Can a Neural Network Model Effectively Classify Images of Plant Seedlings?

By: Kevin Sandoval

Part I: Research Questions

A1.

The research question I am choosing is “Can a Neural Network Model Effectively Classify Images of Plant Seedlings?” This could have many real-world applications, as classifying images can have a huge impact on health or recognition. This is especially clear in the healthcare industry, where a model could potentially be scaled to find abnormalities on scans or other images. In this case, it will be a simpler model to try and create an effective image classifier for plants.

A2.

The goal(s) of this analysis is to be able to effectively classify different images. The model will train on a variety of different plant seedlings and then must predict on new images. The results will be graded, and the model will be evaluated. Ideally the model will be able to effectively classify a variety of different images.

A3.

The type of neural network that can perform the classification task for the image files I chose will be a convolutional neural network (CNN).

A4.

According to the WGU Course Material, the definition of CNNs is: “CNNs are specialized for processing grid-like data, such as images. They use convolutional layers to automatically and adaptively learn spatial hierarchies of features. CNNs are widely used in computer vision tasks, including image classification, object detection, and facial recognition.” The reason I believe that an CNN is the optimal model for video, is that they can handle grid-like data. This makes them a great specific case use for computer vision tasks such as image classification.

Part II: Data Preparation

B1a.

The visualization of the distributions of the different classes is shown below.

A graph of different colored bars

AI-generated content may be incorrect.

B1b.

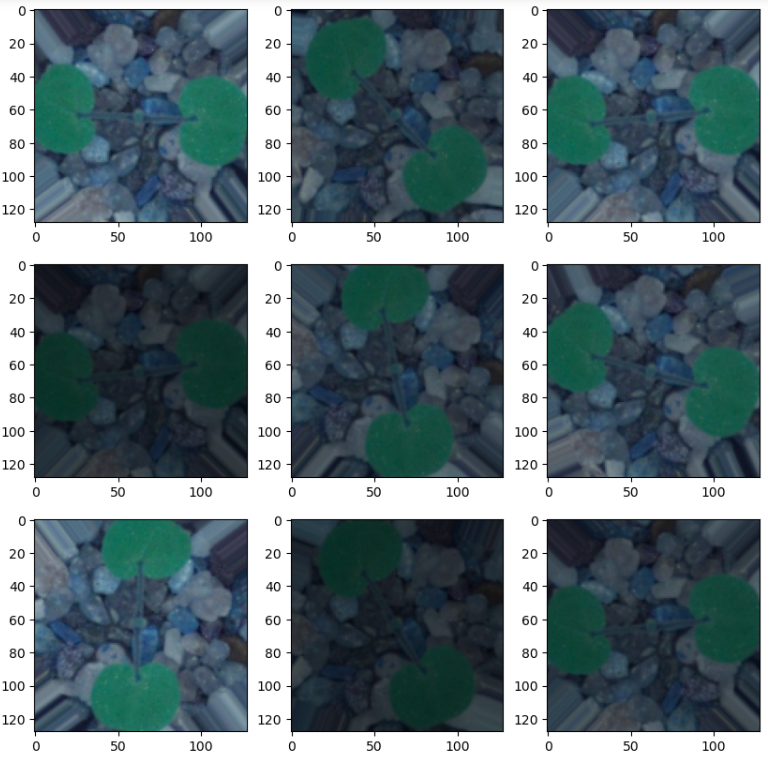
The sample images with associated labels is shown below

A close-up of different plants

AI-generated content may be incorrect.

B2.

In order to perform the data augmentation, I used the ImageDataGenerator function from the tensorflow.keras.preprocessing.image package. I decided to edit the brightness, rotation range, and to fill in the nearest pixels if they were missing after the edits. Below is a sample screenshot of the augmented images.



Editing this image by changing the brightness and rotation angle allows the user to create a significant amount of data points from just one image. This can help tremendously when there is little data available.

B3.

To normalize the images, I just divided the “images” array by 255. This changes all values in the array from 0 to 255, to 0 to 1 while maintaining the correct proportionality.

B4.

To perform the train test split, I called the function train\_test\_split on the images array and the encoded labels. I chose a size of 0.3 for the test set, thus 0.7 for the training set. Since we also need a validation set, I performed another split on the newly created 0.3 size test set. I set the size equal to 0.5 so that way the validation and test sets would both be 0.15 the amount of the original dataset.

B5.

I used the function LabelEncoder() to fit transform the different labels of seedlings. I also used the function to\_categorical on the target features for the training, testing, and validation sets.

B6.

All datasets created are provided in the submission.

Part III: Network Architecture

E1.

Output of the model summary is shown below.

A screenshot of a computer

AI-generated content may be incorrect.

E2a.

As shown above, there are 8 layers in this neural network. They consist of two convolutional layers, two pooling layers, a flatten and dropout layer, and then the fully connected and output layers. This amount of layers allows the neural network to effectively build a functioning model.

E2b.

The first type of layer used is the convolutional layer. These are used to “scan the input data using filters (kernels) to detect patterns like edges, textures, or shapes.” (upGrad, 2025). The second type of layer used is the pooling layer. They are used to “reduce the size of volume which makes the computation fast reduces memory and also prevents overfitting.” (GeeksforGeeks, 2024). The flattening layer is used to turn the feature maps into a 1 dimensional vector so they can be passed into a linked layer for categorization. (GeeksforGeeks, 2024). The dropout layer is used to decrease the number of nodes that are “on” in order to decrease overfitting. The first dense layer is also known as the fully connected layer which “takes the input from the previous layer and computes the final classification or regression task.” (GeeksforGeeks). The last layer is the second dense layer, also known as the output layer. The output layer takes “The output from the fully connected layers which is then fed into a logistic function for classification tasks like sigmoid or softmax which converts the output of each class into the probability score of each class.” (GeeksforGeeks, 2025)

E2c.

The number of nodes for each layer is shown below. This is a result of the effect each layer has on the model respectively.

1st Convolutional Layer: 32 nodes

1st Pooling Layer: 32 nodes

2nd Convolutional Layer: 64 nodes

2nd Pooling Layer: 64 nodes

Flatten Layer: 57,600 nodes

Dropout Layer: 57,600 nodes

1st Dense Layer: 128 nodes

2nd Dense Layer: 12 nodes

E2d.

The number of parameters for each layer is shown below. This is a result of the effect each layer has on the model respectively.

1st Convolutional Layer: 896 parameters

1st Pooling Layer: 0 parameters

2nd Convolutional Layer: 18,496 parameters

2nd Pooling Layer: 0 parameters

Flatten Layer: 0 parameters

Dropout Layer: 0 parameters

1st Dense Layer: 7,372,928 parameters

2nd Dense Layer: 1,548 parameters

E2e.

The activation function for each layer is shown below.

1st Convolutional Layer: ‘relu’. This is used for efficiency in the training and computations

1st Pooling Layer: None

2nd Convolutional Layer: ‘relu’. This is used for efficiency in the training and computations

2nd Pooling Layer: None

Flatten Layer: None

Dropout Layer: None

1st Dense Layer: ‘relu’. This is used for efficiency in the training and computations

2nd Dense Layer: ‘softmax’. This takes the output and returns probabilities, which is then used for the classification.

E3a.

For the backpropagation process, the initial test accuracy of the model was very low, at 6.45%. After the model was fit on the training data and using epochs = 20 and callbacks based on the best weights, the model performed significantly better. The loss function chosen was ‘Binary\_crossentropy’. This was the function selected because it works well in classification situations where the class is determined based on relative probability.

E3b.

The optimizer chose was ‘adam’ or Adaptive Moment Estimation. This was chosen as adam is often considered the default option for choosing an optimizer, as it is fast and robust. Since I had no crucial reason to switch off of the default option, I kept it as is.

E3c.

The learning rate is used in conjunction with the optimizer to speed up convergence. Since I already was using the default optimizer option, I decided to keep the learning rate as the default option as well inside the optimizer.

E3d.

As part of the fitting the model, I used the function called EarlyStopping to help with the stopping criteria. I decided to use a patience level of 5. This should help prevent overfitting in the model. This forces the model to stop if the validation loss is increasing 5 times in a row. The screenshot below shows that at epoch 8, the validation loss is 0.1337, then 0.1303 at epoch 9, but rises to 0.1408 at epoch 10. And we can see that for epochs 11, 12, 13, and 14 the validation loss keeps increasing, and with patience level of 5 it stops after the 14th epoch.

A screenshot of a computer

AI-generated content may be incorrect.

E4.

The confusion matrix is shown below.

A chart of different types of food

AI-generated content may be incorrect.

From the confusion matrix, we can see that the model appears to struggle the most with predicting the Loose Silky-bent vs. the Black-grass. The model also appears to predict the Common Chickweed very often. This allows the model to have a high correct pick of the Common Chickweed at 88 correct labels, but it also lends it to being wrong often, as we can see with the section of 14, 5, 8, and 4 incorrect labels. Similarly, the model also appears to predict the Loose Silky-bent very often. This allows the model to have a high correct pick of the Loose Silky-bent at 86 correct labels, but it also lends it to being wrong often, as we can see with the section of 28, 0, 0, 1, 19, and 11 incorrect labels.

Part IV: Model Evaluation

F1a.

As part of the fitting the model, I used the function called EarlyStopping to help with the stopping criteria. I decided to use a patience level of 5. This should help prevent overfitting in the model. This forces the model to stop if the validation loss is increasing 5 times in a row. The screenshot below shows that at epoch 8, the validation loss is 0.1337, then 0.1303 at epoch 9, but rises to 0.1408 at epoch 10. And we can see that for epochs 11, 12, 13, and 14 the validation loss keeps increasing, and with patience level of 5 it stops after the 14th epoch.

A screenshot of a computer

AI-generated content may be incorrect.

F1b.

When I created my confusion matrix, I also decided to create a classification report to see how the trained model performed on the validation set. The report is shown in a screenshot below.

A screenshot of a computer screen

AI-generated content may be incorrect.

Here we can see that the accuracy of the model is 0.71, which is moderately successful for a classification. We can also see that the model performed poorly on F1-score for Black-grass as expected.

F1c.

The training loss versus the validation loss is plotted and shown below.

A graph with blue line and red line

AI-generated content may be incorrect.

From this plot, we can see that the loss graphs really divide around the epoch of 8, which is consistent with the around the time that the patience level comes into play.

F2.

The overall fitness of the model can be represented in the screenshot below.



Here we can see that the model has a test accuracy score of approximately 70.7%. Thus, the model seems to be acceptable at predicting and classifying plants. The overfitting issues were addressed in the dropout layer, as well as in the stopping criteria of patience level 5.

F3.

As mentioned above, the model scored an accuracy rating of 0.71, or 71% according to the classification report. Rounding is the cause of the slight discrepancy between the classification report, and the test accuracy number generated in part F2.

Part V: Summary and Recommendations

G1.

The code to save the model is shown in the screenshot below.

A close-up of a white card

AI-generated content may be incorrect.

G2.

The networks functionality is adequate. The neural network performs well for 8 out of the 12 classifiers. It struggles to correctly identify Loose Silky-bent, Black-grass, Common Wheat, and Common Chickweed. The large issue with Common Chickweed is that the model overpredicts that plant type, while severely underpredicting Common Wheat. This leads to a very low F1-score for the Common Wheat, while also giving a false sense of correctness for the Common Chickweed, as showcased by the 0.96 recall score. The model also has some trouble with differencing between the Loose Silky-bent and the Black-grass. Despite these issues, the model still graded out well with an overall accuracy of approximately 71%.

The neural network architecture helped contribute to the effectiveness of the model. As mentioned in Part E2B, the convolutional layers helped detect patterns in the data such as edges, textures, or shapes. The pooling layers helped prevent overfitting in the model as well as reduced the size of volume which helped reduce memory and increase computational speed. The flattening layer turned the feature maps into a 1-dimensional vector. This allowed them to be passed into a linked layer for categorization. The dropout layer helped decrease the nodes which then helped decrease overfitting. The first dense layer takes the input from the flattening layer and computes the final classification. The last layer is the 2nd dense layer, which converts the output of each class into the probability score of each class. All of these worked together to create the working model.

G3.

I would say that the model is acceptable at answering the business question proposed in A1. There are some clear struggles that the model has, but it also is very effective for some types of plants. Overall, I would say that this model can be beneficial to botanists or scientists, but the predictions should be taken with a grain of salt, especially if predicting one of the 4 plants listed in G2.

G4.

In a real-life scenario, the model could be improved by more data. The model could also be improved by focusing less on certain types of plants and spread the distribution of training out across the rest of the plant types. This could also play into the model performing more accurately.

G5.

I would recommend that the model be implemented, but with supervision. As there are known areas of issues, it is important to take those classifications into account. This model could be used to help verify, or speed up classification of plant types, but the model could also still be improved upon if the time, resources, and cost for the company allows for it.

Part VI: Reporting

H.

Copy of code used to save the train network within the neural network and the output is provided in a PDF.

I/J.

GeeksforGeeks. (March 3rd, 2025). *Optimization Rule in Deep Neural Networks.* Retrieved March 18th, 2024,From <https://www.geeksforgeeks.org/optimization-rule-in-deep-neural-networks/>

GeeksforGeeks. (October 10th, 2025). *Introduction to Convolution Neural Network.* Retrieved March 18th, 2024,From <https://www.geeksforgeeks.org/introduction-convolution-neural-network/>

KGP Talkie. (April 30th, 2023). *Deep Learning Tutorial 4 – Cats vs Dogs Classification using CNN & Image Augmentation.* Retrieved March 14th, 2024,From <https://www.youtube.com/watch?v=7Dbd5DV3Jxk>

Scikit-learn. (n.d.). *ConfusionMatrixDisplay.* Retrieved March 24th, 2024,From <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.ConfusionMatrixDisplay.html>

TensorFlow. (n.d.). *Data Augmentation.* Retrieved March 20th, 2024,From <https://www.tensorflow.org/tutorials/images/data_augmentation>

upGrad. (March 3rd, 2025). *Basic CNN Architecture: A Detailed Explanation of the 5 Layers in Convolutional Neural Networks.* Retrieved March 20th, 2024,From <https://www.upgrad.com/blog/basic-cnn-architecture/>