Project of Algorithms for Massive data

Plant leave recognizer

Name: Giorgio Degl'Innocenti
MSc: Data science and Economics
giorgio.deglinnocenti@studenti.unimi.it

September 14, 2023

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.

Abstract

The goal of this project is to implement a deep-learning-based system classifying pictures of leaves according to the twelve types of plants.

The project is designed to work on Google Colab, downloading automatically the chosen dataset from Kaggle, preprocessing it, building a tuner to select the best set of hyperparameters between the one chosen and finally fitting and evaluating the best model on the test set.

Contents

1 Introduction

The project deals with a classification problem, in particular, the goal is to correctly classify eleven different type of leaves based on the Kaggle dataset Plant Leaves for Image Classification.

In this project, our objective is the creation of an automated tuning system. This system will assist us in identifying the optimal combination of parameters to enhance model performance. Subsequently, we will employ this refined model for training and evaluation on the test dataset.

2 The Dataset

The dataset used, as already said, has been the Kaggle dataset Plant Leaves for Image Classification. The original dataset is composed of eleven different types of leaves from different plants and for each plant, we find images of both healthy and diseased leaves. The eleven plants that we can find are Mango, Arjun, Alstonia Scholaris, Guava, Jamun, Jatropha, Pongamia Pinnata, Basil, Pomegranate, Lemon, and Chinar.

For the project it has been chosen to use only the healthy leaves, hence discarding the diseased part. In total the dataset was composed as such:

1. Train set: 2163 images
2. Validation set: 55 images
3. Test set: 55 images

3 Preprocessing

The preprocessing of the dataset two main things have been made:

1. Discarding the diseased leaves images
2. Resizing the images to a smaller suitable format

The resizing part was done through a function that looped over the whole dataset, resizing all images to a predetermined size (equal to 256), the whole process took around 15/20 minutes to run, after the dataset had been saved on the drive to ease the test part after.

4 Model

As the standard setting the model used to classify images has been a CNN, created using the Tensorflow library for Python. The model is composed of 3 convolutional layers and 1 dense layer and it has been tuned, using the RandomSearch function, according to certain hyperparameters listed as follows:

1. Learning rate : [1e-2, 1e-3]
2. Number of filters for the 1st layer : [24,32]
3. Number of filters for the 2nd layer : [24,32]
4. Number of filters for the 3rd layer : [24,32]

In the following image we can see how the model has been structured:

```
model=Sequential()
model.add(Input(shape = (size, size, 3)))
model.add(RandomRotation(0.2))
model.add(RandomBrightness(0.2))
model.add(RandomContrast(0.2))
model.add(Conv2D(filter_layer_1,kernel_size = (3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2, seed = 42))
model.add(Conv2D(filter_layer_2,kernel_size = (3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2, seed = 42))
model.add(Conv2D(filter_layer_3,kernel_size = (3,3),activation='relu'))
model.add(BatchNormalization())
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.2, seed = 42))
model.add(Flatten())
model.add(Dense(32))
model.add(Dropout(0.5, seed = 42))
model.add(Dense(11,activation='softmax'))
model.compile(loss='sparse_categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])
return model
```

Figure 1: Model structure

As shown we have an initial part of data augmentation with 3 layers adding random rotation, brightness, and contrast. This layer usually serve two main purposes: To improve the learning capabilities of the model mimicking some real-world variability and to prevent overfitting as data augmentation layers increase the dataset size by applying various transformations.

We have then a normal Convolutional layer which has a kernel size of 3x3 and as activation function the Relu function.

It follows a Batch Normalization layer which is typically used to normalize the weights and prevent the problems of vanishing and exploding gradients and also as a form of regularization to prevent overfitting.

Subsequently, there is a MaxPooling layer that is used to reduce dimensionality (which is needed as this downsampling reduces computational complexity in deeper layers and helps manage memory requirements during training) and to introduce translation invariance in the network as it makes the model more robust to small translations or variations

in the position of the features. The model had a MaxPooling size of 2x2 (therefore taking the maximum values in a matrix 2x2).

As the 'last' layer of the convolution part there is a dropout layer which acts as another form of regularization. By preventing any single neuron from becoming too specialized or dominant, it discourages the network from memorizing the training data and so overfits. For the model it has been chosen a 0.2 percentage of dropout in the convolutional part and 0.5 in the dense one.

As usual for this type of tasks, it then follows a flatten layer to transit to a fully connected layer with 32 neurons and after another dropout layer we have another dense layer with 11 neurons used to classify.

Further details can be seen in model.compile() as the loss function (the Sparse Categorical Crossentropy), the optimizer (Adam) and the metrics to optimize (accuracy).

The tuner part as seen has been runned over the learning rate and the number of filter of the different convolutional layers, the tuner did a random search over the different parameters (for a maximum of 10 trials) with the goal to maxime accuracy and running each trial for 10 epochs. The best trial got an accuracy on the validation set of 0.8%, with the following parameters:

1. Learning rate = 0.001
2. Number of filters for the 1st layer = 32
3. Number of filters for the 2nd layer = 24
4. Number of filters for the 3rd layer = 32

These hyper-parameters have been used to finally build the model that presented a total of 937.123 parameters. The final model has been then run over 50 epochs achieving a final accuracy on the validation set of = 0.964%.

The following image show the loss and accuracy history of the model:

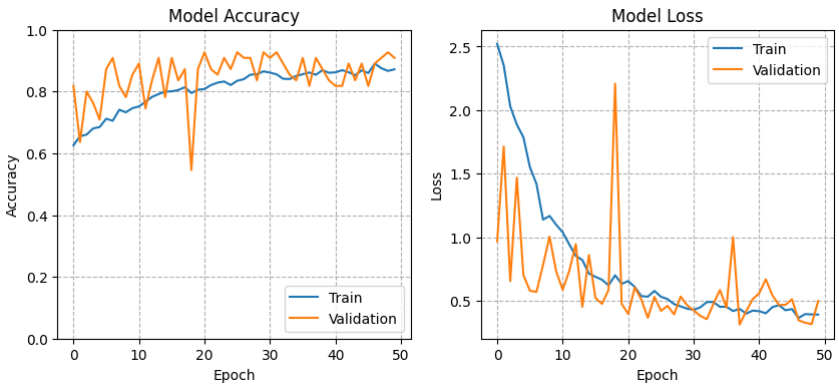

Figure 2: Loss and accuracy history

5 Results

The model has been finally tested on the test set achieving the following results:

- Test Loss: 0.15%
- Test accuracy: 0.964%

It has then produced a confusion matrix showing the correct and incorrect classifications:

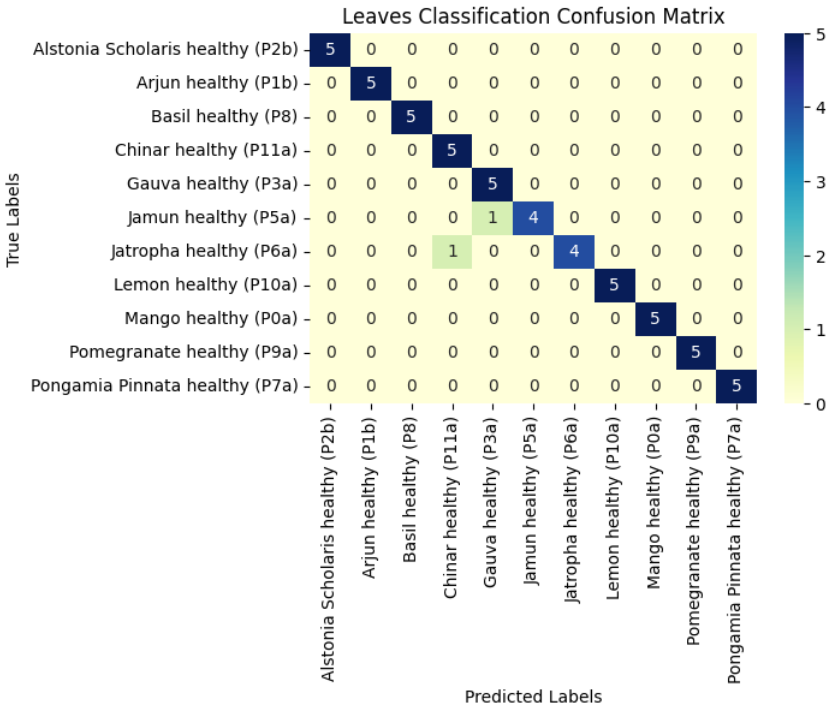

Figure 3: Confusion Matrix

The model correctly classifies all images but for 1 Jamun leave classified as Guava and a Jathropa leave classified as a Chinar.

6 Conclusion

Ultimately, we have achieved success in crafting a deep learning system. Following a tuning phase, this system effectively learned to identify different types of leaves.

Potential avenues for further development include refining the system through more detailed tuning, given the relatively limited exploration of settings thus far. Another option is expanding the system's capabilities to handle the added complexity of identifying diseased leaves.