Artificial Neural Network and Deep Learning

**Image Classification**

Simone Sorrenti

**ABSTRACT**

This report discusses the image classification of different types of leaves. The images of leaves consist of 17728 images of size 256x256 pixel and are divided into 14 classes: “Apple”, “Blueberry”, “Cherry”, “Corn”, “Grape”, “Orange”, “Peach”, “Pepper”, “Potato”, “Raspberry”, “Soybean”, “Squash”, “Strawberry”, “Tomato”. There are 2 steps: feature extraction and classification. Feature extraction obtains features automatically using convolutional neural network (CNN). The CNN model was tested is the Inception-V3. Based on the testing results, the accuracy of the model has come out to be 84% for the leaf identification**.**

# Dataset

The dataset provided is a relatively small dataset, consists of 17728 leaf images from 14 species of size 256x256 pixels, color space RGB and file format JPG. Each class contains approximately from 264 to 5693 images. All the leaves are standing on a dark background.

|  |  |  |
| --- | --- | --- |
| **Class** | **Number of images** | **Distribution** |
| Apple | 988 | 5.57% |
| Blueberry | 467 | 2.63% |
| Cherry | 583 | 3.29% |
| Corn | 1206 | 6.80% |
| Grape | 1458 | 8.22% |
| Orange | 1748 | 9.86% |
| Peach | 977 | 5.51% |
| Pepper | 765 | 4.31% |
| Potato | 716 | 4.04% |
| Raspberry | 264 | 1.49% |
| Soybean | 1616 | 9.12% |
| Squash | 574 | 3.24% |
| Strawberry | 673 | 3.80% |
| Tomato | 5693 | 32.11% |

Dataset 2.0: For some tests I used a different dataset since the dataset provided is an imbalanced dataset (some classes of leaves have a much higher number of examples than others which have relatively few).  
I applied the Random Oversampling on the dataset; it randomly duplicates examples from the minority class and adding them to the training dataset.  
Moreover, in duplicate examples was applied the data augmentation to try to provide examples different from the original ones to avoid overfitting the model.  
The Random Undersampling on the dataset was ignored as deleting examples could risk deleting relevant information and why it would further reduce the available data.

In conclusion, this new dataset has 79702 leaf images and each class 5693 images.

# Augmentation Process

# The data augmentation was applied to increase the examples during training and validation stages and introduce slight distortion to the images which helps in reducing overfitting during the training stage.

# To determine the most suitable transformations to be applied to the images, I relied on research using scientific documents that dealt with the same problem. From this research it emerged that the transformations by which the best performances were obtained were:

|  |  |
| --- | --- |
| Data Augmentation Techniques | Parameters |
| Rotation | [-170, 170] |
| Brightness | [1, 5] |
| Width Shift | [-0.2, +0,2] |
| Height Shift | [-0.2, +0,2] |
| Zoom | [0.5, 1.5] |
| Fill mode | Constant with black pixel |

# Image pre-processing

Graphical user interface, text

Description automatically generated with medium confidenceTo preprocess the data, I used the default pre-processing implemented for the CNN Inception V3 model.

# CNN Model

# Searching through scientific papers addressing this challenge, it emerged that the use of Fine-Tuning with CNN popular models allowed for remarkable performance in terms of image classification accuracy of plant leaves.

# Among the various CNN models, one we covered in class was the CNN Inception V3 model, so I used this model to carry out the various tests. I preferred to focus on a single CNN model due to the limited number of computational resources and the time available (a single training required several hours).

# I did not include the top of the inception v3 as I needed a softmax output for 14 classes, so I proceeded to create a custom top as follows:

|  |
| --- |
| GlobalAveragePooling2D |
| Dropout with 0.2 probability |
| Dense with 1024 neurons using swish and kernel initialized |
| Dropout with 0.2 probability |
| Dense with 14 neurons using softmax and kernel initialized |

# Training

The training was carried out with the aim of minimizing the Categorical Cross-entropy loss function as it was dealing with a classification of several classes. In addition, 2 inherent metrics were used: accuracy and F1 measure, and on them was applied early stopping with a patience from 10 to 20.

Different learning rates and epochs were applied in a range from 1e-3 to 1e-5 and in a range from 50 to 100.  
As for the dataset, it was split into training set and validation set respectively with the following proportions 80% and 20%. As regards the dataset, it was split into training set and validation set (Hold out) with the following proportions 80% and 20% respectively; it would have been appropriate to apply K-fold Cross-Validation but due to limited computational resources and time it was not possible.

Also, for some tests, class\_weight was applied to balance the unbalanced dataset during learning

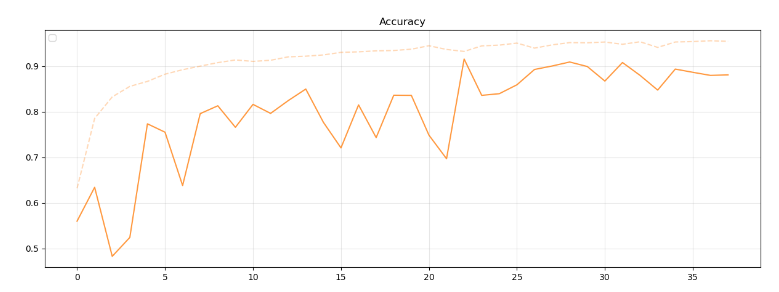
Finally, a fixed batch of 32 was applied due to limited memory.

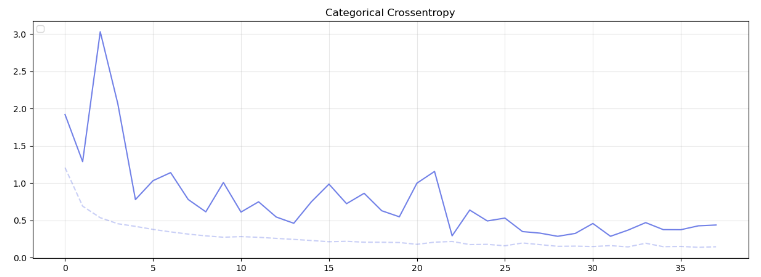
|  |  |
| --- | --- |
| Loss function | Categorical Cross-entropy |
| Metrics | Accuracy, F1-Measure |
| Early Stopping | Accuracy, F1-Measure with patience [10, 20] |
| Learning rates | [1e-3, 1e-5] |
| Epochs | [50, 100] |
| Cross-Validation | Hold out with 80%/20% |
| Balance classes | Class\_weight |
| Batch | 32 |

# Results

The best model (Model 3) scored 83.21% on the first test set and 83.96% on the second test set.

Below are the graphs of the loss function and accuracy at each epoch:





|  |  |
| --- | --- |
| **Model** | **Accuracy on test set 1** |
| Model 1 | 68.30% |
| Model 2 | 80.94% |
| Model 3 | 83.20% |
| Model 4 | 77.73% |
| Model 5 | 81.13% |
| Model 6 | 82.83% |