“Plant Seedlings Classification”

By

Jay Joshi (200440993)

jjm376@uregina.ca

Dhruval Patel (200438935)

Dpp504@uregina.ca

Of

Faculty of Graduate Studies and Research

University of Regina

Abstract: Agriculture plays a vital role for continuing human existence and is a prominent contributor to economy in under developed as well as developing countries [1]. The demand for food and cash crops is continuously increasing due to rise in world population, hence there is urgent need to increase crop production with reduced cost. Instrument vision methods used previously has faced a lot of challenges for efficient and trustworthy weed recognition. In this paper, plant seedlings classification approach is presented using a data set that contains approximately 5000 images of 12 unique plant species at their different growth stages. Convolutional Neural Networks with fine hyper parameter tuning technique is developed from scratch to classify plant seedlings from their corresponding images. We have used three hyper-parameter tuning for experimental purpose namely Random Search, Bayesian Optimization and Early Stopping. The experimental results shows that hyper parameter tuning with Bayesian Optimization gives best generalization accuracy of 85% in optimal period of time. Hence, plant seedlings classification using CNN deep learning technique is robust and has improved crop productivity and harvest techniques if implemented correctly.

List of Keywords: Plant Seedlings Classification, Deep Learning, Convolutional Neural Networks, Random Search, Bayesian Optimization, Early Stopping

1. Introduction

Weed management and control is extremely necessary in modern farming, since the damage of crops because of poor weed control techniques is significant (23% -71% on crops and location). Since, on field herbicides spraying is ban in modern agriculture, it is extremely difficult to implement on site weed control techniques. Image processing techniques were used earlier and has significantly shown better results.

This project proposes one approach of designing a deep convolutional neural network from scratch using three hyper-parameter tuning techniques i.e. Random Search, Early Stopping and Bayesian Optimization to classify crop seedlings from their corresponding images. The main concept of using deep convolutional layers is to build a stack of self-learned features due to which deep convolutional layers are less affected by varied colors and compositions of crops. Hence, our project is successful in achieving the main purpose i.e. to classify plant species despite of their varying colors, compositions and features. Also, the ability of deep neural networks to self-learn the features of new plant species from their images eliminates the need of implementing a feature descriptor.

This paper is organized in appropriate divisions and sub divisions so readers can easily comprehend our project. The section 2 of this report will introduce our readers with the possible problem statements that we are trying to solve. Followed by this section 3 will provide a broad overview of all the literatures that we have referred for designing our system. Further, in section 4 we will discuss in detail about various phases of system design including data set collection, implementing CNN architecture and various types of hyper-parameter tuning techniques. Section 5 briefly illustrates our experiment using fine tuning techniques and later we compare results through graphs and tables wherever required. Lastly, we conclude our project by proposing few possible extensions on our limitations.

2. Problem Statement

Since many years, scientist are trying to build systems which aims at performing site specific weed controls. The different techniques used are, real time precision spraying using ground based platforms equipped with high resolution imagery sensors to constructing range weed maps with remote sensing data. All the approaches mentioned above are trying to detect weeds either in patches or from single plants. Although there are many commercial techniques available for classification, but no such techniques can ensure promising results because general approach which enables effective classification is yet to be discovered. Botanists are dealing with species classification since many years and substantial development is seen in content based image retrievals and their respective analysis [1]

After successful implementation of this project we will be able to successfully answer questions like, ***Can we differentiate a weed from a crop seedling?*** The ability to perform this task successfully means better crop yields and better management of environment. Hence our deep CNN architecture will be able to classify plant species from its corresponding images.

For example,

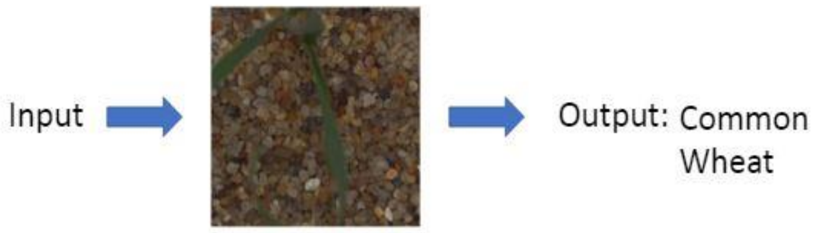


Fig 1 Input / Output Example

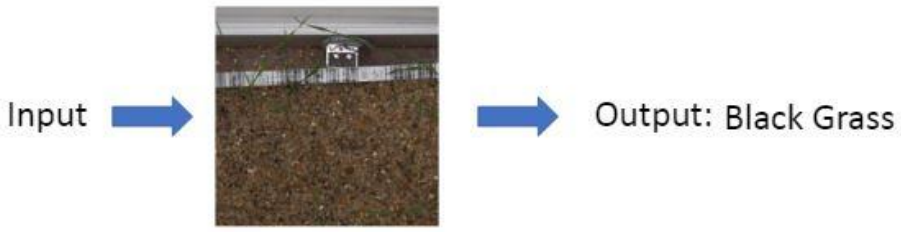


Fig 2 Input / Output Example

3. Literature Review

Deep learning technique using Convolutional Neural Network was presented in [2] which is applied on similar data set that we have considered for our research. They have used pre-trained VGG-NET16 CNN architecture for fine tuning and training layers from scratch. They have trained their model on two different data sets, one balanced data set which was achieved by data augmentation and one unbalanced data set and achieved accuracy of around 99% in both the models [2].

A deep neural network approach was applied on data set consisting of 10,000 images of 22 plant species in [3]. They have considered images in two different forms, one set consist of images captured by using camera stabilisation methods and the other set consists of images from hand-held mobile phones under different lightning and soil conditions. The overall accuracy achieved was 86.4% [3].

A Philippine plant seedlings classification system consisting of 5 plant species was studied in [4]. Pre-trained CNN models namely GoogleNet and RestNet were used for building CNN architecture. The final model gave an overall accuracy of 90% on test data set [4].

A 26-layer CNN model was developed in [5] to classify 100 plant species in natural environment. The resulting model gave an accuracy of 91.48% on unseen data.

An improvised deep learning technique was studied in [6] to train a 7-layer CNN and RNN model for classifying 10 different plant species. The experimented lasted for more than 2 hours and accuracy of model was 90.41% on test data. [6]. Table 1 summarizes all the literature works in brief.

Source	Number of Plant Species	Accuracy	Approach
Plant Seedlings Classification Using Deep Learning [2]	12 plant species (same as our data set)	99.48% on hold out test set	Convolutional Neural Network
Plant species classification using deep convolutional neural network [3]	22 species	86.4% on test set	Convolutional Neural Network
Philippine Indigenous Plant Seedlings Classification Using Deep Learning [4]	5 Philippine plant species	90% overall accuracy on validation set, 98% accuracy with Rest Net model	Fine tuning of pre-trained models like Google Net, Rest Net
Deep Learning for Plant Identification in Natural Environment [5]	100 plant species in natural environment	91.48% accuracy	26-layer Convolutional Neural Network
An Improved Deep Neural Network for Classification of Plant Seedling Images [6]	10	90.15% on test data set	7- Layer CNN as well as RNN

Table 1 Plant Seedlings Literature Review

4. Proposed Methodology

4.1 Data Set Description

Data for plant species classification is collected by the Signal Processing group at Aarhus University has released a data set of 960 unique plants consisting of 12 different plant species at several different growth stages. This dataset is available at following link Dataset-Link. The different plant species are Black-grass, Charlock, Cleavers, Common chickweed, Common wheat, Fat hen, Loose silky-bent, Maize, Scentless Mayweed, Shepherds Purse, Small-flowered Cranesbill, Sugar beet. The data set is in png Image format and is of total 2 GB.

4.2 Design Considerations

- **Data Augmentation.**

In order to get best accuracy out of possible examples available for training set, we have used data augmentation to balance the data set. To achieve this, we have augmented data with various random transformations such as re-scaling, rotating images up to 180 degrees, performing horizontal as well as vertical shifts, flips etc. Data Augmentation also prevents overfitting because smaller data sets are more prone to overfitting. Hence, this improves the models efficiency to generalize on unseen data. Fig 3 and Fig 4 respectively visualizes number of classification labels in each of 12 species before and after performing data augmentation.

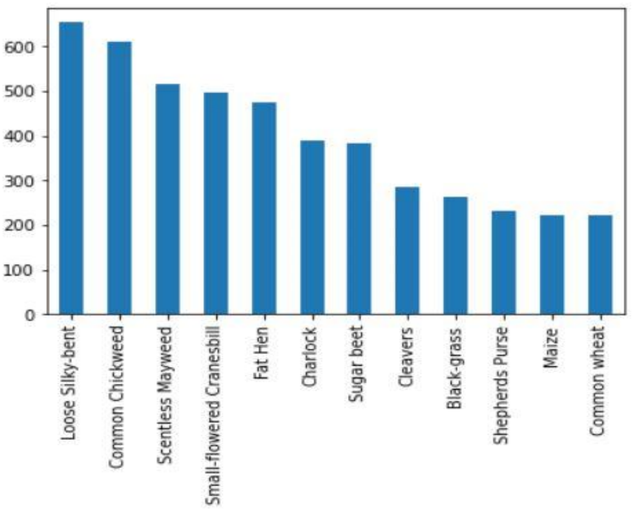


Fig 3 Data Visualisation before Augmentation (Imbalanced data set)

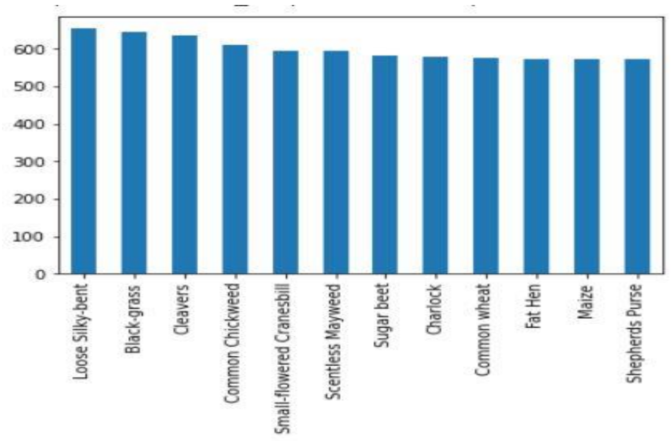


Fig 4 Data Visualisation after Augmentation (Balanced data set)

- **One-Hot Encoding**

Many machine learning algorithms only work when output data is of type numerical. So, to convert ordinal (categorical variables) into numerical variables “one hot encoding” is used. Since, integer encoding can result in poor performance because of pre-defined order, one hot encoding assumes natural ordering between categorical values, and represents each value with a binary vectors of 0’s and 1’s. Here each image corresponds to one index or element in the vector and the uplink of the vector depends upon the total number of images provided as an input. Below is the example of one hot encoding

[1,0,0] [0,0,1] [0,0,1]

- **Data Pre-processing**

The idea of Image Segmentation is presented in [1] using naïve Bayes classifiers and stratified sampling. Since, we are building our system in local cloud environment, image segmentation using naïve Bayes algorithm will require powerful machines. So, we have considered slightly different approach for isolating green plants and background soil. We have considered a similar approach that is described in [7] to differentiate green plants from background soil. One conclusion of this study was images consisting of green plants as foreground than Ex-G (Excessive Green) is a threshold and Ex-G – Ex-R being boiled to G-R acts as a static threshold of 0 and this method performed well with accuracy of 0.87 [7]. We have considered following steps for masking and removing background noise

- Normalizing pixel values in range [0, 1] for better training of CNN model.
- Re-sizing Images in 70x70 pixels to reduce hardware stress during training of our model.
- **Removing background noise with Gaussian blur**

Gaussian blur is a type of image blur technique in which it reduces the components of high frequency, hence Gaussian blur is a low pass filter. It convolve an image with a Gaussian function. Hence it calculates transformation to apply on each pixel in an image. Two dimensional Gaussian distribution is given by,

$$G(x,y) = 1\Big/\sqrt[2]{2\pi\sigma^2}\; e^{-x^2+y^2\big/2\sigma^2} \qquad (1)$$

Were n is two dimensions, it is the product of two such Gaussian functions, one in each dimension, where *x* is the distance from the origin in the horizontal axis, *y* is the distance from the origin in the vertical axis, and *σ* is the standard deviation of the Gaussian distribution [9]

- **Conversion of RGB to HSV Space**:

HSV (Hue, Saturation, and Value) is an alternative for RGB space. It is a cylindrical model that converts RGB colors into dimensions that is interpreted easily by human eye. In Fig 7, Hue is a angular dimension, were red primarily starts at 0^o, with green at 120^o and blue at 240^o degrees respectively. Since color range is easily represented in HSV space especially ExG, we have converted RGB dimension into HSV space.

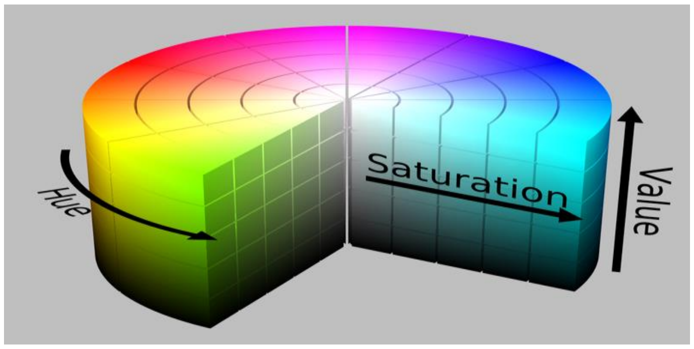


Fig 5 HSV Cylindrical Space [8]

- **Creating Mask:**

Masking is a most common way of defining that whether a segment of image is in use or not. After conversion into HSV dimension we create a mask by empirically selected green range from HSV dimension. Maximum green value ranges in (75, 255, 255) and Minimum green value ranges in (25, 40, 40). Hence, mask is created on green plant whose pixel value lies in above range. The concept of masking helps in saving a lot of training time.

- **Boolean Mask:**

It is used to isolate a green plant seedling region from background soil.

Fig 6 below indicates different stages of image pre-processing steps, from applying Gaussian blur to Boolean masking in pre-processing stage and Fig 7 visualizes some crop seedlings after isolation of background soil and green plants in all images.

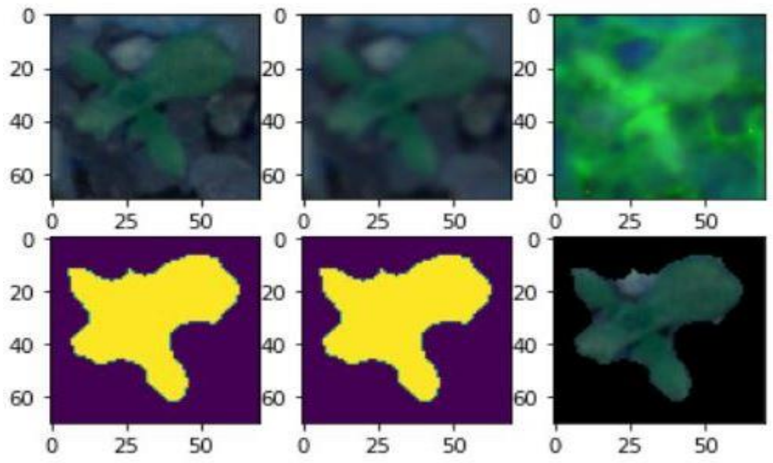


Fig 6 Different stages in image pre-processing

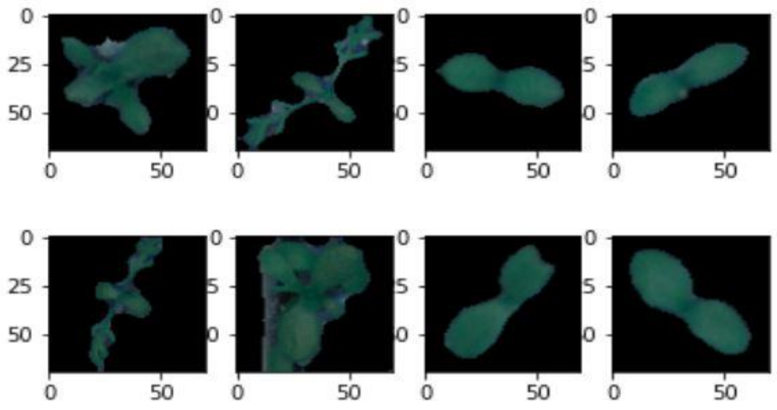


Fig 7 Plant Species Visualisation after pre-processing

4.3 *Model Architecture*

CNN is a well-documented and globally used architecture for classification of images. CNN are most likely as neural networks, where each neuron receives input and produces a class probability for classification. The whole network still perform single differentiable function i.e. transforming pixel by pixel information from images into class probabilities by connecting through some non-linear layers in between. Given below, figure gives an overview of CNN architecture.

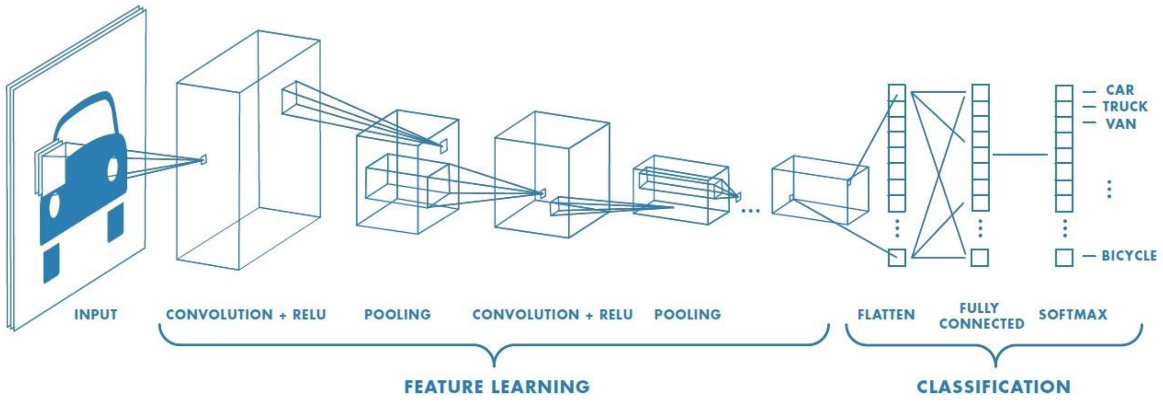


Fig 8 CNN Architecture [11]

The main building blocks of CNN architecture are as follow

- **Convolutional Layer**:

Convolution filters are used to extract certain kind of information from images, like detecting vertical edges, horizontal edges etc. Output from this layer will be of size (70, 70, 12) if we decide to use 12 convolutional filters

- **Max Pooling Layer**:

Max pooling operation refers to downward sampling. It reduces feature matrix by capturing most of required information.

- **Fully Connected Layer**:

Fully connected layers are used to apply weights to features obtained from Convolution and Max pooling layers to predict correct class probabilities.

- **Batch Normalisation**

Batch Normalisation ensures that each layers always falls within identical range even when layers in previous iteration have different range. According to [12] batch normalisation also helps in reducing total number of iterations of training, even though the output is same as that of without normalisation. It is described by equation,

$$y = \frac{x-\mu}{\sqrt{\sigma^2+\epsilon}} \tag{2}$$

Were μ and σ are mean and standard deviation of current iterations of x and y. ϵ is added constant to avoid division with zero

- **Activation Functions**

Activation functions will add decision boundaries in classification problem depending upon the bias and the variance. A Rectified Linear Unit (ReLU) is used in deep learning applications which is much more robust activation function compared to other non-linear functions like tanh, sigmoid etc. ReLU function is defined as follows,

$$f(x) = \begin{cases} x \; if \; x > 0 \\ 0 \; if \; x < 0 \end{cases} \tag{3}$$

4.4 Model Selection and Hyper-parameter Tuning.

- **Training from scratch**.

We have built our CNN architecture from scratch without taking into consideration pre-trained weights of models like VGG16, Rest Net which are trained on larger data sets for purpose of plant seedlings classification. Hence, during training at each epoch our model will learn features from training data and will optimize weights during backpropagation. This method is significant and will give better results if you are training your model on relatively smaller data sets. Since our balanced data set has around 7000 images after data augmentation, we have trained our model from scratch. But, this method still forms an underlying basis of comparison with other algorithms.

In deep learning, hyper-parameter are variables which forms underlying network architecture and also governs the learning process until algorithm converges to some optimal point. Hyper-parameters which define the network architecture are as follows.

- **Number of Convolutional Filters**

Convolutional filters in a convolutional layer convolves around the image for the purpose of detecting horizontal edges, vertical edges etc. Neural Networks learns this filters during training to adjust values in this filters according to features in image.

- **Number of Hidden Layers and units**

In CNN, hidden layers are part of fully connected structure which lies between input features and the output layer. Number of hidden layers in a network structure depends on type of data and the size of input as well as output layer. Number of units in hidden layer is approximately 2/3rd the size of input of layer (reference). The general idea is to keep increasing the number of hidden layers until the test error improves. If there are small number of hidden layers than it will cause the problem of under fitting while, if there are many hidden layers than your network structure will be complex and your model will over fit. Hence, it is extremely important to choose optimal number of hidden layers so, we can balance bias as well as variance.

- **Regularization / Drop out**

As we have discussed earlier, model complexity will increase variance and hence your model will overfit. Also, your model weights are likely to be large if model has high variance. So, with drop out, we decrease model complexity by penalizing weights. One option to do so is to randomly drop some hidden layers and filters from network architecture which will decrease model complexity as well as test error. Below given figure shows, the concept of regularization in neural networks.

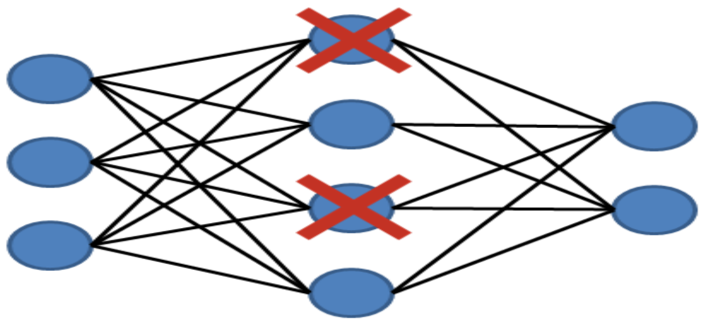


Fig 9 Regularization in Neural Networks [10]

- **Learning Rate**

The value of learning rate determines how fast the algorithm update its parameters to reach some optimal value. If the value is too small, algorithm will take more iterations to converge. If the value is too high, algorithm will never converge to some minimum value.

- **Number of epochs**

Basically, number of epochs refers to total number of times the training data is shown to network while training of model. The training process is stopped if validation accuracy starts decreasing and the training accuracy keeps increasing from a specific value of epoch.

We have implemented following three hyper-parameter tuning techniques from scratch

- **Random Search**

Since, grid search method performs exhaustive search in entire hyper-parameter sub space, it is difficult to implement in local environment were hardware and gpu usage is the major constraint on large data sets. Hence, we have considered slightly better version of grid search that is Random Search which randomly takes combination for hyper-parameters from pre-defined search space. Random Search has been proven effective than Grid Search in many other domains. However, one limitation of using random search is it induces high variance among models during search process, because it randomly selects values for hyper-parameters.

- **Early Stopping**

Early stopping is a state-of-art method which evaluates training process and finding optimal values of hyper-parameters. Experimentally it is found that Early Stopping can increase model training by 30% compared to random search depending upon the underlying structures of neural networks. In Early stopping, the concept of regularization is used to prevent overfitting, because overfitting increases training accuracy at cost of models ability to generalize the data.

- **Bayesian Optimization**

In contrast to random search, Bayesian Optimization evaluates search space in an informed manner. As a result it spends more time in evaluating the most promising values, which results in, convergence of algorithm in optimal time and better generalisation. Before evaluating a set of hyper parameters, Bayesian Optimization calculates the conditional probability P (validation accuracy | hyper parameter values) i.e. probability of set of hyper-parameter values to give higher validation accuracy. It evaluates model further if and only if the value of conditional probability is greater than some specified threshold. Hence, Bayesian Optimization takes much smarter decision than Random Search and Early Stopping

5. Experiments and Results

In a fine tuning experiment, we have done hyper-parameter tuning from scratch of a CNN network consisting of three Convolutional Layers and three fully Connected Layers using three above mentioned techniques.

All the values were derived empirically by performing experiment on google colab for 12 hours by connecting to virtual machines. It is necessary to note that all the values were obtained during search of best set of hyper-parameters and training of all the models. Training of all the models were done from scratch on google cloud (colab) using virtual machines since it’s very difficult to train a convolutional neural network on a 2 GB data set in a local system.

- Training of Best Model retrieved from Random Search

Below figure 9 shows training cycles of best evaluated model using Random Search technique. The maximum validation accuracy is achieved at 30th epoch nearly equal to 79%. From this point model starts overfitting on the training data which gives training accuracy nearly equal to 90% at 40th epoch.

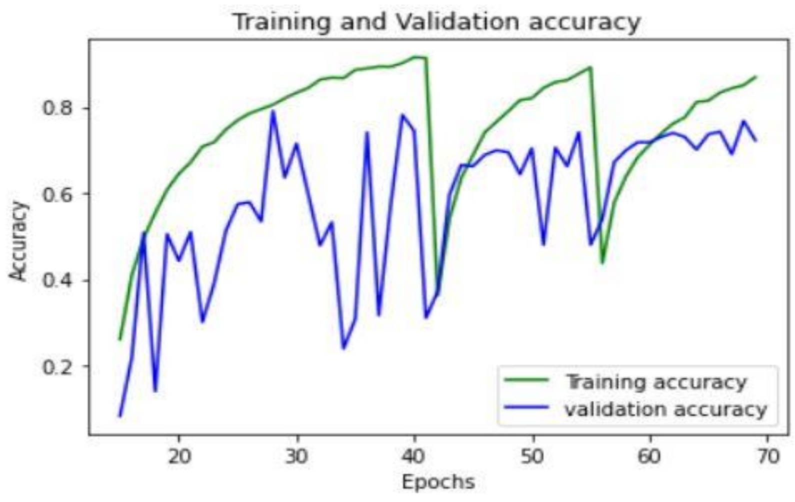


Fig 9 Training vs Validation Accuracy (Random Search)

- Training of Best model retrieved from Early Stopping

Below figure 10 shows training cycles of best evaluated model using Early Stopping technique. The validation accuracy of best evaluated model is around 84% which does not improves over training period. This necessarily implies that efficiency of model is not improved when the best set of hyper-parameters of search space are trained at point after completing a cycle from Early Stopping.

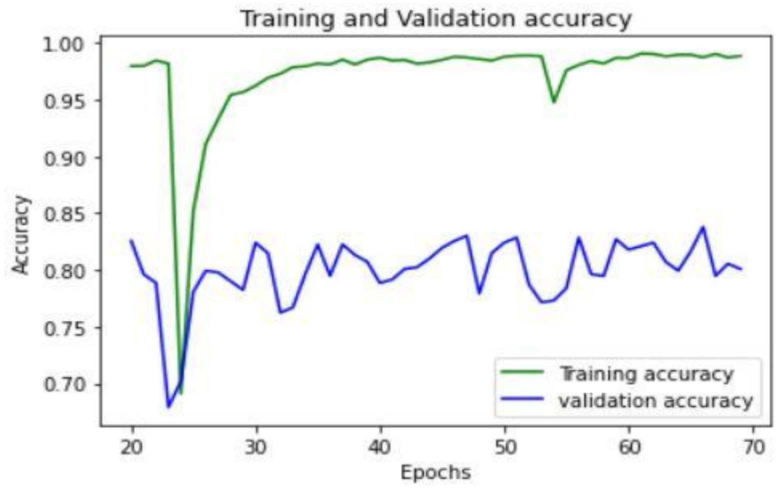


Fig 10 Training vs Validation Accuracy (Early Stopping)

- Training of Best Model Retrieved from Bayesian Optimization

Below fig 11 shows training cycle of best evaluated model using Bayesian Optimisation. This is the most effective technique for finding optimal set of values. Maximum validation accuracy equals to 84%

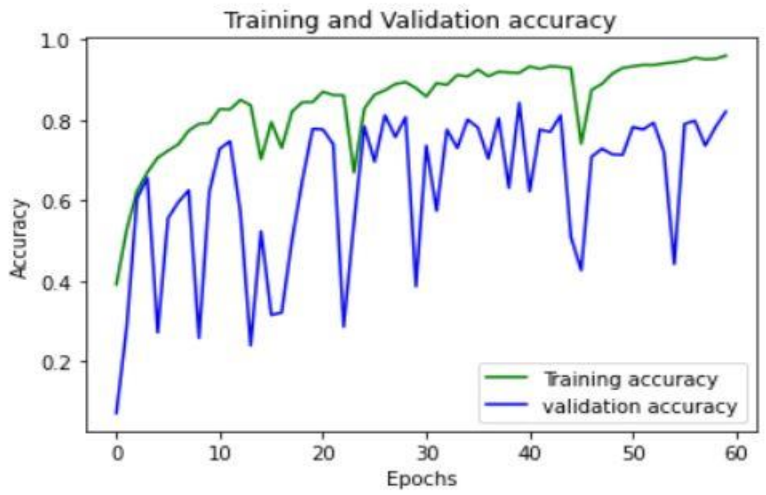


Fig 10 Training vs Validation Accuracy (Bayesian Optimization)

5.1 Model Evaluation

The raw data set consists of nearly 5000 images comprising of 12 classification labels. To balance each classification label we have used data augmentation technique. After balancing our data set, we have in total 7000 images in various forms. We have divided our data set into train, validation and test set. 10% of data is used for validation purpose and 10% of data is used for test purpose.

Basically, the selection of evaluation metric for model evaluation completely depends upon the type of business problem and the domain expert. Domain scientists are the people who decide which accuracy metric is suitable for their model.

Confusion Matrix is plotted for each of above hyper-parameter tuning techniques which is defined with following parameters.

- **True Positive (TP)**

Number of records which were classified as True when they were actually true

- **False Positive (FP)**

Number of records which were classified as true but were actually false

- **True Negative (TN**)

Number of records which were classified as false and were actually false

- **False Negative (FN)**

Number of records which were classified as false but were actually true

The scientists which proposed data set in [1] also mentioned the underlying evaluation metrics for this classification model. The measures which are described in [1] are

- **Cross Validation Accuracy**

This metric is particularly used to avoid overfitting in model. The main aim of cross validation accuracy is to evaluate model's ability to predict on unseen data, in order to regulate model complexity, to balance bias and variance.

- **Precision**

Precision is fraction of relevant instance from total retrieved results. Precision is calculated as,

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

- **Recall**

Recall is fraction of total amount of relevant instances retrieved that were actually present in a data set. Recall is calculated as,

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

- **F1-score**

F1-score is harmonic mean of Precision and Recall. F1-Score is calculated as

$$f1 - score = \frac{2*precision*recall}{precision+recall} \tag{6}$$

We have considered two additional measures as evaluation metric in our project

- **Macro-Average F1-Score**

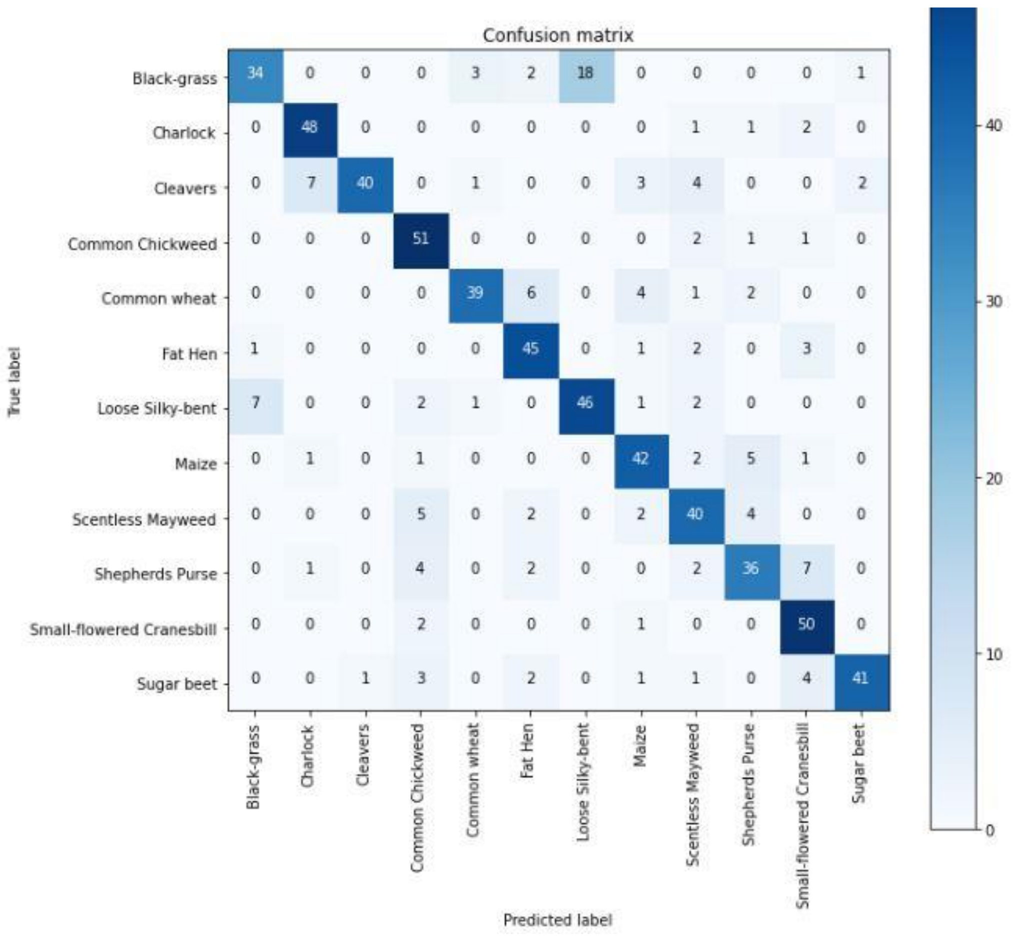
This value is the average of F1-Score of each classification labels without considering proportion of each classification label in data set.

- **Weighted Average F1-Score**

This value is weighted average of F1-Score of each classification labels with considering proportion of each classification label in data set.

Given below, are the listed figures of confusion matrix and f1-score of each of hyper-parameter tuning techniques on validation data. Confusion Matrix and F1-score for test are can be found in appendix section.

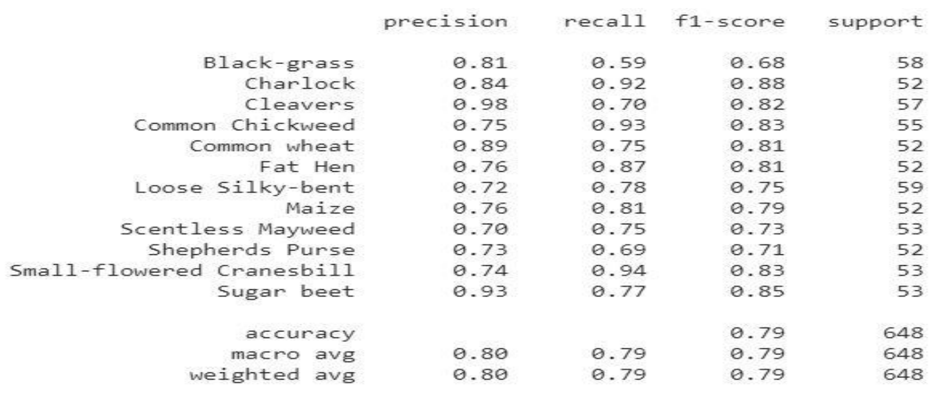
- Confusion Matrix for Random Search



True label \ Predicted label	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	34	0	0	0	3	2	18	0	0	0	0	1
Charlock	0	48	0	0	0	0	0	0	1	1	2	0
Cleavers	0	7	40	0	1	0	0	3	4	0	0	2
Common Chickweed	0	0	0	51	0	0	0	0	2	1	1	0
Common wheat	0	0	0	0	39	6	0	4	1	2	0	0
Fat Hen	1	0	0	0	0	45	0	1	2	0	3	0
Loose Silky-bent	7	0	0	2	1	0	46	1	2	0	0	0
Maize	0	1	0	1	0	0	0	42	2	5	1	0
Scentless Mayweed	0	0	0	5	0	2	0	2	40	4	0	0
Shepherds Purse	0	1	0	4	0	2	0	0	2	36	7	0
Small-flowered Cranesbill	0	0	0	2	0	0	0	1	0	0	50	0
Sugar beet	0	0	1	3	0	2	0	1	1	0	4	41

Fig 11 Confusion Matrix for Random Search

- Precision, Recall F1-Score for Random Search



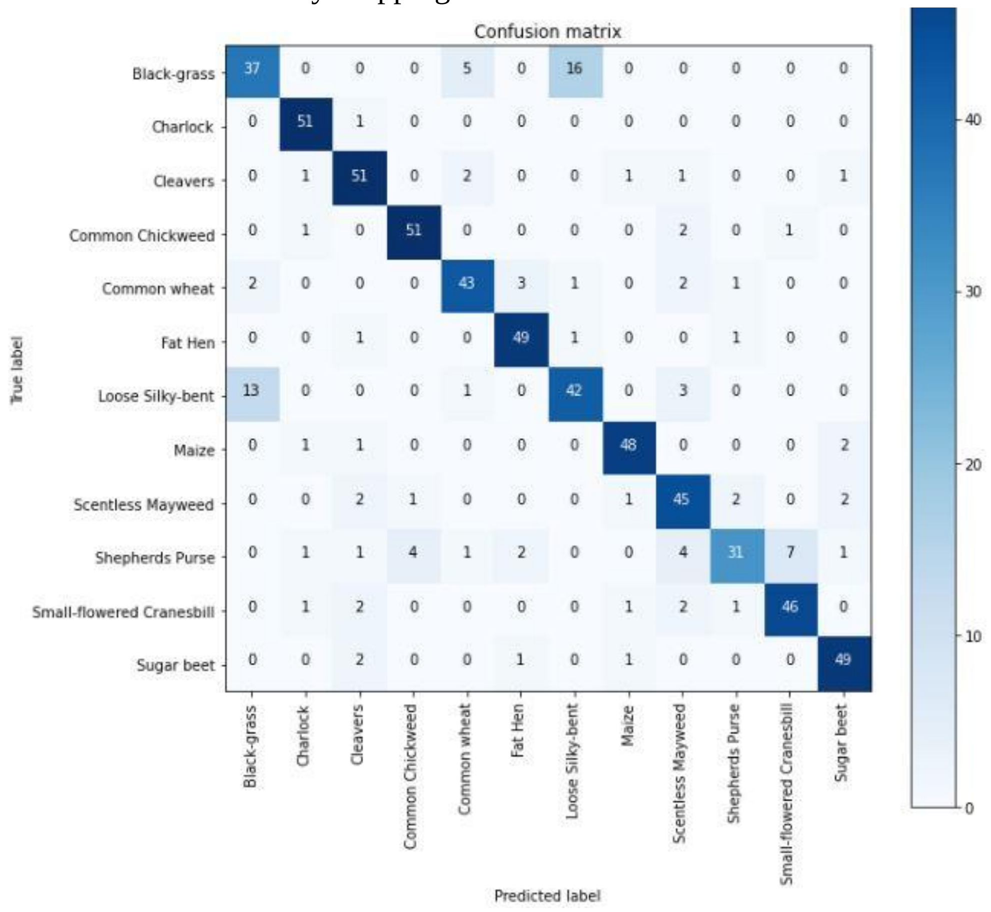
```
                           precision    recall  f1-score   support

              Black-grass       0.81      0.59      0.68        58
                 Charlock       0.84      0.92      0.88        52
                 Cleavers       0.98      0.70      0.82        57
         Common Chickweed       0.75      0.93      0.83        55
             Common wheat       0.89      0.75      0.81        52
                  Fat Hen       0.76      0.87      0.81        52
         Loose Silky-bent       0.72      0.78      0.75        59
                    Maize       0.76      0.81      0.79        52
        Scentless Mayweed       0.70      0.75      0.73        53
          Shepherds Purse       0.73      0.69      0.71        52
Small-flowered Cranesbill       0.74      0.94      0.83        53
               Sugar beet       0.93      0.77      0.85        53

                 accuracy                           0.79       648
                macro avg       0.80      0.79      0.79       648
             weighted avg       0.80      0.79      0.79       648
```

Fig 12 Precision, Recall, F1-Score (Random Search)

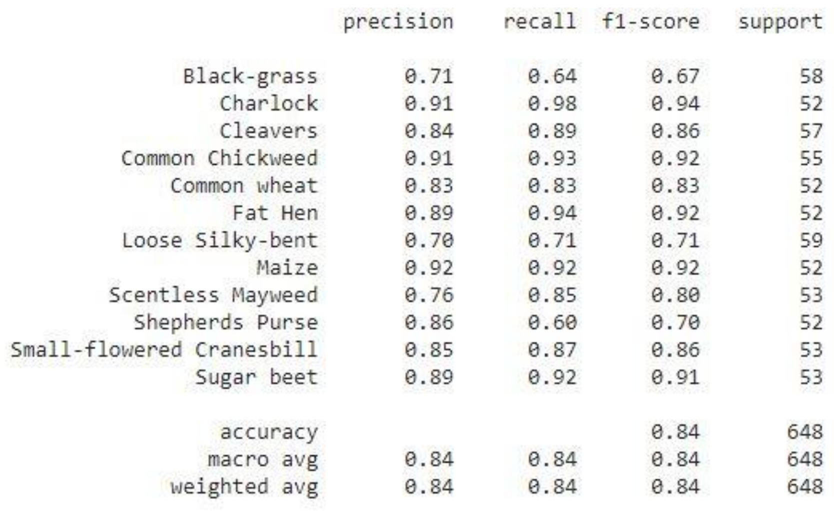
- Confusion Matrix for Early Stopping



True label \ Predicted label	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	37	0	0	0	5	0	16	0	0	0	0	0
Charlock	0	51	1	0	0	0	0	0	0	0	0	0
Cleavers	0	1	51	0	2	0	0	1	1	0	0	1
Common Chickweed	0	1	0	51	0	0	0	0	2	0	1	0
Common wheat	2	0	0	0	43	3	1	0	2	1	0	0
Fat Hen	0	0	1	0	0	49	1	0	0	1	0	0
Loose Silky-bent	13	0	0	0	1	0	42	0	3	0	0	0
Maize	0	1	1	0	0	0	0	48	0	0	0	2
Scentless Mayweed	0	0	2	1	0	0	0	1	45	2	0	2
Shepherds Purse	0	1	1	4	1	2	0	0	4	31	7	1
Small-flowered Cranesbill	0	1	2	0	0	0	0	1	2	1	46	0
Sugar beet	0	0	2	0	0	1	0	1	0	0	0	49

Fig 13 Confusion Matrix for Early Stopping

- Precision, Recall, F1-Score for Early Stopping



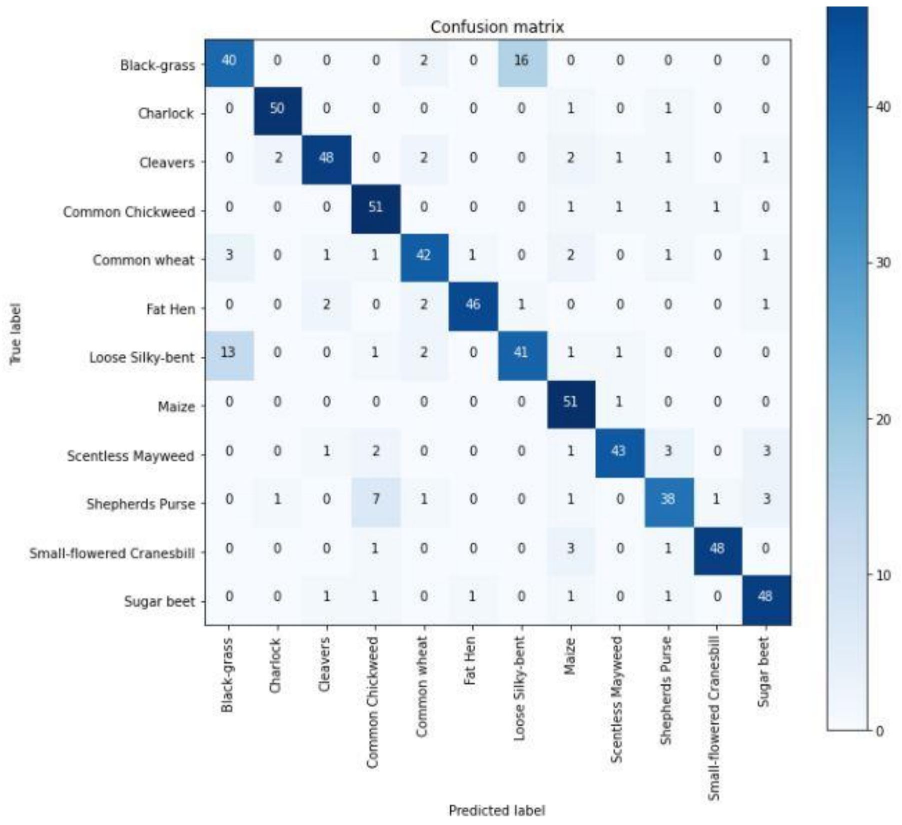
```
                           precision    recall  f1-score   support

              Black-grass       0.71      0.64      0.67        58
                 Charlock       0.91      0.98      0.94        52
                 Cleavers       0.84      0.89      0.86        57
         Common Chickweed       0.91      0.93      0.92        55
             Common wheat       0.83      0.83      0.83        52
                  Fat Hen       0.89      0.94      0.92        52
         Loose Silky-bent       0.70      0.71      0.71        59
                    Maize       0.92      0.92      0.92        52
        Scentless Mayweed       0.76      0.85      0.80        53
          Shepherds Purse       0.86      0.60      0.70        52
Small-flowered Cranesbill       0.85      0.87      0.86        53
               Sugar beet       0.89      0.92      0.91        53

                 accuracy                           0.84       648
                macro avg       0.84      0.84      0.84       648
             weighted avg       0.84      0.84      0.84       648
```

Fig 14 Precision, Recall, F1-Score (Early Stopping)

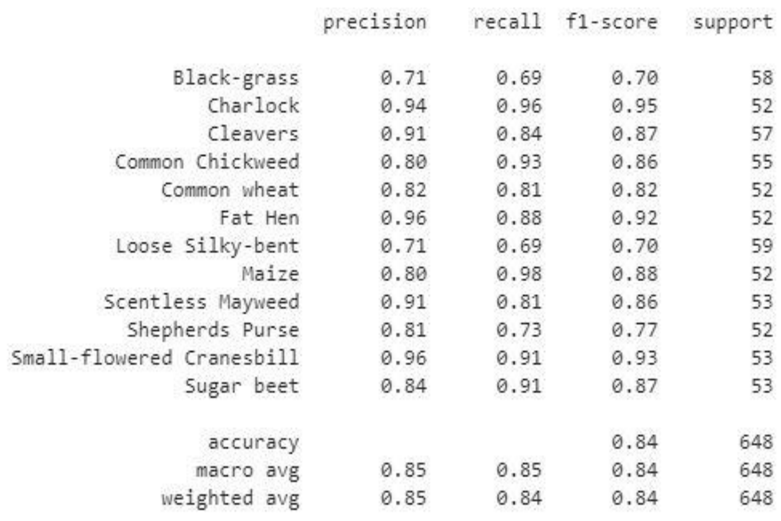
- Confusion Matrix for Bayesian Optimization



True label \ Predicted label	Black-grass	Charlock	Cleavers	Common Chickweed	Common wheat	Fat Hen	Loose Silky-bent	Maize	Scentless Mayweed	Shepherds Purse	Small-flowered Cranesbill	Sugar beet
Black-grass	40	0	0	0	2	0	16	0	0	0	0	0
Charlock	0	50	0	0	0	0	0	1	0	1	0	0
Cleavers	0	2	48	0	2	0	0	2	1	1	0	1
Common Chickweed	0	0	0	51	0	0	0	1	1	1	1	0
Common wheat	3	0	1	1	42	1	0	2	0	1	0	1
Fat Hen	0	0	2	0	2	46	1	0	0	0	0	1
Loose Silky-bent	13	0	0	1	2	0	41	1	1	0	0	0
Maize	0	0	0	0	0	0	0	51	1	0	0	0
Scentless Mayweed	0	0	1	2	0	0	0	1	43	3	0	3
Shepherds Purse	0	1	0	7	1	0	0	1	0	38	1	3
Small-flowered Cranesbill	0	0	0	1	0	0	0	3	0	1	48	0
Sugar beet	0	0	1	1	0	1	0	1	0	1	0	48

Fig 15 Confusion Matrix for Bayesian Optimization

- Precision, Recall, F1-Score for Bayesian Optimization



```
                           precision    recall  f1-score   support

              Black-grass       0.71      0.69      0.70        58
                 Charlock       0.94      0.96      0.95        52
                 Cleavers       0.91      0.84      0.87        57
         Common Chickweed       0.80      0.93      0.86        55
             Common wheat       0.82      0.81      0.82        52
                  Fat Hen       0.96      0.88      0.92        52
         Loose Silky-bent       0.71      0.69      0.70        59
                    Maize       0.80      0.98      0.88        52
        Scentless Mayweed       0.91      0.81      0.86        53
          Shepherds Purse       0.81      0.73      0.77        52
Small-flowered Cranesbill       0.96      0.91      0.93        53
               Sugar beet       0.84      0.91      0.87        53

                 accuracy                           0.84       648
                macro avg       0.85      0.85      0.84       648
             weighted avg       0.85      0.84      0.84       648
```

Fig 16 Precision, Recall F1-Score (Bayesian Optimisation)

5.2 Comparisons

Method	Total Number of Models Evaluated	Time taken to evaluate all models	Validation Accuracy of best evaluated model (%)	Training Accuracy (%)	Validation Accuracy after training model (%)	Test Accuracy (%)	Epoch
Random Search	20	2h 40m	81	89	79	81	70
Early Stopping	30	2h 7m	83	99	83	84	70
Bayesian Optimisation	5	39m	84	97	84	85	70

Table 2 Comparing Results of fine tuning techniques

The above mentioned table 2 compares three hyper-parameter tuning techniques on the basis of their search strategy. All the values were derived experimentally.

From the above table, it can be inferred that total time spent by each model in finding best set of hyper-parameters is in order Random Search > Early Stopping > Bayesian Optimisation. The number of models evaluated by Early Stopping is greater than Random Search in a given specific period of time. However, the general idea of overfitting being one of criteria to prune search spaces in Early Stopping technique forms underlying basis of arguments because training accuracy is slightly higher than validation accuracy though model generalises well on unseen data. Also, Bayesian optimization surpasses all the other technique by giving better generalisation accuracy in optimal time. Hence, Bayesian optimization is more useful if training time is major constraint in your experiments and still it performs better on unseen data.

However, the comparisons made above on the basis of above empirical values is limited to our experiment and domain. This idea cannot be generalised for all the problem statements.

Table 3 mentions each of fine tuning techniques with their corresponding weighted f1-score on validation set. We have use "Weighted F1-Score has evaluation metrics" in our system. This measure is slight modification of F1-score which takes into account proportion of each label in data set. From the given table it can be inferred that Bayesian optimization has best weighted f1-score on unseen data compared to other remaining methods.

Algorithm	Weighted F1-Score
Random Search	0.79
Early Stopping	0.84
Bayesian Optimization	0.85

Table 3 Comparing three fine-tuning techniques

6. Conclusion & Future Work

In conclusion, a convolutional neural network was constructed for differentiating images of twelve plant species at their growth stages. An experiment was conducted to train 8-layer convolutional neural network from scratch by three hyper-parameter tuning techniques namely Random Search, Early Stopping and Bayesian Optimization. Experimental results shows that Bayesian optimization surpasses both the techniques by giving better generalisation accuracy of 85% in optimal time. The generalisation accuracy by Random Search and Early Stopping technique was 81% and 84% respectively. Hence, it can be concluded that Bayesian optimization should be used in solving domain problems where training time is major constraint in building network architecture.

For future work we have considered possible extensions in three main directions,

- Tuning of Hyper-parameters in larger sub spaces. Hyper-parameter tuning in larger spaces is challenging task in local systems with low gpu and hardware powers. Hence, fine tuning of hyper-parameters in larger sub spaces will give better generalised results enabling a more robust classifier.
- Training of model on larger data sets in segmented form. A naïve Bayes image segmentation technique is described in [1] for image pre-processing. Training a CNN model on segmented images can give better results on unseen data.
- Testing our model on multiple plant species in single images. A better insightful analysis can be done by testing our model with multiple plant species in single images. Since, we have used data augmentation with random operation on images, our model must be able to identify plant multiple species in an image despite of their position and location in image.

References

[1] Giselsson, Thomas & Jørgensen, Rasmus & Jensen, Peter & Dyrmann, Mads & Midtiby, Henrik. (2017). A Public Image Database for Benchmark of Plant Seedling Classification Algorithms.

[2] Ashqar, B. A. M., Abu-Nasser, B. S., & Abu-Naser, S. S. (2019). Plant Seedlings Classification Using Deep Learning. International Journal of Academic Information Systems Research (IJAISR), 3(1), 7-14.

[3] Dyrmann, Mads & Karstoft, Henrik & Midtiby, Henrik. (2016). Plant species classification using deep convolutional neural network. Biosystems Engineering. 151. 72-80. 10.1016/j.biosystemseng.2016.08.024.

[4] Villaruz, Jolitte & Salido, Julie Ann & Barrios, Dennis & Felizardo, Rogelio. (2018). Philippine Indigenous Plant Seedlings Classification Using Deep Learning. 1-5. 10.1109/HNICEM.2018.8666412.

[5] Sun, Yu & Liu, Yuan & Wang, Guan & Zhang, Haiyan. (2017). Deep Learning for Plant Identification in Natural Environment. Computational Intelligence and Neuroscience. 2017. 1-6. 10.1155/2017/7361042.

[6] C. R. Alimboyong and A. A. Hernandez, "An Improved Deep Neural Network for Classification of Plant Seedling Images," 2019 IEEE 15th International Colloquium on Signal Processing & Its Applications (CSPA), Penang, Malaysia, 2019, pp. 217-222, doi: 10.1109/CSPA.2019.8696009.

[7] Meyer, G. and Neto, J. (2008). Verification of color vegetation indices for automated crop imaging applications. Computers and Electronics in Agriculture, 63(2):282–293.

[8] Retrieved from https://en.wikipedia.org/wiki/HSL_and_HSV

[9] Retrieved from https://en.wikipedia.org/wiki/Gaussian_blur

[10] Retrieved from https://towardsdatascience.com/introduction-to-dropout-to-regularize-deep-neural-network-8e9d6b1d4386

[11]Retrieved from https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

[12] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep training by reducing internal covariate shift. Arxiv.

7. Appendices

- ***User's Manual & Implementation Manual***

Plant Seedling Classification.rar