Machine Learning Engineer Nanodegree

Capstone Project

Charles Brands
July 10th, 2018

I. Definition

Project Overview

Farmers have been spraying their fields with herbicides for decades to reduce weeds. So far one herbicide mixture was used to spray an entire field. This is not a very effective approach. Massive usage of herbicides are expensive, bad for the environment and may lead to resistance in the weeds. Also a herbicide mixture that is effective against one species of weed can have little or no effect on another species. Not to mention that the herbicide may be damaging to the crop it is supposed to protect. A better approach would be a system that can recognise weeds from crops and only spray at places where it is needed.

Aarhus University in Denmark is working on a project to create weed maps of a field from a tractor to pinpoint where a certain species of weed resides on a field. They have released a dataset of images of 960 unique plants belonging to 12 species at several growth stages.

Kaggle is hosting this dataset as a Kaggle competition. in order to give it wider exposure and to give the community an opportunity to experiment with different image recognition techniques.

Problem Statement

For this project we have to detect which species of plant is in each image of the dataset described above. Each image contains one seedling which has to be classified into one of twelve species.
The solution to this problem is a model trained to predict the plant species for a given image.

- Input: An image of a plant.
- Output: The name of the plant species on the image.

Steps taken to complete this task

1. Imported the required data sets.
2. Displayed some sample images for each plant to get a viual idea of the problem.
3. Separate the data set into training, validation, and testing sets.
4. Do some preprocessing converting to tensors and normalization and so on.
5. Augmenting and oversampling the dataset by using horizontal flipping, rotation, zoom and so on.
6. Create a few models using the transfer learning approach with pretrained network as wel as a neural network started from scratch.
7. Fine tune the model(s) using different optimizers and adjusting multiple parameters.
8. The F1-score will be calculated as described in the section "Evaluation metrics" above.
9. Finally my score will be compared to the leaderboard on Kaggle.

The desired outcome of this project is to show that it is indeed possible to automattically detect weeds so the spraying of herbicides can be applied as sparingly and as effectively as possible.

Metrics

I used the mean multi-class F1 score as the metric to evaluate my solution.
See scikit-learn.org

> The F1 score can be interpreted as a weighted average of the precision and recall, where an F1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the F1 score are equal. The formula for the F1 score is:

$$F_1 = \frac{2 * precision * recall}{precision + recall}$$

> In the multi-class and multi-label case, this is the weighted average of the F1 score of each class.

Precision is the ability of the classifier not to label as positive a sample that is negative. In formula:

$$precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

Recall is the ability of the classifier to find all the positive samples. In formula:

$$recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

The F1 score is also used by Kaggle as the metric to evaluate the solutions for this problem.link

Loss Function: Categorical Cross Entropy

Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. The lower the value the better the predicted probability matches the label. A perfect model would have a log loss of 0.
The formula for Categorical Cross Entropy is:

$$\sum_{c=1}^{M} y_{o,c} ln(p_{o,c})$$

where …

- M - number of classes (dog, cat, fish)
- ln - the natural log
- y - binary indicator (0 or 1) if class label c is the correct classification for observation o
- p - predicted probability observation o is of class c

More information can be found here and here.

II. Analysis

(approx. 2-4 pages)

Data Exploration

The train and test datasets can be found on kaggle. There are 4751 images for training and 794 images for testing. Each image has a filename that is its unique id. The dataset comprises 12 plant species. The list of species is as follows:

Danish	English	Latin	EPPO code
Majs	Maize	Zea mays L.	ZEAMX
Vinterhvede	Common wheat	Triticum aestivum L.	TRZAX
Sukkerroe	Sugar beet	Beta vulgaris var. altissima	BEAVA
Lugtløs kamille	Scentless Mayweed	Matricaria perforata Mérat	MATIN
Fuglegræs	Common Chickweed	Stellaria media	STEME
Hyrdetaske	Shepherd's Purse	Capsella bursa-pastoris	CAPBP
Burresnerre	Cleavers	Galium aparine L.	GALAP
Agersennep	Charlock	Sinapis arvensis L.	SINAR
Hvidmelet gåsefod	Fat Hen	Chenopodium album L.	CHEAL
Liden storkenæb	Small-flowered Cranesbill	Geranium pusillum	GERSS
Agerrævehale	Black-grass	Alopecurus myosuroides	ALOMY
Vindaks	Loose Silky-bent	Apera spica-venti	APESV

EPPO codes are computer codes developed for plants, pests (including pathogens) which are important in agriculture and plant protection.			

The images are of differend sizes. The smallest 49 X 49 the largest 3991 X 3457. In order to use a Convolutional Neural Network, images are rescaled to one size (224 px by 224 px).

Below is an image of each species.

black grass	charlock	chickweed	cleavers
			

common wheat	cranesbill	fat hen	loose silky bent
			

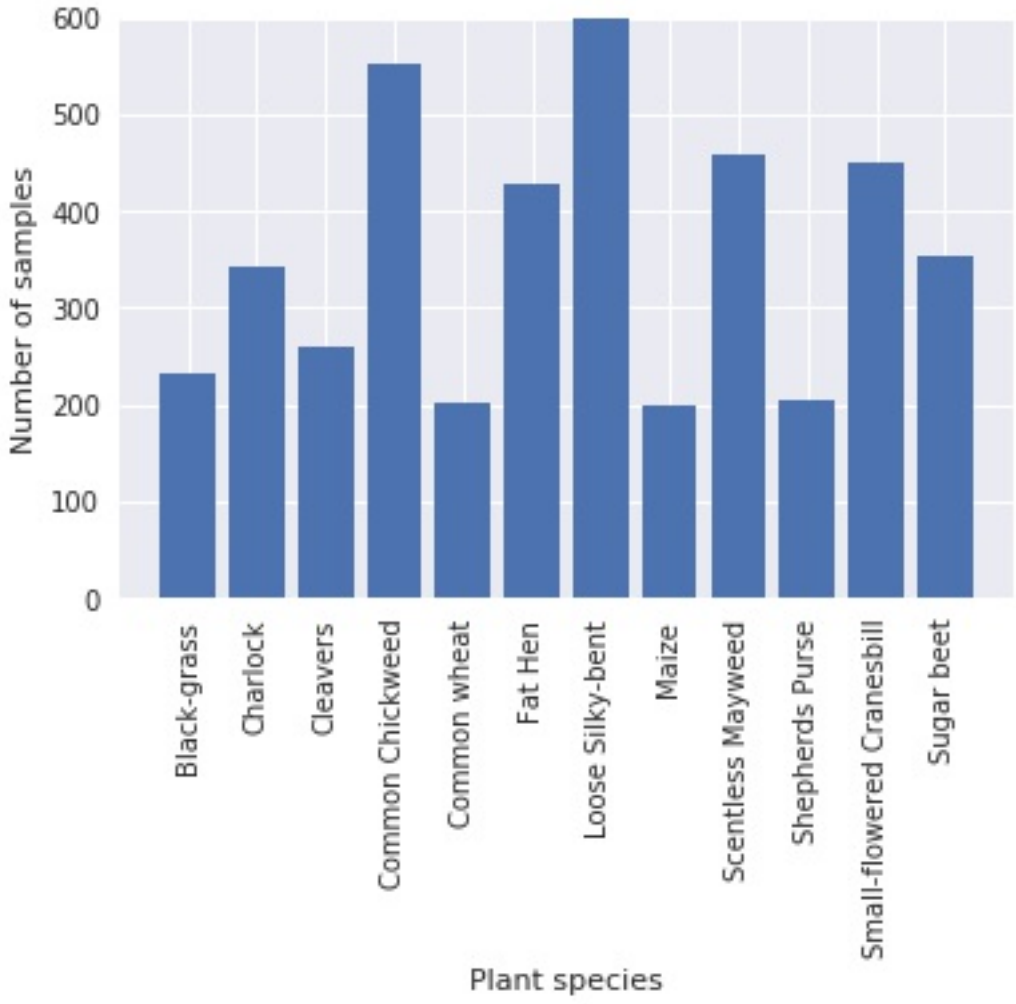
maize	scentless mayweed	shepherds purse	sugar beet
			

This is not a trivial problem Some species are very hard to to tell apart. Even for a human being!

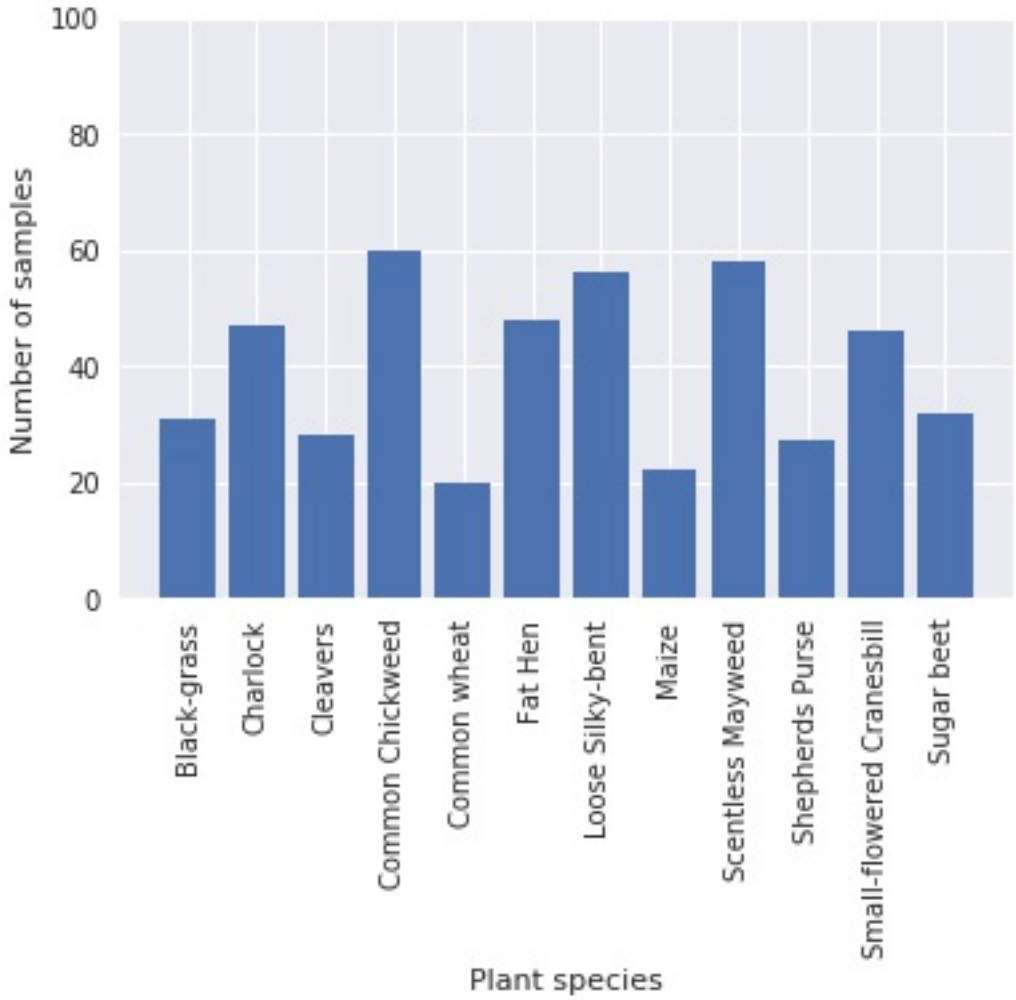
Exploratory Visualization

After loading the data in a jupyter notebook I split the data into a training set and a test set. Below are the distribution plots of these sets.

Traning set

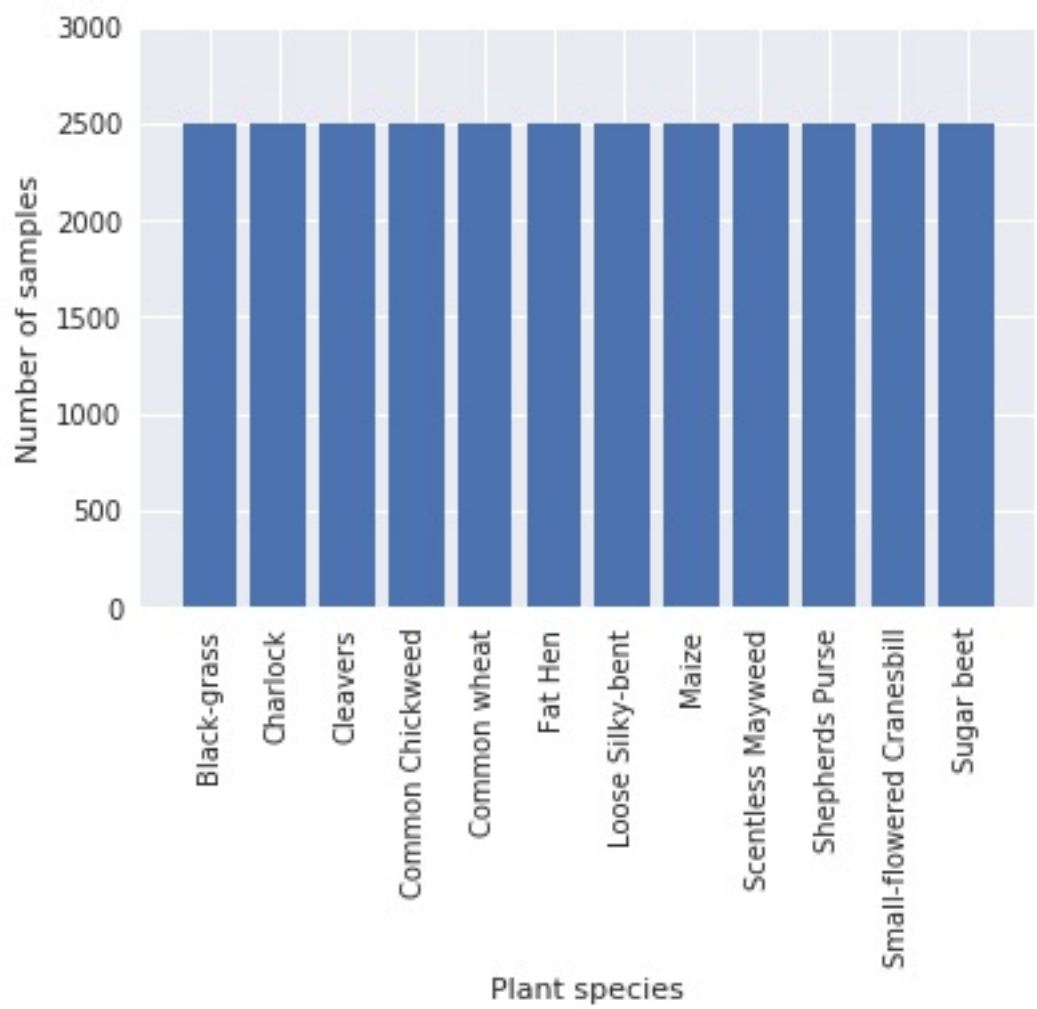


Testing set



There is a large mismatch in the set. For some species there are only 200 samples in the set and for others 600. A factor of 3. Also the complete data set could be larger. So I augmented the data set using techniques shown here and here.

This created a an even distribution with a lot more data as shown below.



Algorithms and Techniques

I created Convolutional Neural Networks to classify the images using the following approaches:
I used three approaches approaches to create a Convolutional Neural Network to classify images:

- Building a CNN from scratch.
- Transfer Learning on the VGG16 network.
- Transfer Learning on the VGG19 network.
- Transfer Learning on the ResNet50 network.

CNN from scratch

I started with the neural network I used in the dog classification project. However the results were not great. So after a lot of experimentation I ended up with the network below.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_1 (Conv2D)            (None, 223, 223, 16)      208
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 111, 111, 16)      0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 110, 110, 32)      2080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 55, 55, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 54, 54, 64)        8256
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 27, 27, 64)        0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 26, 26, 128)       32896
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 13, 13, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 12, 12, 256)       131328
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 6, 6, 256)         0
_________________________________________________________________
global_average_pooling2d_1 ( (None, 256)               0
_________________________________________________________________
dense_1 (Dense)              (None, 12)                3084
=================================================================
Total params: 177,852
Trainable params: 177,852
Non-trainable params: 0
```

I also tried drop out layers but they had a negative impact on the result.

Transfer Learning

For the transfer learning experiments (VGG16, VGG19, and ResNet50) I followed the approach I learned in the CNN section of the Udacity course.

- I removed the toplayers, set the lower layers to nontrainable to keep features like edges and shapes.
- Added a new fully connected layer with 12 nodes. The 12 is the number of classes in the dataset.
- Train the network to update the weights of the new fully connected laye

Benchmark

The results from my models were benchmarked against other submissions in the Kaggle Leaderboard.
Also some Kaggle visitors are hosting Kaggle kernels after completing this project I used their

code as a benchmark.

III. Methodology

(approx. 3-5 pages)

Data Preprocessing

In this section, all of your preprocessing steps will need to be clearly documented, if any were necessary. From the previous section, any of the abnormalities or characteristics that you identified about the dataset will be addressed and corrected here. Questions to ask yourself when writing this section:

- *If the algorithms chosen require preprocessing steps like feature selection or feature transformations, have they been properly documented?*
- *Based on the **Data Exploration** section, if there were abnormalities or characteristics that needed to be addressed, have they been properly corrected?*
- *If no preprocessing is needed, has it been made clear why?*

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

- *Is it made clear how the algorithms and techniques were implemented with the given datasets or input data?*
- *Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution?*
- *Was there any part of the coding process (e.g., writing complicated functions) that should be documented?*

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant

intermediate results as necessary. Questions to ask yourself when writing this section:

- *Has an initial solution been found and clearly reported?*
- *Is the process of improvement clearly documented, such as what techniques were used?*
- *Are intermediate and final solutions clearly reported as the process is improved?*

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

- *Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate?*
- *Has the final model been tested with various inputs to evaluate whether the model generalizes well to unseen data?*
- *Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results?*
- *Can results found from the model be trusted?*

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

- *Are the final results found stronger than the benchmark result reported earlier?*
- *Have you thoroughly analyzed and discussed the final solution?*
- *Is the final solution significant enough to have solved the problem?*

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

- *Have you visualized a relevant or important quality about the problem, dataset, input data, or results?*
- *Is the visualization thoroughly analyzed and discussed?*
- *If a plot is provided, are the axes, title, and datum clearly defined?*

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

- *Have you thoroughly summarized the entire process you used for this project?*
- *Were there any interesting aspects of the project?*
- *Were there any difficult aspects of the project?*
- *Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?*

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

- *Are there further improvements that could be made on the algorithms or techniques you used in this project?*
- *Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how?*
- *If you used your final solution as the new benchmark, do you think an even better solution*

exists?

Before submitting, ask yourself. . .

- Does the project report you've written follow a well-organized structure similar to that of the project template?
- Is each section (particularly **Analysis** and **Methodology**) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification?
- Would the intended audience of your project be able to understand your analysis, methods, and results?
- Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes?
- Are all the resources used for this project correctly cited and referenced?
- Is the code that implements your solution easily readable and properly commented?
- Does the code execute without error and produce results similar to those reported?

References

- Kaggle seedlings evaluation
- scikit-learn.org f1 score