Computer Vision: Seedling Classification

PGP AI/ML for Business Applications

Nicholas Lutostanski

5.1.2024

Contents / Agenda

- Executive Summary
- Business Problem Overview and Solution Approach
- EDA Results
- Data Preprocessing
- Model Performance Summary
- Conclusion
- Appendix

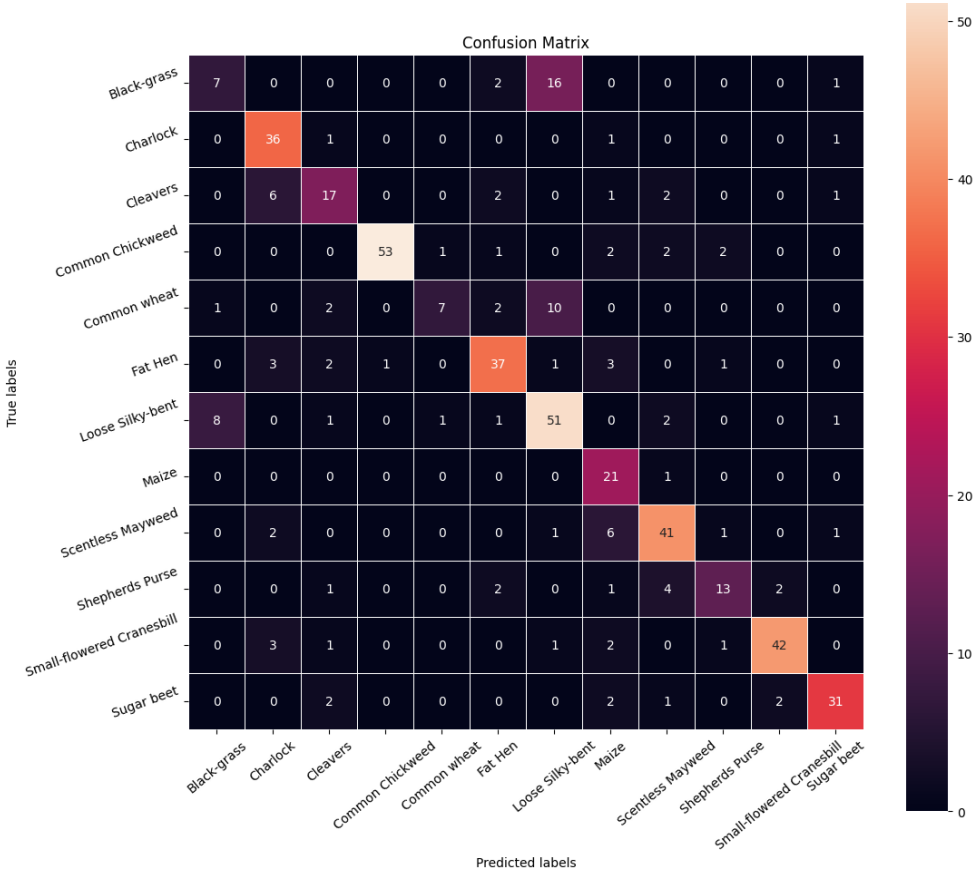
Executive Summary

We chose the second model that attempted to balance out the category samples using data augmentation and reduced the learning rate on plateau using ReduceLROnPlateau. This model also used batch normalization, spatial dropout to reduce computational power (model 2 epochs were executed in about half the time as model 1)

By doing these things, we were able to get our model to correctly identify black-grass, where the previous model always misclassified it as loose silky-bent.

The accuracy between the original model and the improved model increased by 4%.

Further fine tuning with the dropout rate, image rotation degrees used in data augmentation, and playing with the learning rate might be necessary to squeeze out a little more accuracy.



Data Dictionary

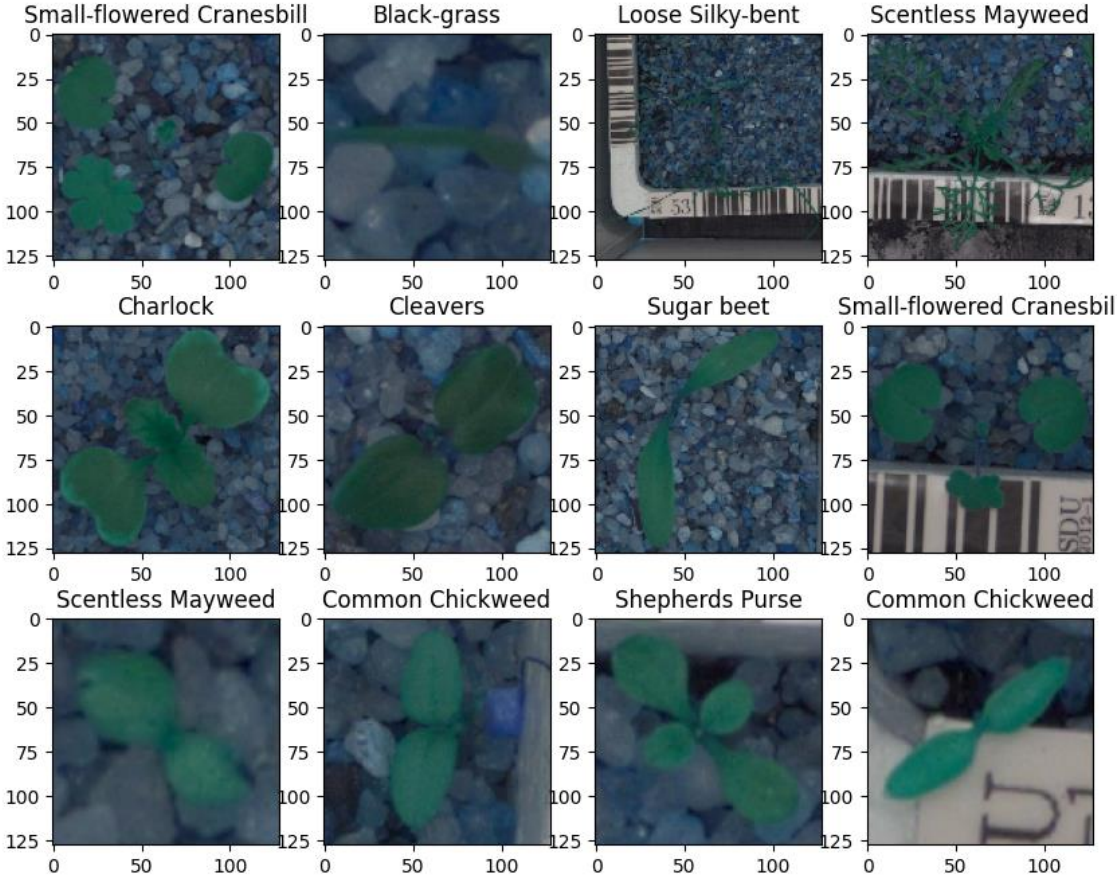
The Aarhus University Signal Processing group, in collaboration with the University of Southern Denmark, has recently released a dataset containing images of unique plants belonging to 12 different species below:

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

Business Problem Overview and Solution Approach

- The potential is ripe for this trillion-dollar industry to be greatly impacted by technological innovations that cut down on the requirement for manual labor, and this is where Artificial Intelligence can benefit the workers in this field, as the time and energy required to identify plant seedlings will be greatly shortened by the use of AI and Deep Learning. This project aims to build a Convolutional Neural Network to classify plant seedlings into their respective categories.
- We are going to build two CNN models – one with the raw data and a second one that we will tune with several different techniques – We will use data augmentation, Batch Normalization, Spacial Dropout, and we will reduce the learning rate with the ReduceLROnPlateau().

EDA Results – Image Samples



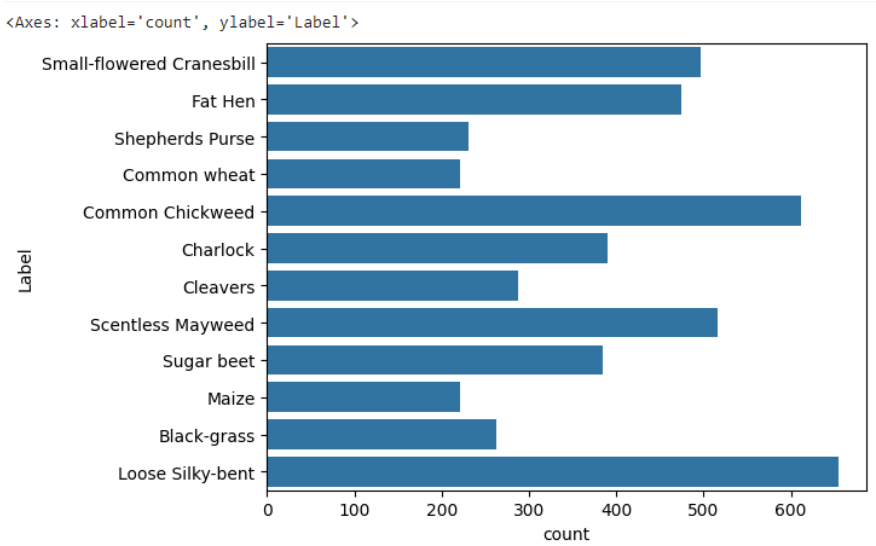
Here we have printed 12 random sample images. These are the original images in 128x128 pixels.

During data preprocessing, to reduce computational load of the Convolutional Neural Network, we will reduce this to 64x64 pixels.

EDA Results – Balance

- After loading the dataset and image files, we check the balance. In practice, it is best to use a somewhat-balanced data set where each plant species is represented about the same for each. Otherwise, you risk overfitting the data to certain over-represented species which will cause your model to misidentify plant-types or species.

- One way to solve this is to use data augmentation to manipulate the images to create artificially “new” images to use to balance the data. We will create a CNN using the data as is and see what impact it does to the accuracy.



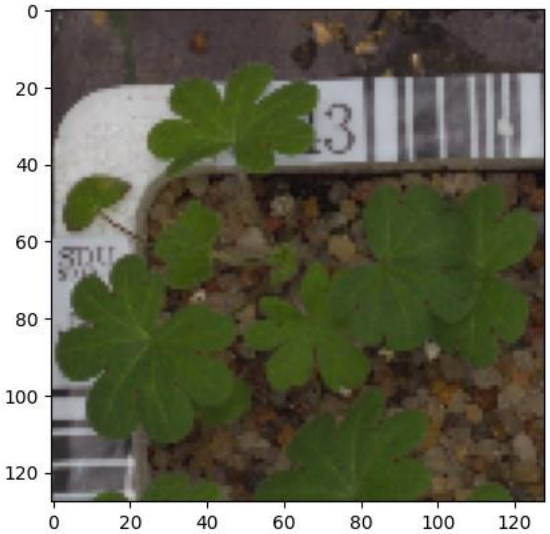
Data Preprocessing

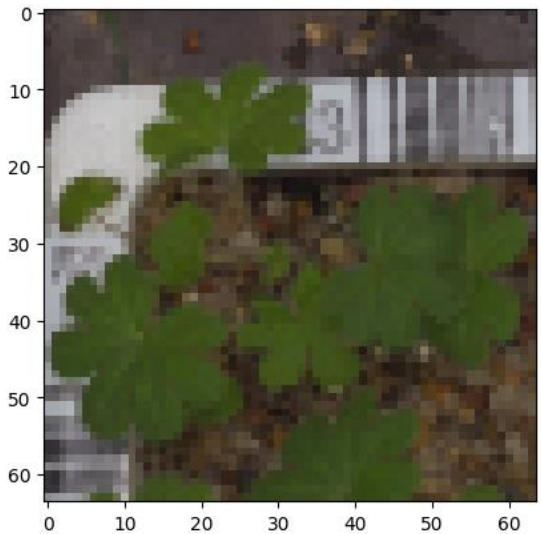
For our task, we will pre-process the data in the following ways:

- Convert BGR images to RGB images –
- We will shrink the resolution of the images from 128 x 128 to 64 x 64 to reduce the computational load for deep learning
- Train-test-split will utilize 80% of the data for model training, 10% for validation, and 10% will be reserved to test the model on unseen data.
- We will use LabelBinarizer() for hot-encoding the target variables
- Finally, we are going to normalize the pixel color range to reduce computational complexity for deep learning. We will scale all values from 0-255 to be a ratio between 0 and 1.

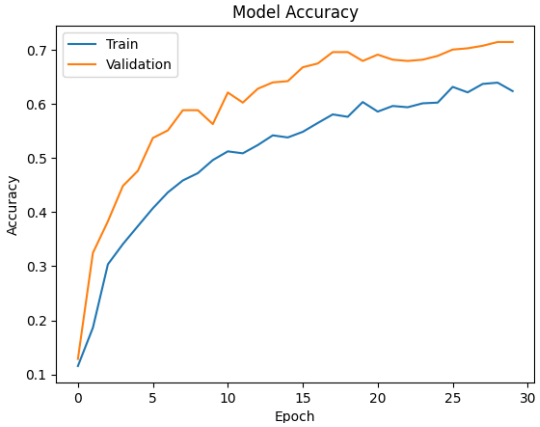
Data Preprocessing – Image Resizing

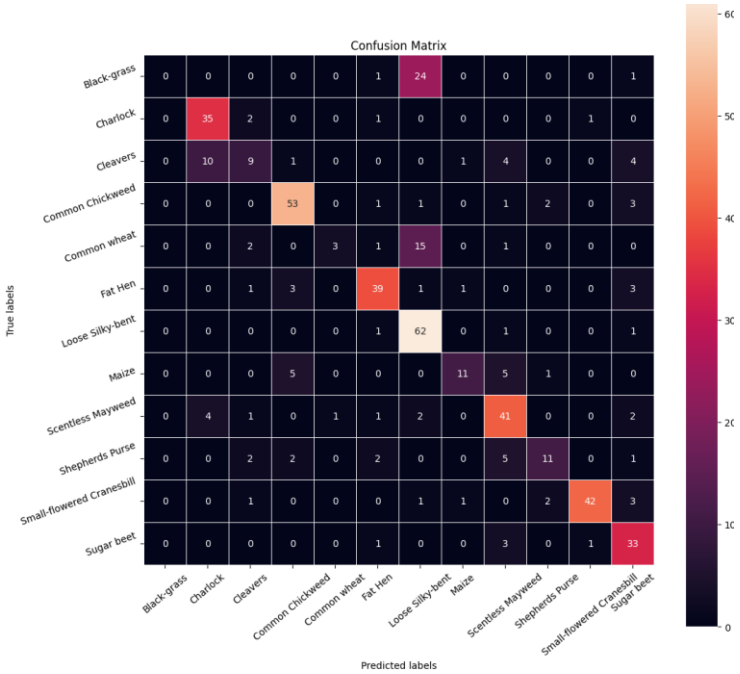
We are reducing the size from 128x128 pixels to 64 x 64 pixels to reduce the computational complexity of the model so that it runs faster. You can see the original image on the left side and the reduced sized image on the right side.





Model Performance Summary – Model 1





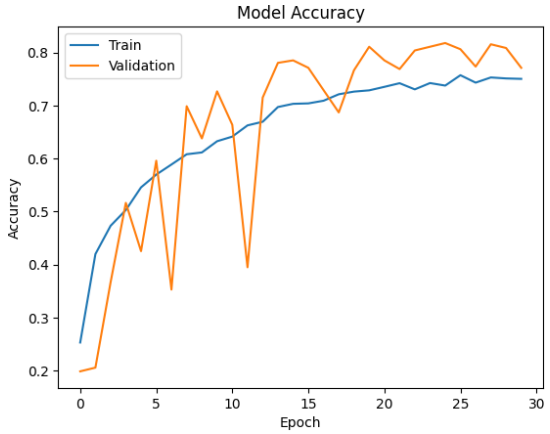
	precision	recall	f1-score	support
0	0.00	0.00	0.00	26
1	0.71	0.90	0.80	39
2	0.50	0.31	0.38	29
3	0.83	0.87	0.85	61
4	0.75	0.14	0.23	22
5	0.81	0.81	0.81	48
6	0.58	0.95	0.73	65
7	0.79	0.50	0.61	22
8	0.67	0.79	0.73	52
9	0.69	0.48	0.56	23
10	0.95	0.84	0.89	50
11	0.65	0.87	0.74	38
accuracy			0.71	475
macro avg	0.66	0.62	0.61	475
weighted avg	0.69	0.71	0.68	475

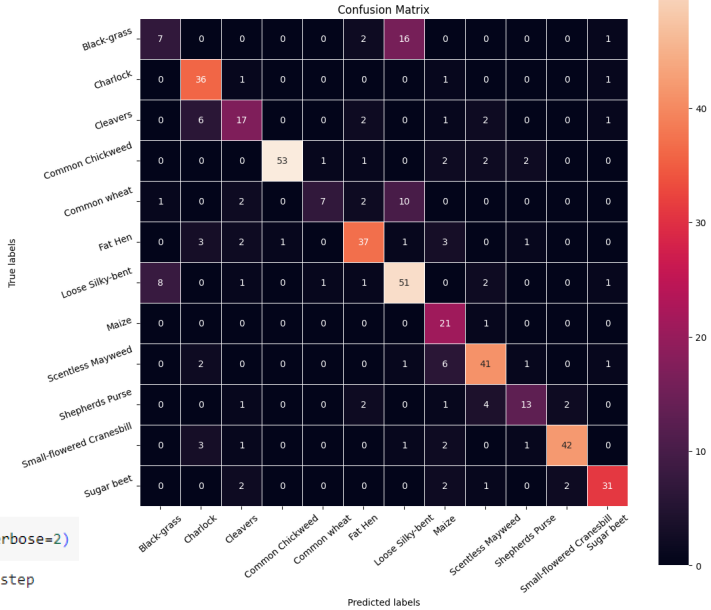
```
accuracy = model1.evaluate(X_test_normalized, y_test_encoded, verbose=2)
```

```
15/15 - 3s - loss: 0.9227 - accuracy: 0.7137 - 3s/epoch - 171ms/step
```

The first model produced an accuracy of 0.71, which is decent and it performs similarly on the test data as it did the validation data. Looking at the confusion table, however, shows that there were several types of grass that were mischaracterized. Ideally, all of the “heat’ in the map should be on the diagonals where true and predicted are equal. In this case, black-grass was mischaracterized most with Loose Silky-bent (and it predicted 0 black-grass correctly). Common wheat was also mischaracterized as Loose Silky-bent. Loose Silky-bent has the most samples in our dataset and both black-grass and common wheat are among the bottom 4 sample images in the dataset. Data augmentation might be useful for helping to identify these correctly by creating more “images” we will rotate them 20 degrees

Model Performance Summary – Model 2





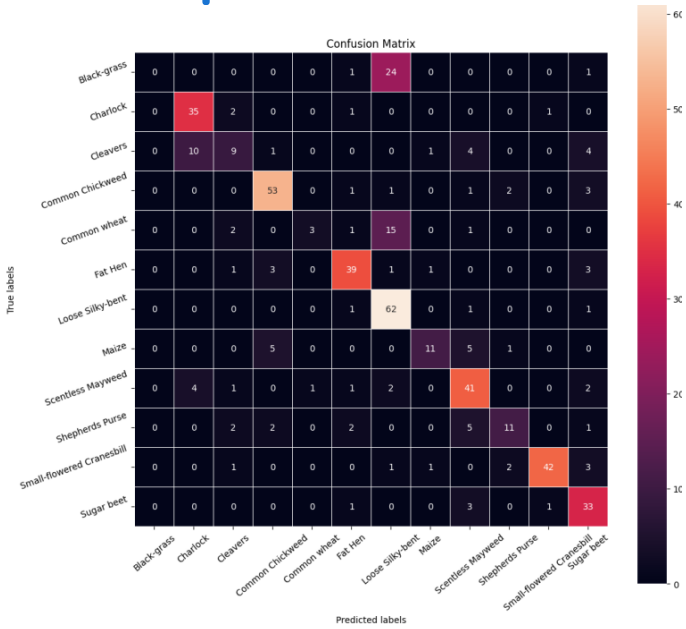

```
accuracy = model2.evaluate(X_test_normalized, y_test_encoded, verbose=2)
```

```
15/15 - 2s - loss: 0.8453 - accuracy: 0.7495 - 2s/epoch - 114ms/step
```

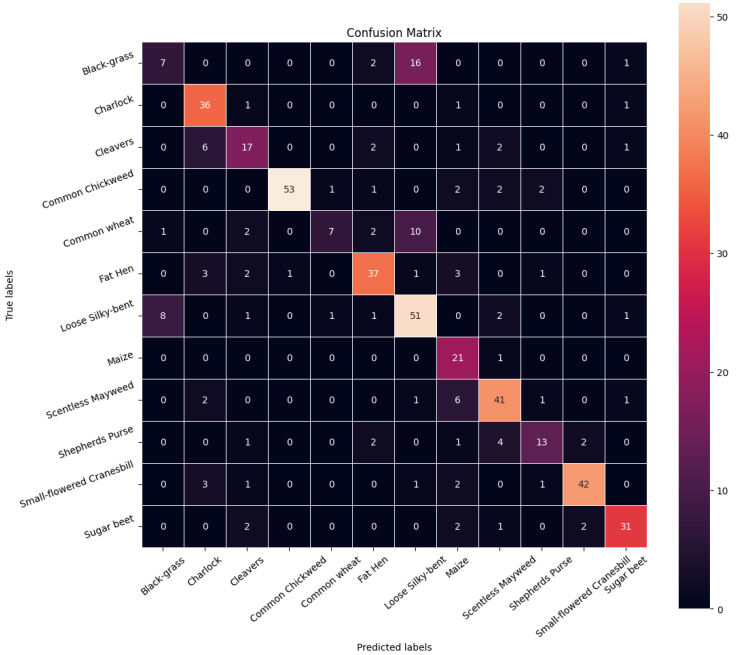
	precision	recall	f1-score	support
0	0.44	0.27	0.33	26
1	0.72	0.92	0.81	39
2	0.63	0.59	0.61	29
3	0.98	0.87	0.92	61
4	0.78	0.32	0.45	22
5	0.79	0.77	0.78	48
6	0.64	0.78	0.70	65
7	0.54	0.95	0.69	22
8	0.77	0.79	0.78	52
9	0.72	0.57	0.63	23
10	0.91	0.84	0.87	50
11	0.86	0.82	0.84	38
accuracy			0.75	475
macro avg	0.73	0.71	0.70	475
weighted avg	0.76	0.75	0.74	475

Using data augmentation to help balance out the number of samples for each type of grass and reducing the learning rate, our results are slightly better. Accuracy has increased from 0.71 to 0.75 and performs well on the unseen test data. You can notice that the model was better at identifying black grass (the previous model identified 0 correctly, this one identified 7 correctly) and common wheat. So what we have done is reduced the overfitting of the model to some plant types which caused them to be mischaracterized. Note: for the data augmentation, we chose a 20 degree rotation to create the artificially “new” images. We also used spatial dropout and batch normalization in this model.

Model Comparison



Model 1



Model 2

Comparing the two confusion matrices, you can see that in model two (with data augmentation), the diagonal of the second model has more “heat” to it, indicating that the model is better at identifying plants correctly, although further tweaking might be necessary to improve it to be even better. Of interest is that using data augmentation, more loose-silk-bent were mischaracterized as black grass than occurred in the first model, even though our overall model is more accurate.

Conclusion

We chose the second model, which was improved from the original using data augmentation and lowered learning rate on plateau, spatial dropout, and batch normalization. The results improved the accuracy of the model by 4% and the model was able to differentiate black-grass from Loose silk-bent, where the original model failed 100% to correctly identify black-grass.

Further manipulation might be required to fine-tune the model and increase accuracy. In using spatial dropout, reducing the learning rate, data augmentation, and batch normalizations, we might play around with the parameters a bit, rerun the models, and see what kind of improvements we can see.

APPENDIX

Data Background and Contents

- Please mention about the data background and contents

Great Learning

Happy Learning !