Introduction to Computer Visions

Prj 8: Plant Seedling Classification

PGP-AIML-BA-UTA-Mar23

Oct 7, 2023

Brent Jones

Contents

1. **Executive Summary / Problem Statement**
2. **Define the problem statement -** Read the dataset properly - Check the shape of the data
3. **Perform an Exploratory Data Analysis and get the insights on the images** - Plot random images from each of the classes and print their corresponding labels. - Count Plot for each category - Key meaningful observations from EDA
4. **Data preprocessing**- Convert the BGR images to RGB images - Resize the images - Plot the images before and after the pre-processing steps - Split the data into train and test - Encode the target variables - Apply the normalization
5. **Model Building** - Build Convolution Neural Network - Evaluate the model on different performance metrics and comment on the performance. For example precision, recall, accuracy. - Plot confusion matrix
6. **Model Performance Improvement and Final Model Selection** - Build another model using data augmentation to overcome the imbalance problem. - Evaluate the model on different performance metrics and comment on the performance. - Plot confusion matrix - Choose the best model from the ones built with proper reasoning.
7. **Actionable Insights & Recommendations** - Conclude with the key takeaways for the business - What would be your recommendations to the business?

1. Executive Summary / Problem Statement

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*. The ability to do so far more efficiently and even more effectively than experienced manual labor could lead to better crop yields, the freeing up of human involvement for higher-order agricultural decision making, and in the long term will result in more sustainable environmental practices in agriculture as well.

Objective: Create a classifier capable of determining a plant's species from an image.

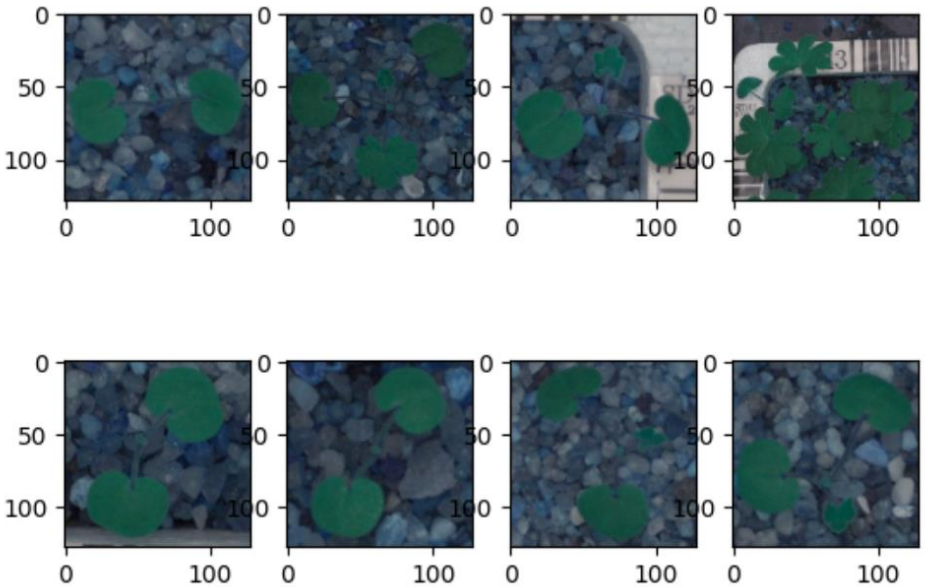
The Aarhus University Signal Processing group, in collaboration with the University of Southern Denmark, has provided the data containing images of unique plants belonging to 12 different species. You being a data scientist, need to build a Convolutional Neural Network model which would classify the plant seedlings into their respective 12 categories.

Data Dictionary:

- Common Wheat
- Black-grass
- Charlock
- Cleavers
- Common Chickweed
- Shepards Purse
- Sugarbeet
- Fat Hen
- Loose Silky Bent
- Maize
- Scentless Mayweed
- Small flower Cranesbill

2. Define the problem - Read the dataset properly - Check the shape of the data

We have 4750 samples (images). Each image has a resolution of 128x128 pixels. Each image has 3 channels, which indicates that these are RGB (colored) images. If it were a grayscale image, you'd expect a single channel. Each image has one corresponding label. The three channels will typically pass through three separate filters initially or will be handled by multi-channel filters.



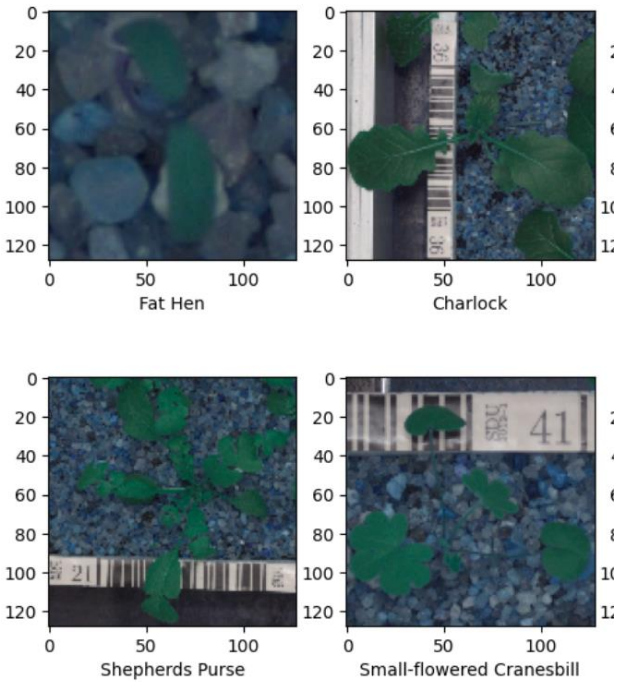
```
array([[[[ 35,  52,  78],
         [ 36,  49,  76],
         [ 31,  45,  69],
         ...,
         [ 78,  95, 114],
         [ 76,  93, 110],
         [ 80,  95, 109]],

        [[ 33,  46,  68],
         [ 37,  50,  73],
         [ 48,  65,  83],
         ...,
         [ 81,  96, 113],
         [ 74,  89, 105],
         [ 83,  95, 109]],

        [[ 34,  50,  68],
         [ 35,  52,  72],
         [ 70,  85, 101],
         ...,
         [ 83,  97, 112],
         [ 79,  94, 108],
         [ 79,  94, 107]],
```

3. Perform EDA and get insight - Plot random images from each of the classes and print their corresponding labels. - Count Plot for each category - Key meaningful observations from EDA

Observations

•Each data point is a matrix representing an image. The innermost lists have three values, corresponding to the Red, Green, and Blue channels of the image, respectively.
•The values are in the range of 0-255, which is the typical range for pixel intensity in images.
•The **dtype=uint8** further confirms that the values are 8-bit unsigned integers, which is standard for image data.
•The visuals on the right represent the RGB Distribution
•Number of unique images: 4747
•Number of duplicates: 3
•When checking for under or over exposed images we got Counter({'Good': 4483, 'Underexposed': 267})
•The mean, median and standard deviation for each channel is
•Mean Channel Values (R, G, B): [52.74117098 73.66265106 83.72709033]
•Median Channel Values (R, G, B): [46. 71. 80.]
•Standard Deviation Channel Values (R, G, B): [32.55663286 28.19947915 26.76837361]
•There are more "Loose Silky bent" plant samples than other types.
•"Common Wheat" and "Maize" have the fewest samples.
•The dataset is uneven with varying image counts for each category.
•The loose silky-bent seedling has the most images at 654, with common chickweed next at 611.
•Shepherds purse, common wheat, cleavers, maize, and black grass have the fewest, with counts of 231, 221, 287, 221, and 263 respectively.

```
random_images(images,labels)
```



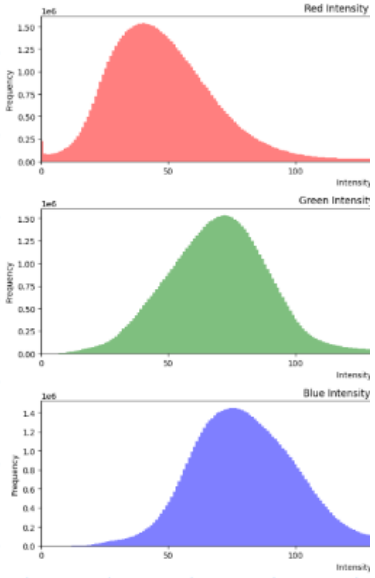
	Label
0	Small-flowered Cranesbill
1	Small-flowered Cranesbill
2	Small-flowered Cranesbill
3	Small-flowered Cranesbill
4	Small-flowered Cranesbill
...	...
4745	Loose Silky-bent
4746	Loose Silky-bent
4747	Loose Silky-bent
4748	Loose Silky-bent
4749	Loose Silky-bent

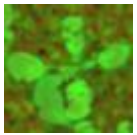
4750 rows × 1 columns

4 Data Preprocessing and Feature Engineering

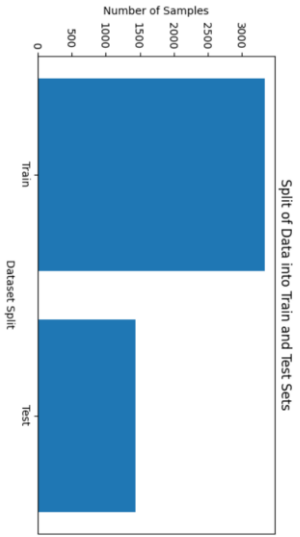
Convert the BGR images to RGB images - Resize the images - Plot the images before and after the pre-processing steps - Split the data into train and test - Encode the target variables - Apply the normalization

- Plotted the distribution of pixel intensities for each channel (R, G, B)
- Resized the images with before and after preprocessing
- Split the data into Test and Train
- Encoded the target variables and applied normalization






```
Shape of y_encoded: (4750, 2)
First 5 rows of y_encoded:
[[1 1]
 [1 1]
 [1 1]
 [1 1]
 [1 1]]
Row 10 of y_encoded:
[1 1]
```

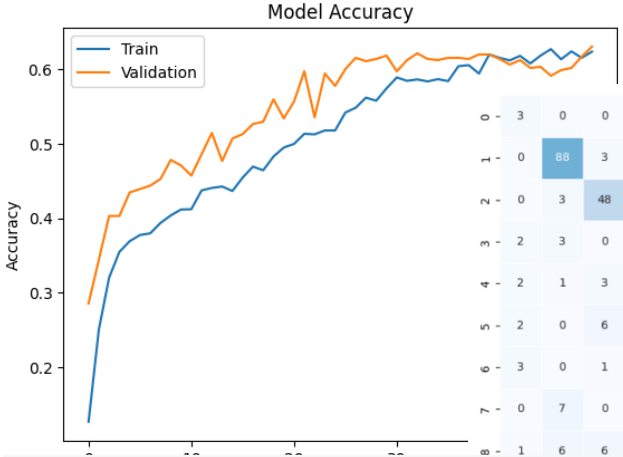


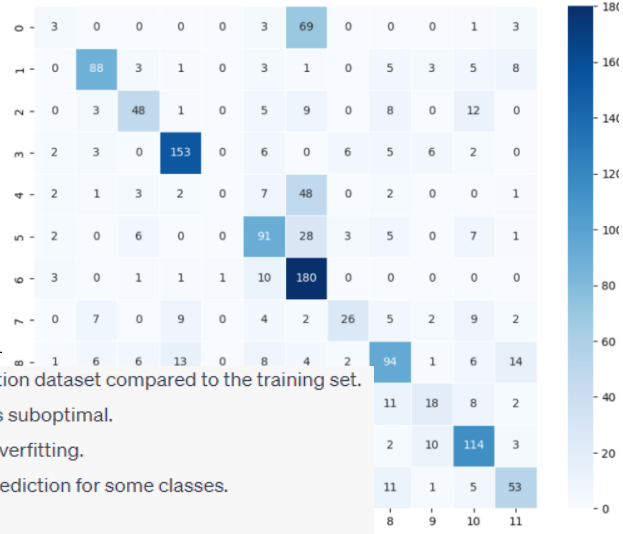

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 128)     3584

 max_pooling2d (MaxPooling2  (None, 64, 64, 128)       0
 D)

 conv2d_1 (Conv2D)           (None, 64, 64, 64)        73792

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 64)        0
 g2D)

 conv2d_2 (Conv2D)           (None, 32, 32, 32)        18464

 max_pooling2d_2 (MaxPoolin  (None, 16, 16, 32)        0
 g2D)

 flatten (Flatten)           (None, 8192)              0

 dense (Dense)               (None, 16)                131088

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 12)                204

=================================================================
Total params: 227132 (887.23 KB)
Trainable params: 227132 (887.23 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

5. Model Building

Build Convolution Neural Network - Evaluate the model on different performance metrics and comment on the performance. For example precision, recall, accuracy. - Plot confusion matrix

- Built a CNN, fit the model on the training and test data using 50 epochs
- Compared Model Accuracy and Validation: Train: accuracy: 0.8425, Test: accuracy: accuracy: 0.6091
- Built a confusion matrix, evaluate & suggest accuracy on unseen/test data is a mere 48%. To enhance performance, we plan to implement techniques such as data augmentation, Batch Normalization, Dropout, and early stopping. Additionally, we'll explore Transfer learning to determine if it can further boost the model's performance.
- Created another model with Batch Normalization, Early Stopping & Drop out layer (next slide)





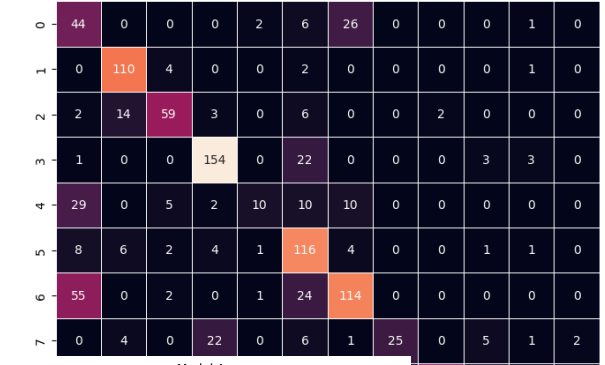
- The model displays higher accuracy on the validation dataset compared to the training set.
- However, its performance on the testing dataset is suboptimal.
- This discrepancy in performance might indicate overfitting.
- We notice that the model struggles with proper prediction for some classes.
- Classes 3, 5, 6, 8, and 10 are correctly classified.
- On the other hand, Classes 0, 1, 2, 4, 7, 9, and 11 exhibit significant misclassification, with each showing poor recall scores.

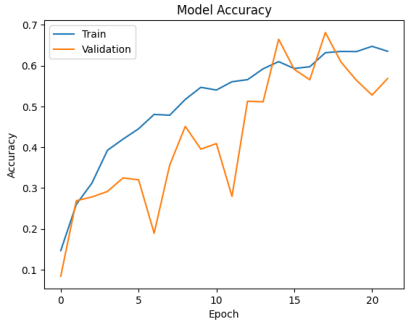
```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 128, 128, 64)      1792

 max_pooling2d (MaxPooling2  (None, 64, 64, 64)        0
 D)

 conv2d_1 (Conv2D)           (None, 64, 64, 32)        18464

 max_pooling2d_1 (MaxPoolin  (None, 32, 32, 32)        0
 g2D)

 batch_normalization (Batch  (None, 32, 32, 32)        128
 Normalization)

 flatten (Flatten)           (None, 32768)             0

 dense (Dense)               (None, 16)                524304

 dropout (Dropout)           (None, 16)                0

 dense_1 (Dense)             (None, 12)                204

=================================================================
Total params: 544892 (2.08 MB)
Trainable params: 544828 (2.08 MB)
Non-trainable params: 64 (256.00 Byte)
_________________________________________________________________
```

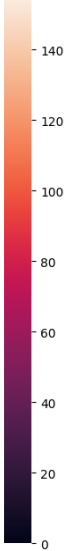
6. Model Performance Improvement and Final Model Selection

Build another model using data augmentation to overcome the imbalance problem. - Evaluate the model on different performance metrics and comment on the performance. - Plot confusion matrix - Choose best model with proper reasoning.

- Created Model 2 with Batch Normalization, Early Stopping & Drop out layer: accuracy: 0.6733 (Train) 0.5937 (Test)
- Precision, recall and accuracy evaluated: Outshining its predecessor, Model 2 boasts superior accuracy both overall and specifically on the test dataset. Yet, it's not all smooth sailing— we've noticed some unexpected twists in the validation dataset's performance. But on the bright side, the classification game has stepped up! Classes 1, 2, 3, 5, 6, 8, 10, and 11 are hitting the mark. Let's keep an eye on classes 0, 4, 7, and 9 where there's room for improvement.
- Model 3: : accuracy: 0.7673 (Train) 0.6288 (Test): classes 0, 2, 4, 7, and 9 are still misclassified, though recall for classes 0 and 4 has improved.







Model 2

	precision	recall	f1-score	support
0	0.31	0.56	0.40	79
1	0.57	0.94	0.71	117
2	0.67	0.69	0.68	86
3	0.75	0.84	0.79	183
4	0.71	0.15	0.25	66
5	0.36	0.81	0.50	143
6	0.73	0.58	0.65	196
7	0.76	0.38	0.51	66
8	0.86	0.36	0.51	155
9	0.51	0.38	0.43	69
10	0.84	0.81	0.82	149
11	0.75	0.10	0.18	116
accuracy			0.59	1425
macro avg	0.65	0.55	0.54	1425
weighted avg	0.67	0.59	0.57	1425

Model 3

	precision	recall	f1-score	support
0	0.54	0.34	0.42	79
1	0.57	0.68	0.62	117
2	0.68	0.31	0.43	86
3	0.66	0.79	0.72	183
4	0.68	0.35	0.46	66
5	0.52	0.65	0.58	143
6	0.67	0.85	0.75	196
7	0.78	0.58	0.66	66
8	0.59	0.54	0.56	155
9	0.00	0.00	0.00	69
10	0.67	0.83	0.74	149
11	0.67	0.78	0.72	116
accuracy			0.63	1425
macro avg	0.59	0.56	0.56	1425
weighted avg	0.60	0.63	0.60	1425

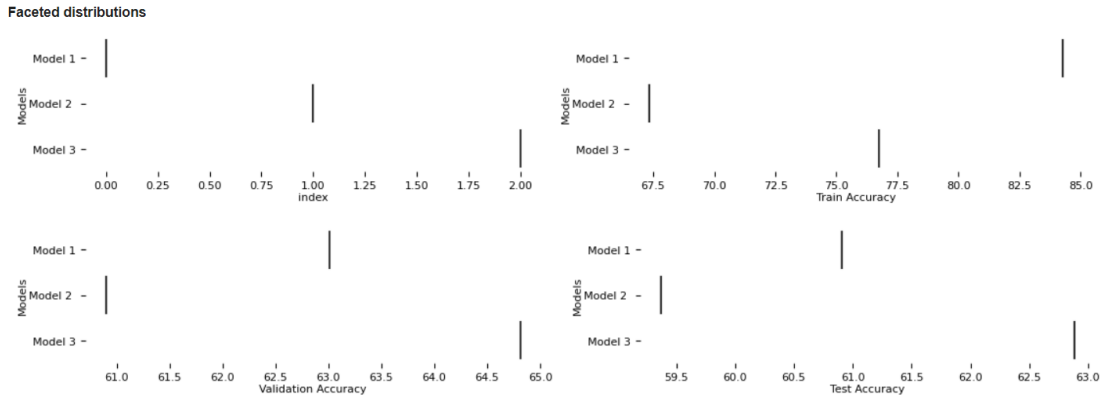
7. Final Observations, Actionable Insights & Recommendations

In this project, we evaluated three distinct models. Out of these, **Model 3** demonstrated the highest accuracy on the test data, achieving 63%. This model provided a more generalized performance compared to the others. All models showcased good results on the training data but exhibited reduced performance on the test datasets, indicating a degree of overfitting. To address this, we employed strategies like Data Augmentation, Early Stopping, Batch Normalization, and the integration of Dropout Layers, which resulted in an enhancement of the model performance.

The application of Transfer Learning not only improved the overall model efficiency but also reduced computational costs, making it a more optimal choice compared to the base CNN model.

Suggestions to improve seed image classifications include adjustments in filter sizes and increasing the number of filters. Additionally, exploring other transfer learning architectures could potentially further refine our CNN models' performance, such as:

- -Enhance the dataset by generating synthetic data especially for classes 0, 2, 4, 7, and 9.
- - Fine-tune the top layers to better align with the dataset
- Using class weights during training, giving under-represented classes more influence on the learning process.
- Use visualization tools to interpret model activations. Understanding which features are heavily relied upon can offer insights into misclassification patterns.
- For model robustness, consider integrating predictions from multiple models. Methods like bagging or boosting may enhance overall performance.



APPENDIX

Great Learning

Happy Learning !