Machine Learning Engineer

Nanodegree Capstone Project Report

Dursun KOÇ March 18, 2018

Project Overview

A seedling is a young plant sporophyte developing out of a plant embryo from a seed. Seedling development starts with germination of the seed. A typical young seedling consists of three main parts: the radicle (embryonic root), the hypocotyl (embryonic shoot), and the cotyledons (seed leaves). The two classes of flowering plants (angiosperms) are distinguished by their numbers of seed leaves: monocotyledons (monocots) have one blade-shaped cotyledon, whereas dicotyledons (dicots) possess two round cotyledons. Gymnosperms are more varied. For example, pine seedlings have up to eight cotyledons. The seedlings of some flowering plants have no cotyledons at all. These are said to be acotyledons.

The plumule is the part of a seed embryo that develops into the shoot bearing the first true leaves of a plant. In most seeds, for example the sunflower, the plumule is a small conical structure without any leaf structure. Growth of the plumule does not occur until the cotyledons have grown above ground. This is epigeal germination. However, in seeds such as the broad bean, a leaf structure is visible on the plumule in the seed. These seeds develop by the plumule growing up through the soil with the cotyledons remaining below the surface. This is known as hypogeal germination.

The aim of this project is to build a convolutional neural network that classifies different species of plant seedlings while working reasonably well under constraints of computation with help of transfer learning technique.

Giselsson T. M., Jørgensen R. N., Jensen P. K., Dyrmann M. and Midtiby H. S. in their study[1] of “A Public Image Database for Benchmark of Plant Seedling Classification Algorithms” created a database of seedlings pictures, and tried to distinguish them by naïve bayes with a pre-processing of pictures. Now their database is placed in a kaggle competition[2], and I will be using their database provided in kaggle.

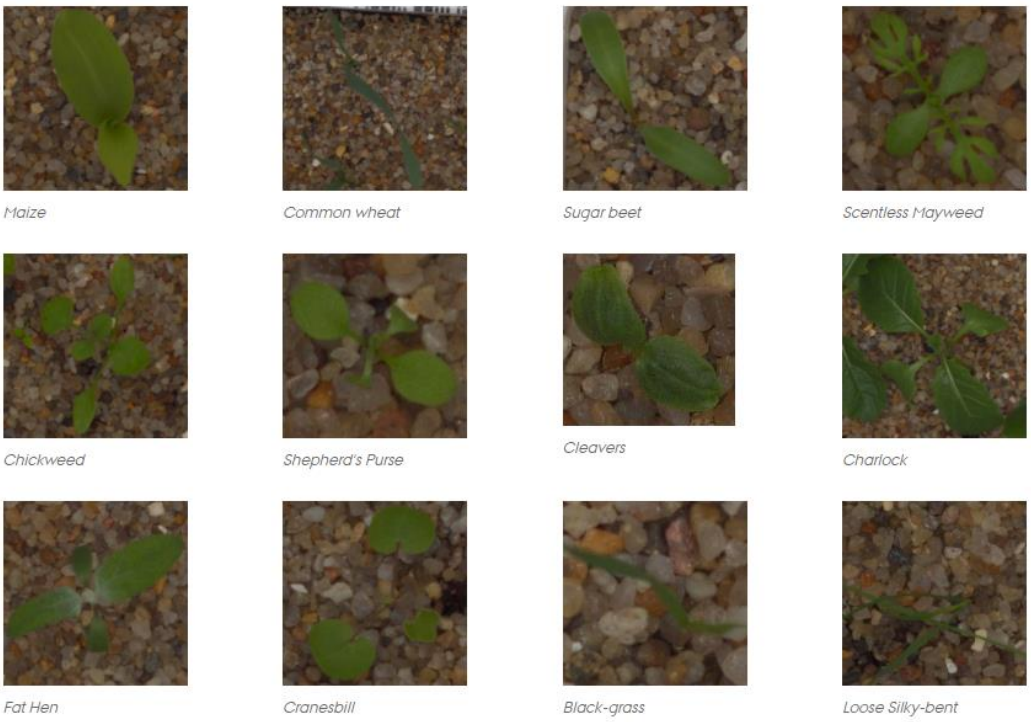
Problem Statement

The goal is to differentiate a weed from a crop seedling, the ability to do so effectively can mean better crop yields and better stewardship of the environment.

The seedling data set is provided by the Aarhus University Signal Processing group, in collaboration with University of Southern Denmark. The dataset containing images of approximately 960 unique plants belonging to 12 species at several growth stages.

Dataset and Data Exploration

As it is not feasible to include too many species, the researchers decided to use only a subset of high importance to the Danish agricultural industry. They have used Styrofoam boxes to grow samples. They had 56 boxes, each containing on average 25 samples with only one specie sown. According to the researchers 80 samples of individual species at the same growth stage is sufficient to capture the main variations within a specie, so 4 boxes of each species are sown to allow for 20% germination failure. Images are recorded multiple times over a 20-day period at an interval of 2 to 3 days, starting a few days after emergence. I will be downloading this dataset[1] from kaggle. Below you will find samples of each species from the database:



The distribution of each specie in the training data is as follows;

```
In [34]: classes = os.listdir("../input/train")

         all_images_class = [os.listdir("../input/train/"+c) for c in classes]

In [38]: df = pd.DataFrame({"n_images": [len(x) for x in all_images_class]}, index=classes)
         df.index.name = "Specie"

In [41]: df.plot(kind="bar", figsize=(15,10))
         df
```

Out[41]:

Specie	n_images
Black-grass	263
Charlock	390
Cleavers	287
Common Chickweed	611
Common wheat	221
Fat Hen	475
Loose Silky-bent	654
Maize	221
Scentless Mayweed	516
Shepherds Purse	231
Small-flowered Cranesbill	496
Sugar beet	385

Some species have more samples than some others; for example, number of “Loose Silky Bent” samples are three times greater than number of “Maize” samples.

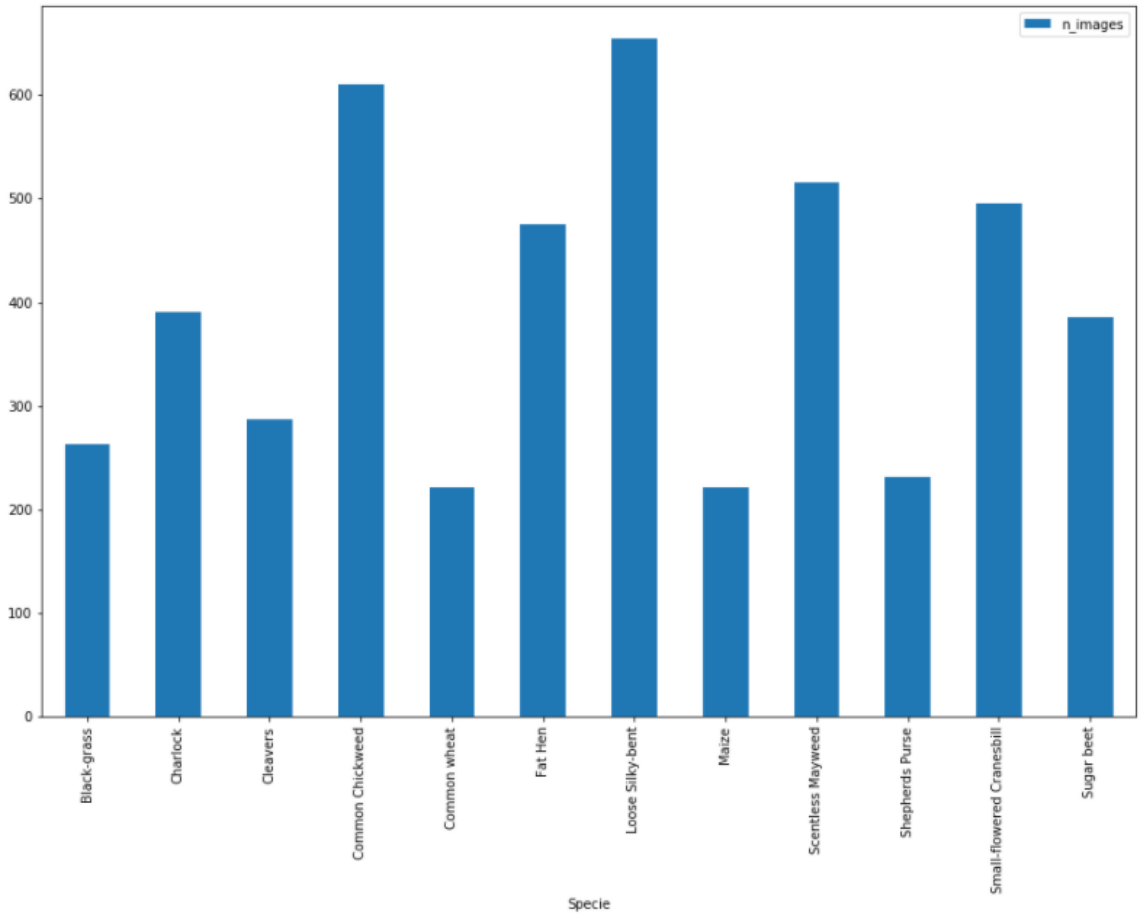


Image sizes varies so we will need to resize them when we input them to the CNN. Lastly; kaggle provides a training set and a testing set, I will use training set to generate my model with success criteria and use the test set for validation.

Metrics

The metric used for this Kaggle competition is MeanFScore, which at Kaggle is actually a micro-averaged F1-score.

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

$$Precision_{micro} = \frac{\sum_{k \in C} TP_k}{\sum_{k \in C} TP_k + FP_k}$$

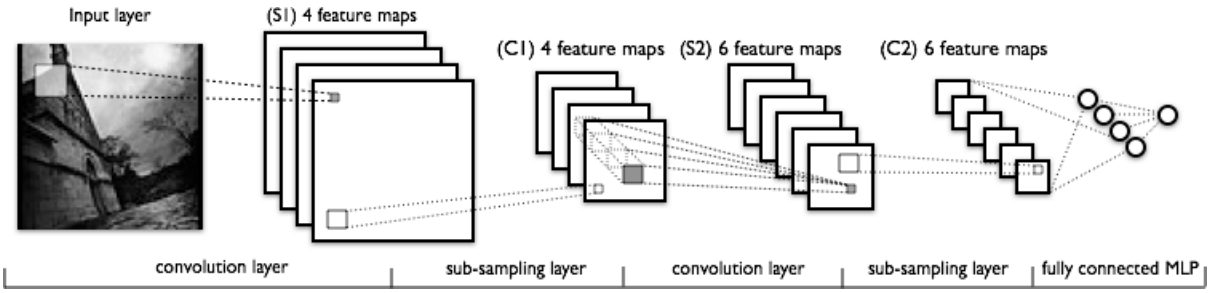
$$Recall_{micro} = \frac{\sum_{k \in C} TP_k}{\sum_{k \in C} TP_k + FN_k}$$

F1-score is the harmonic mean of precision and recall;

$$MeanFScore = F1_{micro} = \frac{2 Precision_{micro} Recall_{micro}}{Precision_{micro} + Recall_{micro}}$$

CNN Architecture

Convolutional Neural Networks (CNN) are a compose of neurons and those neurons have learnable weights and biases. From this point of view, CNN is not much different from an ordinary neural network [2]. A CNN is a multilayer perceptron designed to require minimal processing [3]. You can see a typical CNN topology below [4];



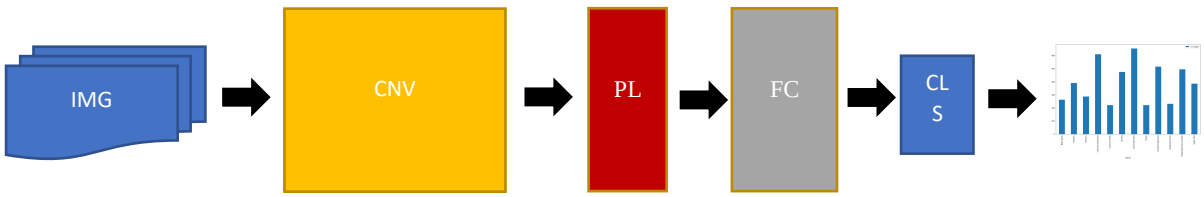
Sparse, convolutional layers and max-pooling are at the heart of the CNN models. The lower-layers are composed to alternating convolution and max-pooling layers. The upper-layers however are fully-connected and correspond to a traditional MLP (hidden layer

+ logistic regression). The input to the first fully-connected layer is the set of all features maps at the layer below.

From an implementation point of view, this means lower-layers operate on 4D tensors. These are then flattened to a 2D matrix of rasterized feature maps, to be compatible with our previous MLP implementation.

Simple CNN Architecture

In our simple CNN, we have used 1 CNV layer followed by 1 PL layer which is followed by 1 FC layer as illustrated below;




```
EPOCHS = 20
INIT_LR = 1e-3
BS = 64

small_conv_nn = Sequential()

small_conv_nn.add(Conv2D(10, (5, 5), padding="same", input_shape=input_shape))
small_conv_nn.add(Activation("relu"))
small_conv_nn.add(MaxPooling2D(pool_size=(4, 4), strides=(4, 4)))
small_conv_nn.add(Flatten())

small_conv_nn.add(Dense(12))
small_conv_nn.add(Activation("softmax"))

small_conv_nn.compile(loss="categorical_crossentropy", optimizer=Adam(lr=INIT_LR, decay=INIT_LR / EPOCHS), metrics=[f1_score])
small_conv_nn.summary()
```

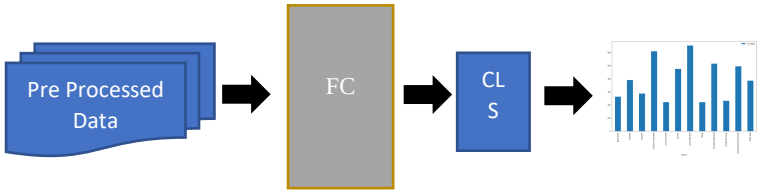
```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 200, 200, 10)      760
_________________________________________________________________
activation_1 (Activation)    (None, 200, 200, 10)      0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 50, 50, 10)        0
_________________________________________________________________
flatten_1 (Flatten)          (None, 25000)             0
_________________________________________________________________
dense_1 (Dense)              (None, 12)                300012
_________________________________________________________________
activation_2 (Activation)    (None, 12)                0
=================================================================
Total params: 300,772
Trainable params: 300,772
Non-trainable params: 0
_________________________________________________________________
```

Using this model, we have received an F1-score of 0.6

Transfer Learning

Pragmatically, most people do not train the whole CNN from scratch as we did in the previous step, they use a ConvNet either as an initialization or a fixed feature extractor for the task of interest. A ConvNet is a previously trained model with a very large dataset.

In this study we are using the Xception pretrained ConvNet. First, we will preprocess our image data with the xception model, after that we feed the resulting array to a simple neural network, illustrated below;




```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_14 (Dense)             (None, 12)                24588
_________________________________________________________________
activation_13 (Activation)   (None, 12)                0
=================================================================
Total params: 24,588
Trainable params: 24,588
Non-trainable params: 0
_________________________________________________________________
Train on 3800 samples, validate on 950 samples
```

Using this model, we have received an F1-score of 0.8 and our processing time decreased from hours to a few minutes.

References

[1] https://www.kaggle.com/c/plant-seedlings-classification/data

[2] http://cs231n.github.io/convolutional-networks/

[3] http://yann.lecun.com/exdb/lenet/

[4] http://deeplearning.net/tutorial/lenet.html