Plant Seedling Classification

AIML course - Computer Vision

Azin Faghihi
Role: Data Scientist
March 2025

Contents / Agenda

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

Executive Summary

Insights:

- Based on our observation, we observe that lowering the **optimizer's learning rate**, adding **batch normalization**, doing **data augmentation**, lowering the number of **convolution layers** and **dense layers** could help improve the **performance** of our model and also reduce **overfitting**.

Recommendations:

- Training multiple models.
- Trying different combinations of convolution, max-pooling, dense, and dropout layers.
- Try different ways of augmenting the data.
- Explore different parameters on learning rate reduction.
- Figuring out the patterns where the models make the wrong prediction.
- Try different ways of regularization.
- Conduct more investigation on individual classes.
- Investigate how steps per epochs affects the training.

Business Problem Overview and Solution Approach

Problem Overview:

- In the problem hand, we have around 5K BRG images of plant seedlings and their labels. There are 12 categories of seedlings. We are tasked to identify them using **Convolutional Neural Network (CNN)**.

Solution Approach:

- We first conduct ***exploratory data analysis (EDA)*** such as viewing the images and investigating the distribution of our classes.
- We also conduct some pre-processing on the images such as ***size reduction***, ***converting colors*** (BRG to RGB).
- We will then pre-process the data to prepare it for modeling. This would involve **rescaling** they data, binary ***encoding*** of the labels and splitting the data into ***train, validation and test***.
- We will train two CNN models with different selection of the parameters such as number/kinds of **layers**, **normalization**, and **data augmentation**.

Business Problem Overview and Solution Approach *continued*

Solution Approach (continued):

- After training the two models, we compare their performance using ***performance metrics*** such as accuracy, recall, precision, and f1-score. We also make sure that our selected model is not suffering from overfitting by investigating the ***loss*** values.
- After conducting the comparison, we select our final model and run the model ***prediction*** on the ***test*** set.
- We also visualize the **confusion matrix** to better understand our model performance.

EDA Results

Data

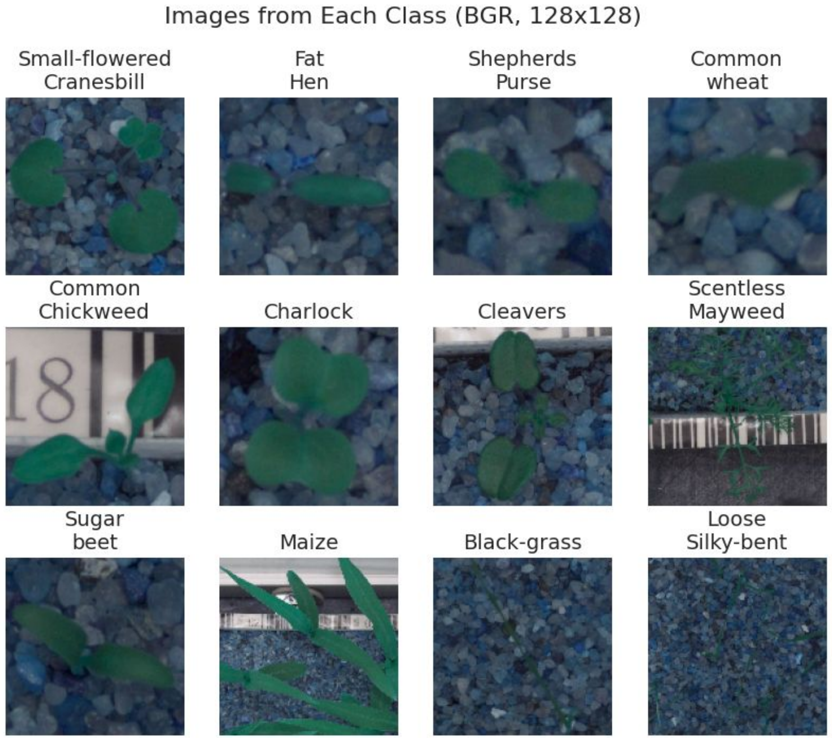
- There are originally 4,750 **images**, each of dimensions *128x128*.
- We are also provided with 4,750 **labels** corresponding to the images.
- There are **12 classes** in the labels.

	# Observations	Height	Width	# Channels	Min	Max	Color
images							
Original	4,750	128	128	3	0	255	BGR

EDA Results

Data - Taking a Look at the Original Images

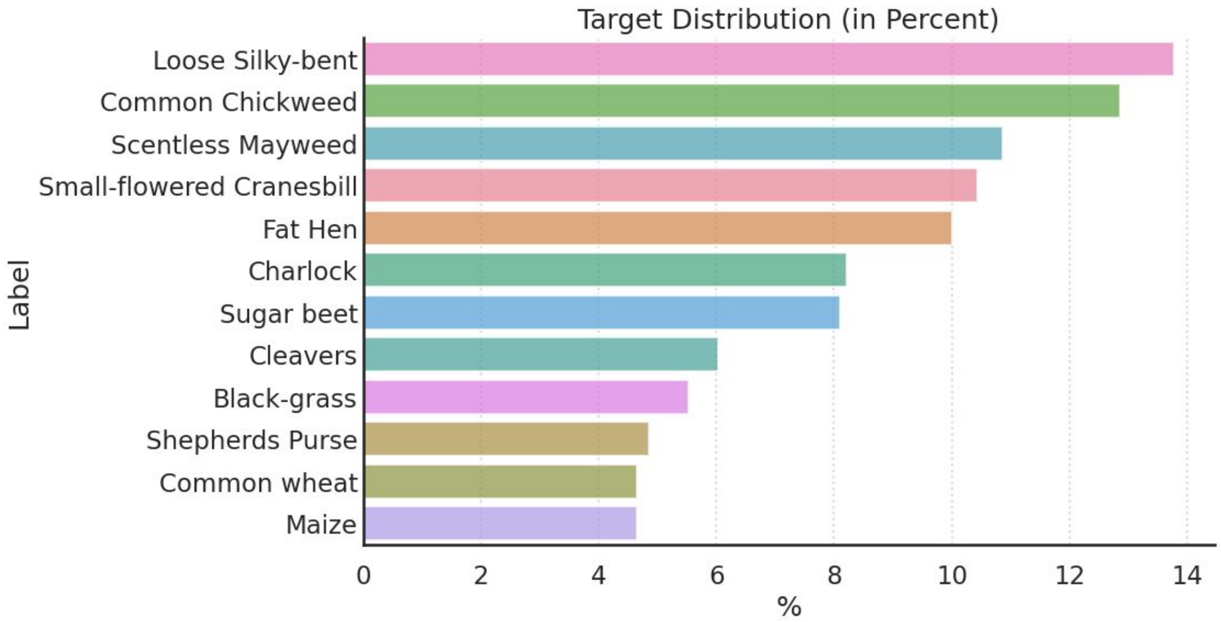
- Below, we see a sample image from each existing class.



EDA Results

Target Distribution

- Below, we see the distribution (in %) of the **12 labels**.
- About 14% of the labels belong to the *Loose Silky-Bent* plant.



EDA Results

Conversion: BGR -> RGB

- The original images are in BGR color and in data preprocessing are converted to **RGB.**

Data Preprocessing

Resizing: Reduction

- The images are resized from 128x128 to **64x64** to reduce the **computational cost**.

Data Preprocessing - *Preparation for Modeling*

Train-Validation-Test Split

- The data is split to 3 parts: **Train** (80%), **Validation** (10%), and **Test** (10%)
- Here's the summary table for the data.

images	# Observations	Height	Width	# Channels	Min	Max	Color
Original	4,750	128	128	3	0	255	BGR
Converted Colors	4,750	128	128	3	0	255	RGB
Resized	4,750	64	64	3	0	255	RGB
Train	3,799	64	64	3	0	255	RGB
Validation	476	64	64	3	0	254	RGB
Test	475	64	64	3	0	255	RGB

Data Preprocessing - *Preparation for Modeling*

Encoding the Target Class

- In order to do **multi-class classification** we binarize the labels similar to one-hot encoding. The table below summarizes the dimensions of the target variable.

target	# Rows	# Columns
Original	4,750	1
Train	3,799	1
Train (Encoded)	3,799	12
Validation	476	1
Validation (Encoded)	476	12
Test	475	1
Test (Encoded)	475	12

Data Preprocessing - *Preparation for Modeling*

Data Normalization

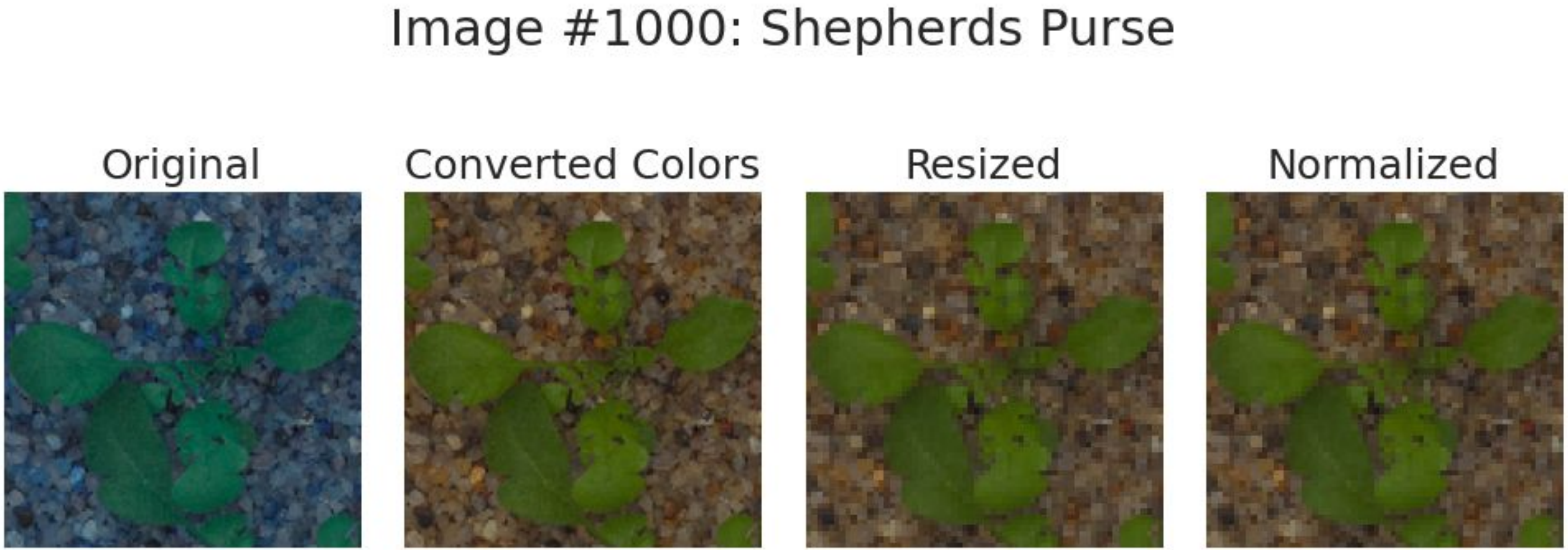
- The original images have values between **0-255**. We scale the data to **standardize** the images to have values between **0-1**.
- The following table shows the min and max value of the data for the normalized data.

images	# Observations	Height	Width	# Channels	Min	Max	Color
Original	4,750	128	128	3	0	255	BGR
Converted Colors	4,750	128	128	3	0	255	RGB
Resized	4,750	64	64	3	0	255	RGB
Train	3,799	64	64	3	0	255	RGB
Train (Normalized)	3,799	64	64	3	0	1	RGB
Validation	476	64	64	3	0	254	RGB
Validation (Normalized)	476	64	64	3	0	0.996	RGB
Test	475	64	64	3	0	255	RGB
Test (Normalized)	475	64	64	3	0	1	RGB

Data Preprocessing

Summary

- Here's the pre-processing demonstrated using a sample image.



Model Performance Summary

Model Parameters

- We have trained two models on this data. Here are the model parameters.

	Model 1	Model 2
# Trainable Params	128,828	151,612
Epochs	30	30
Batch Size	32	64
Augmentation (Rotation)	0	20
Learning Rate Reduction		ReduceLROnPlateau
Steps per Epoch	119	59
# Conv2D Layers	3	2
# MaxPooling2D Layers	3	2
# Dropout Layers	1	1
Normalization Layer		batch normalization
# Neurons (Dense)	16, 12	16, 12
# Filters (Conv)	128, 64, 32	64, 32

	Model 1	Model 2
Activation (Dense)	relu, softmax	relu, softmax
Activation (Conv)	relu, relu, relu	relu, relu
Optimizer	adam	adam
Learning Rate	0.001	0.0001
Regularization	Dropout(0.3)	Dropout(0.3)
Train Loss	1.152	0.901
Validation Loss	1.104	0.901
Train Accuracy	0.568	0.688
Validation Accuracy	0.679	0.71
Test Loss	1.096	0.94
Test Accuracy	0.667	0.678
Elapsed Time (s)	64.02	95.33

Model Performance Summary

Parameter Differences

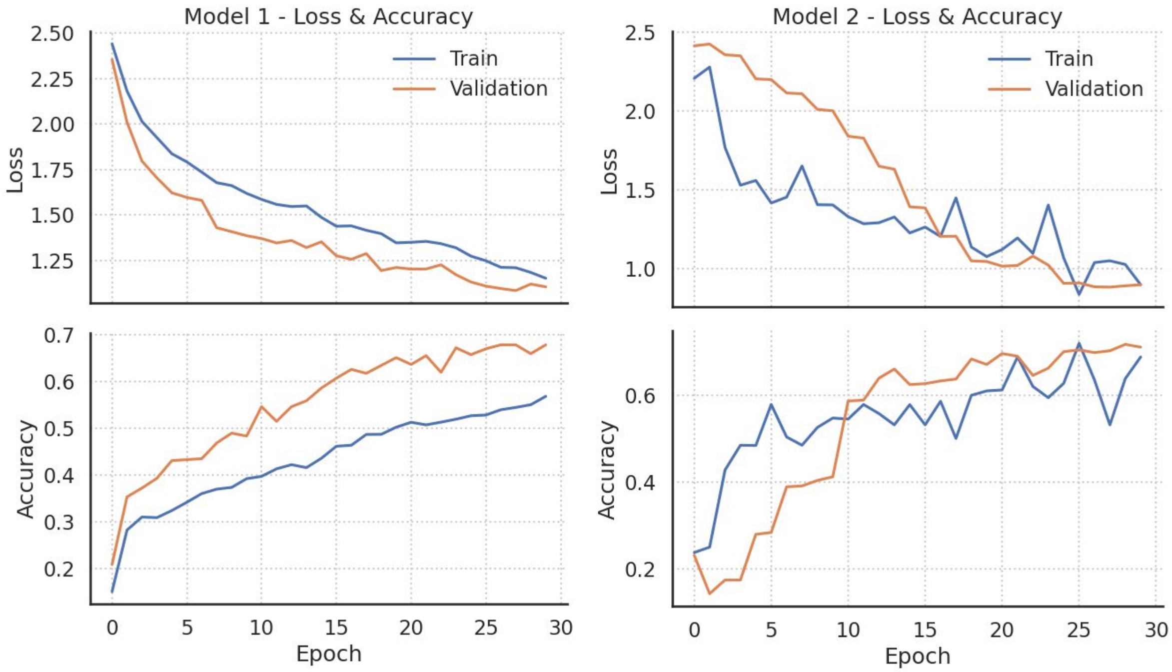
- The table below demonstrates the differences between the parameters of the two models.

Parameter Comparison	Model 1	Model 2	
# Trainable Params	128,828	151,612	Model 2 has more parameters to train.
Batch Size	32	64	Model 2 has a higher batch size.
Augmentation (Rotation)	0	20	The image for training Model 2 are rotated.
Learning Rate Reduction		ReduceLROnPlateau	Model 2 is trained with learning rate reduction.
Steps per Epoch	119	59	Model 2 has a lower number of steps per epoch.
# Conv2D Layers	3	2	
# MaxPooling2D Layers	3	2	
Normalization Layer		batch normalization	Model 2 uses batch normalization.
# Filters (Conv)	128, 64, 32	64, 32	
Activation (Conv)	relu, relu, relu	relu, relu	
Learning Rate	0.001	0.0001	Model 2 optimizer uses a lower learning rate.
Elapsed Time (s)	64.02	95.33	Model 2 takes longer to train.

Model Performance Summary

Loss & Accuracy

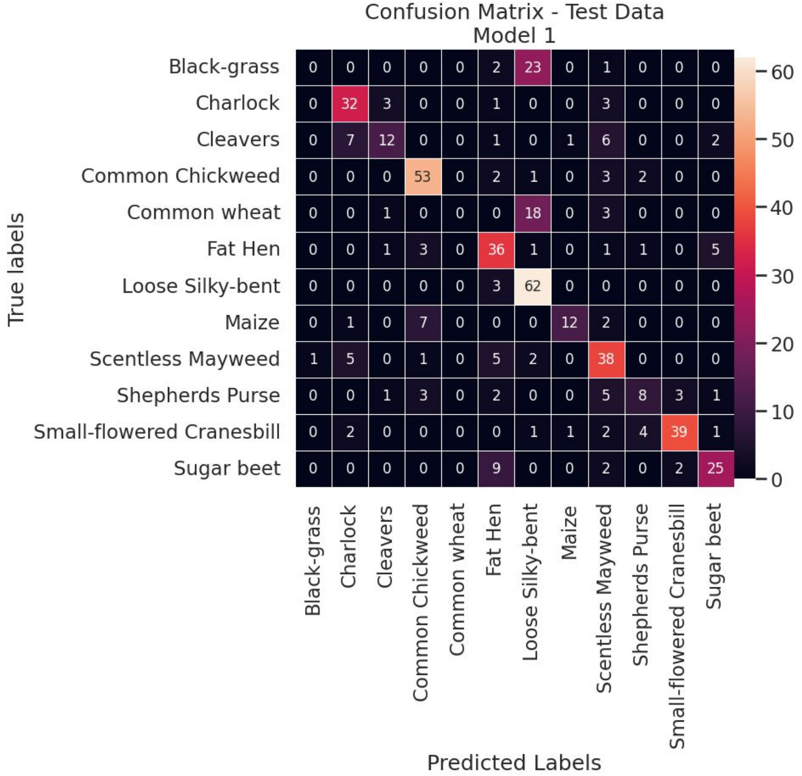
- The loss and accuracy versus the number of epochs is demonstrated here.
- The curves are smoother for Model 1 however the loss approaches a lower number in Model 2.

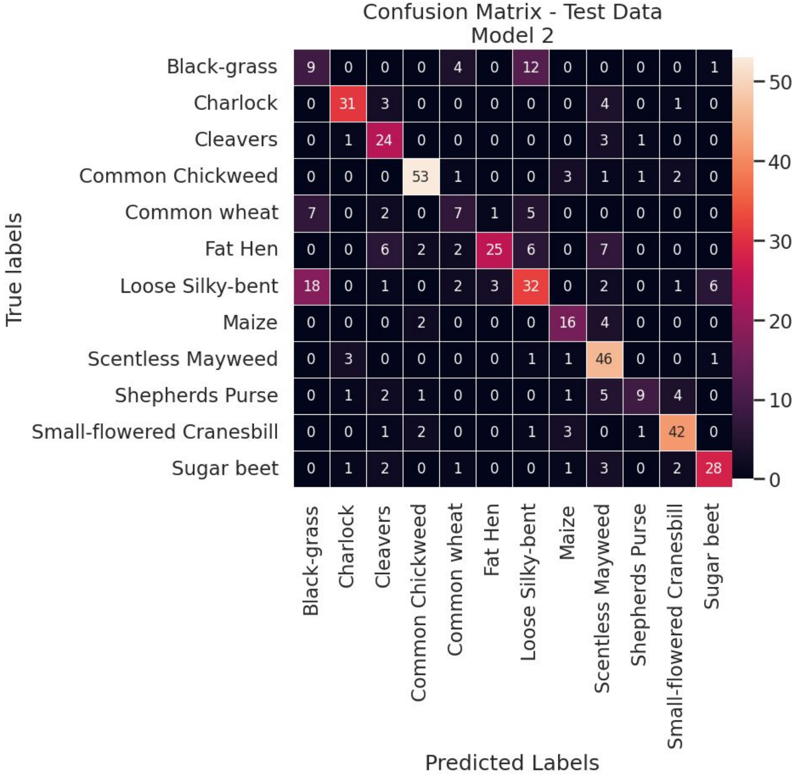


Model Performance Summary

Confusion Matrix - Test Data

- Here are the classification matrices for the two models.

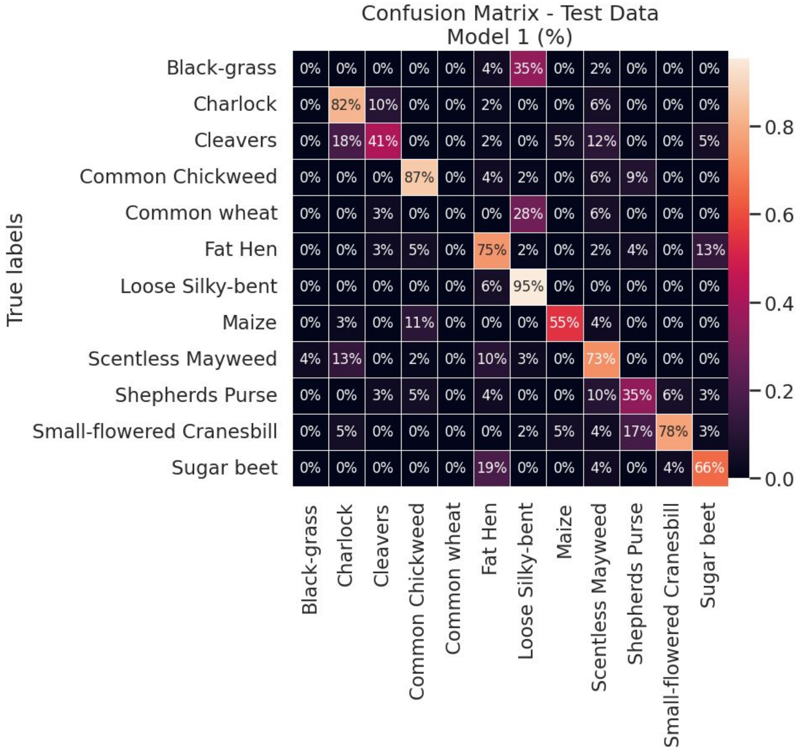


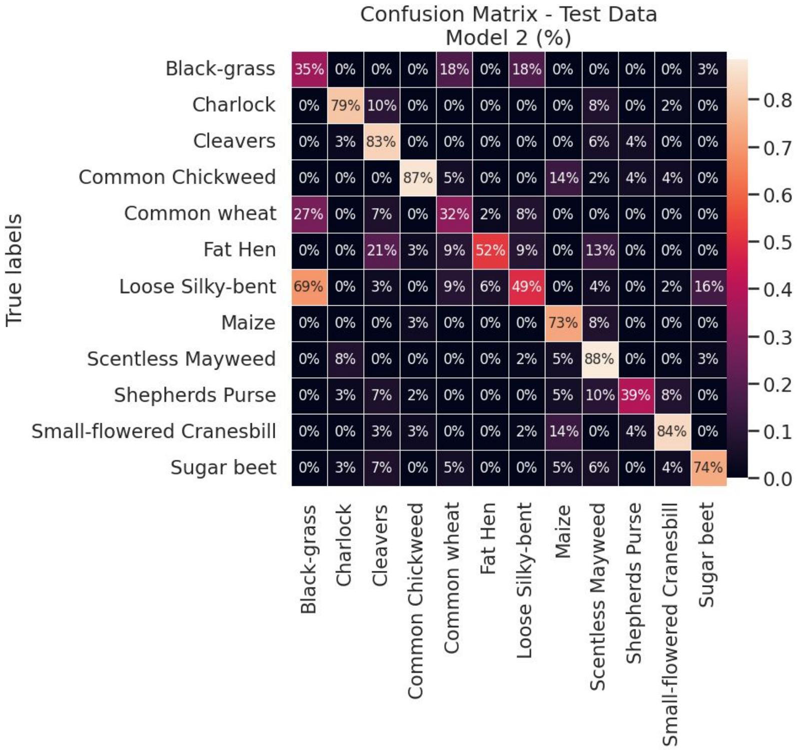


Model Performance Summary

Confusion Matrix (Normalized)- Test Data

- Here are the classification matrices (normalized) for the two models.





Model Performance Summary

Performance Matrix - Test Data

- Here are the classification reports for the two models.

Classification Report (Test Data) – Model 1

	precision	recall	f1-score	support
Black-grass	0.000	0.000	0.000	26
Charlock	0.681	0.821	0.744	39
Cleavers	0.667	0.414	0.511	29
Common Chickweed	0.791	0.869	0.828	61
Common wheat	0.000	0.000	0.000	22
Fat Hen	0.590	0.750	0.661	48
Loose Silky-bent	0.574	0.954	0.717	65
Maize	0.857	0.545	0.667	22
Scentless Mayweed	0.576	0.731	0.644	52
Shepherds Purse	0.533	0.348	0.421	23
Small-flowered Cranesbill	0.886	0.780	0.830	50
Sugar beet	0.735	0.658	0.694	38
accuracy			0.667	
macro avg	0.574	0.572	0.560	475
weighted avg	0.617	0.667	0.628	475

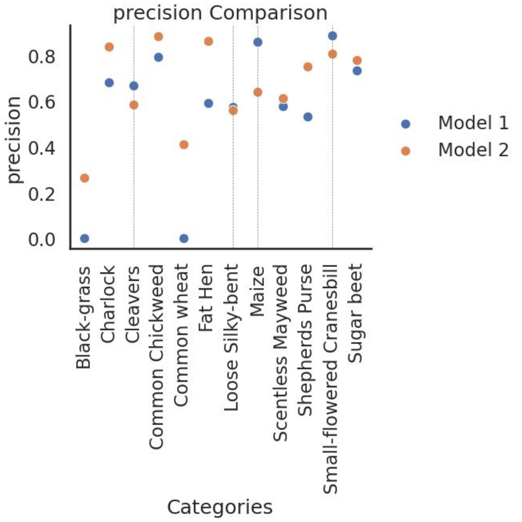
Classification Report (Test Data) – Model 2

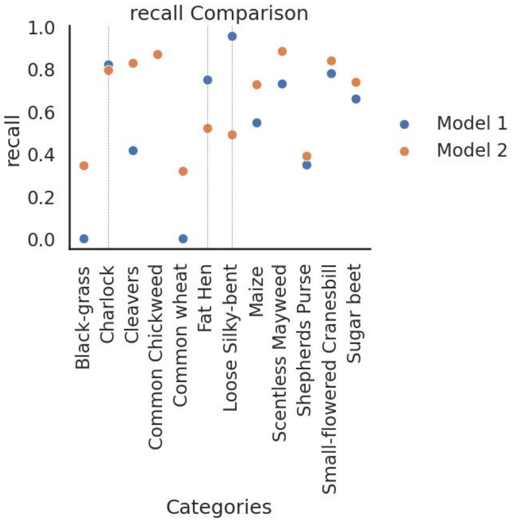
	precision	recall	f1-score	support
Black-grass	0.265	0.346	0.300	26
Charlock	0.838	0.795	0.816	39
Cleavers	0.585	0.828	0.686	29
Common Chickweed	0.883	0.869	0.876	61
Common wheat	0.412	0.318	0.359	22
Fat Hen	0.862	0.521	0.649	48
Loose Silky-bent	0.561	0.492	0.525	65
Maize	0.640	0.727	0.681	22
Scentless Mayweed	0.613	0.885	0.724	52
Shepherds Purse	0.750	0.391	0.514	23
Small-flowered Cranesbill	0.808	0.840	0.824	50
Sugar beet	0.778	0.737	0.757	38
accuracy			0.678	
macro avg	0.666	0.646	0.643	475
weighted avg	0.696	0.678	0.675	475

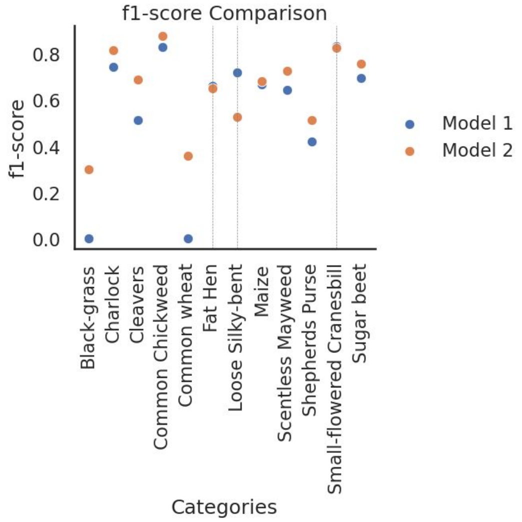
Model Performance Summary

Performance Values - Precision, Recall, f1-score

- The **precision** values for Model 2 is greater than that of Model expect for *Cleavers, Loose Silky-bent, Maize, and Small-flowered Cranesbill.*
- The **recall** values for Model 2 is greater than that of Model expect for *Charlock, Fat Hen, and Loose Silky-bent.*
- The **f1-score** values for Model 2 is greater than that of Model expect for *Fat Hen, Loose Silky-bent, and Small-flowered Cranes.*







Model Performance Summary

Model Comparison - Choosing the final model

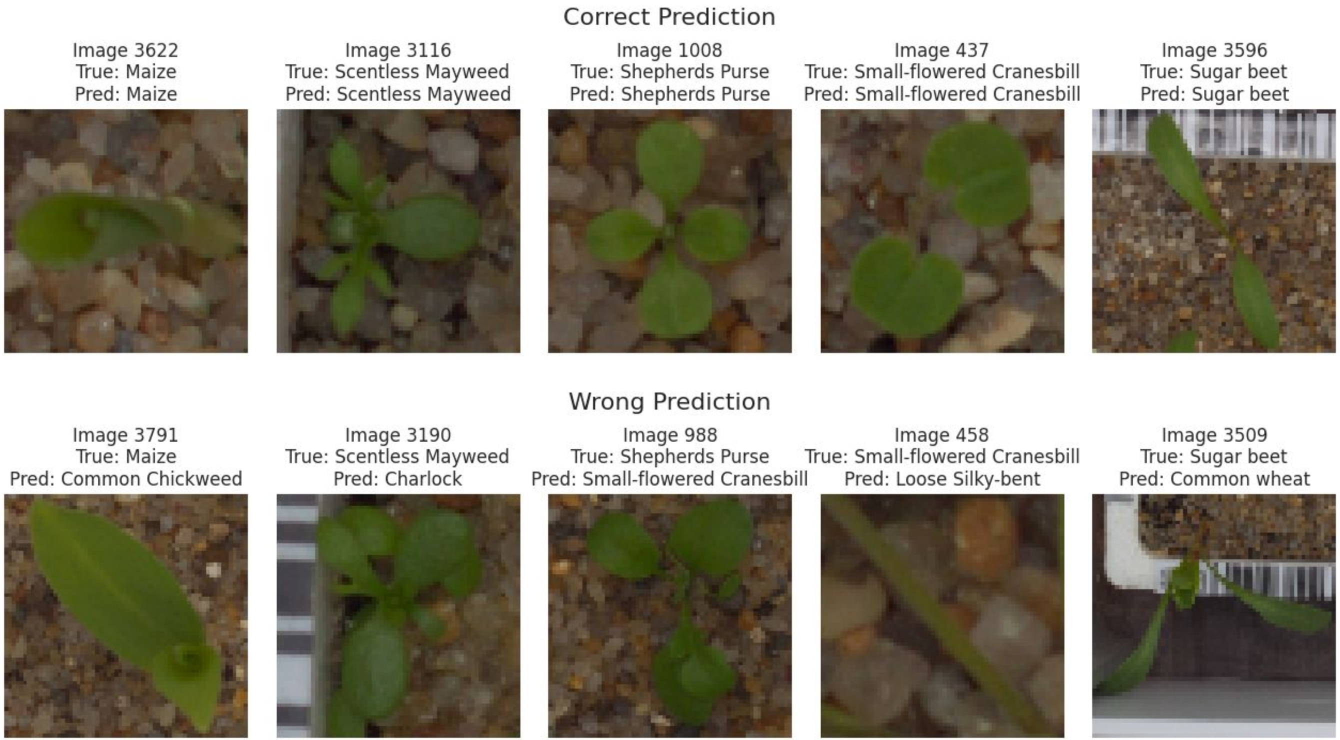
- We choose **Model 2** to be our final model since it has a higher train and validation **accuracy** and less absolute loss difference which shows that it has ***less overfitting***.
- **Model 2** has in higher values for f1-score, precision, and recall for most categories as seen in the previous slide.

	Model 1	Model 2	
Train Loss	1.1517	0.9008	
Validation Loss	1.1043	0.901	
Train Accuracy	0.568	0.688	Model 2 has a higher train accuracy.
Validation Accuracy	0.679	0.71	Model 2 has a higher validation accuracy.
I Train Loss - Validation Loss I	0.0474	0.0002	Model 2 has a smaller absolute loss difference.
Test Loss	1.096	0.94	
Test Accuracy	0.667	0.678	

Model Performance Summary

Visualizing the Prediction

- Below you see the samples of correct and wrong predictions.



Conclusion

Our observation is that lowering the **optimizer's learning rate**, adding **batch normalization**, doing **data augmentation**, lowering the number of **convolution layers** and **dense layers** could help improve the **performance** of our model and also reduce **overfitting**.

APPENDIX

Great Learning

Happy Learning !