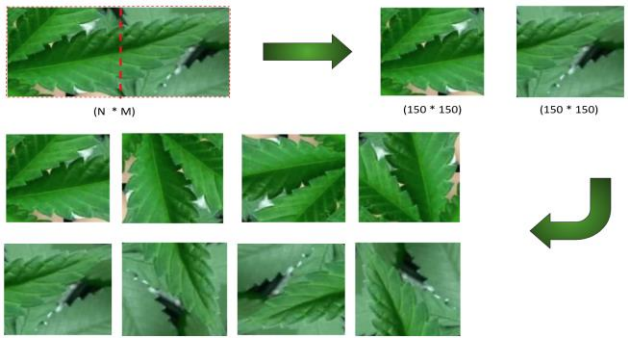


Figure 5. Augmented Images

As part of the process of the augmentation, we have used the complete images as well and produced the augmented images, the augmentation techniques that we have used are as following:

1) rotation_range=40
2) width_shift_range=0.2
3) height_shift_range=0.2
4) shear_range=0.2
5) zoom_range=0.2
6) horizontal_flip=True
7) vertical_flip=True
8) brightness_range=[0.2,0.8],
9) fill_mode='nearest'
10) Noise=.3.

The augmented set of images can be referred to in Figure 6.

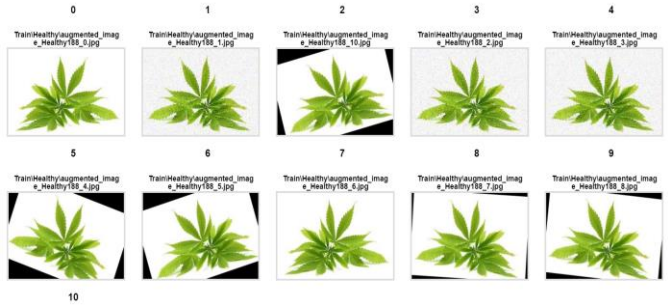


Figure 6. Augmented Images on Full Image

Finally, we performed the normalization on the overall training dataset of collected and augmented images using the formula X = X - Min(X)/Max(X)-Min(X) and it really helped in making the images consistent for the training model.

4. Machine Learning Models

4.1 Convolutional Neural Network (CNN)

4.1.1 Description

For this project, we aim to detect and classify the HEMP diseases and for the same our choice of model is Convolutional Neural Network (CNN). Inspired by human biology, convolutional neural networks (ConvNets) [10], [11] with multi-layer feed-forward architecture that uses the spatial

and invariant features of the image over multiple passes through different stages. Each stage in CNN is typically composed of convolution i.e. sliding window kernel, max pooling and transforming the image data to linear data. In comparison to latest CNNs, the traditional CNNs were used to feed the o/p of the last stage to a classifier. However, the latest CNNs do it in the end of passing through all the stages which allows the classifier to classify low-level features rather than just the high-level features.

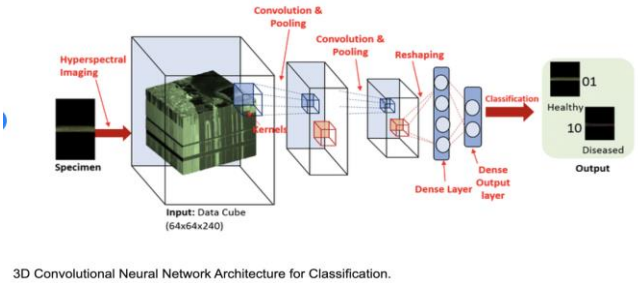


Figure 7. Architecture of CNN Classification

The convolution layer is where image is passed in the form of matrix with pixel values and algorithm further selects the smaller matrix from the complete image matrix (Left most top corner) typically 5×5 or 3×3 to get it multiplied with the filter (or known as the kernel) and summed. This results in producing the convolution i.e. one number value. With the similar operation, by moving the filter along the complete input image by sliding the filter by unit 1 from left to right and from top to bottom, a new matrix is generated which is much smaller than the input matrix.

The CNN consists of several convolutional layers mixed with the non-linear layer and the pooling layer. The non-linear layer is added after each convolution operation, this layer has the activation function that includes the non-linearity property. Non-linearity enables models in classifying the class labels. On the other hand, the pooling layer follows the non-linear layer where image volume gets reduced by down sampling the height and width of the image.

Finally, after a series of convolution, non-linear and pooling layers, output gets attached to a fully connected layer which results in an N dimensional vector where N is the number of classes from which the model identifies the desired class. For this project, we will be using the modified LeNet 5 [10] CNN architecture post HEMP images pre-processing to detect and classify the diseases in HEMP plant images. To create this model, we will employ the following phases: (1) Model Construction, (2) Model Training, (3) Model Testing, (4) Model Evaluation.

For this project the Neural Network is the basis of the algorithm which is implemented in Python. Keras is being used as a framework supported in the backend by Google's TensorFlow. CNN algorithm is as follows:

1) Model object i.e. model = Sequential ()
2) Add the layers to the model with the layer types i.e. model.add(layer_ type())

This model consists of 3 groups of convolutional layers, where the convolution layers (Conv 2D) alternate with the nonlinear layers (Relu) and the pooling layers (Max Pooling 2D). It then follows 2 tightly bound layers (Dense). Conv2D learns a total of 32 filters for the first two layers and from a total of 64 filters for the last layer. These layers use a 3x3 filter size which is the width and height of the convolution window. The activation function for this model is Relu which is f(x) = max (0, x). Max Pooling 2D layer uses 2×2 pool size for the spatial data which halves the input matrix in both the spatial dimensions.

Finally, after 3 consecutive group layers (convolution + activation + max pooling), the 3D output gets flattened in 1D which further gets fully connected to the Dense layer with Relu as activation function followed by dropout to avoid the overfitting. The last fully connected layer has 4 outputs (Healthy Hemp and 3 diseases Grey Powder, Gray Mold and Mosaic Virus) and SoftMax as activation function to classify the Hemp Diseases. For more details, figure 8 is to display the CNN model summary.

For this network, the loss function is opted as "sparse_categorical_crossentropy" and "Adam()" as the optimizer.

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 222, 222, 32)      896
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 32)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 109, 109, 64)      18496
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 54, 54, 64)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 52, 52, 128)       73856
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 26, 26, 128)       0
_________________________________________________________________
flatten_1 (Flatten)          (None, 86528)             0
_________________________________________________________________
dropout_1 (Dropout)          (None, 86528)             0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                5537856
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260
=================================================================
Total params: 5,631,364
Trainable params: 5,631,364
Non-trainable params: 0
_________________________________________________________________
```

Figure 8. CNN Model Summary

For training the model, 80% of the 16,801 training examples are used and 20% of shuffled data is being used for the validation, these training examples comprise all types of the HEMP diseases as well as the disease-free HEMP images. The number of Epochs that we go for is 30 and 32 is used as a batch size. Mitigation of Overfitting:

- L2 Regularizers - In convolutional layers, L2 regularizes of value .001 have been used to avoid overfitting during the feature extraction
- Dropout Layers - Added dropout layers for the hidden layer to drop 50% of the features passed from the flattened layer
- Early Stopping Callbacks - Early stopping callback methods have been used with a patience of 10 Epochs and monitor on 'val_loss'.

4.1.2 Justification

CNN relies on the feature extraction algorithm as part of the Convo Group (convolutional + RELU + Max-Pooling) and as input (M×N×3) progresses down the layers the total number of extracted features keeps extracted where dimension keeps decreasing and the depth keeps increasing and output from the convolutional layers get fully connected to the dense layer where we can control the features we want to connect with how many nodes and this way CNNs get effective in getting the more spatial information and reduces the parameters in play without losing much information about the features to classify the images.

For HEMP Classification, we have high dimensionality which is 224×224×3 and CNN helps in retaining most of the information from the images where with training data examples CNN extracts the information about the edges of objects and hence, CNNs are apt for the image classification problems.

Convolutional layers in CNN, extracts the features as part of passing through the convolutional layers and max-pooling layers helps in dimensionality reduction which helps CNN to be used as feature extractions and classification problems.

Convolutional Neural Network extracts the image features by scanning the whole image feature by feature such as the edges, lines etc. Hence, it gets better feature similarities from the training images than processing a whole set of images. Being neural network architecture, transfer learning in CNN helps it in learning more and reducing the error. CNN can be considered as an algorithm to automatically extract the image features. Down-sampling happens at adjacent image pixel information, first at convolution and then at the prediction layer at the end which improves the accuracy by not considering too many features.

To conclude, for image classification where a load of different params take part in the machine learning problem and hence, it suits to be the perfect problem for CNNs.

4.2 Deep Convolutional Neural Network

4.2.1 Description

For this project, I used CNNs + one layer of fully connected layers as my model to train hemp leaves images dataset and do the testing. Artificial neural networks are one kind of mathematical models. The insights came from the network of neurons in our biological brain. Their main and most important feature is the ability to be trained through the process of supervised learning to perform some particular task by using large amounts of data. During the process, neural networks could "learn" to model some systems through being fed by some specific data, then the system will be modeled. As an advanced form of traditional artificial neural networks, CNNs could get good results from applications such as image recognition. With the methodology used in their layering, CNNs manage to drastically reduce the required number of structural elements (number of artificial neurons) in comparison to traditional feedforward neural networks.

4.2.2 Justification

Computers read images as pixels and it is expressed as a matrix (NxNx3) — (height by width by depth). Images make use of three channels (RGB), so that is why we have a depth of 3. Here my input data matrix is 150 x 150 x 3. The Convolutional Layer makes use of a set of learnable filters. A filter is used to detect the presence of specific features, like the hemp diseases spots, present in the original image (input). It is usually expressed as a matrix (MxMx3), with a smaller dimension but the same depth as the input file.

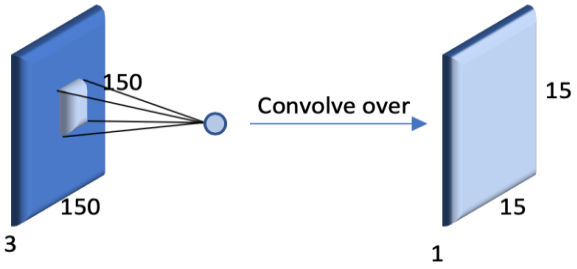


Figure 9. Convolutional Layer

The activation function is a node that is put at the end of or in between Neural Networks. They could help to decide if the neuron net would fire or not.

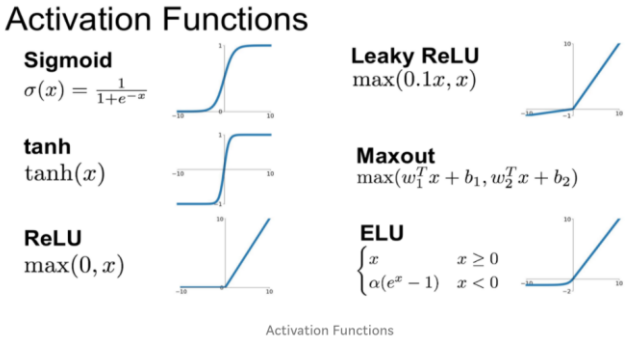


Figure 10. Activation Functions

There are different types of activation functions just as the figure above, but for this project, my focus will be on Rectified Linear Unit (ReLU). ReLU function is the most widely used activation function in neural networks today. ReLU does not activate all neurons at the same time which makes it over other activation functions. From the image for the ReLU function above, we'll notice that it converts all negative inputs to zero and the neuron does not get activated. This makes it very computational efficient as few neurons are activated per time. It does not saturate at the positive region.

The Pooling layer can be seen between Convolution layers in a CNN architecture. The layers help to reduce the number of parameters and then decrease the computation in the networks. At the same time, the layers are useful in controlling overfitting problems by progressively reducing the spatial size of the network. There are two operations in this layer: Average pooling and Maximum pooling. Only Max-pooling will be used

in my project. Max-pooling will take out only the maximum from a pool. This is actually done with the use of filters sliding through the input; and at every stride, the maximum parameter is taken out and the rest is dropped. This actually down-samples the network. Unlike the convolution layer, the pooling layer does not alter the depth of the network, the depth dimension remains unchanged.

Formula for the output after Max-pooling:

(N — F)/ S + 1; where N = Dimension of input to pooling layer

F = Dimension of filter, S = Stride

CNNs are actually made up of some hidden layers and fully connected layer(s). In a fully-connected layer, there is one complete connection to all the activations from the previous layers. Then their activations can be computed with a matrix multiplication followed by a bias offset. This is the last phase for a CNN network.

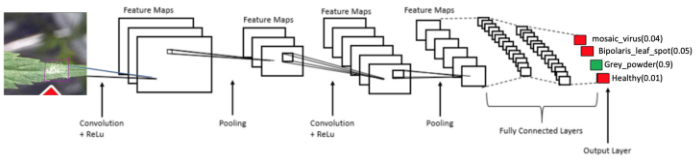


Figure 11. Deep CNNs

4.3 Logistic Regression

4.3.1 Description

A logistic regression model is basically a linear regression model, but it uses a more complex cost function called the "Sigmoid function." The Sigmoid function maps predict values to probabilities, so the hypothesis limits the cost function to values between 0 and 1. In the logistic regression model, there are two types of classification: binary classifications and multinomial classification. Binary classifications only classify 2 classes while multinomial classification classifies more than 2 classes. In this case, this project requires a multinomial classification algorithm since there are more than 2 diseases to classify. For both classifications, the labels must be converted into binary sectors ([01010]). Then, the objective is to minimize the cost value and find the global minimum. To minimize the cost value, a gradient descent is used to update the parameters. Before using the gradient descent, the hypothesis must be computed based on the current parameters. Then, a gradient descent must be computed, which is the hypothesis minus the actual label, multiplied by the transposed inputs [14]. After computing the gradient, the parameters must be adjusted. Finally, the model has to be evaluated by calculating the validation accuracy and looking at the confusion matrix.

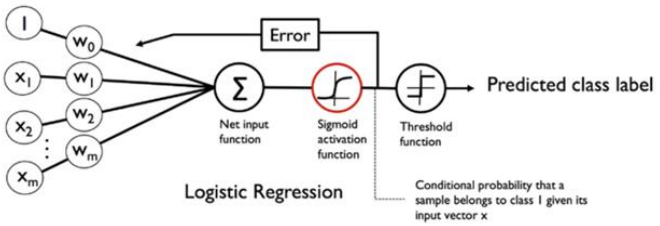


Figure 14. Architecture of Logistic Regression [15]

4.3.2 Justification

Logistic Regression is a widely used algorithm for classification problems; however, most studies use the algorithm where the problems do not require heavy computations. Since this case is an image classification problem, the Logistic Regression algorithm is used as a baseline model. Compared to the other models applied in this study, the Logistic Regression algorithm is the simplest. Also, Logistic Regression does not require standardization and it prevents overfitting. These benefits allow the model to present a rough estimation of the accuracy, making it the ideal baseline model.

4.4 Random Forest Classifier

4.4.1 Description

As the problem we will tackle is to classify different types of hemp disease, it is a supervised, classification machine learning problem. The task requires a model which takes image features as input, and outputs the classification label for each image. Random forest classifier is an ensemble learning technique used for classification problems where it constructs multiple decision trees at training (Figure 12). The objective of building a random forest classifier is to select the attribute with higher information gain as root, which decreases the Gini entropy after each splitting given by Figure 13. The advantage of using random forest classifiers is that the model applies bagging to the data by taking a random sample with replacement. In addition, to avoid overfitting issues, random forest classifier allows bootstrap sampling to the feature variables at each node split which assures feature randomness in the training.

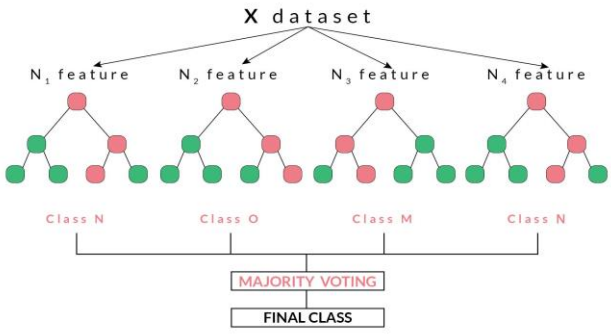


Figure 12. Random Forest Classifier

$$Gain(T,X) = Entropy(T) - Entropy(T,X)$$

- T = target variable
- X = Feature to be split on
- Entropy(T,X) = The entropy calculated after the data is split on feature X

Figure 13. Gini Entropy Formula

4.4.2 Justification

With the bagging and bootstrapping mechanism applied to the classifier, the random forest model balances the bias and variance by reducing the variance of a single decision tree. To justify the performance of the random forest model on this classification task, I chose three approaches to interpret the accuracy. 1) The logistic regression model will be used as a baseline for this task. I will compare my final result with the baseline regarding accuracy improvement. 2) Since proper image preprocessing helps avoid overfitting issues to some extent, I applied standard normalization on the dataset before training. 3) I split 10% of the training dataset as a validation dataset and also tried grid search to fit multiple hyperparameter combinations.

5. Comparative Results and Case Study

5.1 Convolutional Neural Network (CNN)

To test the trained model, we have used 20% of the training set which in terms of the count of the images comes out to be 3361 images. This gave us the validation accuracy of 92%. Below are the plots (Figure 15) for the validation accuracy and the validation loss. It can be seen that the model converged pretty good and considering the high number of augmented data, I think the overfitting mitigation techniques worked pretty well.

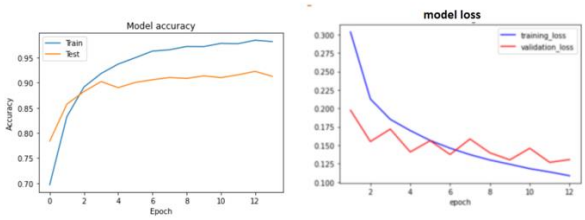


Figure 15. Model Training Validation Accuracy

For testing the CNN model, we used the images which were not included as part of the training/validation examples and used unseen test images. Test data is of a total 4683 images and the same was used to calculate the test accuracy.

For Model Evaluation metrics, we have used classification report, confusion matrices and calculated test accuracy by mismatched predicted vs actual labels for the model.

	precision	recall	f1-score	support
0	0.96	0.93	0.95	1184
1	0.98	0.94	0.96	751
2	0.92	0.86	0.89	700
3	0.93	0.98	0.95	2048
accuracy			0.94	4683
macro avg	0.95	0.93	0.94	4683
weighted avg	0.94	0.94	0.94	4683

Figure 16. CNN Classification Report

It can be observed that with Gray Mold, the model has performed the best with 98% of accuracy, followed by Grey Powder of 96% accuracy, and then Healthy Hemp and Mosaic Virus for which model attained 93% and 92% accuracy respectively.

```
confusion_matrix = metrics.confusion_matrix(y_true=y_test,
                                            y_pred=y_pred)
confusion_matrix
```

```
array([[1106,    0,   30,   48],
       [  10,  709,    0,   32],
       [  21,    6,  601,   72],
       [  17,    9,   23, 1999]], dtype=int64)
```

Figure 17a. CNN Confusion Matrix

For the same, confusion matrix results (Figure 17a) where “Grey Powder” can be seen mis-classified as “Mosaic Virus” and “Healthy” and similarly, same is the case with “Mosaic Virus” and “Healthy Hemp”, this most logical and probable answer to these misclassification is that these disease features are pretty similar and low level and even some noise could be considered as one of the disease but, overall this model attained the test accuracy of 95%.

Mis-classified Category example: In this Example Figure 18, below, it can be seen that the Mosaic virus is spread uniformly over the complete leaf and the model struggles to detect the right disease.

```
In [37]: results = model.evaluate(X_test, y_test)
         print()
         print('Test accuracy: {0:.2f}%'.format(results[1]*100))

         4683/4683 [==============================] - 15s 3ms/step

         Test accuracy: 95.17%
```

Figure 17b. CNN Confusion Matrix

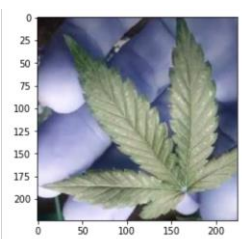


Figure 18. CNN Misclassified Image Example

5.2 Deep Convolutional Neural Network

As introduced before, the final tally data after the data preprocessing and augmentation reached almost 20000 images. But the size of each image is different which is not appropriate for my model. Based on that, I had done the resizing to change all the sizes into 150×150. At the same time unity changed all the image types to "jpg". Then I divided all the dataset to 70% and 30%. As for model training, 70% of the images are used, these training examples comprise all types of the HEMP diseases as well as the healthy HEMP images. 70% of the training dataset will be identified randomly. To test the trained model, I will use 30% of the randomly split dataset, i.e. 6000 images. Then I will get a validation accuracy from it.

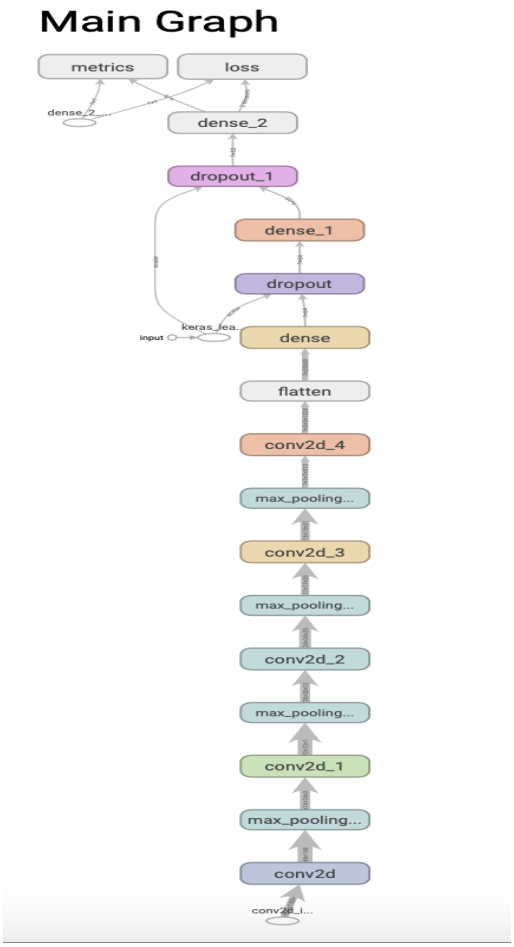


Figure 19. Deep CNNs Model Main Graph

As shown in Figure 20, the model consists of 4 groups of convolutional layers, where the convolution layers (Conv 2D) alternate with the nonlinear layers (Relu) and the pooling layers (MaxPooling2D). Then it is followed by one flatten layer and one tightly bound layer (Dense). In the model I used a learning rate decay. That's because during the learning process the results will back and forth for a long time if without it. So, it could be used to adjust the learning rate to get the accuracy soon. Conv2D learns a total of 32 filters for the first Conv2D and MaxPooling2D. These layers use a 3x3 filter size which is the width and height of the convolution window. The activation function for this model is Relu which is f(x) = max (0, x). Max Pooling 2D layer uses 2×2 pool size for the spatial data which halves the input matrix in both the spatial dimensions.

After above 4 consecutive group layers (convolution + activation + max pooling), the 3D output gets flattened in 1D which further gets fully connected to the Dense layer with Relu as activation function followed by dropout to avoid the overfitting. The last fully connected layer has 4 outputs (Healthy Hemp and 3 diseases Grey Powder, bipolaris leaf spot and Mosaic Virus) and softmax as activation function to classify the Hemp Diseases. The detail was showed in the Figure 20.

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 148, 148, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 72, 72, 128)       36992
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 36, 36, 128)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 34, 34, 256)       295168
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 17, 17, 256)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 15, 15, 512)       1180160
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 1, 1, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 512)               0
_________________________________________________________________
dense (Dense)                (None, 32)                16416
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 4)                 132
=================================================================
Total params: 1,529,764
Trainable params: 1,529,764
Non-trainable params: 0
```

Figure 20. Deep Learning Model

The total parameters used in the model is 1,529,764 which is far fewer than the original model and the layer also decreased a lot. The whole model got simpler, but the results became more accurate. The last result graph shown below:

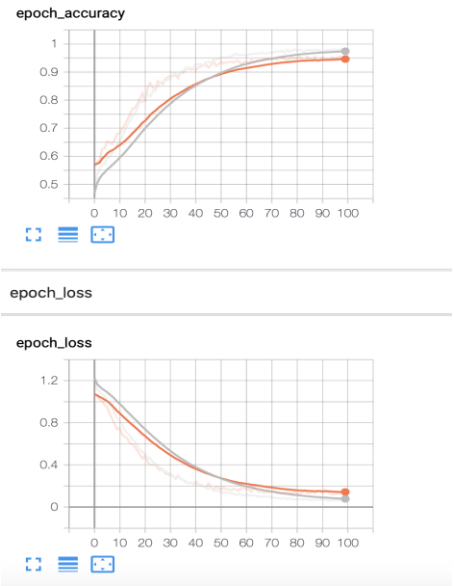
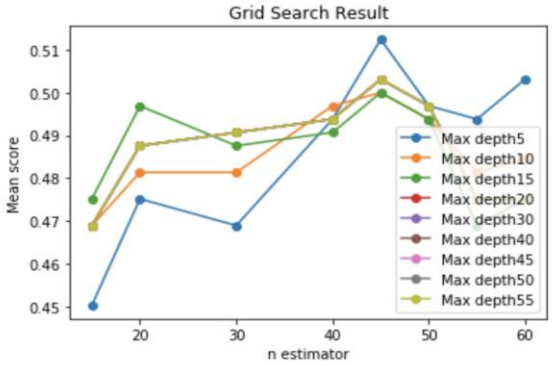


Figure 21. CNNs Model Accuracy

5.3 Random Forest Classifier

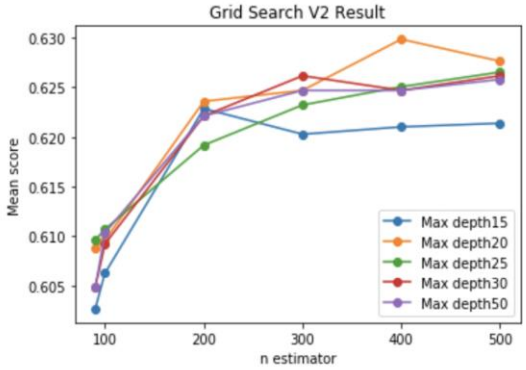
A. Demo Model

As the data preparation section explains the process of building training dataset, tuning hyperparameters on large dataset is not efficient. Instead, I built a training dataset with around 3,000 images for demo. Then I performed grid search validation with 3-fold on two sets of parameter grids. I focused on tuning especially two parameters: max_depth which decides the maximum number of splits, and n_estimators which decides the number of trees. Figure 22a and 22b display the grid search comparison. With a wide range of numbers testing on the two parameters, the model improved slightly on accuracy which implied that it reached the point of diminishing returns of tuning.




```
print (grid_search.best_params_)
print (grid_search.best_score_)
```

```
{'bootstrap': True, 'max_depth': 5, 'n_estimators': 45}
0.5124223602484472
```




```
print (grid_search_2.best_params_)
print (grid_search_2.best_score_)
```

```
{'bootstrap': True, 'max_depth': 20, 'n_estimators': 400}
0.6298200514138818
```

Figure 22a, 22b. Comparison on Grid Search Result

B. Final Results

With performance of grid search cross validation to derive the combination of hyperparameters with the highest score, I fit the classifier again with the augmented dataset. Figure 23 display the shape of training, validation and testing dataset. The model accuracy is 84% on validation dataset and 93% on testing dataset. Figure 24a and 24b are the confusion matrix and classification report.

```
X (16648, 224, 224, 3)
y (16648,)
X_test (4683, 224, 224, 3)
y_test (4683,)

X_val (1665, 150528)
y_val (1665,)
```

Figure 23. Train, Test, Validation Dataset Overview

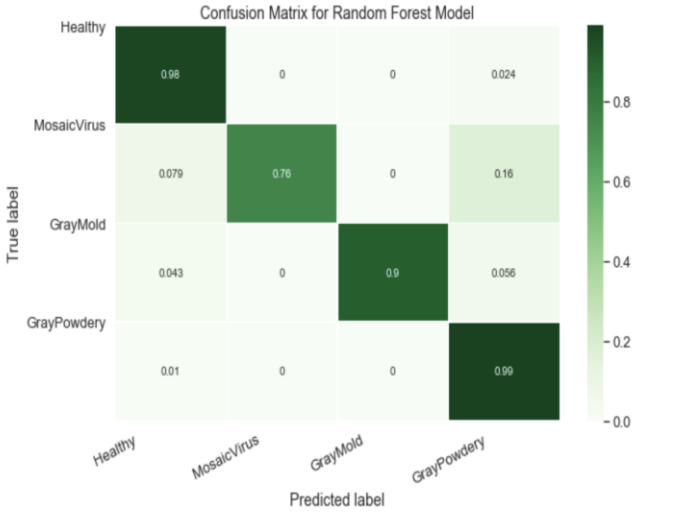


Figure 24a. Confusion Matrix

```
Accuracy:
 0.9359385009609225
Confusion Matrix:
 [[1156    0    0   28]
 [  59  569    0  123]
 [  30    0  631   39]
 [  21    0    0 2027]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      1184
           1       1.00      0.76      0.86       751
           2       1.00      0.90      0.95       700
           3       0.91      0.99      0.95      2048

    accuracy                           0.94      4683
   macro avg       0.96      0.91      0.93      4683
weighted avg       0.94      0.94      0.93      4683
```

Figure 24b. Classification Report

C. Tree Visualization

With the final setting of the random forest classifier, I used the graph viz library to visualize the model with 150528 features in total. Figure 25a displays the general view of all trees where figure 25b is a close examination on some single trees.

Figure 25a. Random Forest Visualization

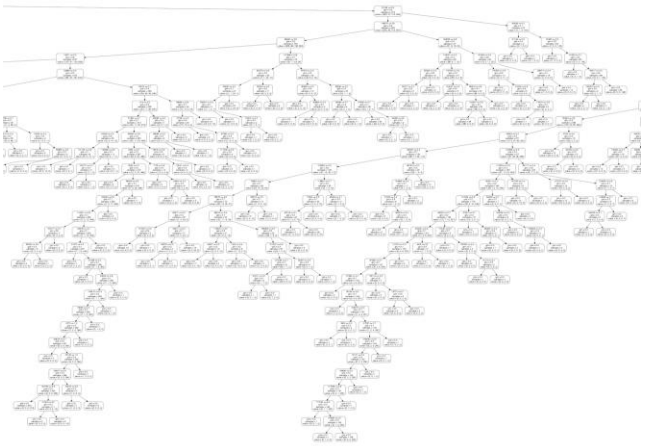


Figure 25b. Details on some trees

5.4 Multinomial Logistic Regression

After running the training dataset through the model and fitting multiple hyperparameter combinations, the overall accuracy came out to be 71%. Also, validation accuracy came out to be 69%. According to the confusion matrix in Figure 26a, the classes Grey Powdery Mildew and Grey Mold have the most correctly classified images. The class Mosaic Virus had the highest misclassified images. The images of Mosaic Virus were incorrectly classified as Grey Mold. As shown in the classification report in Figure 26b, the class Grey Mold showed the best accuracy with the accuracy of 82% compared to the other classes. Even though Grey Mold had the least number of images, it still did the best. A possible reason why it did the best is the number of images augmented. While the other classes had a greater number of augmented images that contributed to the total image count, their precision were still lower than Grey Mold's. To check how effective the model is, a Receiver Operating Characteristic Curve (ROC AUC) score is calculated. The score came out to be .79, which has acceptable discrimination between the different classes.

Figure 26a. Confusion Matrix Visualization

```
print(metrics.classification_report(y_test, y_pred))
```

	precision	recall	f1-score	support
Grey Mold	0.82	0.88	0.85	205
Grey Powdery Mildew	0.66	0.57	0.61	166
Healthy	0.58	0.50	0.54	161
Mosaic Virus	0.67	0.76	0.71	218
avg / total	0.69	0.69	0.69	750

```
print("Multinomial Logistic regression Test Accuracy :",
    metrics.accuracy_score(y_test, mul_lr.predict(X_test_std)) *100)
```

```
Multinomial Logistic regression Test Accuracy : 70.66666666666667
```

Figure 26b. Classification Report with Total Accuracy

6. Comparison of Different Models

6.1 Evaluation of Classification Results by classification report

- The class Grey Mold had the best precision accuracy compared to the other classes.
- The classes Mosaic Virus and Healthy have the lowest.

Table 3: Accuracy Percentage by Class

Model	Positive			Negative
	Grey Mold %	Grey Powdery	Mosaic Virus%	Healthy %
CNN	98	96	92	93
Deep CNN	95	93	92	96
Logictic Regression	82	66	58	67
Random Forest	100	91	100	91

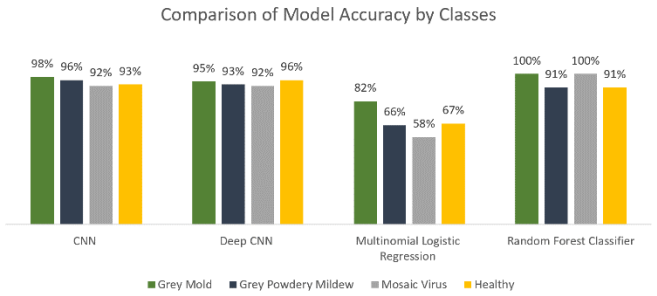


Figure 27. Comparison of Model Accuracy by Class

6.2 Evaluation of Model Accuracy on validation/test dataset

- Multinomial Logistic Regression had the lowest validation and test accuracies, it served as a good baseline model in comparison to the other models.
- The validation and test accuracy for CNN performed the best as expected.
- The accuracies for Deep CNN and Random Forest Classifier closely follow behind CNN's.

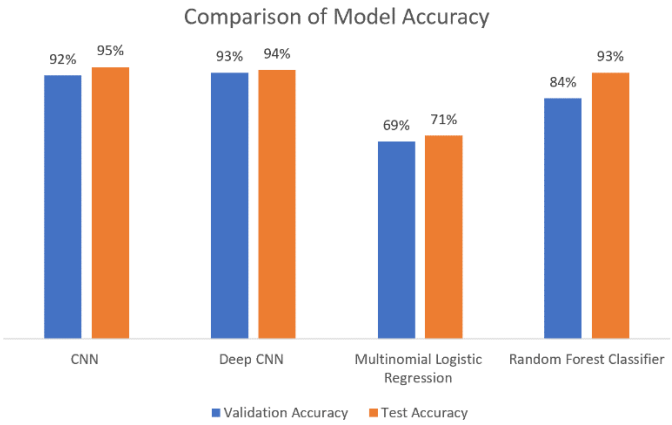


Figure 28. Evaluation of Classification Result

7. Lessons learned and Challenges

During this project, our team learned that data collection and preprocessing take up the most time and are the most important steps for a machine learning project. Although there were a number of photos on the web for each hemp disease, the number of images was not enough to achieve the optimal accuracy. Data collection was probably one of the hardest tasks of the project because we had to make sure there were not any duplicates and they had to be the correct disease. If we accidentally classify the wrong image of disease, then the algorithm has a high probability of making the same mistake. Those were some tips we had to keep in our heads as we collected data. Due to the lack of original photos on the web, we had to do some data preprocessing, which took a good amount of trial and error. For instance, we first experimented running our models with 6 classes; however, we achieved a much lower accuracy. To improve that, we decided to remove the 2 classes with the lowest precision percentages. This was just one of the adjustments we made. We had to do other adjustments such as rotations and mirroring to gather a larger dataset and improve the accuracy.

References

[1] B.Woods, “Newly legalized hemp industry set to create a jobs boom in the US,” 2019.

[2] D. Samanta, P. P. Chaudhury, and A. Ghosh, “Scab diseases detection of potato using image processing,” International Journal of Computer Trends and Technology, vol. 3, pp. 109–113, 2012.

[3] P. J. Herrera, J. Dorado, and Á. Ribeiro, “A novel approach for weed type classification based on shape descriptors and a fuzzy decision-making method,” Sensors, vol. 14, no. 8, pp. 15304–15324, 2014.

[4] B. Cheng and E. T. Matson, “A feature-based machine learning agent for automatic rice and weed discrimination,” International Conference on Artificial Intelligence and Soft Computing, pp. 517–527, 2015.

[5] S. Sankaran and R. Ehsani, “Comparison of visible-near infrared and mid-infrared spectroscopy for classification of Huanglongbing and citrus canker infected leaves,” Agricultural Engineering International: CIGR Journal, vol. 15, no. 3, pp. 75–79, 2013.

[6] A. dos Santos Ferreira, D. Matte Freitas, G. Gonçalves da Silva, H. Pistori, and M. Theophilo Folhes, “Weed detection in soybean crops using ConvNets,” Computers and Electronics in Agriculture, vol. 143, pp. 314–324, 2017.

[7] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, and D. Stefanovic, “Deep Neural Networks Based Recognition of Plant Diseases by Leaf Image Classification,” Computational Intelligence and Neuroscience, vol. 2016, Article ID 3289801, 11 pages, 2016.

[8] S. P. Mohanty, D. P. Hughes, and M. Salathé, “Using deep learning for image-based plant disease detection,” Frontiers in Plant Science, vol. 7, article no. 1419, 2016.

[9] Image Preprocessing. (n.d.). Retrieved from https://keras.io/preprocessing/image/

[10] LeCun, Y, Bottou, L, Bengio, Y, and Haffner, P. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11):2278–2324, November 1998.

[11] Jarrett, K, Kavukcuoglu, K, Ranzato, M, and LeCun, Y. What is the best multi-stage architecture for object recognition? In Proc. International Conference on Computer Vision (ICCV’09). IEEE, 2009.

[12] A. S. Tulshan and N. Raul, “Plant Leaf Disease Detection using Machine Learning,” 2019 10th International Conference on Computing, Communication and Networking Technologies (ICCCNT), 2019.

[13] S. Rathore, M. Hussain, M. A. Iftikhar, and A. Jalil, “Ensemble classification of colon biopsy images based on information rich hybrid features,” Computers in Biology and Medicine, vol. 47, pp. 76–92, 2014.

[14] M. Lind, "Using Logistic Regression to Classify Images", Mmlind.github.io, 2017.

[15] Raschka, S.; Mirjalili, V. Python Machine Learning, 2nd ed.; Packt Publishing Ltd.: Birmingham, UK, 2017