Plants leaves classification

Matteo Tarenzi n°986974

Abstract

This work consists in the creation of two Convolutional Neural Networks used for images classification. It is divided into two parts: the classification of plant types and the identification of the health status of the leaves. For each of the two parts, a deep learning algorithm has been trained over a data set that has been changed based on the purpose of the network. Finally, the models have been tested and the results have been plotted and described.

Introduction

The project was born from the idea that not everyone has the capability or the knowledge to be able to identify a certain type of plant. It might be helpful then to construct a model that is able, given an image of one leaf from that plant, to tell the type and suggest information. This model could be used by curios or by who is trying to grow something at home or in his garden. For this purpose, associated with the model that suggests type, it has been consider to construct another one that can classify leaves into healthy or diseased. This can help people understand if their plants are in good condition or there are some of them that might need care.

The data set

The data set has been obtained from *Kaggle*[1]. It has been downloaded using the *Kaggle* API through Python code. Was already divided into three folders: the training one, the validation one and the test one. Each of them contains images of different leaves, grouped on the base of two criteria: the type of the plant from which they come and the health status of the leaf. Inside each folder, there were images from ten types of plants that had both healthy and diseased leaves, and two types (Bael and Basil) that had respectively only diseased or healthy ones . This is the list of all the twenty two groups:

- Alstonia Scholaris diseased & Alstonia Scholaris healthy
- Arjun diseased & Arjun healthy
- Bael diseased
- Basil healthy
- Chinar diseased & Chinar healthy
- Gauva diseased & Gauva healthy
- Jamun diseased & Jamun healthy
- Jatropha diseased & Jatropha healthy
- Lemon diseased & Lemon healthy
- Mango diseased & Mango healthy
- Pomegranate diseased & Pomegranate healthy
- Pongamia Pinnata diseased & Pongamia Pinnata healthy

Try to classify twenty two classes could be very hard, thus, the project has been divided into smaller classification problems. For each of these tasks, the directories have been adjusted and new classes have been created. The first task deals with the classification of the type of plants collected in the data set, while the second task is focused on the classification of the plants health. Both of the two topics will be analyzed deeper in the dedicated chapters.

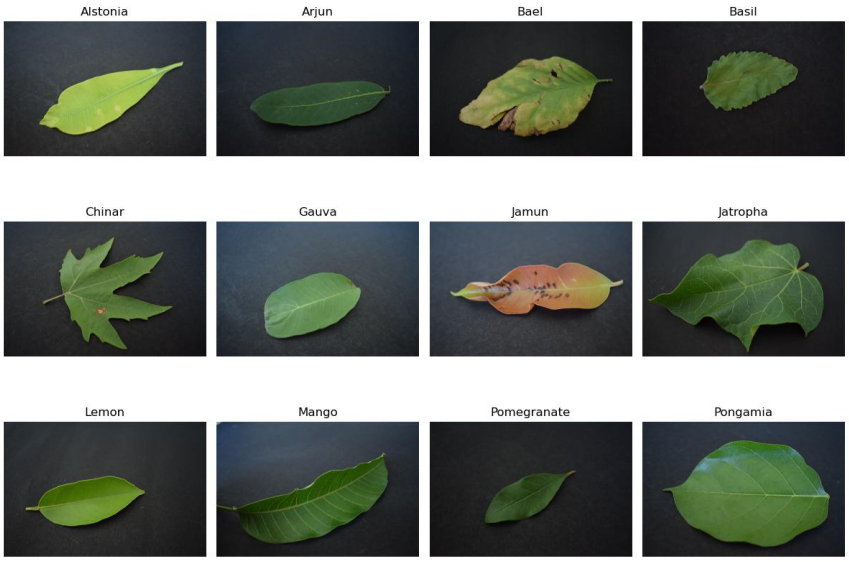
[1] https://www.kaggle.com/datasets/csafrit2/plant-leaves-for-image-classification

First part: "Type classification"

This part aims to build a model able to recognize the plants type using the images of their leaves. For doing that, it has been created a Convolutional Neural Network that has been trained with the images downloaded from *Kaggle*. The model will use both healthy and diseased photos to classify the type. This could decrease the final performance of the model when tested over a test set but the model would benefit in the future if applied to reality. It has to be consider, in fact, that if applied to the real world, a model that has considered only healthy leaves will perform worse than one that has been already trained over both healthy and diseased ones. For this first part, the health status will not be considered.

Data preparation

Since the labels of the original data set were not suitable for this task, it was firstly necessary to group the images of healthy and diseased plants. This can be done manually or using code[2]. Once that the folders have been adjusted, each of the 3 main folders (train,validation and test) contained twelve classes of images:



[2]This project has used code through the os package in Python

The implementation of the model

The model that has been built is a Convolutional Neural Network. It has been implemented using *Tensorflow* package within *Python*. Its architecture is shown in **Table 1**.

First, since the number of classes were quite high and the data set was unbalanced, it has been used a batch size of 128 that should allow to have inside the batches at least some images from each of the classes . For validation and testing, instead, the use of batches has not been applied. Images have been shuffled when loaded and then the batches have been shuffled again among themselves. After that, the images have been resized to a dimension of (64,64) and the measures of the three color channels have been normalized. Three augmentation layers have been introduced to deal with overfitting. They are shown below, in details: a Zoom, a flip, and a rotation.

Data augmentation layer

```
data_augmentation = keras.Sequential(
 [
   layers.RandomZoom(0.2),
   layers.RandomFlip("horizontal_and_vertical"),
   layers.RandomRotation(0.1)
 ])
```

The rest of the model is made of three convolutional layers and seven fully-connected layers, considering also the final one used for the classification. The Neural Network uses a combination of *relu* and *tanh* activation functions that seems to perform nicely. The parameters inside the layers has not been object of cross validation, instead, they were manually changed after each training and testing based on the results. The model showed in the **Table 1** is the one with the best learning curves and the best final performance. To deal with overfitting, after each layer it has been introduced Dropout with a value of 0.2.
It has been created a learning rate decaying function that from 0.00125 decrease constantly at every epoch. This is its form:

$$0.00125\text{-}(\text{epoch}*0.0000025)$$

Table 1

Model: model_div

Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 64, 64, 3)	0
sequential (Sequential)	(None, 64, 64, 3)	0
conv2d (Conv2D)	(None, 60, 60, 25)	1900
average_pooling2d	(None, 30, 30, 25)	0
dropout (Dropout)	(None, 30, 30, 25)	0
conv2d_1 (Conv2D)	(None, 26, 26, 50)	31300
average_pooling2d_1	(None, 13, 13, 50)	0
dropout_1 (Dropout)	(None, 13, 13, 50)	0
conv2d_2 (Conv2D)	(None, 9, 9, 75)	93825
dropout_2 (Dropout)	(None, 9, 9, 75)	0
flatten (Flatten)	(None, 6075)	0
dense (Dense)	(None, 450)	2734200
dropout_3 (Dropout)	(None, 450)	0
dense_1 (Dense)	(None, 350)	157850
dropout_4 (Dropout)	(None, 350)	0
dense_2 (Dense)	(None, 240)	84240
dropout_5 (Dropout)	(None, 240)	0
dense_3 (Dense)	(None, 120)	28920
dropout_6 (Dropout)	(None, 120)	0
dense_4 (Dense)	(None, 60)	7260
dropout_7 (Dropout)	(None, 60)	0
dense_5 (Dense)	(None, 60)	3660
dropout_8 (Dropout)	(None, 60)	0
dense_6 (Dense)	(None, 12)	732

Total params: 3143887 (11.99 MB)
Trainable params: 3143887 (11.99 MB)
Non-trainable params: 0 (0.00 Byte)

The results

The figure below shows the learning curves of the training procedure. The loss functions are pictured on the left while the accuracy is shown on the right. The training has been stopped at 400 epochs because, even if the validation accuracy is still improving a bit, the validation loss was flattening. Moreover, even if the validation loss is decreasing, the pattern is a bit messier; it could be smoother.

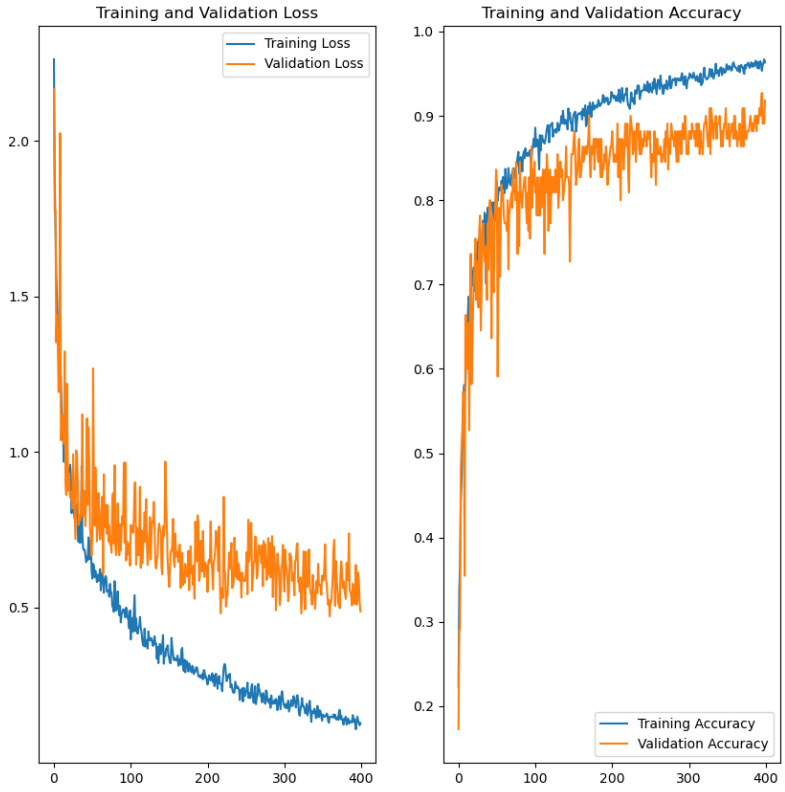


Figure 2

Once that the model has been trained, the performance was evaluated using the images within the test set. The loss and the accuracy of the model are:

Loss: 0.4296 - Accuracy: 0.9273

Furthermore, using the predictions, it has been built a confusion matrix with all the twelve types of plants in the data set. This matrix is shown in **Figure 3**.

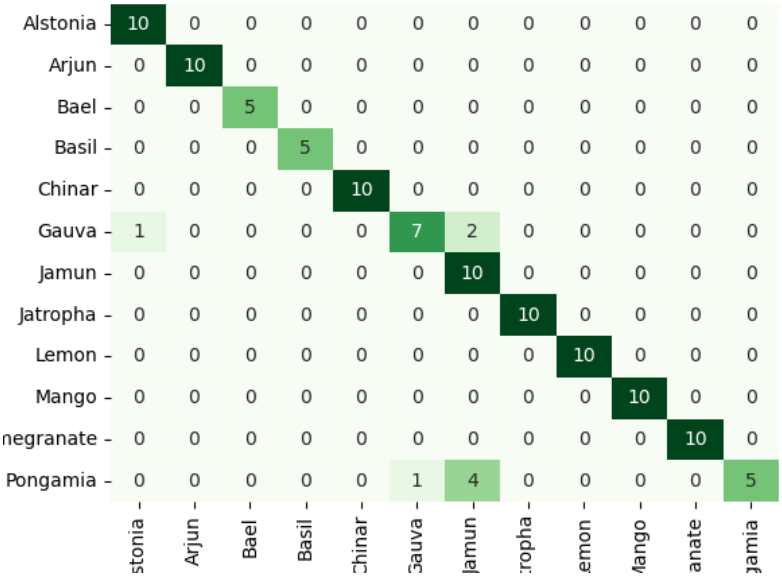
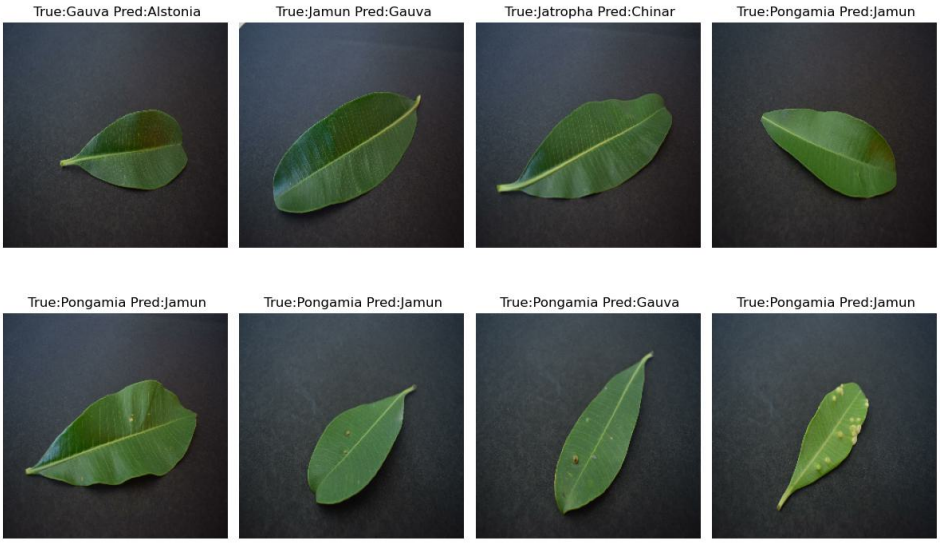


Figure 3

It can be observed that the model makes a good job in classifying the majority of the classes. What creates some problems, and then some misclassification, are three plants: Guava, Jamun and Pongamia. What is happening is that, the model is capable of predictiong the Jamun leaves, but misclassify the Guava and the Pongamia leaves for the ones of Jamun. In particular, the performance for the Pongamia class is so bad that only the 50% of its images have been classified right. **Figure 4** shows the images that have been wrongly classified.

Figure 4



Compared to the photos in the Jamun folder it is understandable why the Pongamia, that is usually characterized by larger leaves, have so many misclassified images. The leaves in **Figure 4** are very similar to the Jamun leaves, the only difference is the veining, something that can be detected only by a more powerful model.

First part conclusions

The results are not completely satisfactory. Even if the model is doing good with a lot of plant types, the results for the remaining categories can not be considered appropriate, resulting in a 93% accuracy. There are four main reasons that can explain this failure.

Model is too simple: the difference between Pongamia and Jamun is not in the shape but in the veining. This might reveal that the model should consider to utilize a greater size of images and improve the convolutional layers both in number and size. Since this project has been implemented on a personal computer, some choices were the effect of computational power and time limits. With more of the two it is surely possible to construct and train a bigger model that will be able to recognize between Pongamia and Jamun.
Poor definition of the learning rate decay: the starting learning rate, even if is the best one recorded, it was a little low. This could lead the loss to a point of relative minimum instead of absolute minimum. Moreover, it could be useful to try different learning rate decay functions instead of the step function utilized in the project.
Construction of the batches: the way the batches were built could be improved. Since the data is unbalanced, it could be useful to use mini-batches with the exact number of images among the different types.
The contents of the training set: the content of the Pongamia folder used for training should be checked for errors in the labeling procedure. The images are very different one from another.

Even if the project has not an high accuracy, it can be considered as a starting point for further studies. All the problems listed before can be take into consideration to improve the model and then the final performance of the classification. One aspect that can be underlined is that the number of images in the training set should be increased. In particular it is suggested to increase the images of the type of plants that are responsible of the misclassification: Guava, Pongamia and Jamun.

Second part: "Plants health"

This second part of the project is focused on the detection of healthy or diseased leaves. Again, the original data set is the one obtained from *Kaggle*, but the classes had to be adapted. This time the type of plant will be ignored and we will focus on the health of the leaves. This implementation could be useful to people who need help in detecting plants that need care and plants that are, in fact, dying.

Data preparation

The *Kaggle* dataset was modified to keep only two classes. For both the training, validation and testing folder, the different types have been merged into two new categories: healthy plants and diseased plants. Again, this can be done manually or via code. Once preparation was complete, each set contained two classes. Within each class there were leaves from different types of plants. The following figure shows some images.

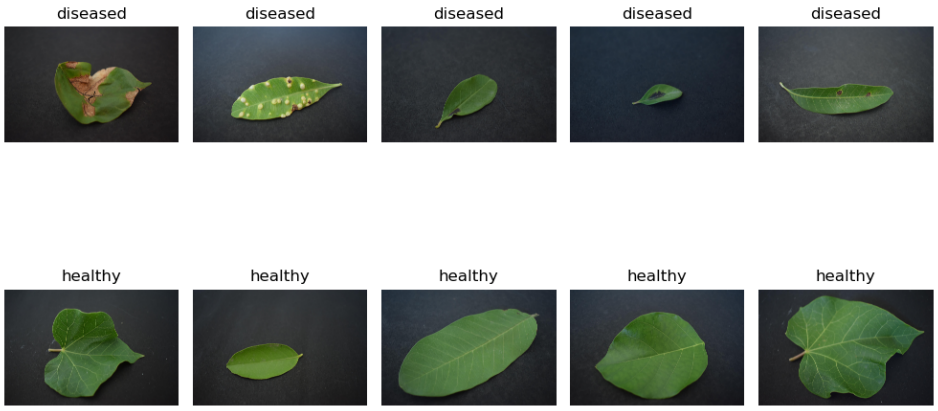


Figure 5

The implementation of the model

It has been used again a Convolutional Neural Network, but this time, the model was more complex. The reason is that while for the types of plants it was a matter of shapes and colors, this time the neural network needed to detect smaller pattern inside the images.

The size of images used for this model was (128,128) but the normalization procedure remained the same. Like is shown below the augmentation layers was composed only by two layers: a flip and a rotation.

Data augmentation layer

```
  data_augmentation = keras.Sequential(
[
  layers.RandomFlip("horizontal_and_vertical"),
  layers.RandomRotation(0.1)
])
```

The rest of the model was made of five convolutional layers with four average pooling layers, and six fully connected layers. Again, the activation functions were a combination of *relu* and *tanh*. Finally, to deal with overfit it has been introduces *Dropout* of 0.2 after each fully-connected or convolutional layer. **Table 2** summarize the architecture of this model.
Also this time, the learning rate decaying function is a step decreasing function, but its steeper than the first one:

0.0012-(epoch*0.000004)

Table 2

Model: "model_ht"

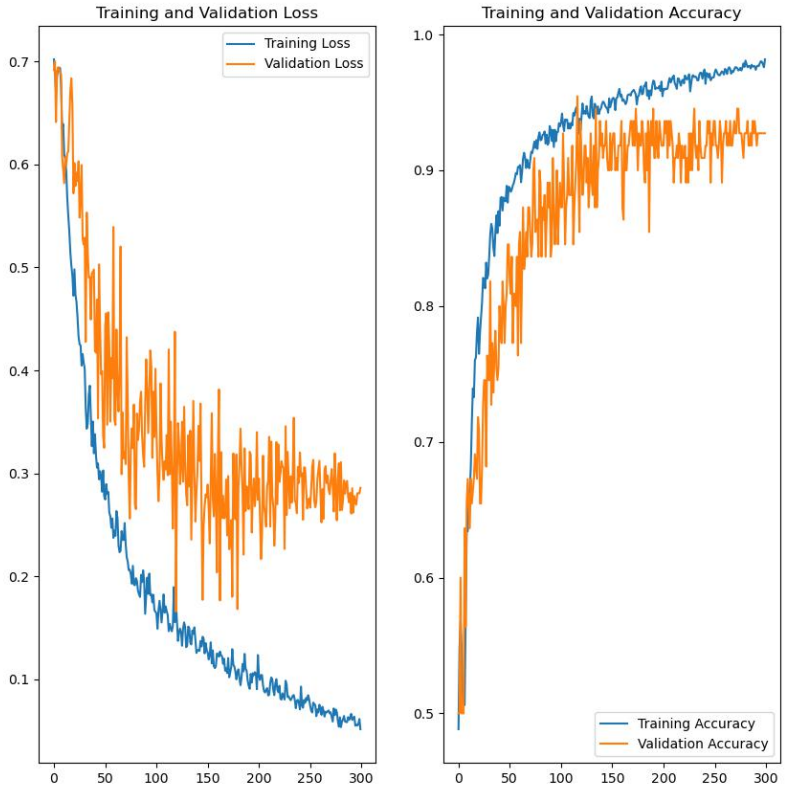
Layer (type)	Output Shape	Param #
rescaling (Rescaling)	(None, 128, 128, 3)	0
sequential (Sequential)	(None, 128, 128, 3)	0
conv2d (Conv2D)	(None, 124, 124, 25)	1900
average_pooling2d	(None, 62, 62, 25)	0
dropout (Dropout)	(None, 62, 62, 25)	0
conv2d_1 (Conv2D)	(None, 58, 58, 50)	31300
average_pooling2d_1	(None, 29, 29, 50)	0
dropout_1 (Dropout)	(None, 29, 29, 50)	0
conv2d_2 (Conv2D)	(None, 25, 25, 75)	93825
average_pooling2d_2	(None, 12, 12, 75)	0
dropout_2 (Dropout)	(None, 12, 12, 75)	0
conv2d_3 (Conv2D)	(None, 8, 8, 100)	140700
average_pooling2d_3	(None, 4, 4, 100)	0
dropout_3 (Dropout)	(None, 4, 4, 100)	0
conv2d_4 (Conv2D)	(None, 1, 1, 100)	90075
dropout_4 (Dropout)	(None, 1, 1, 100)	0
flatten (Flatten)	(None, 75)	0
dense (Dense)	(None, 240)	18240
dropout_5 (Dropout)	(None, 240)	0
dense_1 (Dense)	(None, 120)	28920
dropout_6 (Dropout)	(None, 120)	0
dense_2 (Dense)	(None, 60)	7260
dropout_7 (Dropout)	(None, 60)	0
dense_3 (Dense)	(None, 30)	1830
dropout_8 (Dropout)	(None, 30)	0
dense_4 (Dense)	(None, 12)	372
dropout_9 (Dropout)	(None, 12)	0
dense_5 (Dense)	(None, 2)	26

Total params: 414448 (1.58 MB)

The results

The following figure shows the training procedure. This time it was stopped at 150 epochs, because after that, the validation loss stopped decreasing. As for the previous model, the validation loss is not as smooth as we would like.

Figure 6



Using the test set, the performance of the model has been evaluated. Loss and accuracy are:

Loss: 0.3518 - Accuracy: 0.8909

Then, predictions have been collected to build a confusion matrix. The matrix in **Figure 7** show the number of true positives, true negatives and the number of false positives and false negatives

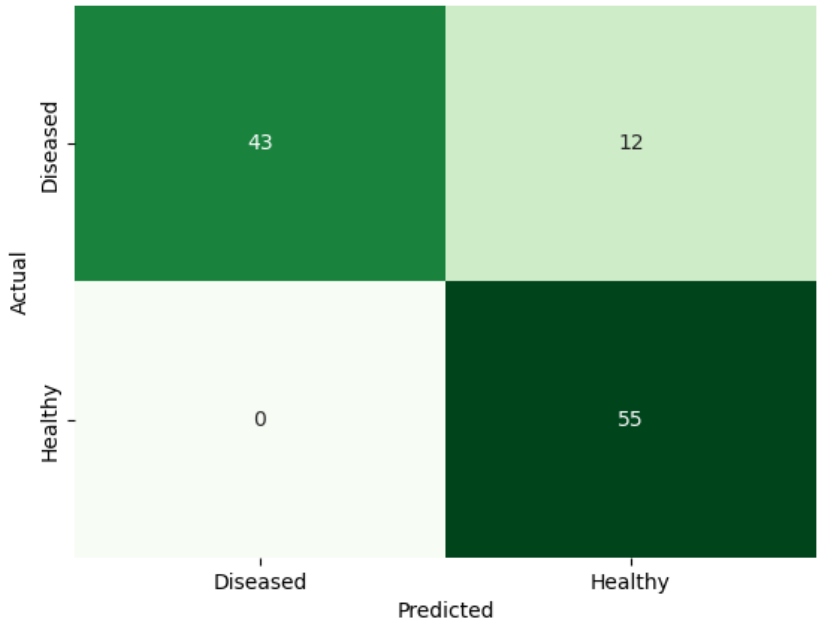
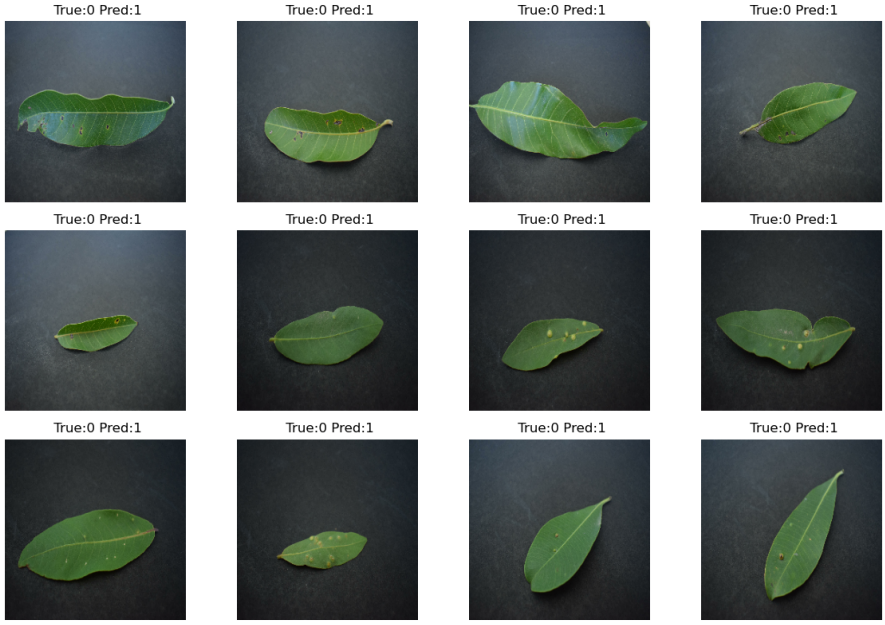


Figure 7

Taking the healthy class as the positive class, it can be observed that all the errors are false positives. In general, the model is good at estimating diseased leaves but it tends to overestimate a lot the number of healthy leaves.

Figure 8 shows the misclassified images within the test set. Looking at the images it is possible to notice why they have been wrongly classified. Although some of them are diseased, they present the same color and shape of healthy ones, with only some small differences like holes or cuts.

Figure 8



Second part conclusions

Once again, the results are not completely satisfactory. The accuracy of 89% is quite low for an image classification problem.

The reason is that using only images for this classification is quite difficult. Although some diseased plants visually change in color and shape, there are other leaves that remain very similar to healthy ones. To complicate the problem even more, exist some healthy leaves with different shapes and veining that reminds of diseased ones.
There are some improvements that can possibly increase the accuracy of the model. First, is essential to study in more detail, maybe with cross validation, the model parameters and its architecture; might not be enough. Once again, the size of the training set should be increased if we want to decrease the validation loss and obtain a better performance. Finally, the learning rate decaying function might need some further studies or some adjustments to make the training procedure even better.

Conclusions

Even if the models have obtained some good results, it is necessary to point out that they need more studies and more improvements to be able to reach the performance that an image classification network requires.
Some of the main problems are the effects of the fact that the project has been implemented on a personal computer that requires a lot of time when training the Neural Networks. The image size, the parameters inside the layers and the learning rate needs some further tuning, possibly using cross validation. Although there are other problems, the biggest one is the dimension of the training set for both of the two algorithms. Associated with the use of mini-batches, the number of images need to be increased by a lot if we want to obtain the expected results. More images might allow to increase the complexity while preserving the model from overfitting. With more complexity, the models could be able to detect the differences that they are now neglecting. For the classification of plants types, the network should be able, with a bigger model, to find the aspects that differentiate the veining in the Pongamia leaves from the one in the Jamun leaves. For the health status, the complexity might help to find those cuts and holes that are associated with diseased plants, stopping the model from overestimating the number of healthy leaves.

In conclusion, the models presented in this project could be seen as a good starting point for the construction of bigger networks. With some work, and the right amount of images, time and computational power it will be possible to obtain the results that we want and give birth to the model that we hoped for in the introduction.

"I/We declare that this material, which I/We now submit for assessment, is entirely my/our 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/our work. I/We 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/us or any other person for assessment on this or any other course of study."