Machine Learning Engineer Nanodegree

Capstone Project

Rishi Ohri
April 12, 2019

I. Definition

Project Overview

The proposed project is to differentiate a weed (i.e. unwanted plant seedling) from a crop seedling. The ability to do so effectively can mean better crop yields and better care for the environment.

The project problem and dataset are taken directly from Kaggle Competition: Plant Seedlings Classification – Determine the species of a seedling from an image.

The Aarhus University Department of Engineering Signal Processing Group, in collaboration with University of Denmark, has shared a dataset of images (on Kaggle) belonging to 12 species of plant seedlings at various stages of growth.

Related academic research: A Public Image Database for Benchmark of Plant Seedling Classification Algorithms. In this paper, a benchmark based on f1 scores is proposed to standardize the valuation of classification results obtained with the provided database.

Problem Statement

The problem here is a close resemblance among different species of plant seedlings. A potential solution to resolve this problem is to design an Image Classification model using Convolution Neural Network. The model should be able to determine the species of a plant seedling from an image, with a good metrics score.

Metrics

As per Kaggle, the evaluation metrics to be used is MeanFScore. So, I have used the same evaluation metrics.

Given positive/negative rates for each class *k*, the resulting score is computed this way:

$$Precision(micro) = \frac{\sum_{k \in c} TP_k}{\sum_{k \in c} TP_k + FP_k}$$

$$Recall(micro) = \frac{\sum_{k \in c} TP_k}{\sum_{k \in c} TP_k + FN_k}$$

Where TP – True Positive, FP – False Positive and FN – False Negative

F1-score is the harmonic mean of precision and recall

MeanFScore:

$$F1_{micro} = 2\ Precision_{micro}\ Recall_{micro} / (Precision_{micro} + Recall_{micro})$$

Also, the training dataset is unbalanced i.e. there is significant difference in number of images for different seedlings. So, I think it is appropriate to use F1 score as evaluation metric. Additionally, I have used confusion matrix to describe the performance of a classification model on set of test data, for both benchmark model as well as InceptionV3 model.

II. Analysis

Data Exploration

Dataset is taken directly from Kaggle which provides a training set and a test set of images of plant seedlings at various stages of growth. Each image has a filename that is its unique id. The dataset comprises of 12 plant species:

- Black-grass
- Charlock
- Cleavers
- Common Chickweed
- Common wheat
- Fat Hen
- Loose Silky-bent
- Maize
- Scentless Mayweed
- Shepherds Purse
- Small-flowered Cranesbill
- Sugar beet

Here, training dataset contains images of plant species organized by folder (i.e. dataset consists of subfolders with name of plant species). However, the test dataset consists of images (test images are not classified into species) for which species need to be predicted by our trained model and the predictions need to be submitted to Kaggle competition.

Dataset provided by Kaggle consisted of 2 parts:

- Training set (1.61GB of data) contains 4750 Images in 12 folders (for 12 species)
- Testing set (87MB of data) contains 794 files in 1 folder (not divided into 12 species)

Since, testing set didn't label the files with plant species, so I have used custom subset of data for this project. I have split the total dataset (training dataset of Kaggle) into three parts:

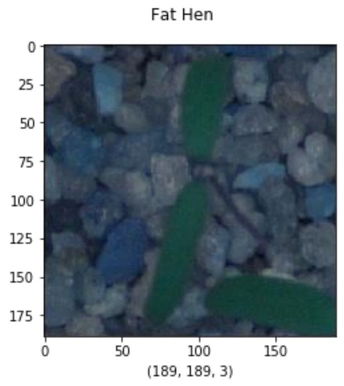
Dataset	% data	Images
training set (to train model)	60%	2850
validation set (to cross-validate model)	20%	950
testing set (to evaluate performance of model)	20%	950

Training and Testing dataset contains images of different sizes so I rescaled all the images to same size (such as 224*224 or 299*299).

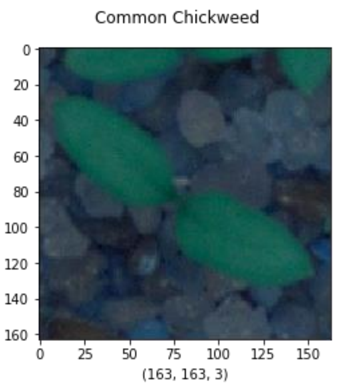
Exploratory Visualization

As a first step for visualization, I displayed 5 images picked randomly from the training set. Clearly, all the images are of different sizes so I rescaled all the images to same size (i.e. 224 * 224). Following are the 5 random images to justify the above observation:
(Note: Below images are taken from one execution. There were multiple runs)

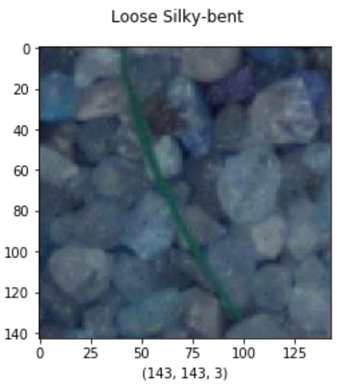
```
Image 1464 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
```



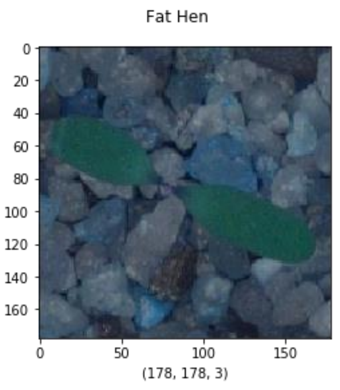
```
Image 1072 [0. 0. 0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]
```



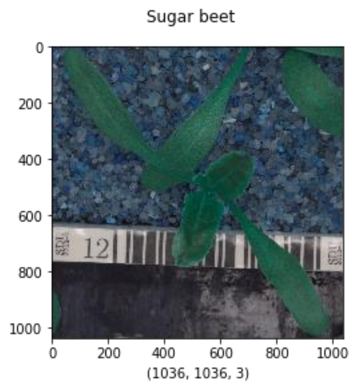
```
Image 798 [0. 0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0.]
```



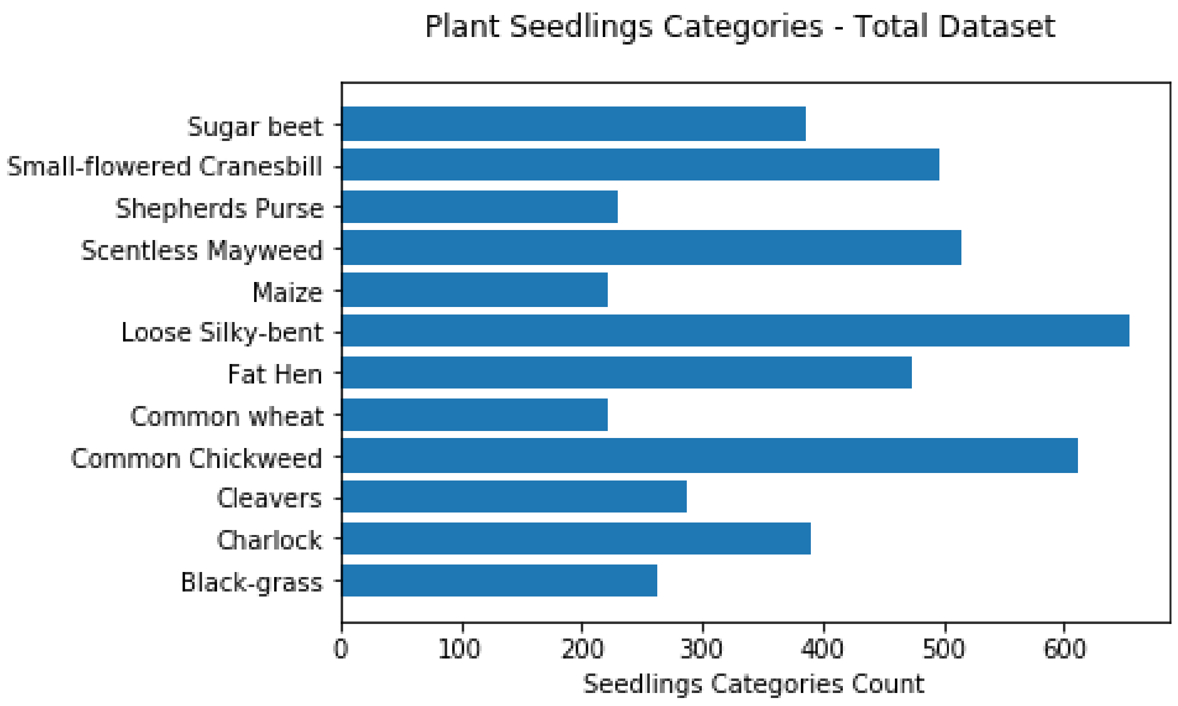
```
Image 2368 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
```

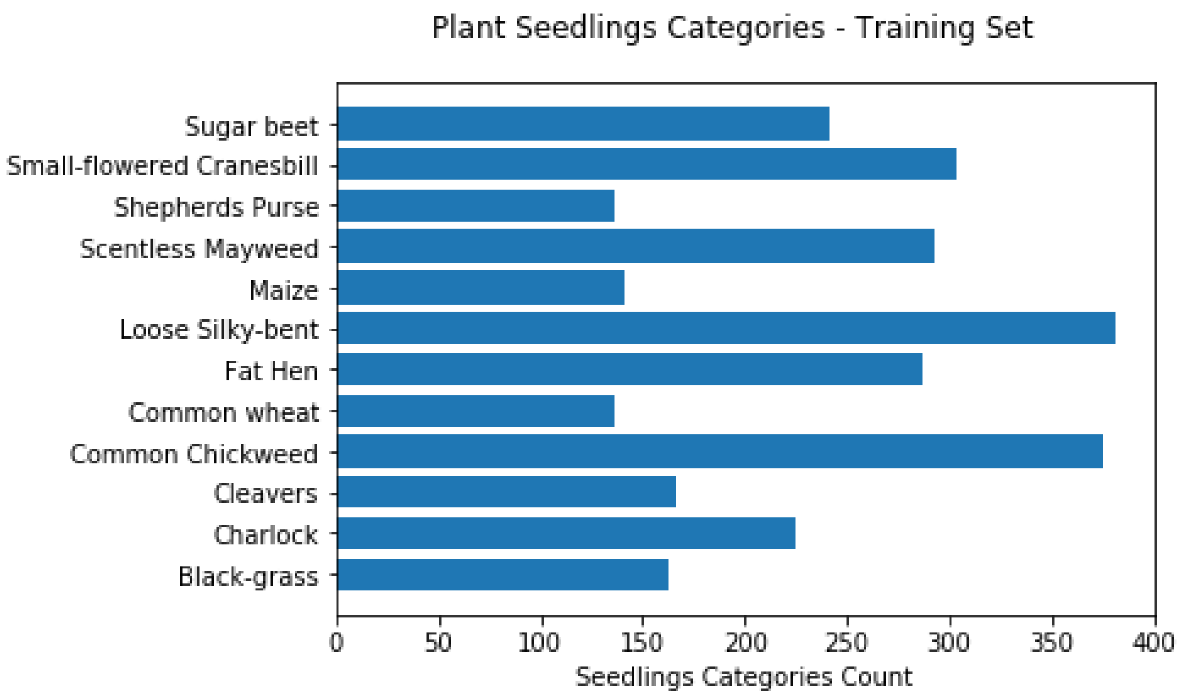


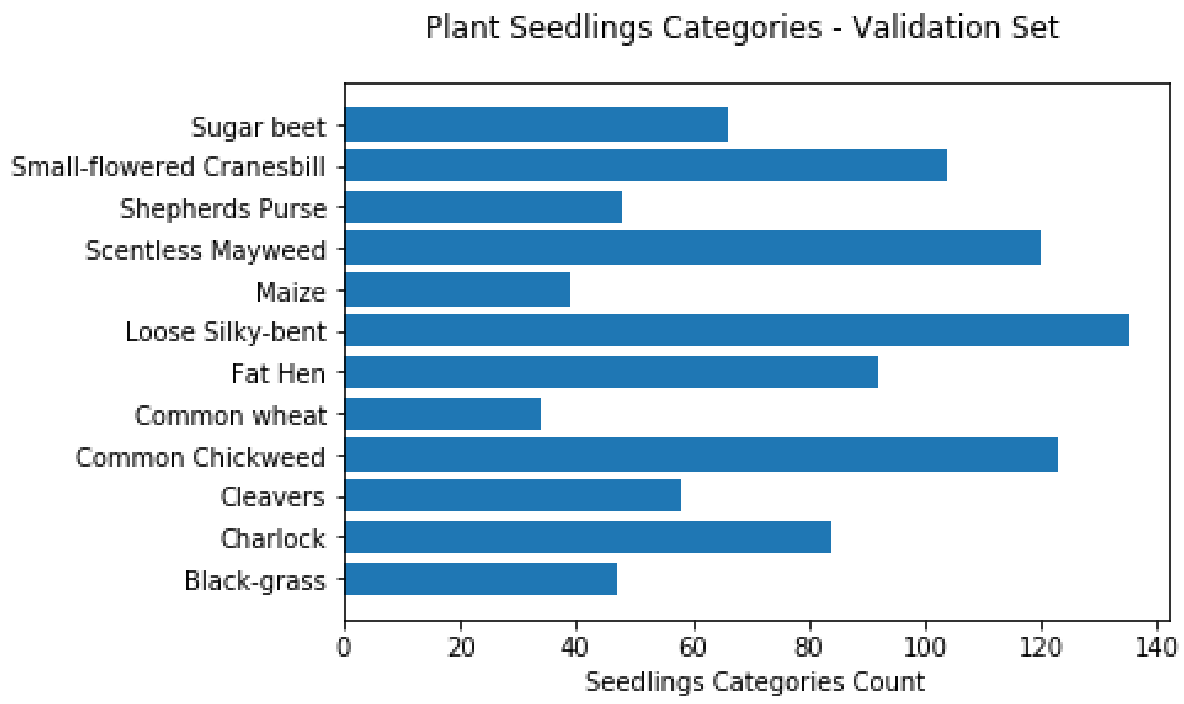
```
Image 1299 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
```

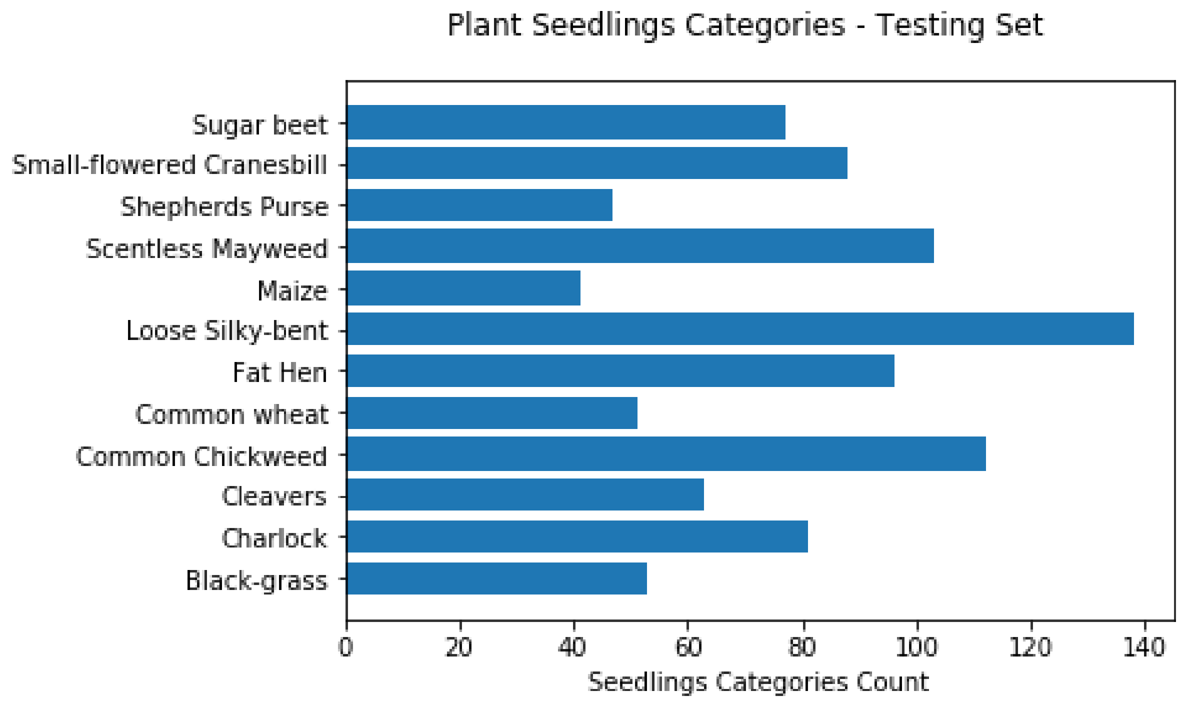


Below is the visualization for number of plant seedlings categories for total set and for each of the sets – training set, validation set and testing set. Clearly, the training dataset is unbalanced i.e. there is significant difference in number of images for different seedlings. So, I have used F1 score as evaluation metric. Additionally, I have used confusion matrix to describe the performance of a classification model on set of test data, for both benchmark model as well as InceptionV3 model.






Plant Seedlings Categories - Validation Set
Sugar beet
Small-flowered Cranesbill
Shepherds Purse
Scentless Mayweed
Maize
Loose Silky-bent
Fat Hen
Common wheat
Common Chickweed
Cleavers
Charlock
Black-grass
0
20
40
60
80
100
120
140
Seedlings Categories Count


Plant Seedlings Categories - Testing Set
Sugar beet
Small-flowered Cranesbill
Shepherds Purse
Scentless Mayweed
Maize
Loose Silky-bent
Fat Hen
Common wheat
Common Chickweed
Cleavers
Charlock
Black-grass
0
20
40
60
80
100
120
140
Seedlings Categories Count

Algorithms and Techniques

I built my project in following broad steps:

Step 1: Import Plant Seedlings dataset. Used np_utils.to_categorical to one hot encode the target values and store them into a NumPy array.

Step 2: Pre-process data:
- Split the seedlings data,
- Visualize the images,
- Rescale the images,
- Visualize the seedlings count for total dataset and each of the sets – training set, validation set and testing set.
- Calculate F1 score for using custom metrics in Keras models.

Step 3: Create a Convolution Neural Network to classify plant seedlings (from scratch):
- Create Model Architecture
- Compile the model
- Train the model
- Load the model with best validation loss
- Test the model – calculate F1 score and create confusion matrix

Step 4: Create a Convolution Neural Network to classify plant seedlings (using transfer learning):
- Create a basic InceptionV3 Model Architecture
- Compile the model
- Train the model
- Load the model with best validation loss
- Test the model – calculate F1 score and create confusion matrix

Step 5: Compare the F1 scores for the two models

Benchmark

I built a simple Convolution Neural Network model as a benchmark model, to confirm that problem is actually 'solvable'. Following is the architecture of the model:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_9 (Conv2D)            (None, 224, 224, 16)      208
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 112, 112, 16)      0
_________________________________________________________________
```

```
dropout_5 (Dropout)          (None, 112, 112, 16)      0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 112, 112, 32)      2080
_________________________________________________________________
max_pooling2d_10 (MaxPooling (None, 56, 56, 32)        0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 56, 56, 64)        8256
_________________________________________________________________
max_pooling2d_11 (MaxPooling (None, 28, 28, 64)        0
_________________________________________________________________
dropout_6 (Dropout)          (None, 28, 28, 64)        0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 28, 28, 128)       32896
_________________________________________________________________
max_pooling2d_12 (MaxPooling (None, 14, 14, 128)       0
_________________________________________________________________
global_average_pooling2d_3 ( (None, 128)               0
_________________________________________________________________
dense_3 (Dense)              (None, 12)                1548
=================================================================
Total params: 44,988
Trainable params: 44,988
Non-trainable params: 0
_________________________________________________________________
```

III. Methodology

Data Pre-processing

Following are the steps for data pre-processing:

- Import the plant seedlings data set:
    - Created a load_dataset function to load all the files in a NumPy array "seedlings_files" and
    - One hot encoded the target seedlings names using function np_utils.to_categorical and save them in NumPy array "seedlings_targets"
    - Count the seedlings categories and images
    - Print the seedlings names

- Split the total dataset into three parts:

Dataset	**% data**	**Images**
training set (to train model)	60%	2850
validation set (to cross-validate model)	20%	950
testing set (to evaluate performance of model)	20%	950

- Visualize the images: All the images were of different sizes.
- Rescale the images:
    - Rescale the images to a size 224*224 using image.load_img function from keras.preprocessing module.
    - Convert the image to 3D tensor of (224, 224, 3)
    - Convert the 3D tensor to 4D tensor with shape (1, 224, 224, 3)
    - Finally, divide each value in tensor by 255 to get values between [0,1].
- Visualize the data for number of plant seedlings categories for total set and for each of the sets – training set, validation set and testing set. Clearly, the training dataset is unbalanced i.e. there is significant difference in number of images for different seedlings. So, I have used F1 score as evaluation metric. Additionally, I have used confusion matrix to describe the performance of a classification model on set of test data, for both benchmark model as well as InceptionV3 model.

Implementation

Algorithm used for CNN from scratch:

- Input consists of 224*224*3 tensor.
- And network consists of following 4 convolution layers each followed by 1 max pooling layer. Also, added two dropout layers to prevent overfitting.
- Convolution layer with 16 filters, 2 kernel size and 'relu' activation function.
- Max pooling layer with pool size of 2.
- Drop out layer with rate of 0.2.
- Convolution layer with 32 filters, 2 kernel size and 'relu' activation function.
- Max pooling layer with pool size of 2.
- Convolution layer with 64 filters, 2 kernel size and 'relu' activation function.
- Max pooling layer with pool size of 2.
- Drop out layer with rate of 0.3.
- Convolution layer with 128 filters, 2 kernel size and 'relu' activation function.
- Max pooling layer with pool size of 2.
- Global average pooling layer.
- Dense layer with 12 filters (for 12 seedlings) and 'softmax' activation function.
- This model architecture included total of 44,988 trainable params.

Algorithm used for CNN with transfer learning:

- Input consists of 224*224*3 tensor.
- Create a base model with pre-trained weights from 'imagenet'.
- Freeze the first 200 layers of the network.
- Global average pooling layer.
- Fully connected layer with 256 filters and 'relu' activation function.
- Dense layer with 12 filters (for 12 seedlings) and 'softmax' activation function.

- This model architecture included total of 15,331,916 trainable params.

Refinement

I implemented following algorithms and techniques before reaching the final solution:

- Freeze first 143 layers of ResNet model and tried to train rest of the model after adding one fully connected layer with 512 filters (‘relu’ activation function and one dense layer with 12 filters (‘softmax’ activation function). It took lot of time to train (30+ mins per epoch) and validation score was not better than benchmark model. Also, tried training ResNet model after removing few top layers and adding series of dense layers and found results to be similar.
- Freeze first 85 layers of Xception model and tried to train rest of the model after adding one fully connected layer with 256 filters (‘relu’ activation function and one dense layer with 12 filters (‘softmax’ activation function). It took lot of time to train (approximately 30 mins per epoch) and validation score didn’t improve from 0.6151. Final F1 score was 0.63368 approx.
- I also tried training the networks on AWS but there was not much improvement in training time for this network.
- Finally, I used InceptionV3 model in which I froze first 200 layers and trained rest of the model after adding one fully connected layer with 256 filters (‘relu’ activation function and one dense layer with 12 filters (‘softmax’ activation function). Each epoch took around 15 minutes to train and the final result gave F1 score of 0.806 approx.

IV. Results

Model Evaluation and Validation

Final model has F1 score of 0.80631 which I think is reasonable enough to predict many of the seedlings correctly. Also, the confusion matrix for the final model reiterates that model generalizes well with unseen data. I understand that models results can be trusted for most of the seedlings categories except for these four categories: “Black-grass”, “Common wheat”, “Maize” and “Shepherds Purse”.

Justification

The results from final model (F1 score = 0.806) are much better than that of benchmark model (F1 score = 0.611). The improvement of F1 score in final model can be considered significant enough to have solved the problem.

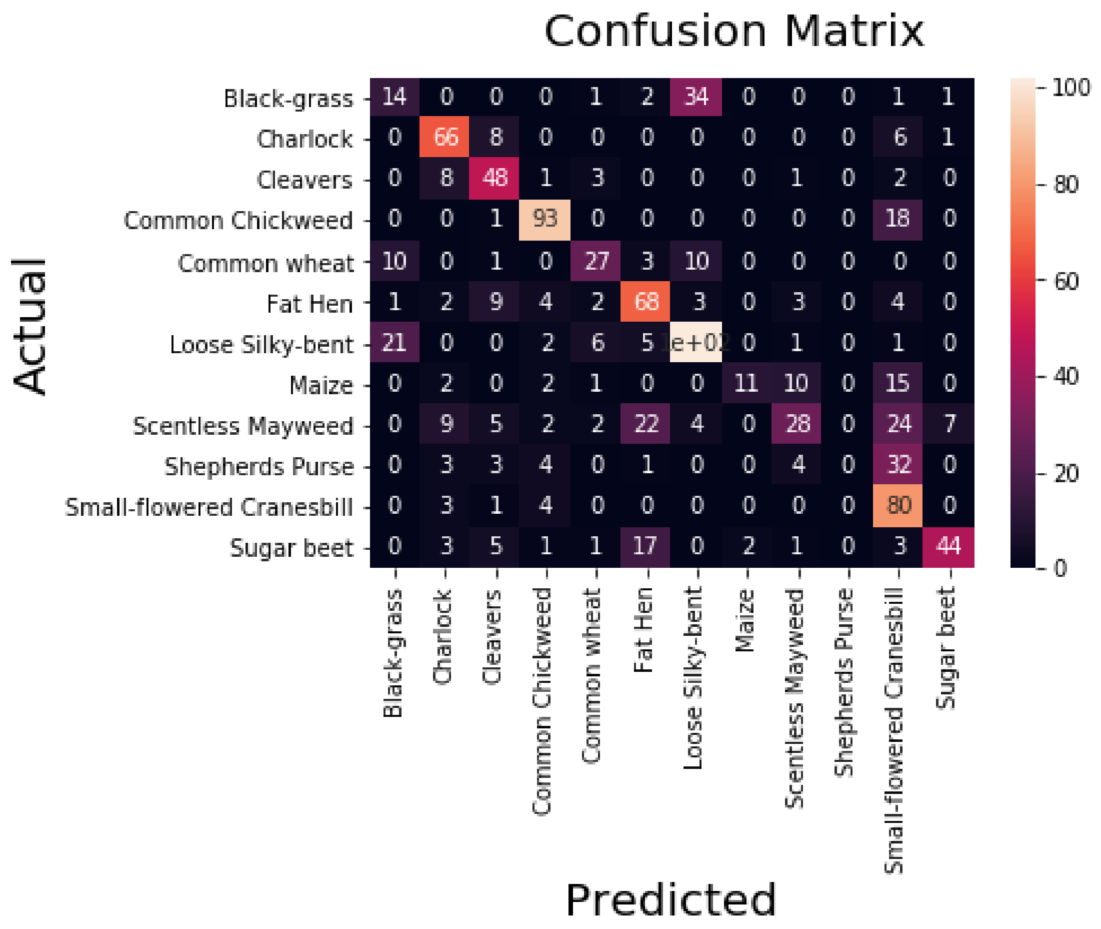
V. Conclusion

Free-Form Visualization

I used the following visualization techniques: one to analyse data and the other to measure the performance of the model.

- Plotted the seedlings image count for total dataset and each of the sets – training set, validation set and testing set.
  (Note: plots shown in Exploratory visualization section above)

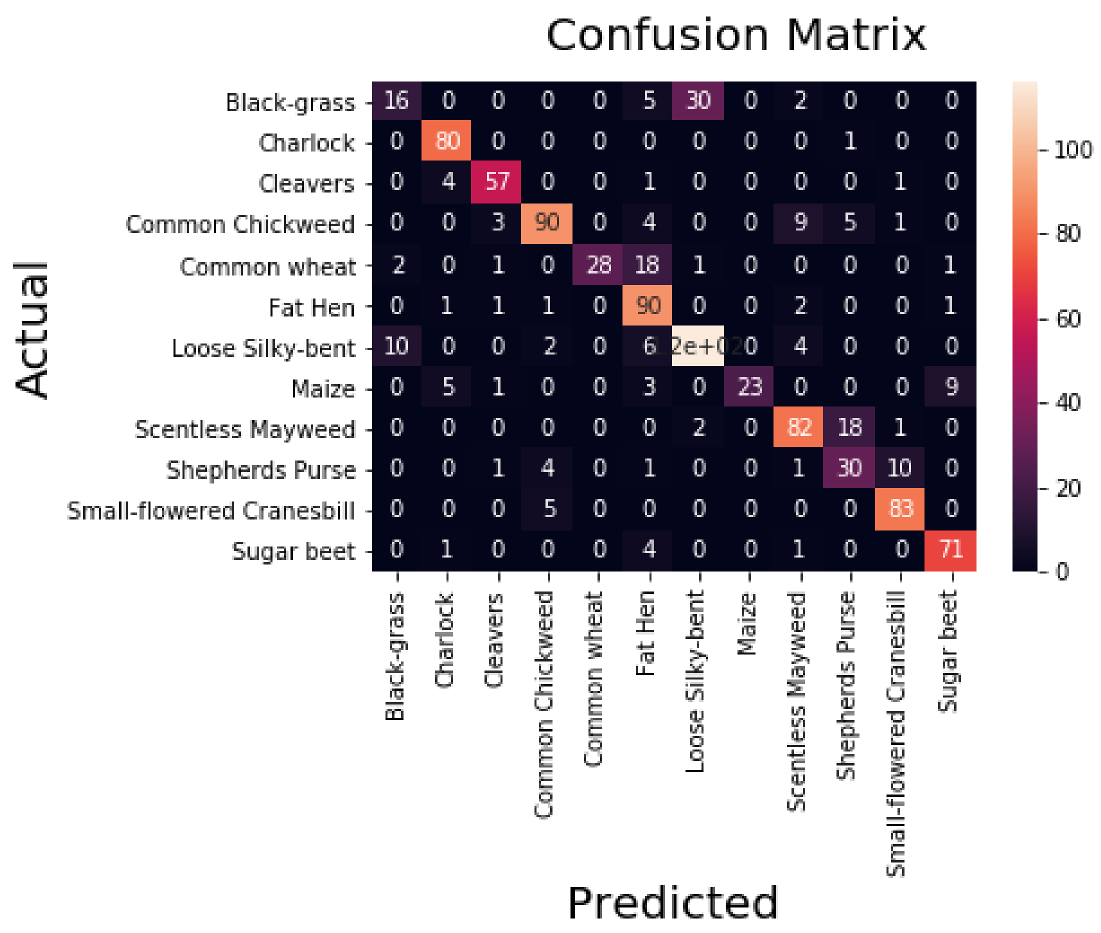
- Plotted the confusion matrix to get insight of classification errors made by our classifier.
  Confusion Matrix for Benchmark Model



Actual \ Predicted	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	14	0	0	0	1	2	34	0	0	0	1	1
Charlock	0	66	8	0	0	0	0	0	0	0	6	1
Cleavers	0	8	48	1	3	0	0	0	1	0	2	0
Common Chickweed	0	0	1	93	0	0	0	0	0	0	18	0
Common wheat	10	0	1	0	27	3	10	0	0	0	0	0
Fat Hen	1	2	9	4	2	68	3	0	3	0	4	0
Loose Silky-bent	21	0	0	2	6	5	1e+02	0	1	0	1	0
Maize	0	2	0	2	1	0	0	11	10	0	15	0
Scentless Mayweed	0	9	5	2	2	22	4	0	28	0	24	7
Shepherds Purse	0	3	3	4	0	1	0	0	4	0	32	0
Small-flowered Cranesbill	0	3	1	4	0	0	0	0	0	0	80	0
Sugar beet	0	3	5	1	1	17	0	2	1	0	3	44

-

Confusion Matrix for Final Model



	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	16	0	0	0	0	5	30	0	2	0	0	0
Charlock	0	80	0	0	0	0	0	0	0	1	0	0
Cleavers	0	4	57	0	0	1	0	0	0	0	1	0
Common Chickweed	0	0	3	90	0	4	0	0	9	5	1	0
Common wheat	2	0	1	0	28	18	1	0	0	0	0	1
Fat Hen	0	1	1	1	0	90	0	0	2	0	0	1
Loose Silky-bent	10	0	0	2	0	6	1.2e+02	0	4	0	0	0
Maize	0	5	1	0	0	3	0	23	0	0	0	9
Scentless Mayweed	0	0	0	0	0	0	2	0	82	18	1	0
Shepherds Purse	0	0	1	4	0	1	0	0	1	30	10	0
Small-flowered Cranesbill	0	0	0	5	0	0	0	0	0	0	83	0
Sugar beet	0	1	0	0	0	4	0	0	1	0	0	71

Reflection

In this project, I imported the plant seedlings data set: created a load_dataset function to load all the files in a NumPy array "seedlings_files" and one hot encoded the target seedlings names using function np_utils.to_categorical and saved them in NumPy array "seedlings_targets"

Then, I split the total dataset into three parts: training set (60%), validation set (20%) and testing set (20%). And rescaled the images to 4D tensor with shape (1, 224, 224, 3). Then I trained two CNN models – one from scratch and other using transfer learning on InceptionV3 model.

Before submitting the final solution, I experimented with different models of both simple CNN and one using transfer learning (specially ResNet, Xception and InceptionV3 model). While trying different implementations, I built better understanding of architecture of these models. I also learnt about connecting to AWS EC2 instance and running a Jupyter notebook using Ubuntu commands. Overall, this project gave me good intuition to handle image classification problems in future.

Improvement

Following aspects of implementation could be improved:

- Experiment with different pre-trained models: I have mainly focussed on ResNet, Xception and InceptionV3 model, experimented with different layers and activation functions. It would be a good idea to experiment on VGG16, VGG19 and other Inception models.
- Use Image Augmentation: This will help boost the performance of the model by creating different variants of images using random rotation, shifts, flips etc.
- Convert image from RGB to HSV, apply techniques like masking, segmentation and sharpening.

I would like work further on this project and experiment with above improvements, considering my final solution as a benchmark.