Plant Seeding Classification

Date: 12/14/23

Contents / Agenda

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

Executive Summary

	Observations and Findings
Predictive Model	• Analyzed the dataset containing images of unique plants belonging to 12 different species and applied four different type of techniques to build the Convolutional Neural Network models and selected the best Classification predictive model. • The final model can be deployed and use to identify plant seedlings to free up human involvement for higher-order agricultural decision making.
Model training and performance	• Our models have 128,828 to 151,612 Trainable parameters for the base and Data Augmentation models, respectively. • For training VGG16, we directly used the convolutional and pooling layers and freeze their weights i.e., no training was done on them. • 4 models were built and validated; 4 models were fine tuned to get the best performance for final selection.
Data Augmentation	• The advantage of data augmentation is that all the methods can be used simultaneously, resulting in many distinct data samples from the initial one and exponential growth in the number of combinations possible. • This proved to help reduce the model's overfitting and it outperformed all other models in this analysis.
Transfer learning with the VGG16	• We observed the model built with transfer learning on the pre-built VGG16 model did not turn out to be the best model compared to models built with Data Augmentation and Hyperparameter Search tuning method.
Hyperparameter Search tuning	• Hyperparameter Search tuning was built using the same layers, and parameters as the base model, however, the model seems to be quite overfitted but still provided the second-best accuracy scores among all other models. • RandomSearch technique was used to search for the best combination of parameters from the Keras tuner for building the model.

Business Problem Overview and Solution Approach

Problem Statement

In recent times, the field of agriculture has been in urgent need of modernizing, since the amount of manual work people need to put in to check if plants are growing correctly is still highly extensive. Despite several advances in agricultural technology, people working in the agricultural industry still need to have the ability to sort and recognize different plants and weeds, which takes a lot of time and effort in the long term.

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.

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. The object of this analysis is to build a Convolutional Neural Network model which would classify the plant seedlings images into their respective 12 categories.

.

Solution Approach

Identity focus area

To extract actionable insights from the image collection.

Perform Data Analysis & Build Predictive Model

- Perform EDA on Variables that influence the model categorization.
- Build several sequential Convolutional Neural Network multi-classification models to classify the plant seedlings.
- Optimize the models using various techniques to determine the best predication model.

Summarize Recommendation

Summarize the observation and Conclusion for presentation.

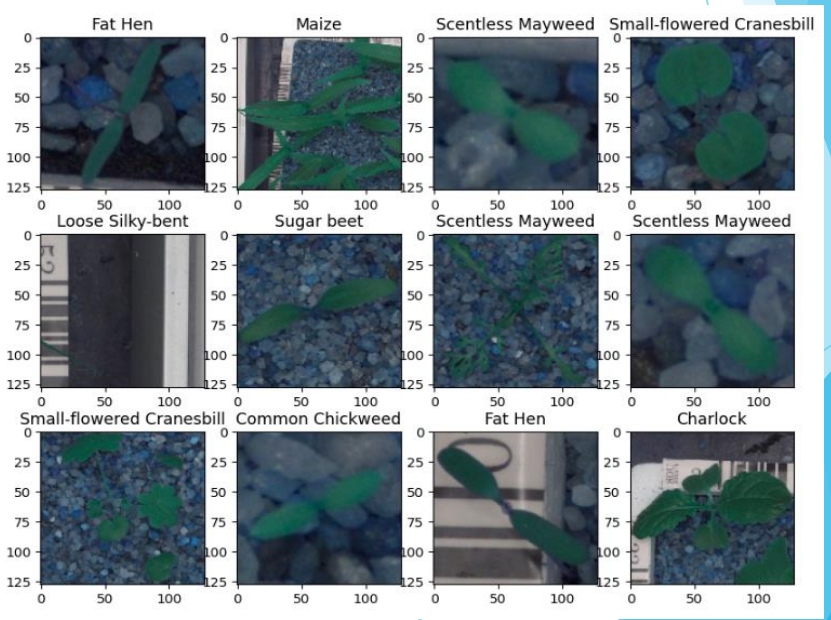
Data Overview

- 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.
- Due to the large volume of data, the images were converted to the images.npy file and the labels are also put into Labels.csv, these files were used as the data input to the model.
- There are 4750 RGB images of shape 128x128x3, each image has 3 channels.

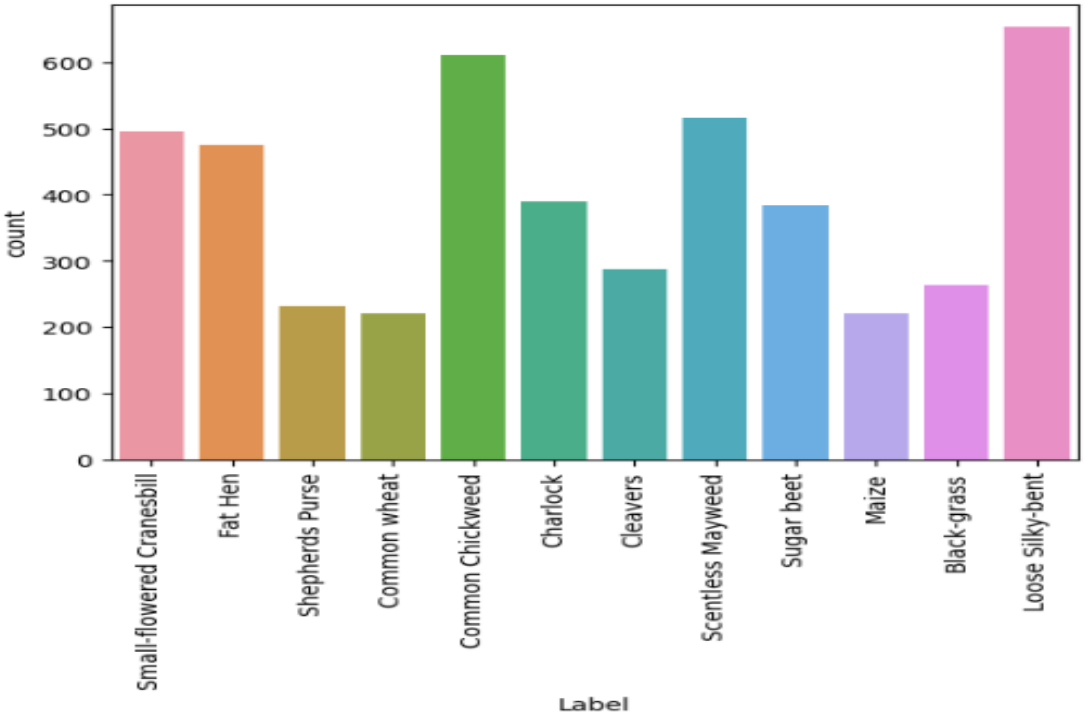
List of Species

1) Black-grass
2) Charlock
3) Cleavers
4) Common Chickweed
5) Common Wheat
6) Fat Hen
7) Loose Silky-bent
8) Maize
9) Scentless Mayweed
10) Shepherds Purse
11) Small-flowered Cranesbill
12) Sugar beet

Images with their Labels



EDA Results - Univariate Data Analysis



Observations:

- As you can see from the above plot, the dataset is not very balanced.
- There are 12 category, with the label 'Loose Silky-bent' has the most images, Common Chickweed comes the second, and 'Maize' has the lease images

Data Preprocessing

- Resizing Images – as the size of the images is large, it may be computationally expensive to train on these larger images; therefore, it is preferable to reduce the image size from 128 to 64.
- Images conversion – the images are also converted from BGR to RGB because of the default color format used by different image processing libraries e.g. TensorFlow
- Data preparation for Modeling - as we have less images in our dataset, we will only use 10% of our data for testing, 10% of our data for validation and 80% of our data for training. We are using the train_test_split() function from scikit-learn. Here, we split the dataset into three parts, train, test and validation.
    - Total rows for train data is 3847, for validation data is 428, and for test data is 475 after the two data split executions.
- Encoding the target class – we have converted the labels from names to one hot vector by using an encoding method called labelBinarizer, Labelbinarizer works like onehotencoder. Each class will be represented in the form of array.
- Data Normalization - since the image pixel values range from 0-255, our method of normalization here will be scaling - we have divided all the pixel values by 255 to standardize the images to have values between 0-1.

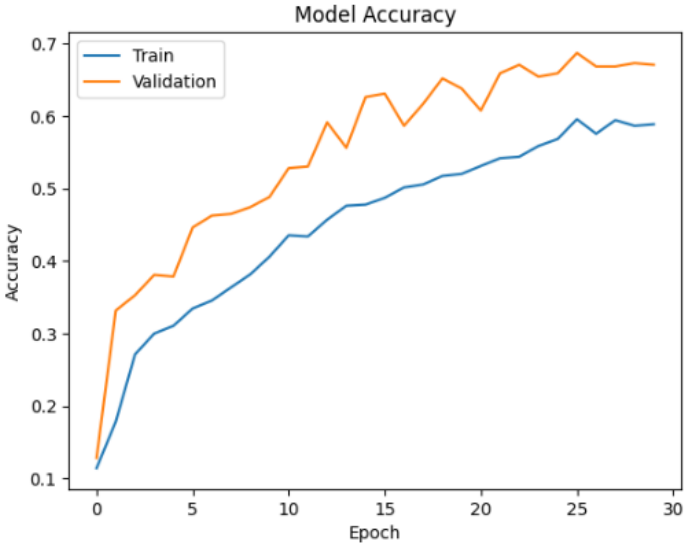
Model Building

The Followings are the multi-classification model building steps :

- a CNN model was created sequentially, where we will be adding the layers one after another.
- First, we need to clear the previous model's history from the session even if a single model can run multiple times on the same data.
- In Keras, we used a special command to clear the model's history, otherwise the previous model history remains in the backend, we also fixed the seed again after clearing the backend.
- We set the seed for random number generators in Numpy, the Random library in Python, and in TensorFlow to be able to reproduce the same results every time we run the code.
- Convolution layers and filter were applied to the images after feature extraction and ReLu activation, then padding and max pooling Nfor isolation important feature.
- We flattened the feature map and feed it into a fully-connected neural network to generate the final predictions.

Model Performance Summary (Model 1)

- CNN Model with the following 2 main parts: 1) The Feature Extraction layers which are comprised of convolutional and pooling layers. 2) The Fully Connected classification layers for prediction. Also, we are choosing accuracy as the metric to measure the performance of the model.
- this CNN model will train and learn 128,828 parameters (weights and biases).

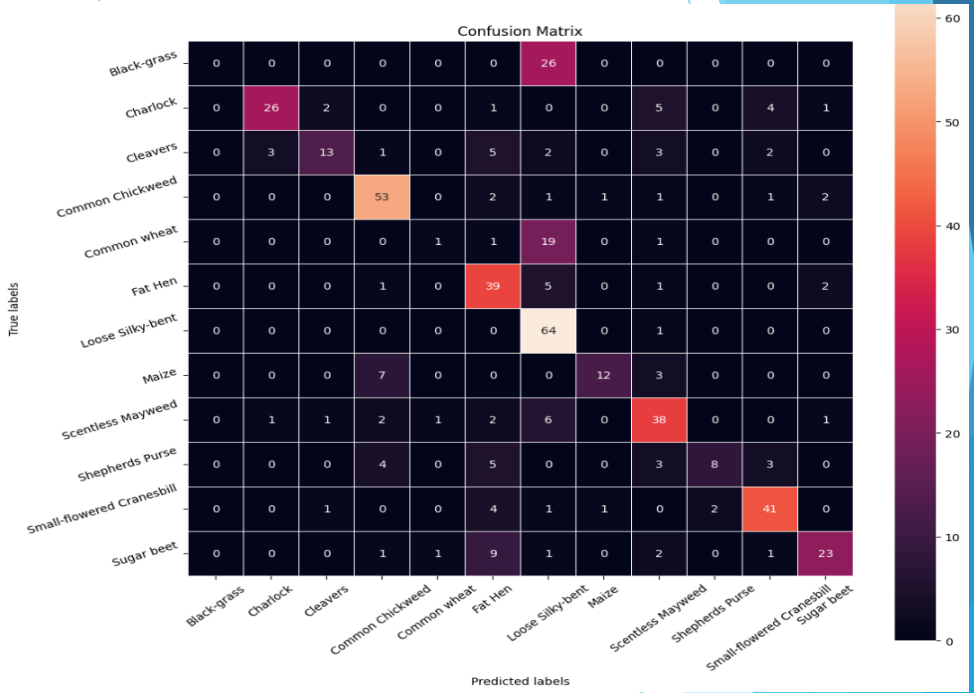



```
] accuracy = model1.evaluate(X_test_normalized, y_test_encoded, verbose=2)

  15/15 - 0s - loss: 1.0719 - accuracy: 0.6695 - 263ms/epoch - 18ms/step
```

Observations:

- We can see from the above plot that the training accuracy of the model was good, but the validation accuracy was not good.
- The model seems to be overfitted on the data.
- The result gave **test accuracy score of 67%.**

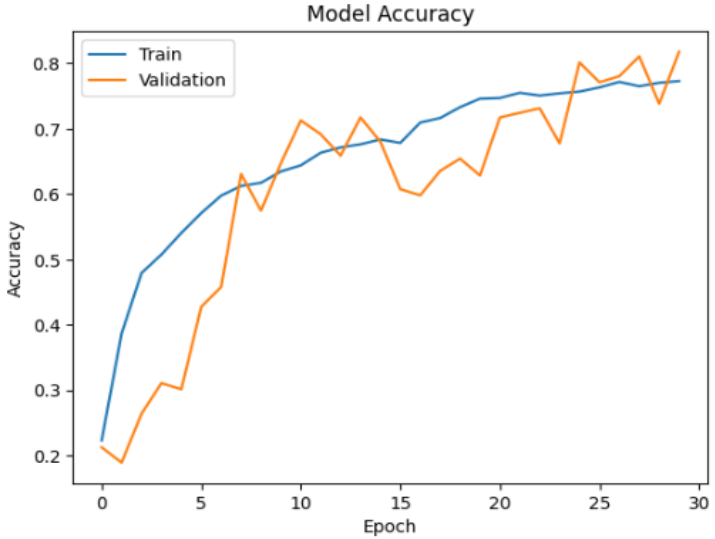


Observations

- Loose Silky-bent, common-chick weed and scentless mayweed, small-flowered cransebill are predicated correctly and well classified.
- Observed the rest of the classes are mostly misclassified.

Performance Summary use Data Augmentation (Model 2)

- In most of the real-world case studies, it is challenging to acquire a large number of images and then train CNNs. To overcome this problem, one approach we use is Data Augmentation. CNNs have the property of translational invariance, which means they can recognize an object even if its appearance shifts translationally in some way. Taking this property into account, we augment the images using different techniques like horizontal flip, vertical flip, etc. This CNN model will train and learn 151,612 parameters (weights and biases).

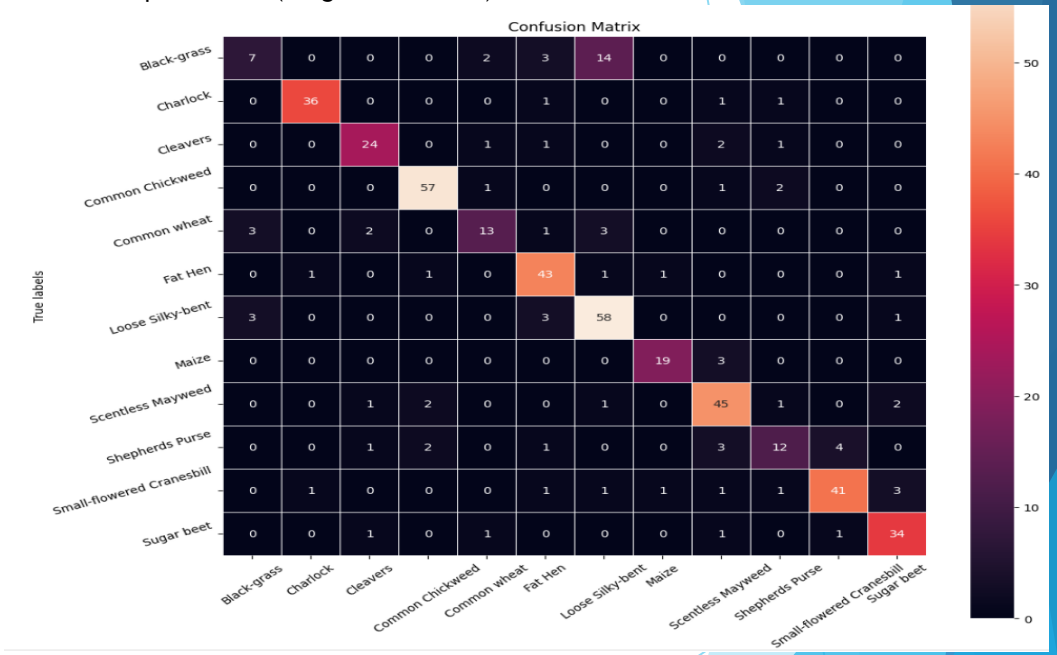



```
accuracy = model2.evaluate(X_test_normalized, y_test_encoded, verbose=2)
```

```
15/15 - 0s - loss: 0.6350 - accuracy: 0.8189 - 70ms/epoch - 5ms/step
```

Observations:

- The model seems to have been improved compared to our previous model.
- The result gave a **test accuracy score of 82%.**

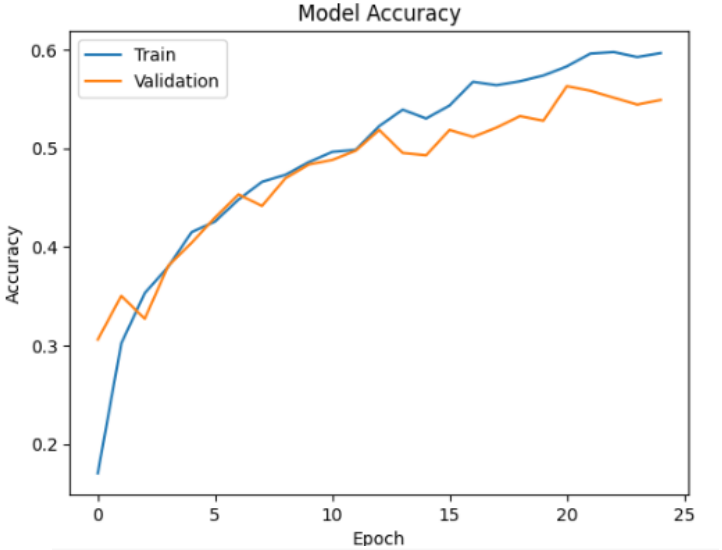


Observations

- The confusion matrix appears to be improving as well, however there is still some confusion with the Black-grass, Common wheat classes.
- We can observe that this model has outperformed our previous model.

Performance Summary use Transfer Learning VGG16 (Model 3)

- Transfer Learning - We loaded a pre-built architecture - VGG16, which was trained on the ImageNet dataset and is the runner-up in the ImageNet competition in 2014.
- For training VGG16, we will directly use the convolutional and pooling layers and freeze their weights i.e. no training will be done on them. For classification, we ed specifically

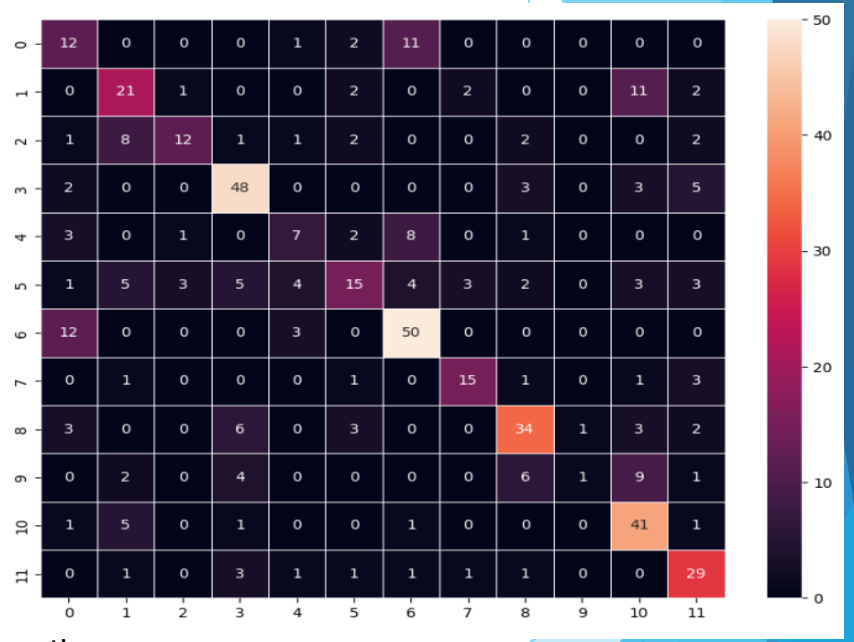



```
accuracy = new_model.evaluate(X_test_normalized, y_test_encoded, verbose=2)
```

```
15/15 - 1s - loss: 1.2334 - accuracy: 0.6000 - 691ms/epoch - 46ms/step
```

Observations:

- We can see from the above plot that the training accuracy of the model was good, but the validation accuracy was not good.
- The result gave a **test accuracy score of 60%.**

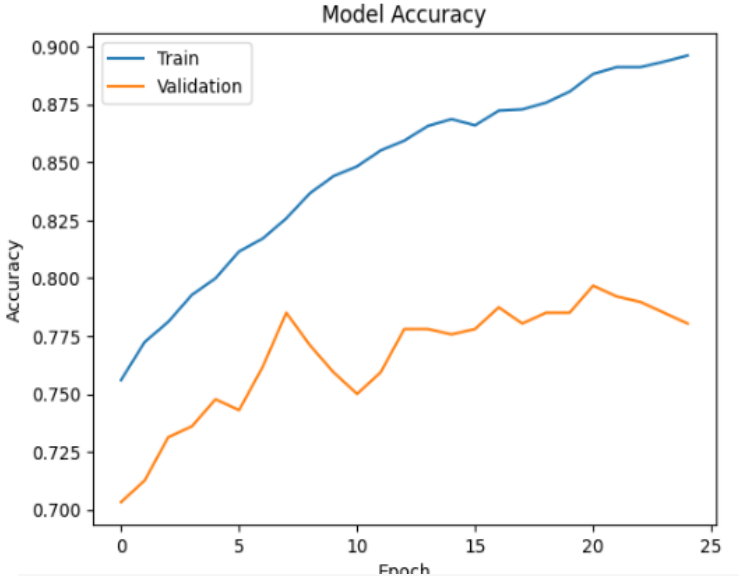


Observations

- The confusion matrix appears to be improving as well, however there are still many classes that are mis-classified.
- Common wheat and Maizeclass are the most confused class among all.
- We can observe that this model has degraded from our previous model.

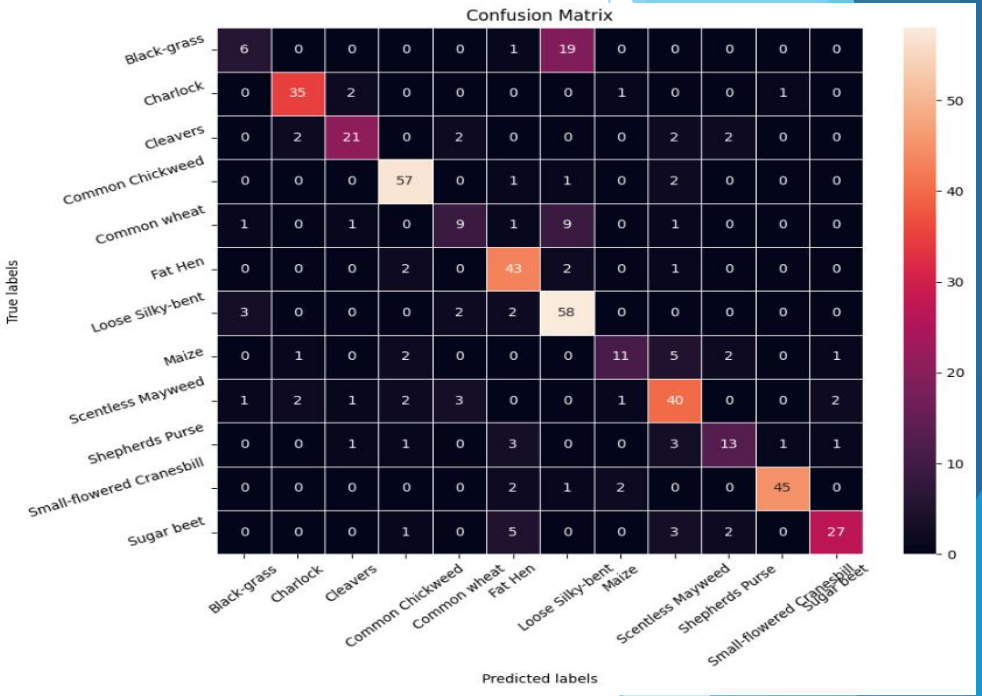
Performance Summary use Hyperparameter Search Tuning(Model 4)

- The model was also trained by utilizing Keras Tuner to systematically search for the best hyperparameters to see if there is better performance gained.




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

```
15/15 - 0s - loss: 0.8257 - accuracy: 0.7684 - 92ms/epoch - 6ms/step
```



Observations:

- We can see from the above plot that the training accuracy of the model was very good, but the validation accuracy was not good.
- The model seems to been overfitted even we have applied Early-stopping, drop out technique.
- The result gave a **test accuracy score of 77%.**

Observations

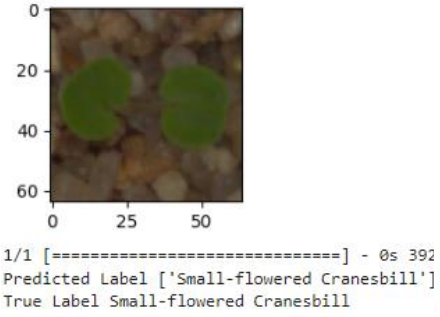
- The confusion matrix appears to be improved a lot in many classes, however there is still some confusion with Black-grass, Common wheat, Shepherds Purse and classes.
- Common Wheat class is the most confused class among all.
- We can observe that this model has outperformed our previous model.

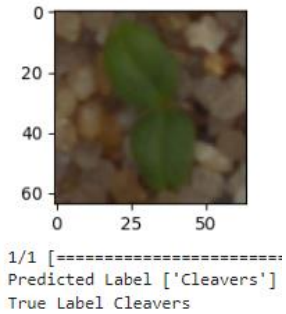
Model Comparison and Final Model Selection

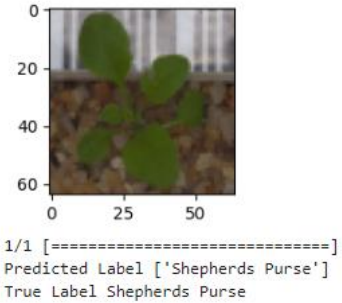
- After building 4 CBB models, it was observed that both CNN with Data Augmentation and CNN with Hyperparameter Search Tuning, exhibited strong performance on both the training and validation datasets.
- We built these models with different convolutional layers, max pooling, and a final dense layer for classification, as well as tuner configuration to improve the model's performance. The result is as followings:

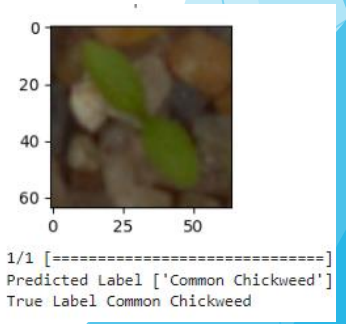
Models	Train Accuracy	Validation Accuracy	Test Accuracy
Base CNN Model	59%	67%	67%
CNN Model with Data Augmentation	77%	82%	82%
Transfer Learning with VGG16 Model	60%	55%	60%
CNN Model with Hyperparameter Search Tuning	89%	77%	77%

- Convolutional Neural Network model trained with **Data Augmentation** got the best accuracy score on unseen test data, therefore it's considered to be the best and final model. We also visualize the prediction from model 2 (Data Augmentation) and got accurate prediction results.









Conclusion

- By comparing the train and test accuracy to find out if the model is overfitting, so adding more layers to the model worked, and based on that we can say that this CNN model is good. We also tried building different models by increasing the hidden layers and see if we get good accuracy.

- Data Augmentation outperformed model built with VGG16 model could be due to the dataset we used may have unique features that a generic pre-trained model like VGG16 does not capture. A model tuned specifically for our dataset can learn these unique features more effectively.

- Hyperparameter tuning searches for the best combination of parameters (like learning rate, number of layers, dropout rate, etc.) for our specific problem. This can lead to a model that is better adapted to the nuances of your dataset compared to the pre-trained VGG16, which was trained on ImageNet and may have a different set of optimal hyperparameters.

- Based on several round of tuning and using different techniques combination to build the models, Data Augmentation outperformed all other models and is the final model we selected for prediction. However, all these models can be further improved by training with different filter sizes and different number of filters.