Advanced Analytics

Task 1

Nathan Hefner

A. Select one of the scenarios and describe the purpose of this data analysis by doing the following:

1. Provide one research question that you will answer using neural network models and computer vision techniques.

One research question I will propose is whether neural network models can predict which plant will come from specific seedlings?

2. Define the objectives or goals of the data analysis

The main objective of this data analysis will be to train a neural network model that can classify and distinguish twelve different plant seedlings. This model will help botanists by optimizing their workflow through the earlier identification of specific seedlings, rather than after they have sprouted and grown. Thus, improving plant yield and saving valuable time during the harvest by knowing the plant type before the seedlings grow.

3. Identify an industry-relevant type of neural network capable of performing an image, audio, or video classification task that can be trained to produce useful predictions on image sequences on the selected dataset.

The neural network I will choose is a Convolutional Long Short-Term (ConvLSTM) network, because it has strong capabilities for image sequence classification.

4. Justify your choice of neural network

ConvLSTM is a specialized type of neural network that combines CNN layers, which allows the model to capture spatial patterns such as shapes, textures, and edges, directly from the image, and RNN layers, which gives the model the ability to detect temporal dependencies such as growth stages, and changes in light and leaf patterns, while utilizing input, output, and forget gates to control how much past information is kept or discarded. With both of these features combined, this neural network becomes the industry standard in agriculture, medical imaging, and video analytics because of its inherent spatiotemporal understanding. A necessity for data that changes over time.

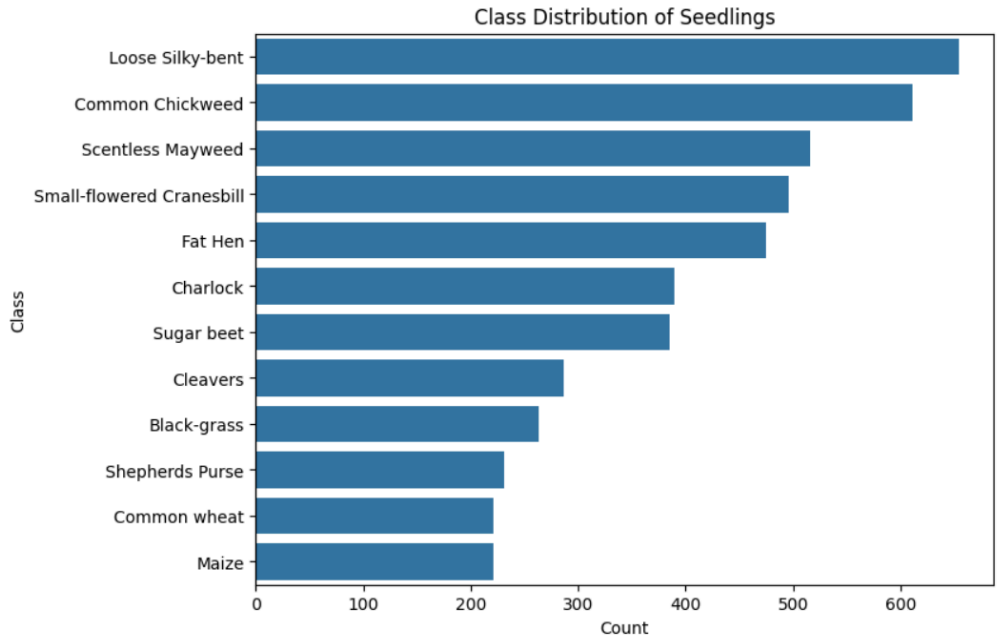
B. Describe the data preparation process by doing the following:

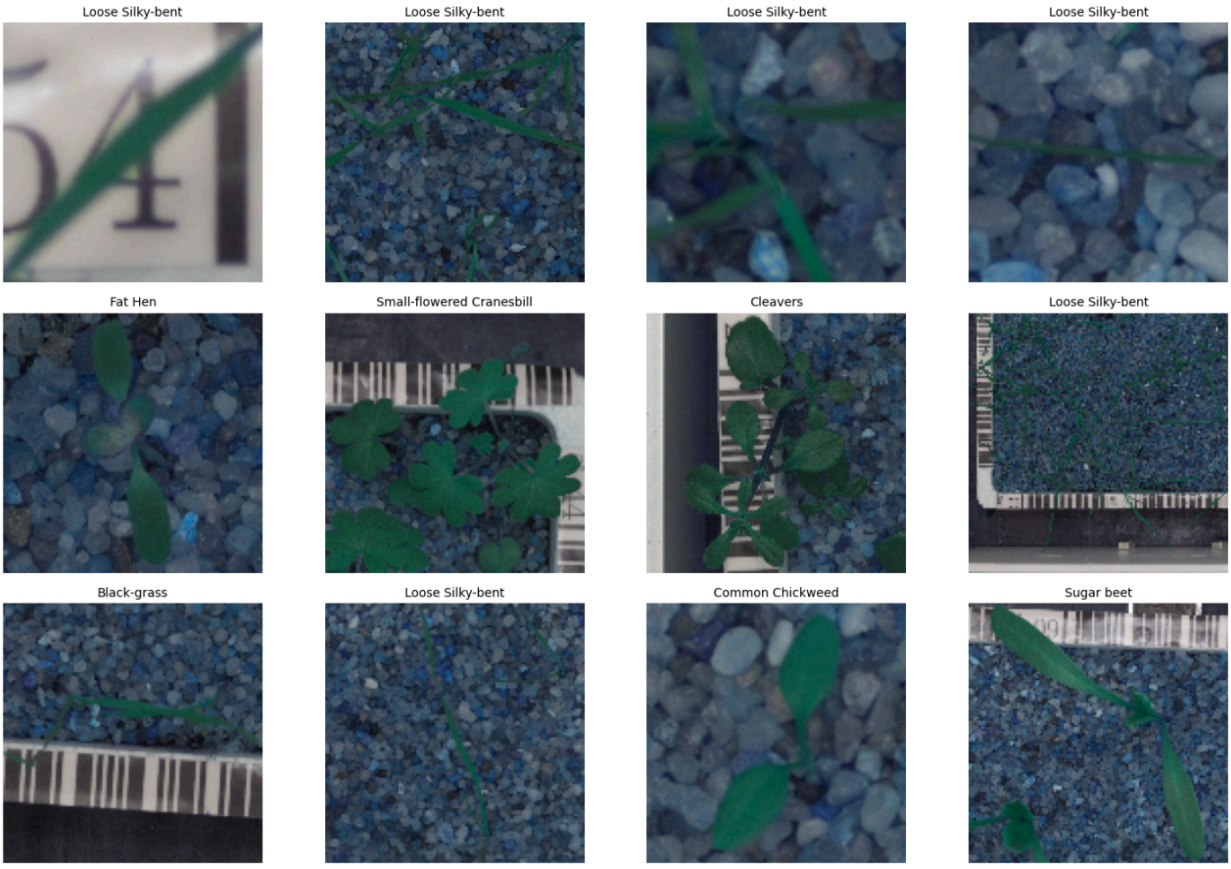
1. Perform exploratory data analysis on the chosen image dataset and provide screenshots of the following:

a. visualization for the distribution of the different classes

b. sample images with associated labels

Below is a simple count plot of the seedling distribution and the sample images with the associated labels.





2. Perform data augmentation and justify the steps taken to augment the images.

Data augmentation is an artificial way to produce more data points to train the models further. It also simulates the wide variety of pictures possible. I'll now go over the augmentations I applied to create more data and why:

Horizontal Flip - Seedlings can grow in any orientation, and leaves are not always symmetric, but a horizontal picture of a plant is still the same plant. For real-world scenarios when the pictures are being taken of the seedlings, the orientation is almost always different; by applying these augmentations, it teaches the model to be invariant to camera orientation.

Rotation - Seedlings rarely grow perfectly upwards; relative to the camera, the soil could be uneven, or leaves could bend. In real-world scenarios, images may be captured in less controlled environments, but rotation augmentations will help keep the model robust to different variations in seedling tilt, camera angles, or leaf droop.

Zoom - Pictures may be taken closer or further away due to hardware inconsistencies. Pictures taken from different camera gear or at different times may experience slight to large differences in distance. By including these augmentations, we will help train the model to recognize these differences.

Width and Height Shift - Most of the time, the seedling will not be perfectly centered. By shifting the frames' width and height, it will help train the model to still recognize the seedling. It will also help by removing some features of the seedling, by losing it through frame shifting. This will help train the model with images that are missing some information and photos that aren't perfectly aligned. The most important aspect is that it will train the model on the seedling instead of the background.

Brightness Adjustments - Lighting can vary dramatically for a number of reasons, whether it be environment or hardware differences. By adjusting the brightness, it will also change hue and color, simulating environmental changes, and will train the model to recognize the seedlings regardless of light levels. By training the model with these augmentations, it will keep the classifier more robust for these uncontrollable factors.

3. Normalize the images and discuss the steps taken for normalization.

In the step above, I rescaled the pixels. This is to normalize the pixels, meaning the value of every pixel is between 0 and 1 instead of 0 and 255. This helps optimize the model by making the training speed faster and more stable.

4. Perform a train-validation-test split and justify your selection of the proportions for the split.

For the training, testing, and validation split, I did a 70%, 15%, and 15% split because the training needs the most data for learning, and the validation and test data sets can be evenly split. I decided on these splits because if there was less training data, the model would show high variance in training, but with less testing/validation data, the model's performance would have

greater variance. So this specific split is the most optimal to avoid both of these problems (Train test validation split: How to & best practices).

```
X_train, X_train_val, y_train, y_train_val = train_test_split(images, labels_encoded, test_size=0.3, random_state=42, stratify=labels_encoded)
X_val, X_test, y_val, y_test = train_test_split(X_train_val, y_train_val, test_size=0.5, random_state=42, stratify=y_train_val)
```

```
np.save("X_train.npy", X_train)
np.save("X_val.npy", X_val)
np.save("X_test.npy", X_test)

np.save("y_train.npy", y_train_encoded)
np.save("y_val.npy", y_val_encoded)
np.save("y_test.npy", y_test_encoded)
```

5. Encode the target feature appropriately for all your datasets and discuss the steps taken.

Encoding the dataset is essential as the deep learning model requires numeric one-hot encoded vectors rather than a string of integers. We will need both label encoding and one-hot encoding to convert the labels into integers and then convert the integers to class vectors for the neural network.

```
y_train_encoded = to_categorical(y_train)
y_val_encoded = to_categorical(y_val)
y_test_encoded = to_categorical(y_test)
```

E. Describe the type of network used by doing the following:

1. Provide the output of the model summary.

Model: "sequential_3"

Layer (type)	Output Shape	Param #
conv2d_8 (Conv2D)	(None, 126, 126, 32)	896
max_pooling2d_8 (MaxPooling2D)	(None, 63, 63, 32)	0
conv2d_9 (Conv2D)	(None, 61, 61, 64)	18,496
max_pooling2d_9 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten_1 (Flatten)	(None, 57600)	0
dense_6 (Dense)	(None, 64)	3,686,464
dropout_3 (Dropout)	(None, 64)	0
dense_7 (Dense)	(None, 12)	780

Total params: 3,706,636 (14.14 MB)

Trainable params: 3,706,636 (14.14 MB)

Non-trainable params: 0 (0.00 B)

2. Discuss components of your neural network architecture and justify the choice of the following:

Number of Layers

The number of layers varies for each reason; the feature extraction uses 2-4 to capture low and high-level spatial patterns. Temporal modeling uses 1-2 layers to capture temporal dependencies and 1-2 layers for dense layers to assist with final decision making. We keep the number of layers in a specific range because if there are too few, then the model may underfit, too many layers, the model may overfit. So keeping a moderate depth balances these two risks.

Types of Layers

There are many types within the neural network, all with unique jobs to accomplish. The convolutional layers (Conv2D) extract spatial features such as edges, textures, and patterns from the different frames. ConvLSTM layers combine convolution with LSTM to capture spatial-temporal features such as motion. Pooling Layers (MaxPooling2D) reduce spatial dimensions, which leads to faster training and translation invariance. Flatten converts 2D feature maps into a 1D vector for dense layers. Dense layers perform final classification by mapping extracted features to class probabilities.

Number of Nodes per Layer

The number of nodes per layer varies for each layer. For Conv layers, the number of nodes starts small (32) but doubles the deeper layers you go because the deeper layers need more filters to capture the more complex patterns. For the ConvLSTM Layer, 64 is a standard amount to help balance accuracy and computational load. The Dense layers have 64 nodes to extract high-level features.

Number of Parameters

Our model has just over one hundred thousand parameters, which helps reduce computational load and overfitting the data by getting the same results many times over if the parameters were higher. CNN layers help carry the learning power because that is when the bulk of feature extraction happens within the convolutional layers. GAP also helps by encoding the features more efficiently for classification, thus not needing as many parameters.

Activation Functions

The layers have different activation functions. For Conv2D and Dense layers, use ReLU (Rectified Linear Unit) because it prevents vanishing gradient problems, sparse activation for efficiency, and fast convergence. For Output, the activation is Softmax because we are using multi-class classification, interpretability, which allows easy thresholding and confidence analysis, and works with categorical cross-entropy loss for multi-class problems.

3. Discuss the backpropagation process and justify the choice of the following hyperparameters:

Backpropagation is the process of computing gradients of the loss function with respect to each trainable parameter and updating them to minimize loss.

Loss Function

Our loss function is Categorical Cross-Entropy because our network is a multi-class classification with one-hot encoded labels. It also directly measures the difference between the true and predicted probability distribution.

Optimizer

The optimizer we will use is Adam (Adaptive Moment Estimation) because it combines the benefits of momentum and RMSProp (adaptive learning rate) and reduces sensitivity to manual learning rate tuning.

Learning Rate

Since we are using Adam, the default learning rate is 0.001. We use this value because if it is too high, there will be divergence, and if it's too low, there will be convergence.

Stopping Criteria

The stopping criteria we will choose is EarlyStopping with patience=5, which will monitor for validation loss. We chose 5 because this will help prevent overfitting by stopping before the model memorizes training data, and with validation-based stopping, this will ensure the model generalizes well.

4. Create, explain, and provide a screenshot of your confusion matrix.

The confusion matrix above is a heatmap showing which seedlings the model predicted to be and what the seedlings actually were. Along the diagonal are the correct predictions; anything outside of that is an incorrect prediction. Overall, the model performed very well; all the incorrect predictions happened a low amount of times, except for Loose Silky-bent being mistaken for Black-grass, this being the highest amount of incorrect predictions. This may be because of the data, the image-generated data points, or something else entirely, and will need to be explored further.

F. Analyze the model by doing the following:

1. Evaluate the model training process and its relevant outcomes by doing the following:

a. Discuss the impact of using stopping criteria to include defining the number of epochs, including a screenshot showing the final training epoch.

```
Epoch 1/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 21s 178ms/step - accuracy: 0.4829 - loss: 1.3929 - val_accuracy: 0.6368 - val_loss: 1.1588
Epoch 2/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 178ms/step - accuracy: 0.4900 - loss: 1.3365 - val_accuracy: 0.6355 - val_loss: 1.1075
Epoch 3/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 178ms/step - accuracy: 0.5237 - loss: 1.2640 - val_accuracy: 0.6539 - val_loss: 1.0920
Epoch 4/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 177ms/step - accuracy: 0.5547 - loss: 1.1848 - val_accuracy: 0.6395 - val_loss: 1.1085
Epoch 5/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 178ms/step - accuracy: 0.5592 - loss: 1.1723 - val_accuracy: 0.6303 - val_loss: 1.1185
Epoch 6/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 178ms/step - accuracy: 0.5832 - loss: 1.0879 - val_accuracy: 0.6737 - val_loss: 1.0360
Epoch 7/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 178ms/step - accuracy: 0.6066 - loss: 1.0502 - val_accuracy: 0.6395 - val_loss: 1.0470
Epoch 8/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 43s 192ms/step - accuracy: 0.6247 - loss: 0.9464 - val_accuracy: 0.6513 - val_loss: 1.1180
Epoch 9/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 23s 195ms/step - accuracy: 0.6484 - loss: 0.9337 - val_accuracy: 0.6395 - val_loss: 1.0715
Epoch 10/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 40s 184ms/step - accuracy: 0.6505 - loss: 0.8758 - val_accuracy: 0.6618 - val_loss: 1.1018
Epoch 11/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 41s 186ms/step - accuracy: 0.6663 - loss: 0.8628 - val_accuracy: 0.7026 - val_loss: 1.0283
Epoch 12/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 22s 185ms/step - accuracy: 0.6755 - loss: 0.8444 - val_accuracy: 0.6737 - val_loss: 1.0623
Epoch 13/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 42s 189ms/step - accuracy: 0.6871 - loss: 0.7759 - val_accuracy: 0.6842 - val_loss: 1.0617
Epoch 14/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 22s 185ms/step - accuracy: 0.7074 - loss: 0.7488 - val_accuracy: 0.6618 - val_loss: 1.1457
Epoch 15/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 40s 179ms/step - accuracy: 0.7158 - loss: 0.7141 - val_accuracy: 0.6618 - val_loss: 1.1937
Epoch 16/50
119/119 ━━━━━━━━━━━━━━━━━━━━ 22s 186ms/step - accuracy: 0.7147 - loss: 0.6935 - val_accuracy: 0.6592 - val_loss: 1.1998
```

With a patience value set to 5, the model trains until it does worse when training on new data. This is why the stopping criteria are important because without them, the model will continue training, even after lowering the accuracy, and will start to overfit the data. By including the stopping criteria, we optimize the accuracy and prevent overfitting.

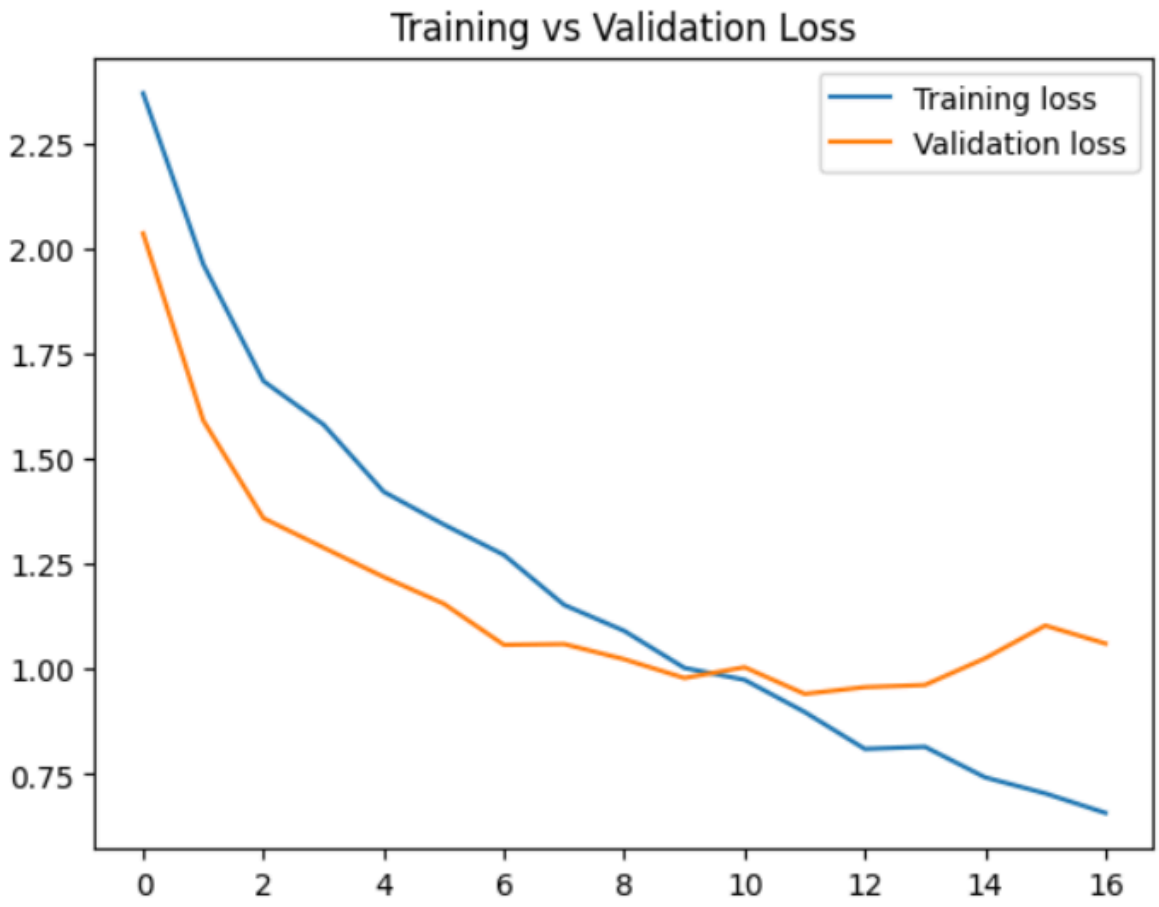
b. Compare the training data to the validation dataset using an evaluation metric such as loss, accuracy, F1, or mean absolute error (MAE) to assess model performance. Include an explanation.

```
y_train_pred = model.predict(X_train)
y_val_pred = model.predict(X_val)

y_train_pred_classes = np.argmax(y_train_pred, axis=1)
y_val_pred_classes = np.argmax(y_val_pred, axis=1)

y_train_classes = np.argmax(y_train, axis=1) if y_train.ndim > 1 else y_train
y_val_classes = np.argmax(y_val, axis=1) if y_val.ndim > 1 else y_val

f1_train = f1_score(y_train_classes, y_train_pred_classes, average='macro')
f1_val = f1_score(y_val_classes, y_val_pred_classes, average='macro')

print(f"F1-score (Train): {f1_train:.4f}")
print(f"F1-score (Validation): {f1_val:.4f}")
```

```
104/104 ━━━━━━━━━━━━━━━━━━━━ 5s 46ms/step
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 45ms/step
F1-score (Train): 0.8840
F1-score (Validation): 0.6549
```

The evaluation metric we used is F1-score, a metric that converts precision and recall of a dataset into one harmonic mean. The F1 score for the training is 0.884, which means the model is doing very well, correctly identifying the seedlings in the training set. However, the validation score is much lower, equaling 0.6549. This is a sign of overfitting within the model, and that the model does not generalize well. Meaning the model is memorizing examples instead of learning patterns. This difference in scores ultimately means the model has room for improvement.

c. Provide a visualization comparing the model's training versus validation loss. Include a screenshot.



This graph shows both the training loss and validation loss both start by decreasing, which is a strong sign of the model learning. The optimal number of epochs appears to be around 9 at the point of intersection. This is when the model is not overfitting and generalizes well. After this, with more epochs, the model starts to overfit as the gap only widens between training and validation loss.

2. Assess the fitness of the model and any actions taken to address overfitting or underfitting.

3. Discuss the predictive accuracy of the trained network using the test set and the chosen evaluation metric from part F1b.

```
test_loss, test_acc = model.evaluate(X_test, y_test_encoded)
print(f"Test Accuracy: {test_acc}")
```

```
23/23 ━━━━━━━━━━━━━━━━━━━━ 1s 51ms/step - accuracy: 0.6816 - loss: 1.0142
Test Accuracy: 0.6816269159317017
```

With our model's test accuracy being 68% the model does an excellent job identifying the majority of the seedlings except for two, being Black-Grass and Loose Silky-bent. This also means the model does not overfit and fits well. Below are measures I took to prevent overfitting:
Early Stopping - By having a low patience score, the model will stop training when the accuracy drops, therefore preventing overfitting as we see in the graph above.
Seedling Distribution - By balancing the number of data points to be relatively even between all the seedlings, this helps distribute the focus and training on all the seedlings instead of having a high density on a few specific seedlings, thus preventing overfitting of data.
Dropout Layer - Helps reduce overfitting by reducing the number of neurons in a fully connected layer.

G. Summarize results by doing the following:

2. Discuss the functionality of your neural network, including the impact of the network architecture.

By being able to consistently identify the correct seedlings, with the exception of two, the neural network is successful. Every part in the architecture played a unique role in achieving this success. The dropout layers helped reduce overfitting to create more accurate predictions overall, and the dense layers combined the data to create fewer parameters and faster computational times. For the data to be processed more easily, it needed to be transformed into an array, which the flattening layer accomplished. Convolutional layers dealt with the actual image detection by helping the model detect the patterns, textures, and edges. All these pieces work uniformly to create a successful model.

3. Discuss the effectiveness of the model in addressing the business problem you identified in part A1.

With a success rate of ⅝, I would say this model is highly effective at addressing my business problem, which is can a neural network can be trained to identify what type of plant the seedlings are. With an accuracy percentage of 68, the model is more right than wrong, even with the two plants it struggles with most.

4. Discuss lessons learned, including how the model might be improved (i.e., if you were to deploy this model in a real-life scenario).

I learned that training a neural network model was harder than first conceived because many more factors weren't initially present. By learning what and how affects the model the most, we can adjust and tweak these parameters to continually improve the model. Such as having a more evenly spread dataset instead of having a highly dense amount in only a select few plants. Also, by gathering more data so as not to rely on image generation too strongly, and by improving the photo captures. All of these factors will contribute to a more accurate and precise model.

5. Recommend a course of action based on your results as they relate to the research question.

A course of action I would recommend is that, since we proved the capabilities of the neural network to be highly accurate, if we wanted to use the model more often and reliably, then I would try to improve the data points used to train the model. To do so, I would gather more real pictures manually, and on top of that, I would generate more images through the varying levels of augmentation we used before. This would exponentially increase the amount of data, leading to a better pool of images to train off of. Along with this, I would clean the data better by focusing on removing class imbalance between the seedling samples and ensuring there is a more even spread within the data, to help the model have a stronger ability to identify specific plants. This will help the model identify plants overall, but more specifically help reduce the error in identifying the Loose Silky-bent and Black-Grass, thus vastly improving the accuracy and the model's capabilities.

Code Sources

https://www.tensorflow.org/tutorials/images/cnn
https://learn.udacity.com/nd222-ent-wgu-data-sci-d604-2?version=1.0.9&partKey=cd1821&lessonKey=35982887-10d0-4335-a3e8-5517b66d4660&conceptKey=c0e54a40-5b92-40cf-9563-5a99fd02c587

Sources

"Train Test Validation Split: How to & Best Practices [2024]." *V7*, www.v7labs.com/blog/train-validation-test-set. Accessed 9 Sept. 2025.