Leaf Image Classification using Deep Learning in Tensorflow with Automatic Model Selection using Keras Tuner

Shihab Hamati

June 12, 2023

1 Introduction

This project creates a fast and scaleable image classifier model for 11 different types of plant leaves. The model is a deep convolutional neural network whose optimal parameters are selected automatically from a set of possible hyperparameters. The model is trained over 58 epochs and achieves 98% accuracy on the test set composed of 55 images uniformly distributed across the classes.

2 Dataset

2.1 Description and Subset Selection

The dataset consists of 7 GB of images of leaves of 11 different plant species. They contain both healthy and diseased images. For this project, only the healthy classes were used to train the model to distinguish among the species. The training split consists of 2,163 and 55 images for each of the validation and testing splits. These are all high quality 24 MP images (i.e., 6000x4000 pixels).

The dataset is downloaded from Kaggle using a JSON token file for authentication. This dataset is then unzipped using Linux commands into the local Colab storage.

2.2 Reducing the Memory Footprint for Speed

The images are then downsized and saved for use thereafter. This significantly reduces the memory footprint and dramatically increases the model's training time. Combined with Tensorflow's `AUTOTUNE` function which is explained later, these two interventions reduce the epoch training time from about 10 minutes to about 6 to 7 seconds - so training this model for 58 epochs would have taken about 10 hours instead of about 6 minutes. In the GitHub repository folder `Model_1` there exists the slower model.

Lastly, as these preprossessing steps take about around 20 minutes, the preprossessed dataset is saved on Google Drive and can be directly used to assess, train, or explore the CNN without incurring again every time the overhead preprocessing time to achieve the same final preprocessed dataset.

3 Model

3.1 Convolutional Neural Network (CNN)

A CNN is a deep learning neural network composed of consecutive convolution blocks followed by a dense layer and an output layer. It utilize the power of convolutions to scan and identify for spatial features across an image input.

3.1.1 Data Augmentation

All the images are first converted to a unified size of `256x256` by the input layer. Prior to passing the images to the neural network, multiple random augmentations are applied to them to prevent the model from overfitting to the training dataset. This has a similar effect of artifically appearing to increase the available training samples as well for the model to focus on learning the characteristic features rather than memorizing over the multiple training epochs.

3.1.2 Convolutional Blocks

The convolutional blocks consititute of a convolutional layer which learns multple filters that are convolved with the input from the previous layer (starting with the original image). The size of these kernels in this project as `3x3`. The nodes are activated or not based on a ReLu activation function. Following this layer is a Max Pooling layer with the default size `2x2`. This layer downsample the convolutional layer out by replacing every 4 pixels with the maximum value among them. Lastly, there is a Dropout layer that randomly turns off a proportion of the computed nodes during each training epoch. This prevents the model from overfitting.

3.1.3 Dense and Output Layers

The output of the final conolutional block is flatted and passed on to a single fully connected (dense) layer. This layer is finally connected to another dense layer with the same number of nodes as the number of classes. The activation function for this layer is a `softmax`, giving the probability of the image being each of the 11 classes. The most likely class is the label assigned to an image.

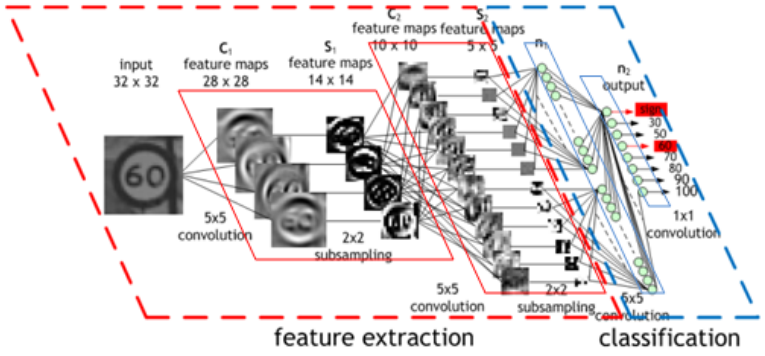

Figure 1: Outline of a Convolutional Neural Network

3.2 Hyperparameter Tuning

The Keras Tuner package was used to create a `tuner` instance. Its `search()` method was passed a custom function `make_model()` containing a range of values for some hyperparameters to optimize the validation accuracy from.

The image size used is `256x256` and the architecture consists of 3 sequential 2D convolutional blocks followed by a dense layer and output layer. Each convolutional block consists, in order, of a convolution layer, a ReLu activation layer, a MaxPooling layer, and a Dropout layer. The dense layer is preceeded by a Flatten layer and proceeded by an output layer of 11 nodes (matching the number of classes) with a Softmax activation.

The hyperparameters optimized by Keras Tuner are

- `DROPOUT`: values between 0.1 and 0.3 for the convolutional layers and between 0.3 and 0.5 for the dense layer, tuneable independently. This is the number of nodes randomly turned off at each training pass. It serves to reduce overfitting.

- `FILTERS`: integer values for each convolutional layer, ranging from 16 to 32 spaced 8 numbers away, depending on the layer, and tuneable independently for each layer. These values are added cumultively on top of the previous layer's to somehow mimic a VGG inspired architechture of progressively higher number of convolutional filters.
- `UNITS`: integer values between 8 and 32 for the number of nodes in the per-output dense layer, spaced 8 numbers away. The maximum number of units is capped at 32 to avoid a blowing number of trainable parameters for the model. This could lead to overfitting. Furthermore, by Occam's razor, a simpler model with a similar accuracy is preferred.
- `OPTIMIZER`: a choice between `adam` and `sgd` for the compiler's learning rate approach.

3.3 Trials Result

The `search()` function ran 10 trails over the hyperparameter for 10 epochs each. The model was trained using the optimal hyperparameters found in the automatic model selection trials:

- Convolutional Layer 1: `24` filters
- Convolutional Layer 2: `48` filters
- Convolutional Layer 3: `64` filters
- Pre-Output Dense Layer: `32` units
- Dropout Rate, Convolutional Blocks: `0.1`
- Dropout Rate, Pre-Output Dense Layer: `0.4`

4 Performance

The `fit()` function is called to run up to 100 epochs. An `EarlyStop` callback is passed to it to allow for stopping the training process after the 50^{th} epoch if no improvement is achieved to the validation accuracy for 5 consecutive epochs. This leads it to stop at the 58^{th} epoch. The parameters of the best model are restored. The model exhibits learning as observed by the improving training accuracy (and decreasing loss). Furthermore, there is not sign of overfitting as the validation curves tracks the training's.

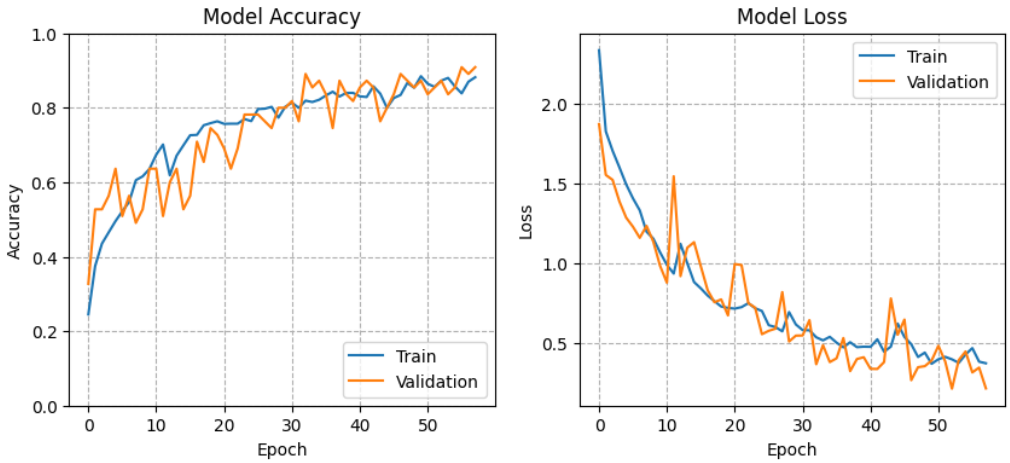

Figure 2: Training History of Accuracy and Loss Curves

Tested on an unseen dataset, it achieves an accuracy of 98%. The classification report exhibits excellent performance across all metrics, and the confusion matrix visually exhibits the excellent performance of the model. It also indicates no single class that is systemically misclassified.

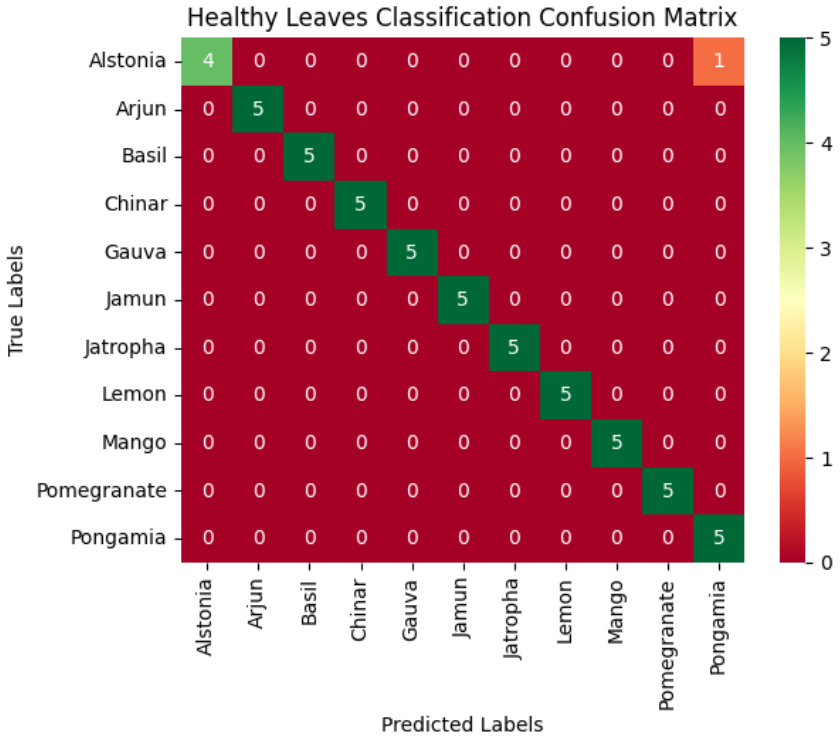

Figure 3: Confusion Matrix on Test Dataset

Naturally with this high rate of accuracy and unprobelmatic confusion matrix, the classification report computes excellent values across all classification metrics.

	precision	recall	f1-score	support
Alstonia Scholaris healthy (P2b)	1.00	0.80	0.89	5
Arjun healthy (P1b)	1.00	1.00	1.00	5
Basil healthy (P8)	1.00	1.00	1.00	5
Chinar healthy (P11a)	1.00	1.00	1.00	5
Gauva healthy (P3a)	1.00	1.00	1.00	5
Jamun healthy (P5a)	1.00	1.00	1.00	5
Jatropha healthy (P6a)	1.00	1.00	1.00	5
Lemon healthy (P10a)	1.00	1.00	1.00	5
Mango healthy (P0a)	1.00	1.00	1.00	5
Pomegranate healthy (P9a)	1.00	1.00	1.00	5
Pongamia Pinnata healthy (P7a)	0.83	1.00	0.91	5
accuracy			0.98	55
macro avg	0.98	0.98	0.98	55
weighted avg	0.98	0.98	0.98	55

Figure 4: Classification Report

Declaration

I declare that this material, which I now submit for assessment, is entirely my own work and has not been taken from the work of others, save and to the extent that such work has been cited and acknowledged within the text of my work. I understand that plagiarism, collusion, and copying are grave and serious offences in the university and accept the penalties that would be imposed should I engage in plagiarism, collusion or copying. This assignment, or any part of it, has not been previously submitted by me or any other person for assessment on this or any other course of study.