Assignment 4.1: Building a Plant Classification Model Using Convolutional Neural Networks

Patricia Enrique

Department of Engineering, University of San Diego

AAI-511: Neural Networks and Learning

Prof. Esnaeilli

July 24, 2023

Methodology

The dataset used to build a plant classification model with convolutional neural networks is downloaded from Flavia and processed by placing the images into their corresponding class folder. Next the dataset is divided into training, validation, and testing datasets in a 70:20:10 ratio respectively. Each dataset contains the 32 classes with 1323 files belonging to the training set, 371 files in the validation set, and 213 files in the testing set (TensorFlow, 2023).

The first layer defined for the model is the processing layer, where the data is converted to grayscale, resized to a fixed size of (32,32), and normalized. Then the pooling layers are structured with the rectified linear unit activation function, followed by the dense layers. The model is created by combining all the layers before being compiled using the Adam optimizer and the categorical crossentropy loss.

The model is then fitted to the training data and evaluated using the validation data on accuracy, precision, and recall. The accuracy is then visualized on a graph (Medium, 2023).

To improve the performance of the model, different parameters are changed, and the resulting metrics are analyzed. The parameters chosen to modify are the number of pooling layers, the learning rate, and the regularization technique. The model is evaluated using 1, 2, 3, and 4 pooling layers starting with 16 filters and doubling the amount for each layer. The learning rates used are 0.001, 0.01, and 0.1 and the three regularization techniques used are Kernel, activity, and bias.

The final model is created from the parameter trends and evaluated using the test dataset.

Results

The results from experimenting with different parameters in the neural network are summarized in the tables that follow. From Table 1 it can be seen that the performance of the models is similar when

comparing loss and accuracy. However, as the layers increase, the precision decreases and the recall increases.

Table 1

Exploration of pooling layers

Performance Metric	1 Layer	2 Layers	3 Layers	4 Layers
Loss	0.6673	0.5942	0.6810	0.6845
Accuracy	0.7736	0.8194	0.7817	0.7951
Precision	0.5476	0.3326	0.1011	0.0923
Recall	0.5580	0.7951	0.9973	1.0000

Table 2 summarizes the results from experimenting with different learning rates. A learning rate of 0.1 performs worst overall and the learning rates of 0.001 and 0.01 achieve similar metrics. The model with a learning rate of 0.001 slightly outperforms when comparing the precision, and the model with a learning rate of 0.01 slightly outperforms when comparing the recall.

Table 2

Exploration of learning rates

Performance Metric	0.001	0.01	0.1
Loss	0.5030	0.6182	3.4671
Accuracy	0.8275	0.8491	0.0404
Precision	0.4452	0.2909	0
Recall	0.6900	0.8922	0

Table 3 summarizes the results from experimenting with different regularization techniques. The three regularizations apply both L1 and L2 regularization penalties. The bias regularization produces the best loss and precision metrics, while the activity regularization produces the best accuracy and recall metrics. The kernel regularization performs the worst overall but has similar recall performance as the activity regularization.

Table 3

Exploration of activation functions

Performance Metric	Kernel_regularizer	bias_regularizer	activity_regularizer
Loss	1.9814	0.6012	1.4965
Accuracy	0.6092	0.7898	0.8248
Precision	0.0880	0.4272	0.1442
Recall	0.9730	0.7278	0.9784

The final model uses the trends found from the parameter exploration to optimize the performance. The model consists of two layers with 16 and 32 filters respectively, a learning rate of 0.001, and bias regularization. The training and testing performances are summarized in Table 4.

Table 4

Final model performance metrics

Performance Metric	Train	Test
Loss	0.2685	0.5085
Accuracy	0.9237	0.8216
Precision	0.3995	0.4028
Recall	0.8980	0.8075

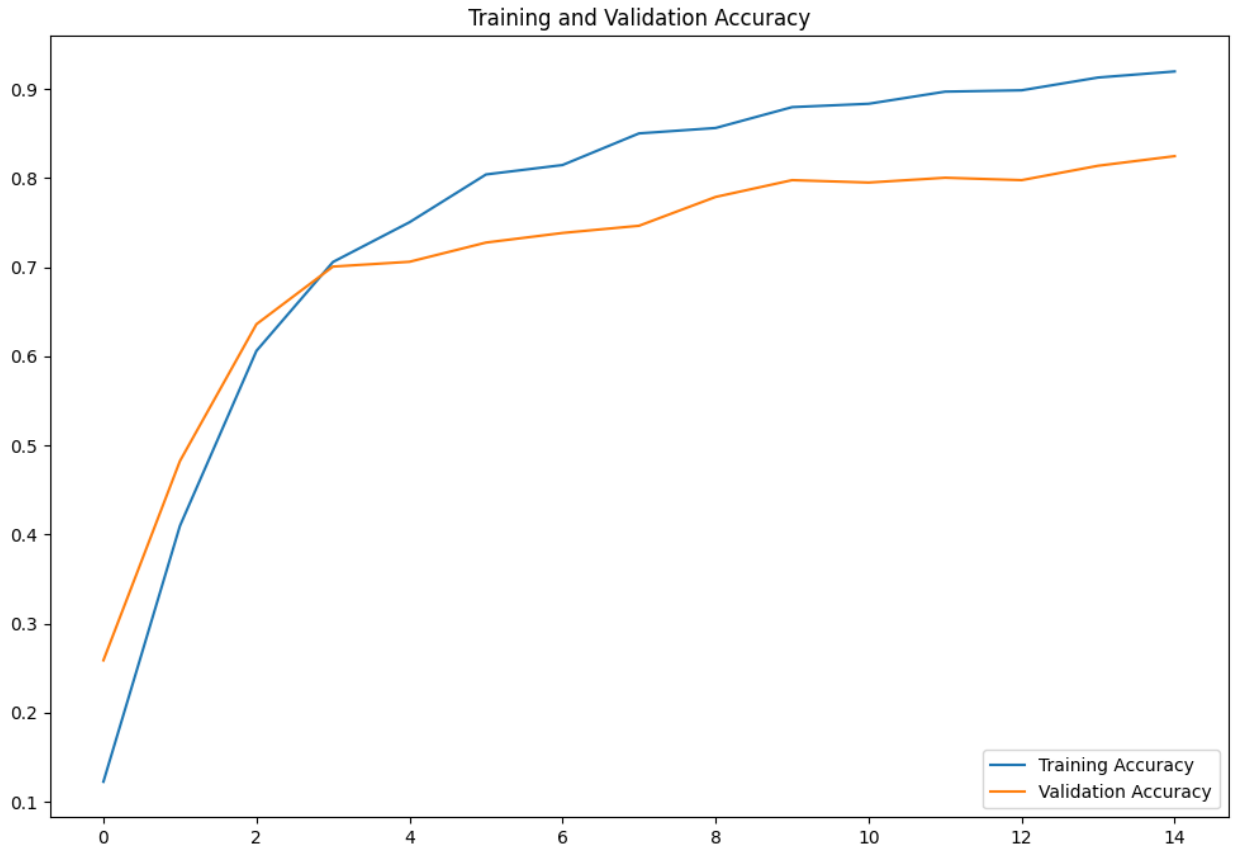
Analysis

The performance metrics remain consistent between the training and the testing dataset indicating that the model is not overfitting or underfitting to the data it is trained on. The accuracy for the model is above 80% for both the training and the testing dataset indicating that the model is able to generalize the predictions to combinations of features it has not yet seen. Precision is the worst performing metric; however, it is similar between training and testing, meaning that when predicting a specific plant species, the model is correct about 40% of the time. The recall metric has an above 80% performance for both datasets indicating that the model can identify the correct class for a leaf image.

The accuracy is the performance metric that has consistently the largest discrepancy between the training dataset and the validation dataset or between the training dataset and the testing dataset. An example of this comparison is shown in Figure 1 which is gathered from the model exploring the effect of the activity regularization. It can be seen that the validation accuracy closely follows that of the training accuracy until an epoch of about 2 or 3 and then the difference between the two increases.

Figure 1

Accuracy comparison between training and validation

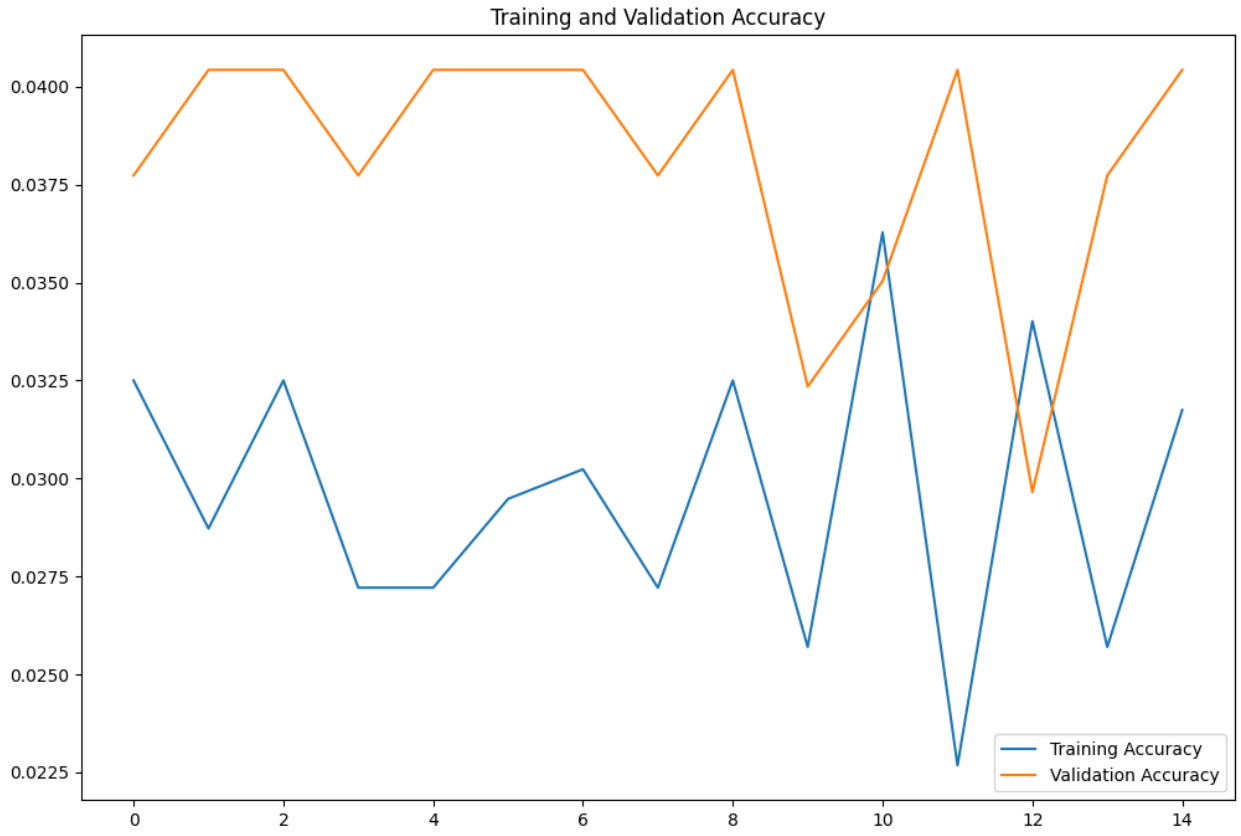


The only exception to the accuracy comparison trend is the model that uses a learning rate of 0.1. The accuracies of both the training and validation sets for this model are shown in Figure 2. From this graph it can be seen that the weights of the convolutional neural network are not being updated sufficiently to

allow for learning to occur. As the learning rate is too large, the model can be converging to a solution that is not optimal too quickly.

Figure 2

Accuracy comparison between training and validation with a learning rate of 0.1



Limitations and Improvements

The parameters explored in this analysis are a few of the ones available. For example, further exploration into the effects of the filters used for each layer as well as increasing or decreasing the filters can be completed. The performance of the final model created is acceptable and does not require in-depth parameter tuning to obtain.

The consistently low precision performance suggests that the dataset is imbalanced and that there are classes with few examples. A technique that can be used to mitigate this issue is oversampling and under-sampling the classes during the preprocessing stage.

References

Medium. (2023, July 22). Image Classification with TensorFlow. https://medium.com/analytics-vidhya/image-classification-with-tensorflow-2a406bdfe0c1

TensorFlow. (2023, July 22). Convolutional Neural Network (CNN). https://www.tensorflow.org/tutorials/images/cnn

TensorFlow. (2023, July 22). tf.keras.utils.image_dataset_from_directory. https://www.tensorflow.org/api_docs/python/tf/keras/utils/image_dataset_from_directory