Università degli Studi di Milano

MSc. in Data Science and Economics

Algorithms for massive datasets

Project 4: Plant leave recognizer

Angela Roberti
angela.roberti@studenti.unimi.it

June 2023

1 Dataset

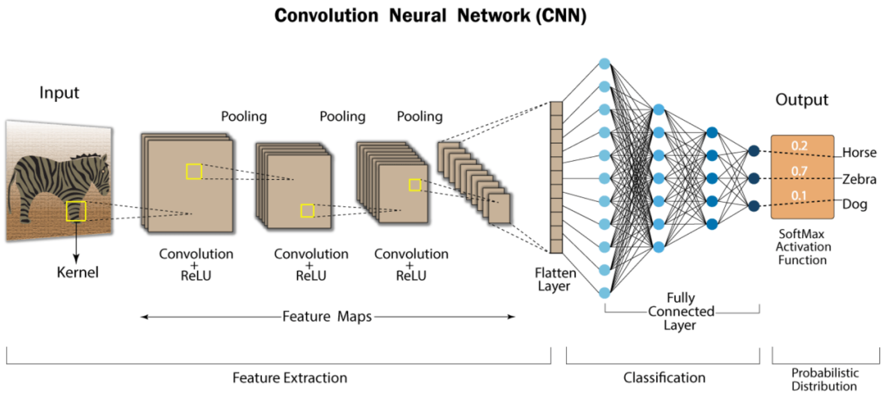
The dataset from Kaggle used for this project initially contained 4503 images of leaves of 11 different types of plants. Of these, 2278 images labelled as healthy leaves images and 2225 images as diseased leaves. For the construction of the model for image classification the dataset has been cleaned by removing the diseased images. Thus, only the healthy leaves were kept and the training folder contained 2163 images while the validation and the test folder 55 images each, 5 for each type of plant.

2 Theoretical framework: why CNN?

Neural Networks are a useful tool to solve different types of problems related to images one of those is image classification, as for this project. A Neural Network is able to recognize and analyse images by learning and identifying complex patterns and features. Neural Networks that have at least one hidden layer which is not either the input nor the output layer are called deep neural networks. In particular Convolutional Neural Network (CNN) is a class of deep NN and is a diffused type of NN used when the goal is to assign a label or a category to a large amount of data.

A CNN consists of 3 main layers:

- **CONVOLUTIONAL LAYER**: it is the core of the structure; it applies a series of filters (for this project with fixed size 3x3) to the image each capturing a specific pattern or a feature. The filter, or kernel, moves across the input image checking if a feature is present in it, as the iterations goes the kernel sweeps the entire image. Convolution operations are nothing but element-wise matrix multiplication between the filter values and the pixels in the image.
- **POOLING LAYER**: as the name suggests conducts the mathematical operation of pooling so the extraction from a given matrix of a new matrix with lower dimensionality which contains only one element for each cluster of the original matrix, in this project the maximum value (max pooling) .This helps decreasing the dimensionality of the input data helping in reducing the numbers of parameters and consequently the computation required.
- **FULLY CONNECTED LAYER (FC)**: or Dense layer, is the last layer, before the output layer,(where image classification happens according to the features extracted in the previous layers). It's called fully connected because here all the nodes from one layer are connected to every activation unit, or node, of the next layer.



In addition to that, the **flatten layer**, that comes before the FC, simply convert the output of the previous layer, which is multi-dimensional, into a one-dimensional vector. It does not introduce any additional parameters or computations, and its purpose is solely to make the input compatible with the subsequent fully connected layer.

The **activation functions** used in this projects are the *Rectified Linear Unit (ReLU)* and the *Softmax function*. The latter is used only for the output layer that outputs the probability that the input image belongs to a particular class. The Rectified Linear Unit, used in the other occurrences, is a non-negative activation function. ReLU does not saturates close to its extreme producing very small gradients that can cause the signal to be slow to propagate especially for larger networks, as the Sigmoid does instead. This helps the backpropagation of the error during training.

3 Model compile

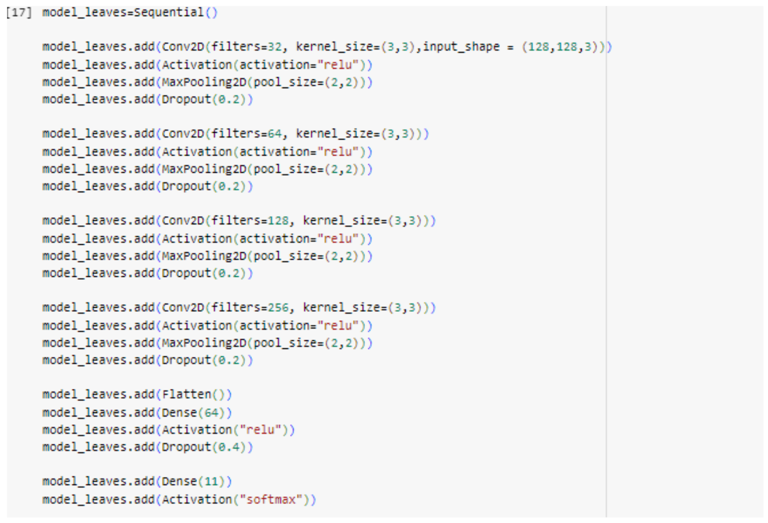
The **learning rate** is a hyperparameter that controls the amount of change of the weights during training in response to the estimated error. Usually deep learning neural networks are trained using the stochastic gradient descent optimization algorithm. For this project I've chosen the *Adaptive learning rate optimization algorithm (Adam)* that combines the advantages of momentum optimization (which helps accelerate convergence) and RMSprop (which adapts the learning rate for each weight based on the gradients' magnitudes). During training, Adam automatically adapts the learning rate for each weight, converging faster compared to traditional SGD with a fixed learning rate.

For the **loss function**, since this is a project related to multiclass classification, I've selected the *Categorical Cross Entropy loss function*, where the function compares each predicted probability with the actual classes, which are mutually exclusive, of output.

Finally, the **metric** chosen is *accuracy*. Accuracy metric computes the accuracy rate so the number of correctly predicted images as a fraction of the total number of predicted images.

4 Model structure

The model used in this project is made by 4 hidden layers. The number of the filters are increasing proportionally 32,64,128,256 (I was inspired from the VGG architecture where the number of filters gradually increases as we go deeper into the network, allowing the model to learn more complex features). Each convolutional layer is followed by an *Activation layer* (ReLu function) and a *Pooling layer* (max pooling, with fixed size 2x2).



```
[17] model_leaves=Sequential()

     model_leaves.add(Conv2D(filters=32, kernel_size=(3,3),input_shape = (128,128,3)))
     model_leaves.add(Activation(activation="relu"))
     model_leaves.add(MaxPooling2D(pool_size=(2,2)))
     model_leaves.add(Dropout(0.2))

     model_leaves.add(Conv2D(filters=64, kernel_size=(3,3)))
     model_leaves.add(Activation(activation="relu"))
     model_leaves.add(MaxPooling2D(pool_size=(2,2)))
     model_leaves.add(Dropout(0.2))

     model_leaves.add(Conv2D(filters=128, kernel_size=(3,3)))
     model_leaves.add(Activation(activation="relu"))
     model_leaves.add(MaxPooling2D(pool_size=(2,2)))
     model_leaves.add(Dropout(0.2))

     model_leaves.add(Conv2D(filters=256, kernel_size=(3,3)))
     model_leaves.add(Activation(activation="relu"))
     model_leaves.add(MaxPooling2D(pool_size=(2,2)))
     model_leaves.add(Dropout(0.2))

     model_leaves.add(Flatten())
     model_leaves.add(Dense(64))
     model_leaves.add(Activation("relu"))
     model_leaves.add(Dropout(0.4))

     model_leaves.add(Dense(11))
     model_leaves.add(Activation("softmax"))
```

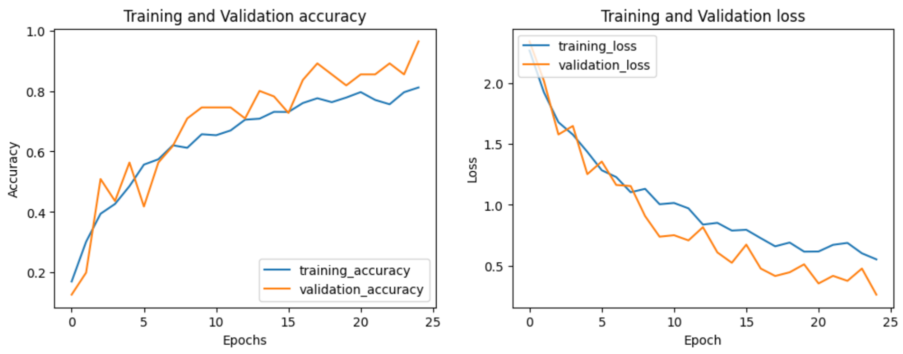
To prevent the model from overfitting, in which the CNN would try to learn too many details in the training data that it is also learning the noise in the data, **the dropout function** has been added in the model.The dropout layer is a mask that nullifies the contribution of some nodes towards the next layer. By randomly deactivating some nodes in the layer, it forces the network to learn with less information and thus prevent overfitting on the training data. Thus, the models has been tested on 0.2 dropout rate for convolutional layers and for the fully connected layer I let the dropout rate be higher,(0.4) since it possesses a higher number of trainable parameters.

Data augmentation, that is added in the *Image Data Generator*, helps the model in reducing the overfitting problem, since it forces the CNN to identify the images when these are being slightly modified. In particular in my example, I've implemented **width shift** with range [-10%, 10%] and

height shift with range[-10%, 10%] and **horizontal flip**. This is supposed to help the neural network model as it expands the training dataset and so the ability of the models to recognize what they have learned to the new images, imposing to the model to not just memorize the trends.

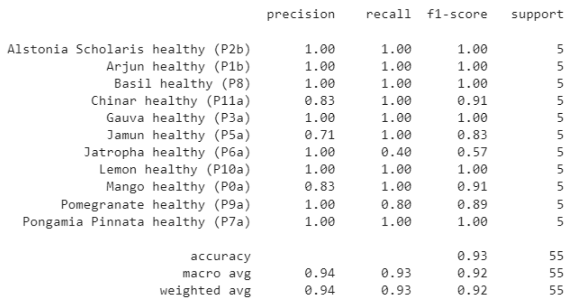
5 Results

Figure shows the accuracy and loss curves over 25 epochs of both training and validation.



The training and validation accuracy curves exhibit a similar upward trend while the loss curves a consistent decreasing one, indicating that the model is effectively learning from the training data. Besides test and validation accuracy curves do not present a big gap between them which means that the model is not suffering from big problem of overfitting. Due to restricions in the GPU access, the model has been trained over 25 epochs. The validation curve peaks at more than 85% of accuracy in the last 5 epochs. Although there is not yet an evident sign of plateau in the curves, this level of accuracy can be considered satisfactory regarding the complexity and difficulty of the problem being solved. Furthermore the test accuracy, performed on unseen images, is 92.73%.

The accuracy score which computes the accuracy of the model's predictions by comparing them to the true labels is above 90%. In addition, precision score, which is the ratio of true positives to the sum of true positives and false positives it's more than 80% for all classes except *Jamun* class, where indeed reached 71%. Recall score, which is the ratio of true positives to the sum of true positives and false negatives, is highest for all classes except *Jatropha* class (0.4) and *Pomegranate* class (0.8). Overall precision,recall and f1-score reached more than 90%.

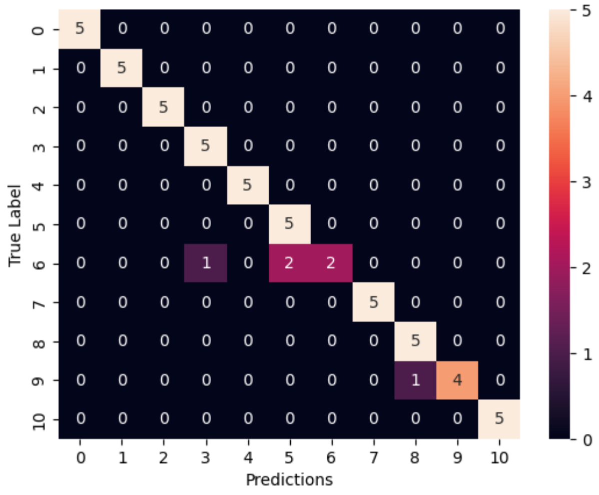


```
                                 precision    recall  f1-score   support

Alstonia Scholaris healthy (P2b)      1.00      1.00      1.00         5
             Arjun healthy (P1b)      1.00      1.00      1.00         5
              Basil healthy (P8)      1.00      1.00      1.00         5
           Chinar healthy (P11a)      0.83      1.00      0.91         5
             Gauva healthy (P3a)      1.00      1.00      1.00         5
             Jamun healthy (P5a)      0.71      1.00      0.83         5
          Jatropha healthy (P6a)      1.00      0.40      0.57         5
            Lemon healthy (P10a)      1.00      1.00      1.00         5
             Mango healthy (P0a)      0.83      1.00      0.91         5
       Pomegranate healthy (P9a)      1.00      0.80      0.89         5
  Pongamia Pinnata healthy (P7a)      1.00      1.00      1.00         5

                        accuracy                          0.93        55
                       macro avg      0.94      0.93      0.92        55
                    weighted avg      0.94      0.93      0.92        55
```

With the heatmap shown below it is possible to interpret the confusion matrix that shows the true positive, false positive, false negative, and true negative counts, clearly. The figure shows that out of the 11 types of plants, 8 of them were classified correctly 5/5. The least correctly predicted class is number 6 *Jatrohpa* class, where only 2 images out of the 5 contained in the test folder were correctly predicted.


True Label
Predictions