Capstone Project

Machine Learning Engineer Nanodegree

Noura Hussein

june 5th, 2018

I. Definition

Image classification has become one of the most important problems nowadays. As computers are getting better at understanding images due to advances in computer vision, solving of image classification problem using Deep learning becoming increasingly realistic.
In This project I will use one of kaggle Competition dataset, this dataset containing images of approximately 960 unique plants belonging to 12 species at several growth stages.
The database have been recorded at Aarhus University Flakkebjerg Research station in a collaboration between University of Southern Denmark and Aarhus University. You can find the dataset here.
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).
This project was inspired by this research center.

Problem Statement:

The goal of this project is to is to be able to differentiate between weed seedling and crop seedling; the tasks involved are the following:
Data exploration

1. Data exploration

- Visualize the distribution of data

2. Data preprocessing

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

2. CNN

- 2.1 Define the model
- 2.2 Set the optimizer and annealer
- 2.3 Data augmentation

4. Evaluate the model

- 3.1 Training and validation curves
- 3.2 Confusion matrix

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

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.

The most important errors are also the most intrigues.

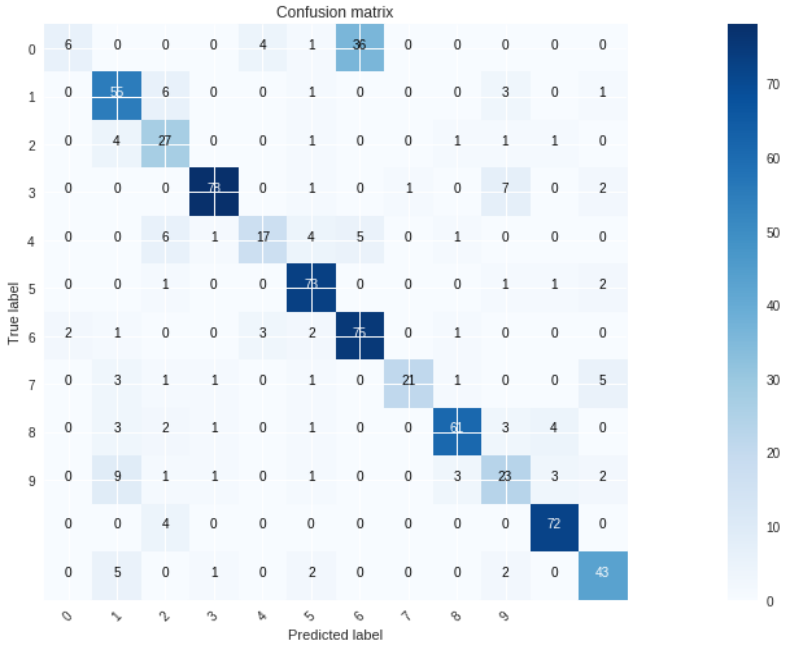
- For CNN I will use the same function provided in Keras API evaluate

Which return the loss value and accuracy matrices for the model in a test mode.

- Accuracy score matrix provided by [confusion matrix sklearn]

Compute confusion matrix to evaluate the accuracy of a classification

Here is a plot for confusion matrix of this model:



II. Analysis

Data Exploration:

Plant seedling data set size is 1.6 GB divided into 12 folder in each one it contain number of image belong to certain class. I will split it later into training set and testing set.

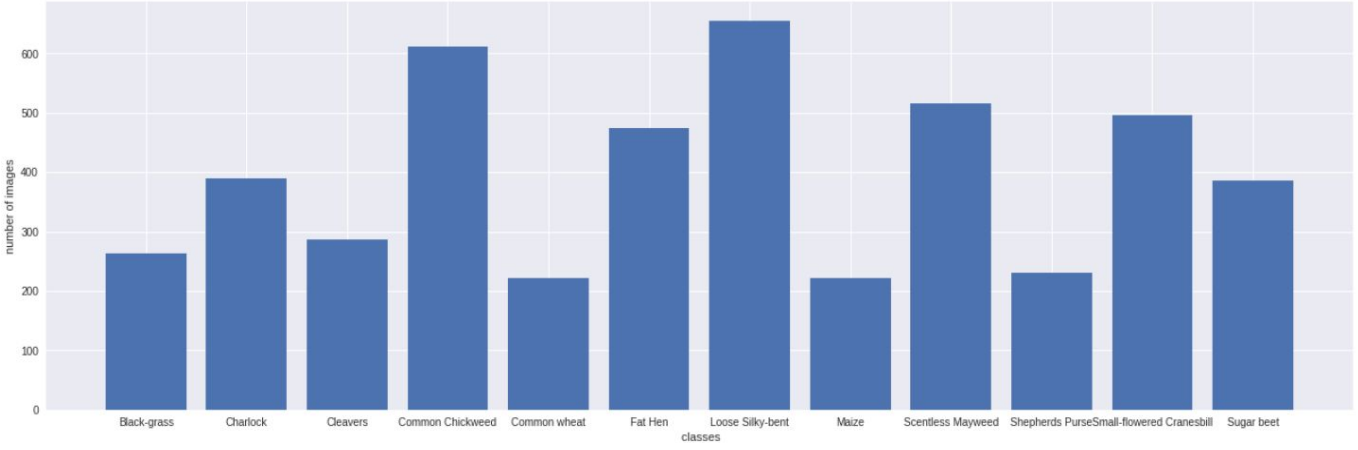
- There is 12 types of plant seedling:

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

- Not all images has the same size (I have been resized all images in the preprocess stage)

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

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



Algorithms and Techniques

The classifier is a CNN (Convolutional Neural Network), which is the state-of-the-art algorithm for most image processing tasks, including classification. It needs a large amount of training data compared to other approaches; and this dataset already are big enough to fit this criteria.
The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positive using a threshold .
(The tradeoff is that this increases the number of false negatives.)
The following parameters can be tuned to optimize the classifier:
❖Classification threshold (see above)

❖ Training parameters
➢ Training length (number of epochs)
➢ Batch size (how many images to look at once during a single training step)
➢ Solver type (what algorithm to use for learning)
➢ Learning rate (how fast to learn; this can be dynamic)
➢ Weight decay (prevents the model being dominated by a few “neurons”)

❖ Neural network architecture
➢ Number of layers
➢ Layer types (convolutional , fully-connected , or pooling)
➢ Layer parameters
❖ Preprocessing parameters (see the Data Preprocessing section)

Benchmark:

Plant seedling classification dataset is very similar to this dataset is very similar to CIFAR-10 dataset.
which used in “image classification” project (has 10 images for different objects) and also this data set has 12 types of seedling plant.

III. Methodology

Implementation:

The implementation process can be split into two main stages:

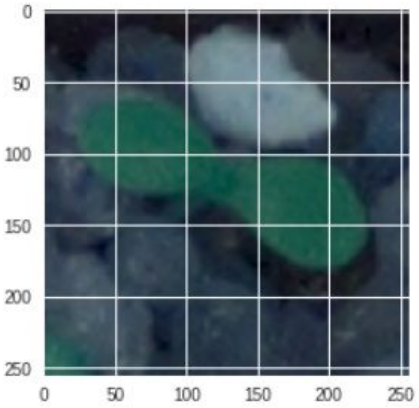
- Preprocessing data stage
- The classifier training stage

Preprocessing:

1- Resizing Images:

Images have not the same size so I have resized the images to 256*256 pixel to feed it later to the neural network

In this figure the image after resize:

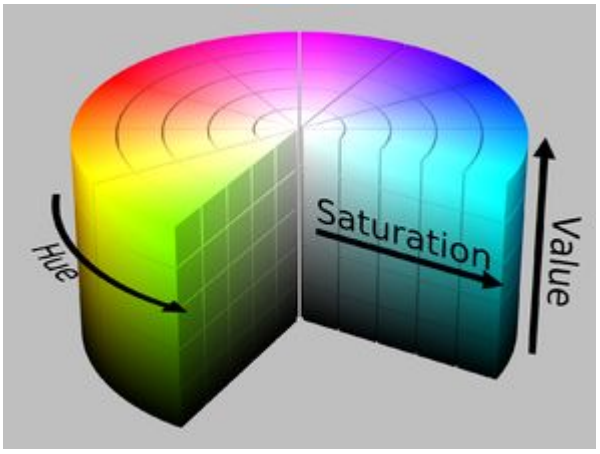


2- Creating mask for the images :

`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. I can create this mask using HSV of the image.

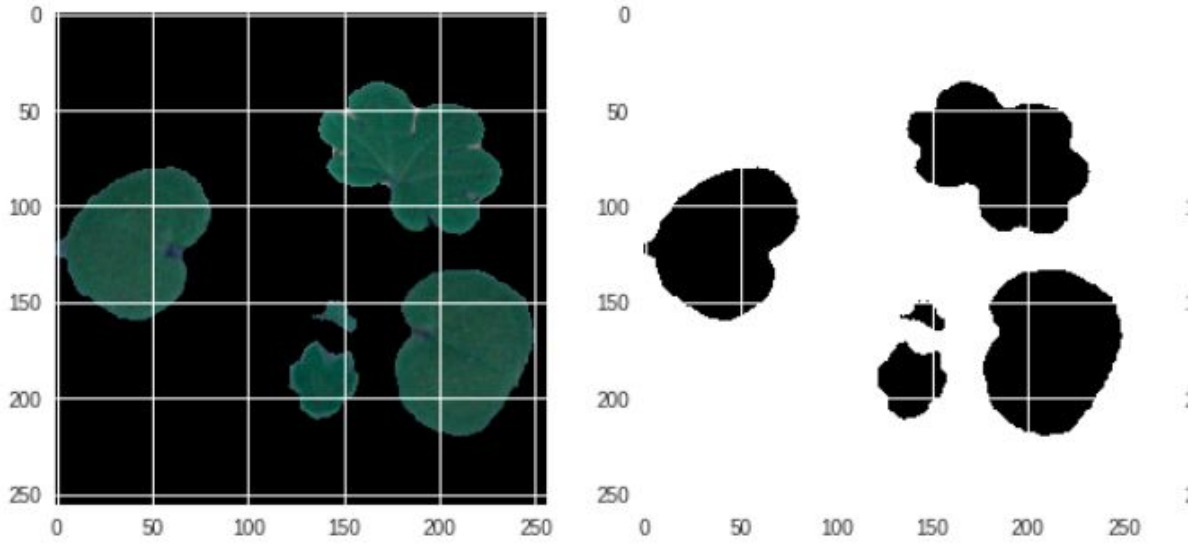
The HSV color-space is suitable for color detection because with the Hue we can define the color and the saturation and value will define "different kinds" of the color. (For example it will detect the red, darker red, lighter red too). We cannot do this with the original BGR color space.

This figure illustrate HSV space of the image.



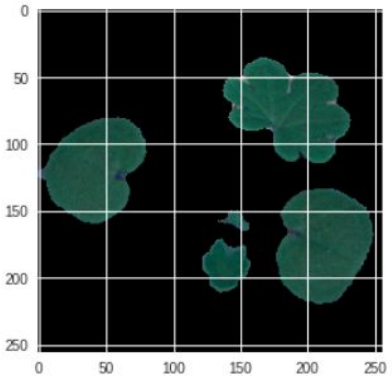
After converting RGB to HSV I started to apply *morphological operations* ***morphological operations*****:** one of most common morphological operation is **closing** : closing used to close the small halls in the images.
This figure below illustrate the image before and after applying closing:

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



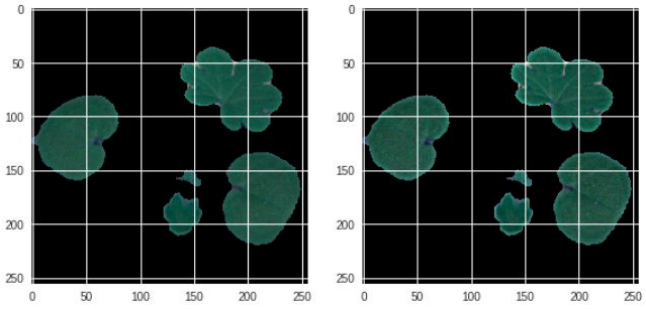
Segmentation:

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:



Sharpening:

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



Label encoding:

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

The classifier training stage

CNN Architecture:

CNN algorithm consists 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.

I 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. I chose 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.

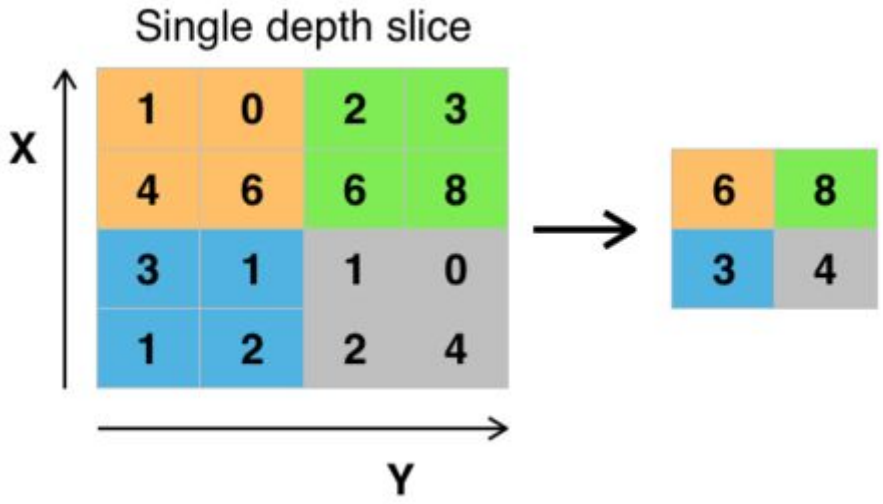
Each filter transforms a part of the image (defined by the kernel size) using the kernel filter. The kernel filter matrix is applied on the whole image. Filters can be seen as a transformation of the image.

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.

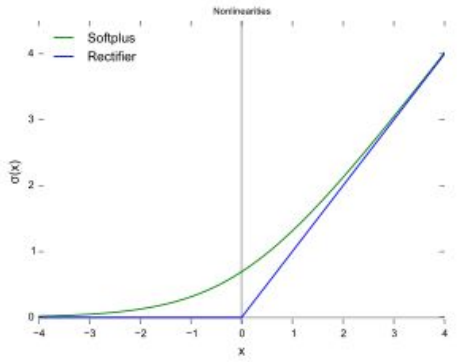
Dropout is a regularization method, where a proportion of nodes in the layer are randomly ignored (setting their weights to zero) for each training sample. This drops randomly a proportion of the network and forces the network to learn features in a distributed way. This technique also improves generalization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

Refinement:

- Adding preprocessing stage before neural network improved my model accuracy from 46% to 59%
- Decreasing learning rate from .01 to .001 avoid me the overfitting of the model.
- By adding data augmentation accuracy increased to : 91%
- Adding some additional layer also improved the model
- Changing dropout from .25 to .3 improved the accuracy for 86%

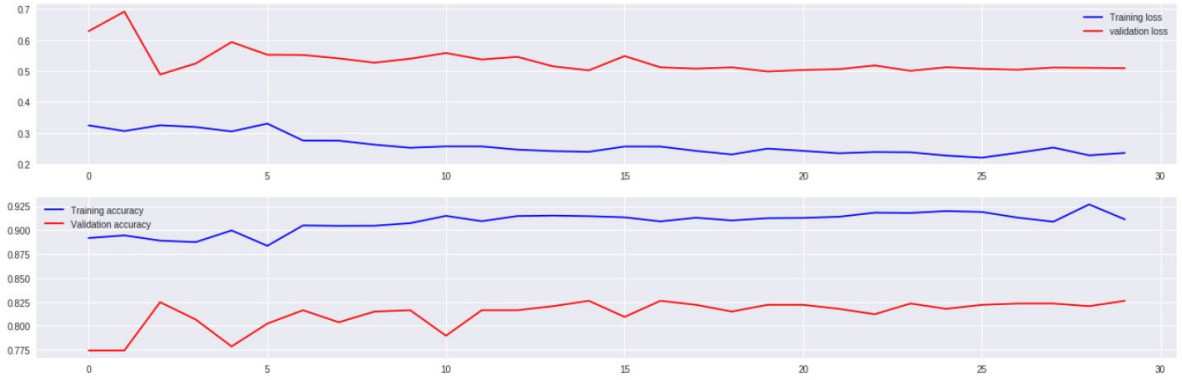
IV. Results

Model Evaluation and Validation:

During development, a validation set was used to evaluate the model.
The final architecture and hyperparameters 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.
- ❖ The first convolutional layer learns 32 filters, the second learns 64 filters the third learns 128 filters.
- ❖ The convolutional layers have a stride of 2*2
- ❖ Like the convolutional layers, the pooling layers halve the resolution too.
- ❖ False positives are rare but present

As for the evaluation and validation, I have used 2 main values to check: Accuracy and validation loss.



Justification

The results of the final classifier are much better than that of the benchmark model. It has score of 86.3% is a decent score when compared to 60.8% (benchmark model's performance). I believe the final solution will definitely contribute significantly towards solving the current problem and also with more training data and more preprocessing stages, there are possibilities of improving the model further.

Conclusion:

To show the model quality we get the images which classified incorrectly in 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.

Reflection:

I wanted to pick a problem that would give me much more clarity and exposure on how to deal with image classification problems, and I believe that choice of this problem has justified that need.
For this problem, after downloading the dataset from Kaggle, I loaded the dataset and converted the categorical file names into array of 1s and 0s using the labelbinarizer . I explored the data by visualizing the categorical distribution of the dataset by plotting a graph (using matplotlib) to check if the dataset is well balanced or not. After splitting the dataset into train, validation sets, the images were converted into 4D tensors for further processing. Once I built CNN models, I fed these tensors and evaluated the model's performance using confusion metrics. I watched, over multiple iterations, how does the number of layers in the convolutional network have an significant effect on the performance of the classifier, how does adding dropout layers reduce potential overfitting, This project gave me a good insight on how to deal with future image classification problems and encouraged me to work on further improving my current model.

Improvement

As an improvement and future work, I would like to try data masking on the training set. Noise from the background of the images can be cancelled by masking images. I believe that without the background noise and restricting the visibility to the green leaves, the model can be trained better, and we may notice significant improvement in the performance.
Another implementation that can be tried is data augmentation. As the dataset is highly unbalanced, augmenting data to the under-represented classes might give a good boost to the total number of training images yielding a well-balanced dataset. Training the model on such dataset may give us significant improvement in the performance.

References:

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