A

Project Report on

DIFFERENTIATE A WEED FROM A CROP BY IMAGE PROCESSING

Submitted in partial fulfilment for the award of

PG DIPLOMA IN BIG DATA ANALYTICS

CENTRE FOR DEVELOPMENT OF ADVANCED COMPUTING (C-DAC)

ADVANCED COMPUTING TRAINING SCHOOL (ACTS)

Knowledge Park, Bangalore - 560038

Submitted By

BHUVNESH SHARMA	PRN: 190250125010
NISHU MITTAL	PRN: 190250125036
JHALAK KHURANA	PRN: 190250125021

Guided By

MRS. SHILPA

CERTIFICATE

This is to certify that, the project report entitled

DIFFERENTIATE A WEED FROM A CROP BY IMAGE PROCESSING

Submitted By

BHUVNESH SHARMA	PRN: 190250125010
NISHU MITTAL	PRN: 190250125036
JHALAK KHURANA	PRN: 190250125021

is the record of bonafide work carried out by them in partial fulfilment of the requirement for the award of **PG Diploma in Big Data Analytics** prescribed by **Centre For Development Of Advanced Computing (C-DAC)**.

Mrs. Shilpa
(Project Guide)

Mrs. M.Savitri
(Course Co ordinator)

ACKNOWLEDGEMENT

We would like to express our sincere gratitude to all the people without whom this project would have been highly impossible.

We would like to devote our first vote of thanks to our guide **Mrs.Shilpa** for her constant support and encouragement. She has a great hand in the firm foundation of this project. We are deeply in debt for his valuable suggestions, scholarly guidance and constructive criticisms along with constant encouragement at each and every step for successful completion of the project.

Also a hearty thanks to our Professor **Mr. Ramesh Naidu Laveti** for his expert teachings and motivation all throughout the course.

Last but not the least we would like to thank all those who assisted us directly or indirectly for their valuable time and help.

Bhuvnesh Sharma
Nishu Mittal
Jhalak Khurana

ABSTRACT

Agriculture is vital for human survival and remains a major driver of several economies around the world; more so in underdeveloped and developing economies.

With increasing demand for food and cash crops, due to a growing global population and the challenges posed by climate change, there is a pressing need to increase farm outputs while incurring minimal costs. Previous machine vision technologies developed for selective weeding have faced the challenge of reliable and accurate weed detection.

We present approaches for plant seedlings classification with a dataset that contains 4,275 images of approximately 960 unique plants belonging to 12 species at several growth stages.

We compare the performances of two traditional algorithms and a Convolutional Neural Network (CNN), a deep learning technique widely applied to image recognition, for this task. Our findings show that CNN-driven seedling classification applications when used in farming automation have the potential to optimize crop yield and improve productivity and efficiency when designed appropriately.

TABLE OF CONTENTS

Chapter 1

INTRODUCTION

Plants continue to serve as a source of food and oxygen for all life on earth. In continents like Africa, where agriculture is predominant, proper automation of the farming process would help optimize crop yield and ensure continuous productivity and sustainability. The transformation of the agricultural sector by use of smart farming methods can power economic growth in many countries.

One major reason for reduction in crop yield is weed invasion on farmlands. Weeds generally have no useful value in terms of food, nutrition or medicine yet they have accelerated growth and parasitically compete with actual crops for nutrients and space. Inefficient processes such as hand weeding has led to significant losses and increasing costs due to manual labour. Precision agriculture, with the goal of defining systems that support decision-making in farm management in order to optimize returns on outputs while preserving resources, and weed control systems have been developed aiming at optimizing yields and costs while minimizing environmental challenges; some robotic systems have been used to do this. The robots and the vision machines need to be able to precisely and reliably detect a weed from the useful plants. Machine vision technologies developed for selective weeding face a challenge of reliable and accurate weed detection. It’s not easy to identify the weeds due to unclear crops boundaries, with varying rocky or sandy backgrounds, and as a result, traditional classification methods are likely to fail on this task.

In this work, we explore the performance of traditional computer vision methods on this task and show that a Deep Convolutional Neural Network (CNN) does the best job at classifying plant seedlings. In computer vision, CNNs have been known to be powerful visual models that yield hierarchies of features enabling accurate segmentation. They are also known to perform predictions relatively faster than other algorithms while maintaining competitive performance at the same time.

Chapter 2

AIM AND OBJECTIVE

The target of this project is to distinguish between the different weed seeding and crop seeding of 12 different plant species, hence it’s a multi-classification problem. As we take a .png image of a weed or a crop seedling and output the correspondent specie from our 12 classes. We’ll be using CNN model to classify the plants. As mentioned further CNN is widely used in the field of computer vision for doing complicated tasks, hence we’ll be using it.

The problem here is the weed seedling is much like crop seedling and our goal is to be able to differentiate between them. It will help farmers to automate this task (classify seedling plants).

Chapter 3

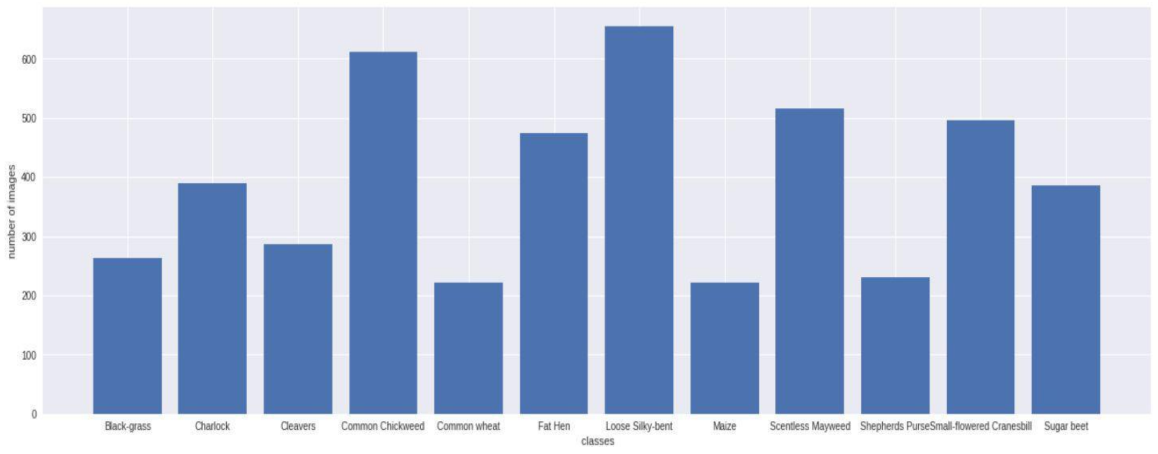
DATA

Plant seedling data set size is 1.6 GB divided into 12 folders in each one it contains number of images belong to certain class. We will split it later into training set and testing set.
There are 12 types of plant seedling:

1- Black-grass
2- Common Chickweed
3- Loose Silky-bent
4- Shepherd's Purse
5- Charlok
6- Common wheat
7- Maize
8- Small-flowered Cranesbill
9- Cleavers
10- Fat Hen
11- Scentless Mayweed
12- Sugar beet

- Here is a sample of one of those classes(charlock) :

- The total number of images is **4750** distributed as follows:



Chapter 4

PROBLEM STATEMENT

Based on the data described in the previous section, the following are the problems to be solved:

- With the given 12 different plant Species we need to build a model to distinguish between the different weed seeding and crop seeding.

Chapter 5

METHODOLOGY

Data exploration

- Visualize the distribution of data

Data pre-processing

- Check for null and missing values
- resize images
- apply segmentation and sharpening for images
- Label encoding
- Split training and validation set

CNN

- Define the model
- Set the optimizer and annealer
- Data augmentation

Evaluate the model

- Training and validation curves
- Confusion matrix

The final model is expected to be useful for classify the 12 different image Species.

Technology:

Programming language used: Python, HTML, CSS
Machine Learning and Data Analysis library: Numpy, Pandas, Scikit-learn, Matplotlib and Flask

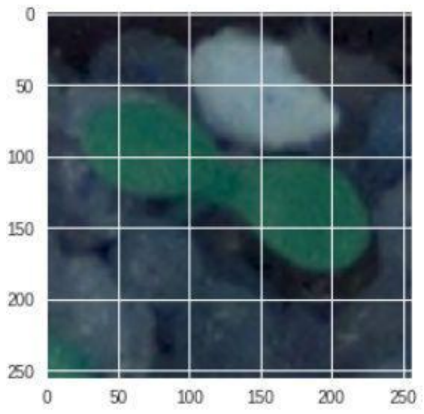
Metrics:

- Accuracy is a common metric for binary classifiers; it takes into account both true positives and true negatives with equal weight.
- accuracy = true positives + true negatives/ dataset size
- **Confusion matrix** can be very helpful to see the model drawbacks.
- a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group i but predicted to be in group j

Pre-Processing:

1- **Resizing Images:**

Images have not the same size so We have resized the images to 256*256 pixel
to feed it later to the neural network
In this figure the image after resize:

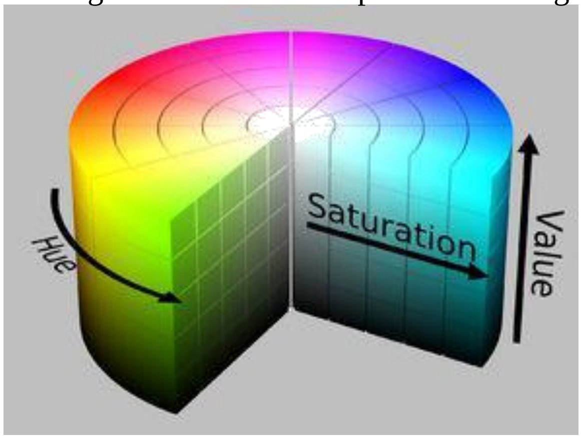


2- Creating mask for the images :

create_mask_for_plant function returns an image mask: Matrix with shape (image_height, image_width). In this matrix there are only 0 and 1 values. The 1 values define the interesting part of the original image. We can create this mask using HSV of the image.

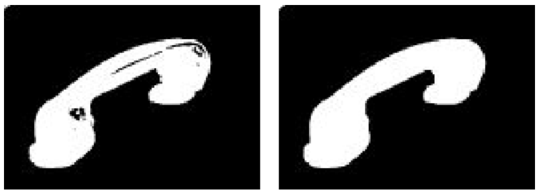
The HSV color-space is suitable for color detection because with the Hue we can define the color and the saturation and value will define "different kinds" of the color. (For example it will detect the red, darker red, lighter red too). We cannot do this with the original BGR color space

This figure illustrate HSV space of the image.

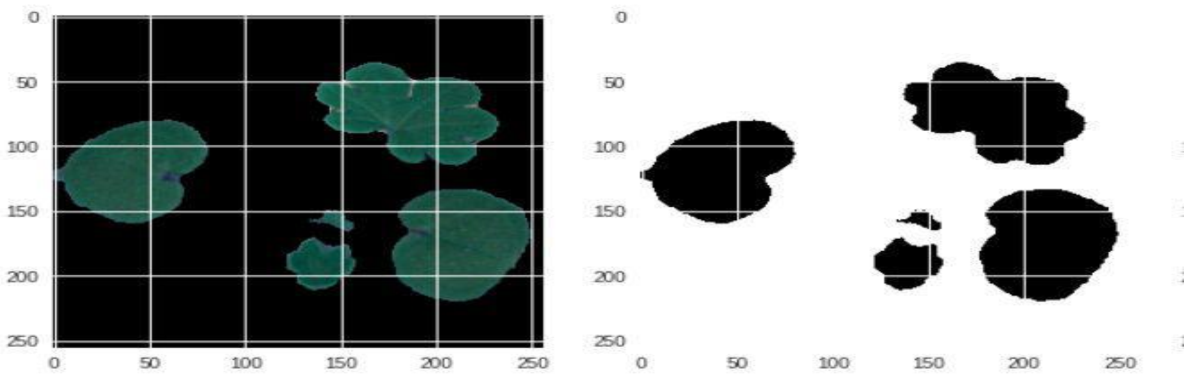


After converting RGB to HSV we started to apply 3 morphlogical operations:
One of the most commom morphological operation is closing used to
close the small halls in the images.

This figure below illustrate the image before and after applying closing:

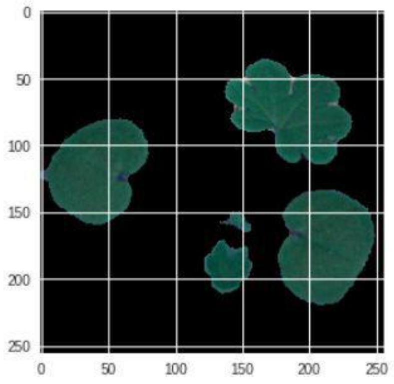


Here is sample after applying the mask on one of images:



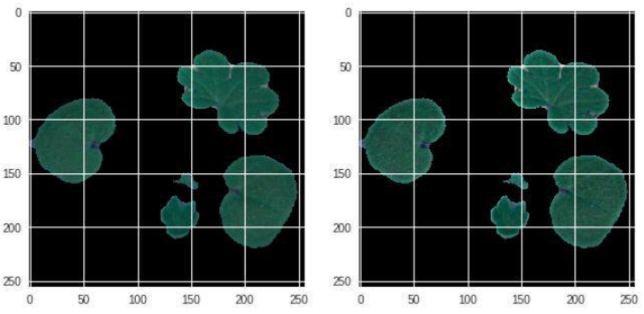
3- Segmentation:

Segmentation is partitioning an image into distinct regions containing each pixels with similar attributes. To be meaningful and useful for image analysis and interpretation, the regions should strongly relate to depicted objects or features of interest. Here is a sample after applying segmentation on one of images:



4- Sharpening:

Sharpening an image increases the contrast between bright and dark regions to bring out features.
Image before and after sharpening:



5- Label encoding:

Using LabelBinarizer Binarize labels in a one-vs-all fashion
Input: the label of image and the output is vector represents the class in binary form

6- Splitting data into training and testing set:

In this step we used sklearn train_test split 70% for training and 30% for testing Then we split testing data into testing and validation set 50% for testing and 50% for validation. The point of using the validation set is to try to avoid overfitting and check if an overfitting may occur.

The classifier training stage

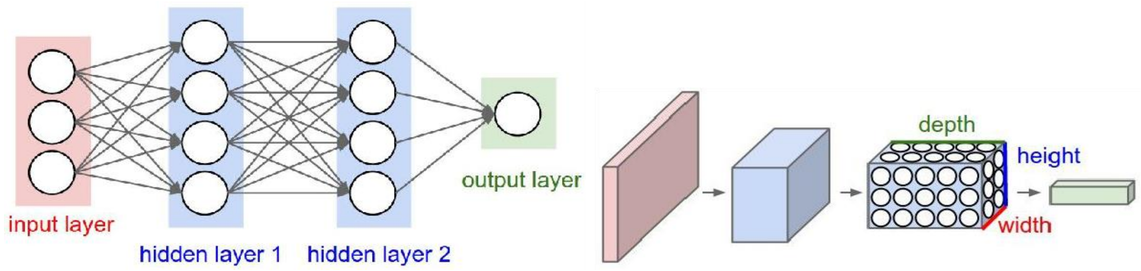
During the first stage, the classifier was trained on the preprocessed training data. It can be further divided into the following steps:

1. Load both the training and validation images into memory, preprocessing them as described in the previous section the data required for a single training step (a batch of images, their labels, and the learning rate)
2. Define the network architecture and training parameters
3. Define the loss function, accuracy
4. Train the network, logging the validation/training loss and the validation accuracy
5. Plot the logged values
6. If the accuracy is not high enough, return to step 3
7. Save and freeze the trained network

CNN Architecture:

In general, Neural networks accept a single vector as input, transform it to a series of hidden layers, which in turn is made up of set of neurons that are fully connected to all neurons in the previous layer. Neurons of the same layer are independent and do not share any connections. After the hidden layers, is the last fully connected layer which is also called the 'output layer', where each node outputs score for each class. The downside of regular neural network is that they don't scale well to full images. It's mainly because with images of decent size, the number of neurons and weights that the network must accommodate becomes unmanageable. This is where Convolutional Neural Network comes to rescue with its neurons arranged in 3 dimensions (width, height, depth).
Each of the layer in CNN accepts 3D input volume and transforms it into 3D output volume
Following is a simple visualization of how CNN arranges its neurons in 3 dimensions

(width, height, depth):



Following are the layers that are used to build a CNN:
Input layer (w,h,d)
Input layer of shape (w,h,d) represents image of size 'w x h' and 'd' number of color channels. For example, for an image of size (256x256) with 3 color channels (HSV), the input layer will hold raw pixel values of the image as a vector of size [256,256,3]

CONV layer:

The CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. For eg, if we use 32 filters, CONV layer will output a volume equal to (256x256x3) Activation functions

Activation layer will apply an element wise activation functions, leaving the volume unchanged.

POOL layer:

Pool layer performs downsampling operation along the spatial dimensions (width, height), outputting a reduced volume than the previous layer. For eg, (128x128x32)

Fully-Connected Layer:

FC layer, also called as the dense layer, each neuron will be connected to all the neurons of the previous layer. FC layer when used as the output layer results in much reduced volume of size [1x1xm], where m is the number of categories that are to be predicted. Each of the nodes of fully connected layer outputs a score corresponding to a class score.

Dropout layer

Dropout layer is used as a method of regularization to combat over-fitting of the training set. It ‘drops’ neurons at random (depending on the probability mentioned) while calculating the forward prop and backward prop, resulting in a simpler version of the CNN for each iteration and hence giving the model a hard time to overfit the training set.

CNN algorithmconsists of several convolution (CNV) operations followed of the image sequentially which is followed by pooling operation (PL) to generate the neurons feed into fully connected (FC) layer.

We used the Keras Sequential API, where you have just to add one layer at a time, starting from the input.

Input:of CNV is typically 2D image data with HSV (hue saturation value)

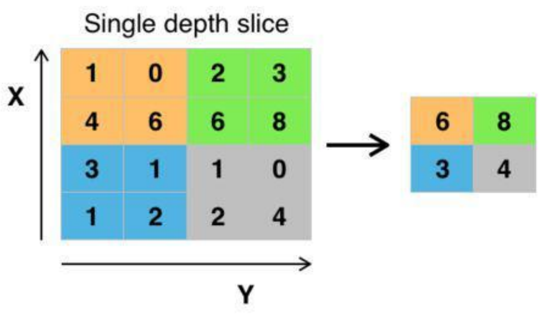
The first is the convolutional (Conv2D) layer: It is like a set of learnable filters. We choose to set 32 filters for the two firsts conv2D layers and 64 filters for the 2nd convolutional layer and 128 filters for the two last ones.

Each filter transforms a part of the image (defined by the kernel size) using the kernel filter. The kernel filter matrix is applied on the whole image. Filters can be seen as a transformation of the image.

The CNN can isolate features that are useful everywhere from these transformed images (feature maps).

The second important layer in CNN is the pooling(MaxPool2D) layer.
This layer simply acts as a downsampling filter. It looks at the 2 neighboring pixels and picks the maximal value. These are used to reduce computational cost, and to some extent also reduce overfitting. We have to choose the pooling size (i.e the area size pooled each time) more the pooling dimension is high, more the downsampling is important.

This figure below illustrate Max pooling with a 2x2 filter and stride = 2:



Combining convolutional and pooling layers, CNN are able to combine local features and learn more global features of the image.

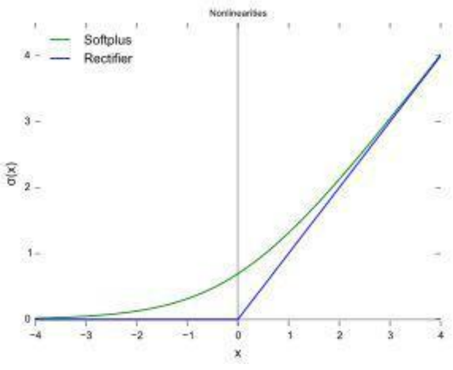
Dropout is a regularization method, where a proportion of nodes in the layer are randomly ignored (setting their weights to zero) for each training sample. This drops randomly a proportion of the network and forces the network to learn features in a distributed way. This technique also improves generalization and reduces the overfitting.

Relu Layer :

ReLU is the abbreviation of Rectified Linear Units. This layer applies the non-saturating activation function. $f(x) = x^{+} = max(0, x)$

It increases the nonlinear properties of the decision function and of the overall network without affecting the receptive fields of the convolution layer.

Figure below illustrate Relu function: Plot of the rectifier (blue) and softplus (green) functions near x = 0



The Flatten layer is use to convert the final feature maps into a one single 1D vector. This flattening step is needed so that you can make use of fully connected layers after some convolutional/maxpool layers. It combines all the found local features of the previous convolutional layers.

In the end we used the features in two fully-connected (Dense) layers which is just artificial neural networks (ANN) classifier.

In the last layer (Dense(10,activation="softmax")) the net outputs distribution of probability of each class.

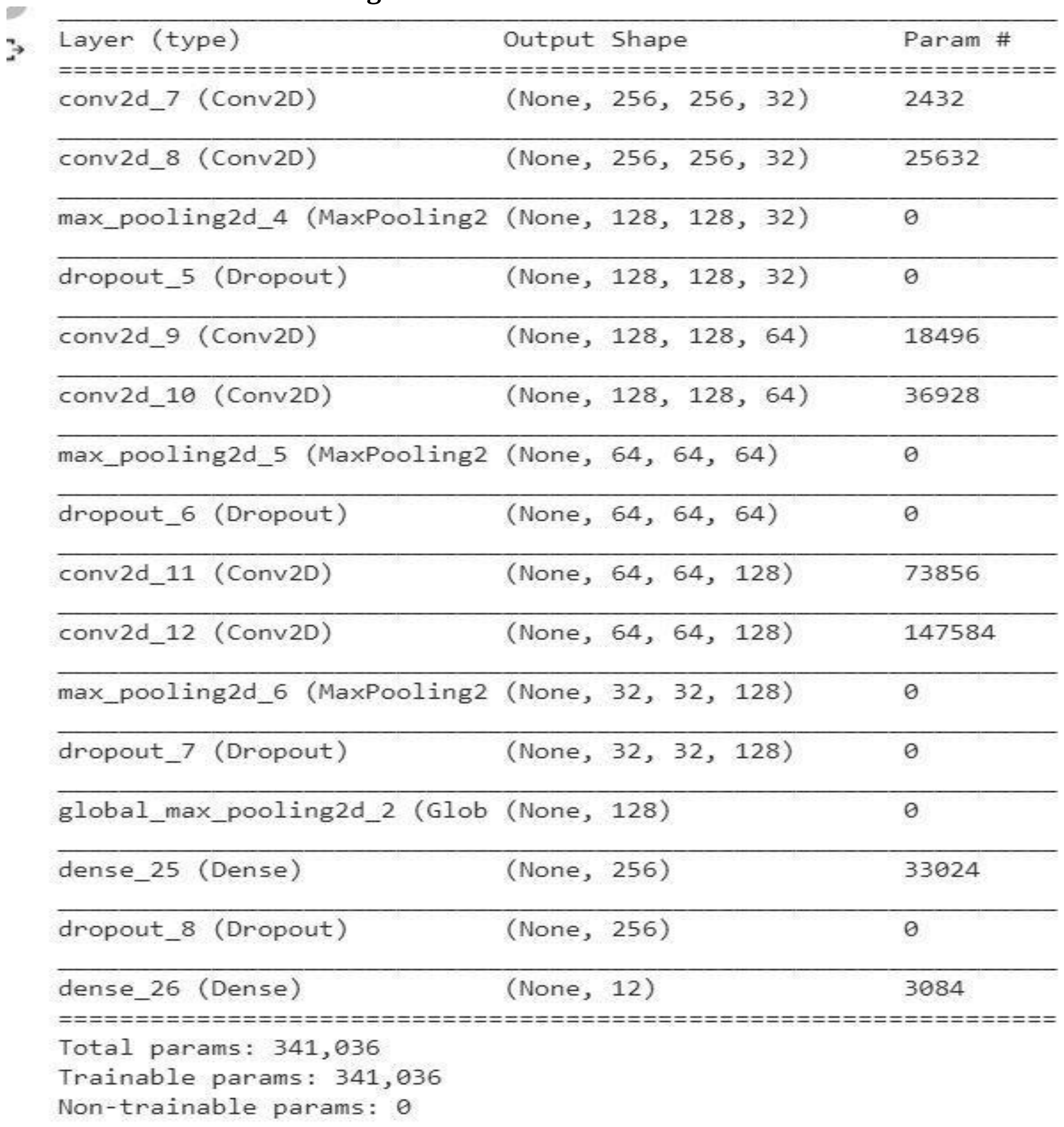
Fully Connected (FC) Layer:

This layer will reduce the size of input data to the size of classes that the CNN is trained for by combining output of CNV layer with different weights. Each neuron at the output of the CNV layer will be connected to all other neurons after weighted properly, Similar to CNV layer, weight of these taps in FC layer is found though backpropagation algorithm.

Classification Layer (CL):

This is the final layer of the CNN that converts the output of FC to probability of each object being in a certain class. Typically soft-max type of algorithms are used in this layer

The model has the following architecture:



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 256, 256, 32)      2432
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 256, 256, 32)      25632
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 128, 128, 32)      0
_________________________________________________________________
dropout_5 (Dropout)          (None, 128, 128, 32)      0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 128, 128, 64)      18496
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 64, 64, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 64, 64, 128)       73856
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
dropout_7 (Dropout)          (None, 32, 32, 128)       0
_________________________________________________________________
global_max_pooling2d_2 (Glob (None, 128)               0
_________________________________________________________________
dense_25 (Dense)             (None, 256)               33024
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_26 (Dense)             (None, 12)                3084
=================================================================
Total params: 341,036
Trainable params: 341,036
Non-trainable params: 0
```

CHAPTER 6

COMPLICATIONS

There are some complications that occurred during the code process:

- When the batch_size was 38, it caused an error Bad Input Shape. We looked in the code and tried to fix it but couldn't, after some trials and exhausting tracing, we eventually realised that it was the batch_size, hence, we assigned it to none.
- We were considering that all images have the same size but when we feed images to CNN but it also arises some errors, so we did resize images to the same size 256x256x3
- We weren't able to get higher accuracy at first. However, after applying 128 filters we were able to get better accuracy of 87% which is acceptable.

Refinement:

- ✓ Decreasing learning rate from .01 to .001 avoided the over fitting of the model.
- ✓ By adding data augmentation accuracy increased to : 87%
- ✓ Adding some additional layer also improved the model
- ✓ Changing dropout from .25 to .3 improved the accuracy for 82%

CHAPTER 7

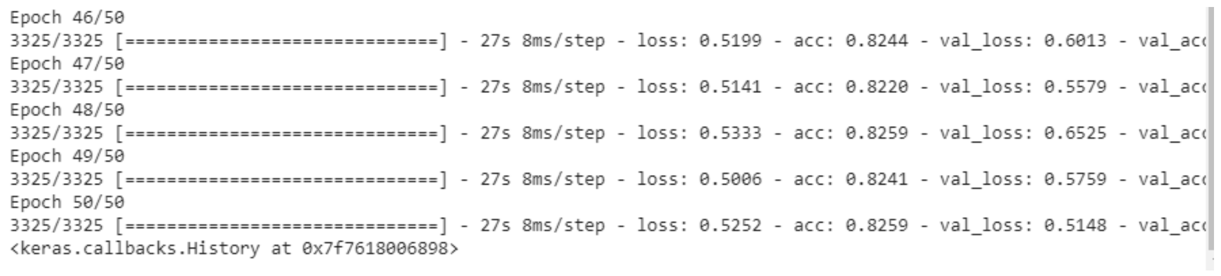
MODEL EVALUATION AND VALIDATION

During development, a validation set was used to evaluate the model. The final architecture and hyper parameters were chosen because they performed the best among the tried combinations.

Complete description of the final model and the training process:

- The shape of the filters of the 1st and 2nd convolutional layers is 5*5 and 3*3 for the rest of convolutional layers.
- False positives are rare but present

After testing on validation set it give the score in the image:

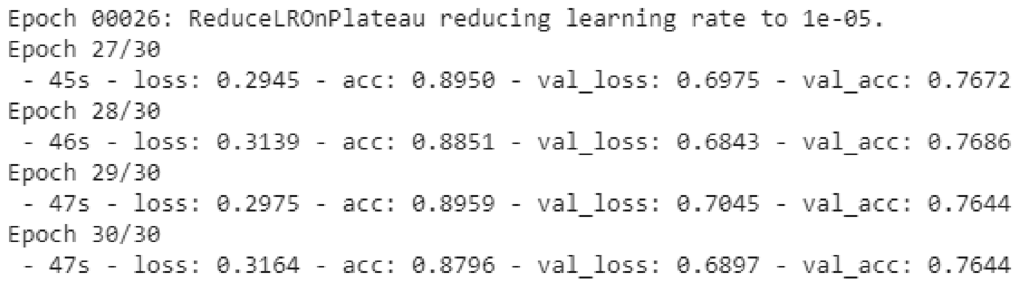


```
Epoch 46/50
3325/3325 [==============================] - 27s 8ms/step - loss: 0.5199 - acc: 0.8244 - val_loss: 0.6013 - val_ac
Epoch 47/50
3325/3325 [==============================] - 27s 8ms/step - loss: 0.5141 - acc: 0.8220 - val_loss: 0.5579 - val_ac
Epoch 48/50
3325/3325 [==============================] - 27s 8ms/step - loss: 0.5333 - acc: 0.8259 - val_loss: 0.6525 - val_ac
Epoch 49/50
3325/3325 [==============================] - 27s 8ms/step - loss: 0.5006 - acc: 0.8241 - val_loss: 0.5759 - val_ac
Epoch 50/50
3325/3325 [==============================] - 27s 8ms/step - loss: 0.5252 - acc: 0.8259 - val_loss: 0.5148 - val_ac
<keras.callbacks.History at 0x7f7618006898>
```

With Test loss: 0.52 and Test accuracy: 0.82

And after applying data augmentation

It gives a test score of .87

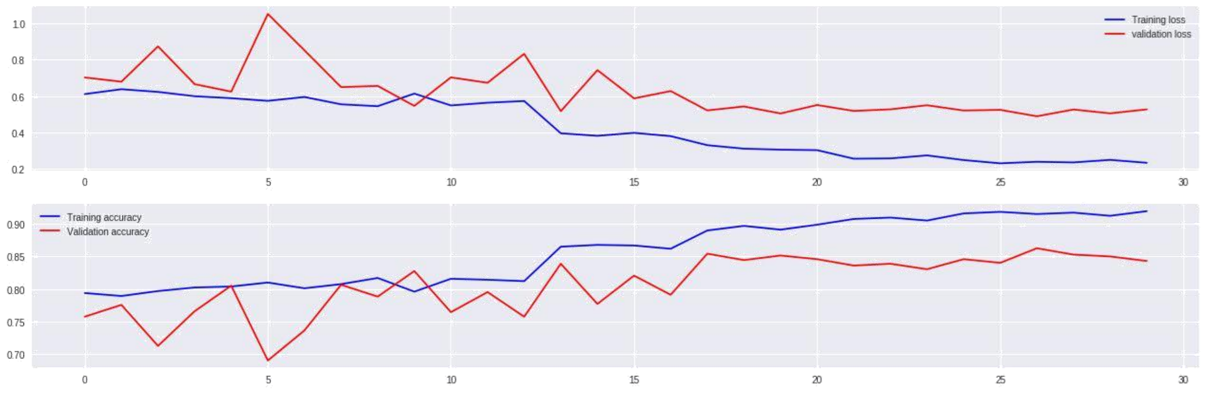


```
Epoch 00026: ReduceLROnPlateau reducing learning rate to 1e-05.
Epoch 27/30
 - 45s - loss: 0.2945 - acc: 0.8950 - val_loss: 0.6975 - val_acc: 0.7672
Epoch 28/30
 - 46s - loss: 0.3139 - acc: 0.8851 - val_loss: 0.6843 - val_acc: 0.7686
Epoch 29/30
 - 47s - loss: 0.2975 - acc: 0.8959 - val_loss: 0.7045 - val_acc: 0.7644
Epoch 30/30
 - 47s - loss: 0.3164 - acc: 0.8796 - val_loss: 0.6897 - val_acc: 0.7644
```

As for the evaluation and validation, we have used 2 main values to check:

Accuracy and validation loss.

Before tuning some parameters (before reducing learning rate)



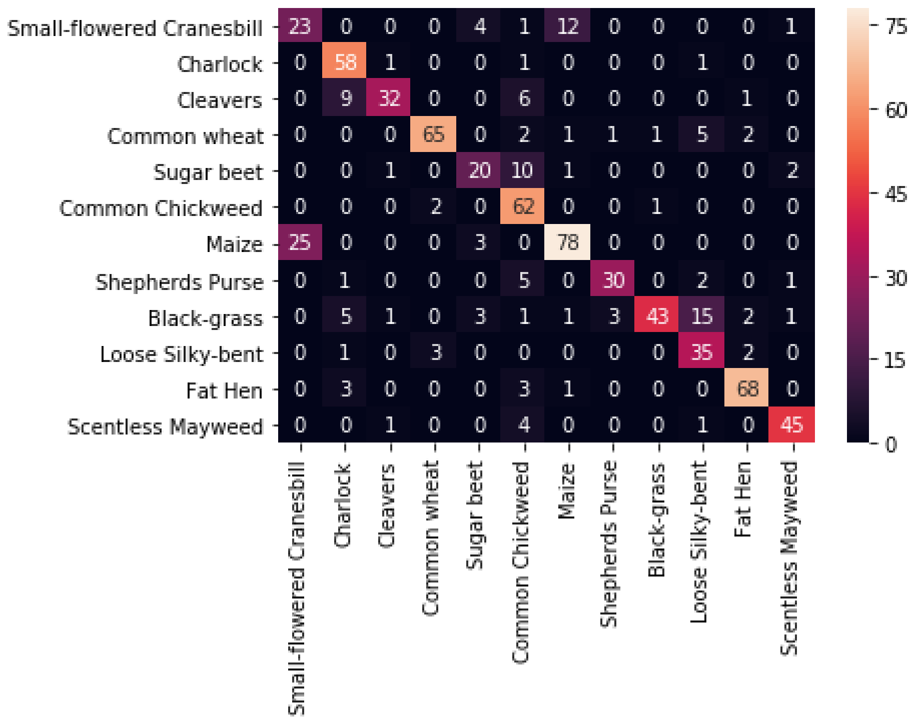
Justification:

The results of the final classifier are much better than that of the benchmark model. It has score of 87% is a decent score when compared to 60.8% (benchmark model's performance).We believe the final solution will definitely contribute significantly towards solving the current problem and also with more training data and more pre-processing stages, there are possibilities of improving the model further.

Chapter 8

CONCLUSION

To show the model quality we get the images which classified incorrectly in the test set this means the images that uncropped correctly can be misclassified
This model can help farmers to automate the task of classifying seedling plants and weed plants.
We plotted the confusion matrix to observe which category is poorly classified by the classifier and observe its performance visually. We can see in the below confusion matrix plot, that the major misclassification happened between Maize and Black-grass. It looks like the classifier is having difficulty classifying these two categories. We plotted the confusion matrix to observe which category is poorly classified by the classifier and observe its performance visually. Hence, this is where the classifier needs improvement as these misclassification amount to 50% of the misclassification overall. We believe overcoming this challenge will boost the F1-score significantly.



Our final scores after model tuning is 0.87

REFERENCES

1. https://en.wikipedia.org/wiki/HSL_and_HSV
2. https://homepages.inf.ed.ac.uk/rbf/HIPR2/close.htm
3. https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html /topic4.htm
4. https://machinelearningmastery.com/evaluate-performance-deep-learning-models-kera s/