# Plant Seedling Classification Using Convolutional Neural Networks

Machine Learning Engineer Nanodegree Capstone Report

Brungi Vishwa Sourab

January, 2019

## I. Definition

**Project Overview:**

For several years, many people have tried to solve the problem of identifying weeds. They used wide range of approaches with the common goal of identifying weeds from normal plants but no system that has made a commercial breakthrough has been developed. There are various reasons for these systems to be not able to solve the problem. One of them is data collection. A research paper on similar problem identifying plant species by the shape of leaf is published by Ji-Xiang Du, Xiao Feng Wang & Guo-Jun Zhang, Department of Automation, University of Science and Technology of China.

I strongly believe that this problem can be solved using deep learning because the technology(hardware) is capable of doing image identification tasks easily and now is the right time for such breakthrough systems because of the latest techniques in deep learning. Until very recently, the main problem was acquisition of the image data to be used to build robust systems. But, in November of 2017, to support and encourage the development of species recognition techniques for the agricultural industry, the Computer Vision and Biosystems Signal Processing Group, Department of Engineering, Aarhus University has collected the data and made it available to the public for free.

**Problem Statement:**

Differentiating a weed from a crop seedling by the image can pave a significant way to intelligent weed control systems and thus eliminating the unwanted plants in its initial stages only. The ability to do so effectively can mean better crop yields and better stewardship of the environment. The Aarhus University Signal Processing group, in collaboration with University of Southern Denmark, has recently released a dataset containing images of approximately 960 unique plants belonging to 12 species at several growth stages. My goal is to build a classifier that classifies the seedling's class based on the image input, using state of the art deep learning techniques. Here are some sample images of seedlings from the dataset.

Figure 1: Sample images from the Seedlings dataset

The details and the workflow is listed below:

- **Exploring the data**
  - Importing libraries and data
  - View a sample of training data
  - View number of images in training data
  - View species vise images count
- **Data Preprocessing**
  - Transforming Images (Resize, Converting to Arrays, Normalizing)
  - Sharpening Images
  - Creating Segments of leaves
  - Splitting data for training and Validation
- **Model Building**
  - Basic CNN for benchmarking
  - Complex CNN architectures

- Hyperparameter Tuning
- Training and Validating the models
- Saving best model and weights to disk

- **Algorithm Building**
  - Creating a pipeline for image classification
  - Loading and predicting the class on saved model

**Metrics:**

Performance is measured using weighted averages of **f1 scores** for each fold. Given positive/negative rates for each class *k*, the resulting score is computed this way:

Precision (P): Precision is a metric that says, Out of all points that are predicted to be positive, how many are actually Positive.

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

Recall (R): Recall is a metric that says, Out of all positive points, how many are actually positive.

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

**F1 Score:** $F_1$ score is the harmonic mean of precision and recall.

$$f1\ Score = 2 * (\frac{Precision*Recall}{Precision+Recall})$$

The Performance of our model is measured using f1 score because it includes both precision and recall just as needed by the problem. Accuracy score also is used to measure

the performance of our model. Classification Report is used to present the report of our model.

The classification report of our model is shown below:

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| 0 | 0.69 | 0.65 | 0.67 | 34 |
| 1 | 0.88 | 0.98 | 0.93 | 58 |
| 2 | 0.98 | 0.86 | 0.91 | 50 |
| 3 | 0.99 | 0.98 | 0.98 | 97 |
| 4 | 0.94 | 0.94 | 0.94 | 36 |
| 5 | 1.00 | 0.94 | 0.97 | 69 |
| 6 | 0.87 | 0.88 | 0.88 | 94 |
| 7 | 0.93 | 1.00 | 0.96 | 27 |
| 8 | 0.93 | 0.99 | 0.96 | 72 |
| 9 | 0.98 | 0.96 | 0.97 | 47 |
| 10 | 0.99 | 1.00 | 0.99 | 76 |
| 11 | 0.96 | 0.94 | 0.95 | 53 |
| avg / total | 0.94 | 0.94 | 0.94 | 713 |

The model we built did pretty good in identifying the weed plants (Class 3) with f1 Score of 0.98.

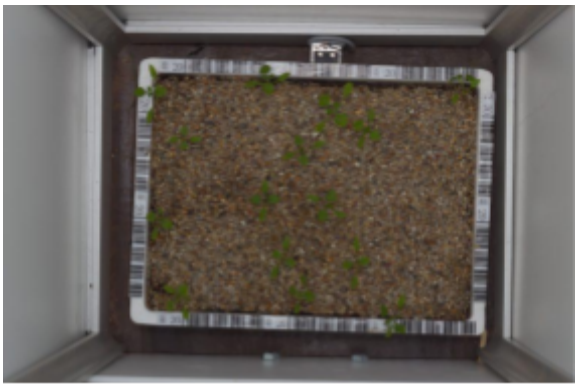
## II. Analysis

**Data Exploration:**

The Plant Seedlings Dataset contains images of approximately 960 unique plants belonging to 12 species at several growth stages. It comprises annotated RGB images with a physical resolution of roughly 10 pixels per mm.

The database have been recorded at Aarhus University Flakkebjerg Research station in a collaboration between University of Southern Denmark and Aarhus University.

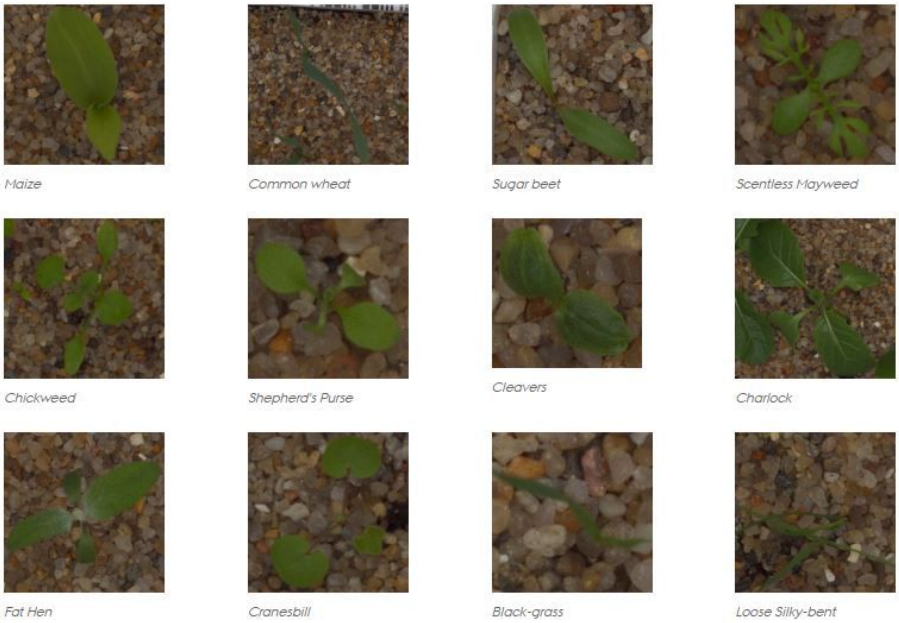
The datasets are available for download in different versions i.e, Raw images (9.7GB), Cropped Images(1.7 GB) and Segmented Images (258 MB). In my project, I use Cropped Images version of the dataset. A sample of raw type of images is shown below:



Raw chickweed image sample

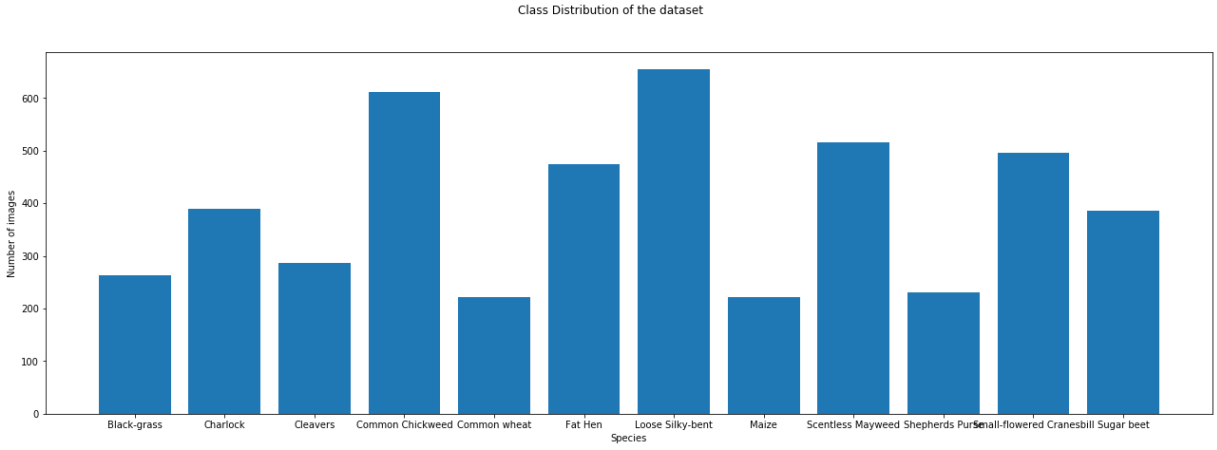
*Image source: Computer Vision and Biosystems Signal Processing Group, Aarhus University*

Since it is not possible to include too many species in the database, only a subset of high importance to the Danish agricultural industry are chosen to make the database. There are 12 species of images in the database:

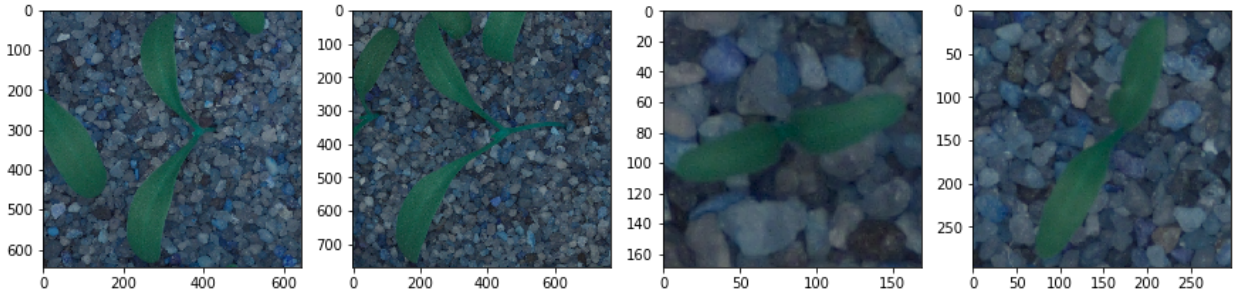



**Exploratory Visualization:**

The data is explored in the notebook named ‘data exploration.ipynb’. The class distribution of 4750 images is plotted below:




In the database, all images are not in same size. We will preprocess the images to solve this problem. Different sizes of images are shown in the below plot with axes.



**Algorithms and Techniques:**

We will use Convolutional Neural Network(CNN) to build our model. CNN are similar to regular neural networks: they are made up of neurons and have learnable weights and biases. The main difference is: the CNN takes advantage of image and accepts an image as a image with width, height and depth while the regular neural nets flattens the input. So, it is not good in identifying some features of an image while CNN easily can. The main building blocks of a CNN architecture are explained in the methodology section.

The classifier I built is a Convolutional Neural Network, which is a state-of-the-art algorithm for image classification problems. It needs a large amount of data. Fortunately, We were having enough data to train our model. This algorithm takes an image as an input, passes through a sequence of layers and returns the probability for each class. There are many Hyperparameters that can be tuned to adjust the performance of our algorithm. Some of them are:

- Number of epochs
- Batch size
- Learning Rate
- Solver (Algorithm used for learning)
- Number of layers in the architecture
- Activation function

**Benchmark:**

To create a benchmark score for the classifier, I used a Convolutional Neural Network architecture. There are some pairs of Conv2d and Maxpooling2d, Dropout layers followed by a GlobalAveragePooling2d and a dense layers. The detailed architecture can be seen in below sections. This model is trained for 20 epochs with RMSprop optimizer with default parameters and achieved an accuracy score of 0.5708. Now, we will try to make some improvements to the model that beats this score. There was also a Kaggle Competition held on this dataset. The benchmark score given by kaggle was 0.09823. I believe, the final solution I give will contribute towards solving this problem.

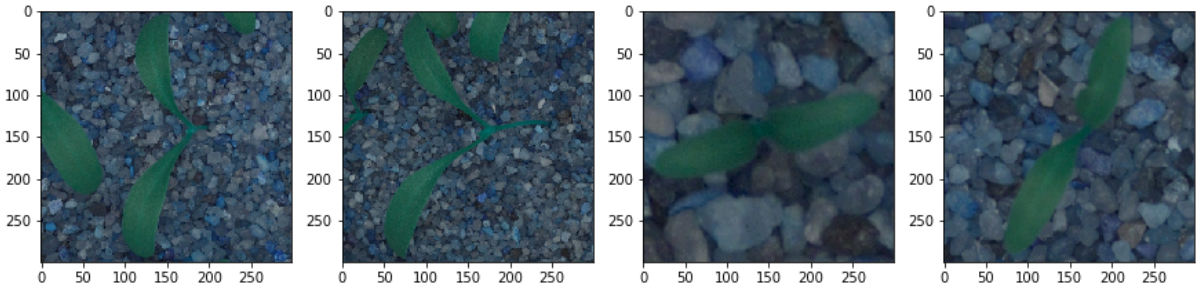
## III. Methodology

**Data Preprocessing:**

The images present in our database are not of same size, we apply some preprocessing techniques before using the data for training.

**Resizing Images:**

We have already seen the different sized images in the database. Now, we resize all the images to the same width and height (256*256). The images after the preprocessing are shown below with axes.



**Encoding targets and data split:**

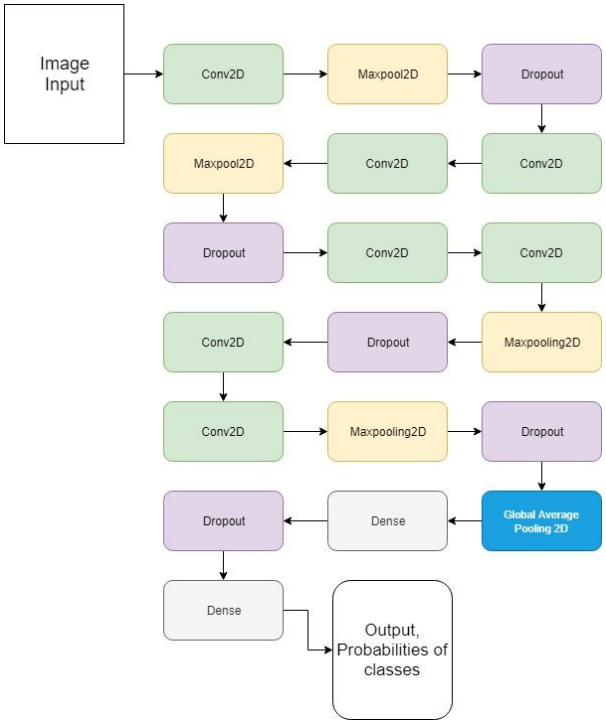
After preprocessing the images, I encoded the class names using to_categorical() from Keras utils. Later, the data is split into training and validation sets in the ratio of 85:15 using train_test_split() from scikit-learn.

**Implementation:**

The tasks required in implementing the project are:

- Loading the data
- Preprocessing the data;
- Defining the network architecture.
- Defining the loss function.
- Keep training the model until the desired accuracy is reached.
- Saving the model to the disk and use it for inferencing.
- Writing an algorithm that predicts plants class given an image input

The Convolutional Neural Network we developed has the following architecture:




Below, you will find explanation of different layers in our architecture.

* Input Layer: This a Conv2d Layer that accepts a (1*256*256*3) shaped image as an input. This layer has 16 filters and a 5*5 kernel size.

* Maxpool2D: Max-pooling layer is used to decrease the dimensions(only width and height) of the image by the pooling_size parameter. Foe ex, the input was of size 256*256*3 and if the pool size is 2 then, the output size would be 128*128*3. In all our Maxpool layers, the pool size is 2. So, after this layer, the width and height of the image are halved.

* Dropout: This layer is used as a regularization method to prevent overfitting. It drops the nodes at random when calculating forward and backward propagations. The dropout probability can be given as a parameter. In our architecture, I have given 0.1 probability to dropout nodes after several layers.

* Output Layer: This layer is a dense layer that outputs class probabilities for 12 classes. This layer has ‘softmax’ activation function.

The function plant_seed_classification() in the notebook takes a image as an input, preprocesses it , loads the model and predicts the class of the given image.

**Refinement:**

As mentioned in benchmark section, the CNN architecture we defined there achieved an accuracy of 57%. Now, the architecture we built has the following refinements:

- Dynamic Learning rate: Whenever the loss function stops decreasing, the learning rate was reduced.
- Data Augmentation: The training data was augmented using ImagedataGenerator in Keras to improve accuracy.

After all these refinements to the new architecture, the model was trained for more 30 epochs and achieved an accuracy score of **93.6%.**

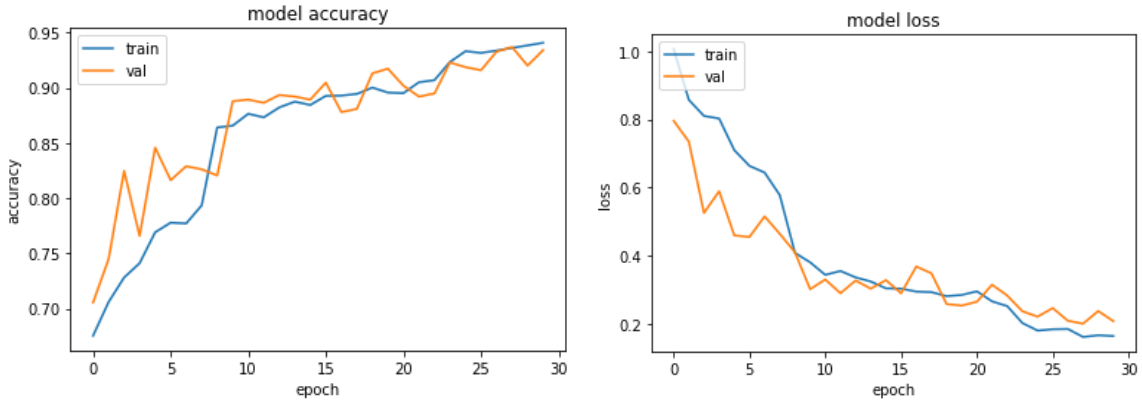
## IV. Results

**Model Evaluation and Validation:**

During the development of the model, a training set is used to train and a validation set is used to validate the model. The model we built might be sensitive to outliers because, the images that are present in the data set have intensities in range of 0-150 only. If a new image apart from these intensities is given for prediction, it may perform poorly.

- There are total 3,52,652 trainable parameters in our model.
- The model is trained for a total 50 epochs (20 without improvements and 30 after some improvements) with a batch size of 32
- I have used model checkpointer to save the model weights to the disk whenever the validation loss is low.

After training the model, the model gave an accuracy score of 0.936 with a loss of 0.2. The model did a good job in identifying the weed plants (class 3) with f1-score of 0.98. The model may give more good score because the curves are still converging. But, it already took around 6 hours to train. So, I stopped for 50 epochs in total. The training curves of accuracy and loss are shown below:




Overall, we can trust this model because it gave good f1-score on our goal, identifying weeds from others. Still, some improvements are to be made to the model to be able to deploy it for production use.

**Justification:**

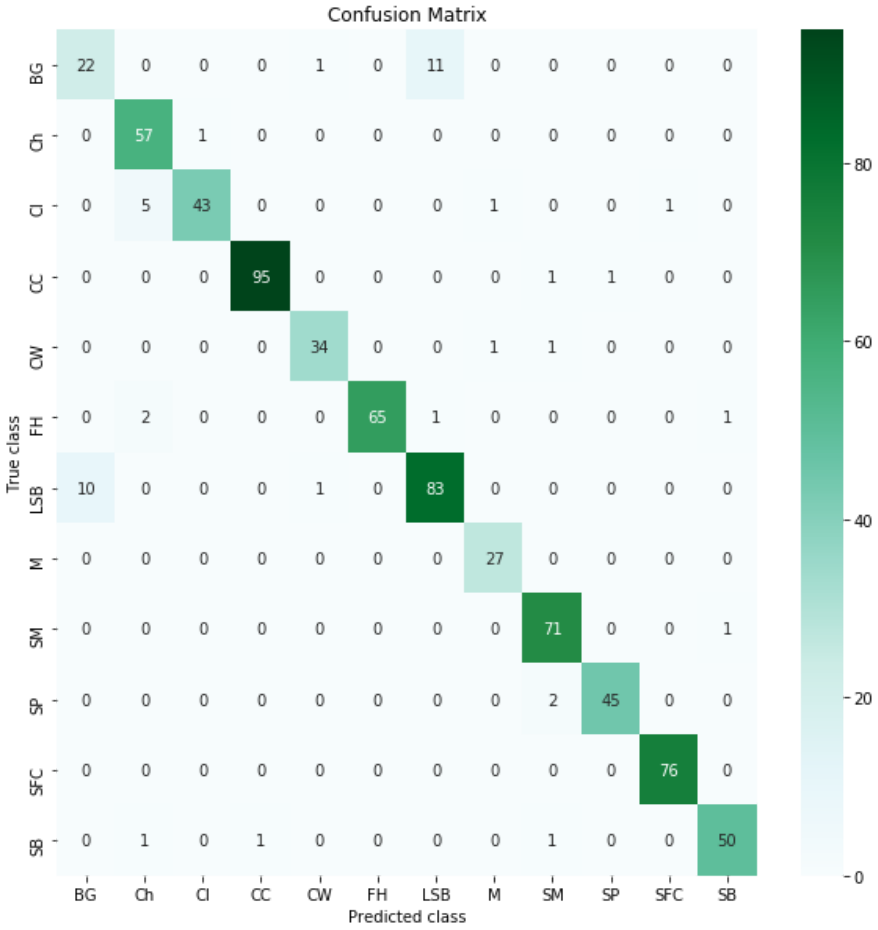
The results of our classifier are sufficient enough for the problem we are solving. Mainly, the classifier we built gave a very good accuracy in identifying weed plants, which was the main goal of our project. So, I strongly believe this model can be used as an application in detecting weed plants when they are small and significantly boost farmers productivity.

| Benchmark Score | Our Model Score | % of improvement |
|---|---|---|
| 0.578 | 0.936 | 61.9 |

Our model Shown a great improvement, almost 62% then the benchmark model.

# V. Conclusion

The model we built was with the aim of identifying weed plants in initial stages. Let us see the confusion matrix and analyze the performance.




- Model did find hard time in identifying Black-Grass from Loose Silky Bent.
- Model did very good job in detecting common chickweed, Loose Silky Bent, and identified Small-Flowered Cranesbill very accurately with f1-score of 0.99.

**Reflection:**

The process used for this project can be summarized as below:

- Problem set identified and relevant datasets are found online
- Data was downloaded and preprocessed
- Benchmark score was obtained
- Built a model with some improvements and trained it for several iterations.
- The best model is obtained and saved to disk
- This model was used for building an algorithm that identifies class based on image input/

Among all these steps, Preprocessing the data was one of the main challenges to me. I was not familiar with image processing techniques before this project. But, after completing this project, I am confident enough to say that I can solve an image classification problem and make results out of it.

**Improvement:**

As the further improvement to our model, we can build a model by over or undersampling the data because of the irregular count in images and build a more generalized model. We can also improve the accuracy by doing some preprocessing techniques such as image segmentation, masking etc. Finally, the model we built can be used for real-time identification of weed plants by the farmers if a mobile application is built on this model.

**References:**

[1] T. M. Giselsson, M. Dyrmann, R. N. J⌀rgensen, P. K. Jensen, and H. S. Midtiby, "A Public Image Database for Benchmark of Plant Seedling Classification Algorithms," arXiv preprint, 2017. 5

[2] "Classification Report - Sklearn." [Online]. Available: learn.org/stable/ modules/generated/sklearn.metrics.classification_report.html

[3] "Conv2D- keras." [Online]. Available: http://keras.io/layers/convolutional/

[4] "MaxPooling2D- Keras." [Online]. Available: http://keras.io/layers/ pooling/

[5] "GlobalAveragePooling2D- Keras." [Online]. Available: http://keras.io/ layers/pooling/