Convolutional Neural Network: Leaves Classification
Artificial Neural Networks and Deep Learning - a.y. 2021/2022

Fabio Tresoldi
M.Sc. Computer Science and Engineering
Politecnico di Milano - Milan, Italy
E-mail: fabio1.tresoldi@mail.polimi.it
Student ID : 10607540
Codalab Nickname: "fabioow"
Codalab Group: "artificial_comrades"

Mirko Usuelli
M.Sc. Computer Science and Engineering
Politecnico di Milano - Milan, Italy
E-mail: mirko.usuelli@mail.polimi.it
Student ID : 10570238
Codalab Nickname: "mirko"
Codalab Group: "artificial_comrades"

***Abstract*—In this report, we explore and compare the development process we had to design a Convolutional Neural Network able to classify 14 different plants species through their leaves images as input.**

1. Introduction

The given dataset is unbalanced with respect to the classes distribution, then we decided to apply a Stratified Sampling procedure in order to preserve the full dataset proportions both in training, validation and testing.

This structural characteristic suggests the usage of the Categorical Crossentropy as loss function, whereas as metrics we kept into account accuracy and F1-score, being the latter more meaningful for unbalanced categories, while the former is used for validation comparisons in this report.

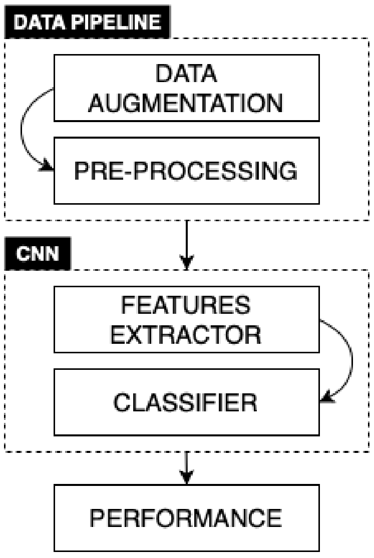

Figure 1. Development Process.

2. Data Pipeline

2.1. Data Augmentation

Due to the reduced dimension of the proposed dataset compared to the parameters complexity of the underlying architecture, Data Augmentation purpouse consists in solving this issue by generating at run time several new transformed images which allow a better generalized performance of the model.
Starting from Traditional Transformations (2.1.1), we tried to furtherly improved the quality of classification by introducing some Advanced Transformations (2.1.2).

2.1.1. Traditional Transformations. Rotating, Zooming, Flipping, Brightness, Shifting.

2.1.2. Advanced Transformations. CutMix, MixUp, CutOut.

However, these last Advance Transformations did not increased the overall performance in our specific case, but indeed, they made the training process slower and not as accurate as without them, therefore we decided to discard these techniques.

In terms of validation accuracy, within 20 epochs:

No Augmentation	99.86%
Traditional Transf.	**99.93%**
Advanced Transf.	99.79%

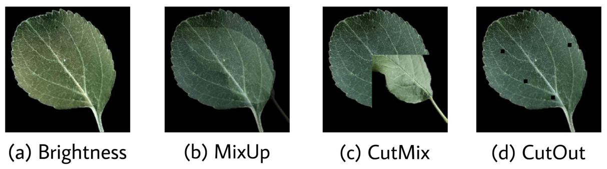

Figure 2. Some examples of Data Augmentation techniques.

2.2. Pre-Processing

Since for the Features Exctractor (section 3.1) we relied on State-of-Art architectures through Fine Tuning, the data pre-processing function adopted is the same suggested by the corresponding model contained in `tensorflow.keras.applications`.

3. Convolutional Neural Network

3.1. Features Extractor

We started comparing the most famous architectures showed during the course lectures through Transfer Learning and a simply GAP + Softmax layers as baseline classifier. Then we plotted both Categorical Crossentropy loss and Accuracy for the validation and training set (dashed line) within 10 epochs:

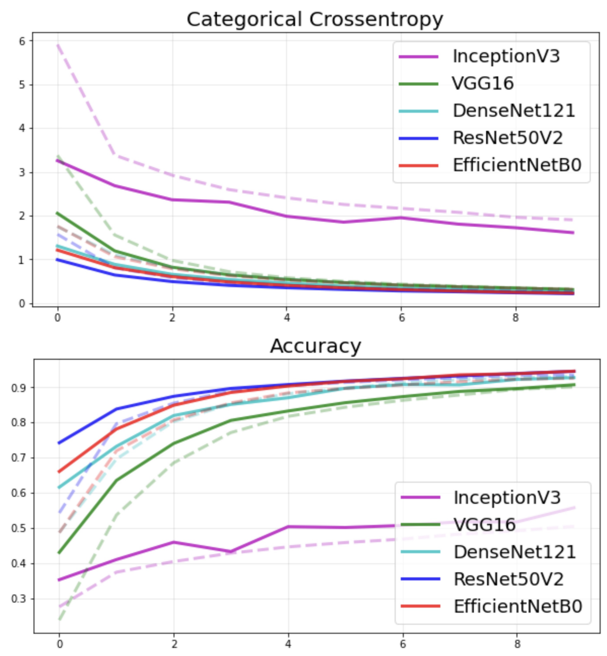

Figure 3. State-of-Art architectures comparison.

As shown in Figure 3, the best models were EfficientNet and ResNet. While the latter converges faster, the former is able to reach a higher peak of performance in both loss and metrics as epochs go further. Thus, we have chosen EfficientNet as Features Extractor.

ResNet50V2	94.54%
EfficientNetB0	**94.57%**
EfficientNetB1	94.40%
EfficientNetB2	94.38%

3.1.1. Ensemble. Then we started introducing the concept of Ensemble in order to reduce the overall variance in prediction. By taking the input multiple times through different Features Extractors, we concatenated the GAP ouput for each model in purpose to feed the final classifier with different conclusions.

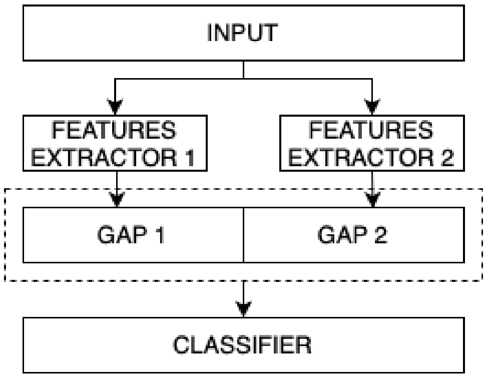

Figure 4. CNN Ensemble.

However, this method performed worse than a single CNN, therefore we discarded this technique.

EfficientNetB0	**99.97%**
EfficientNetB0+B1	99.92%
EfficientNetB0+B1+B2	99.88%

3.2. Classifier

Starting from a general classifier model, with the main layers shown during the course, we analyzed the performance of each of them through Transfer Learning:

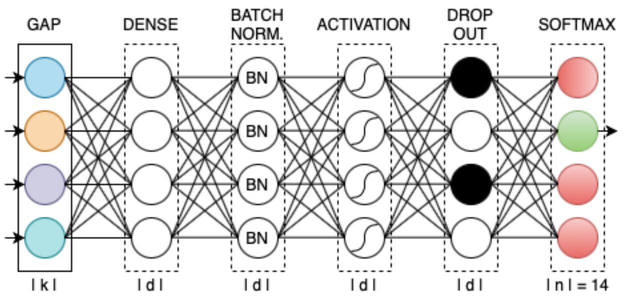

Figure 5. Baseline Classifier Architecture.

3.2.1. Global Average Pooling Layer. this layer is introduced in order to avoid the usage of the Flatten layer, being heavier to manage in terms of complexity and parameters. While GAP is a lightweight layer able to be invariant to intrinsic shift and rotation by structural definition, gaining a more powerful generalization performance.

3.2.2. Dense Layer. The sizes we tested for the dense layer were:

0 (only GAP)	98.89%
64 neurons	98.79%
128 neurons	98.87%
256 neurons	**98.90%**
512 neurons	98.79%
256 neurons (x2)	98.89%

3.2.3. Batch Normalization Layer. We introduced the batch normalization in order to have learnable parameters to be used for the regularization process.

3.2.4. Activation Function Layer. We took into account the main three CNN activation functions in order to find the best matching one:

ReLU	**98.87%**
LeakyReLU	98.73%
GeLU	98.14%

3.2.5. DropOut Layer. We tried DropOut as boosting technique; the hyperparameters we tested were:

No DropOut	**98.93%**
DropOut(0.1)	98.84%
DropOut(0.2)	98.87%
DropOut(0.3)	98.86%

4. Conclusion

4.1. Final Model

The final model setting submitted is:
- **batch_size** : 32
- **loss** : Categorical Crossentropy
- **optimizer** : Adam(1e-4)
- **fine_tuning_layer** : 162 (`block6a_expand_conv`)

Layer (type)	Output Shape	Param
`input`	[(None, 256, 256, 3)]	0
`efficientnetb0`	(None, 8, 8, 1280)	4049571
`global_avg_pool`	(None, 1280)	0
`dense`	(None, 256)	327936
`batch_norm`	(None, 256)	1024
`re_lu`	(None, 256)	0
`softmax`	(None, 14)	3598

Total params:	4,382,129
Trainable params:	3,487,786
Non-trainable params:	894,343

4.2. Performance

The dataset has been split in 80% for training and 20% for validation. The model in training, evaluated with an Early Stopping with patience 10 epochs based on validation loss, reached the peak of performance at epoch 34 with the following indexes:

Train. loss	Train. acc.	Val. loss	Val. acc.
0.0036	99.89%	0.0019	99.97%

4.3. Confusion Matrix

Finally, we generated the confusion matrix to identify the correctness of classification for each class with F1-score, precision and recall, on the dataset split in 60% for training and 20% each for validation and testing. Testing set performance indexes:

- **Accuracy** : 99.77%
- **Precision** : 99.78%
- **Recall** : 99.75%
- **F1-score** : 99.76%

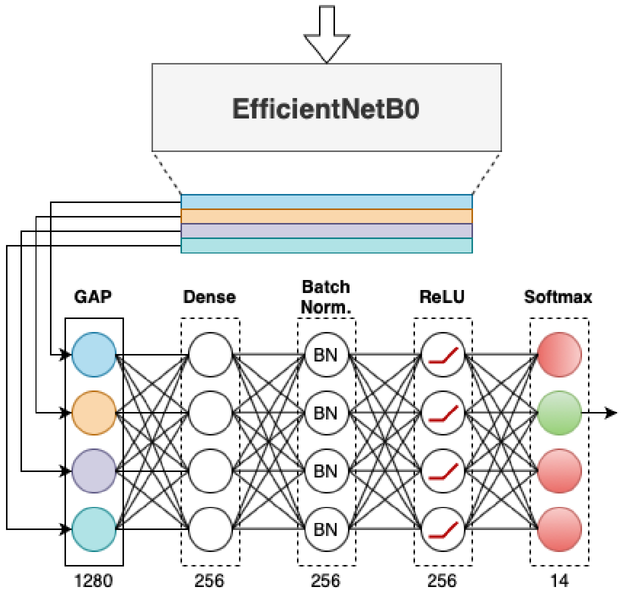

Figure 6. Final model architecture.

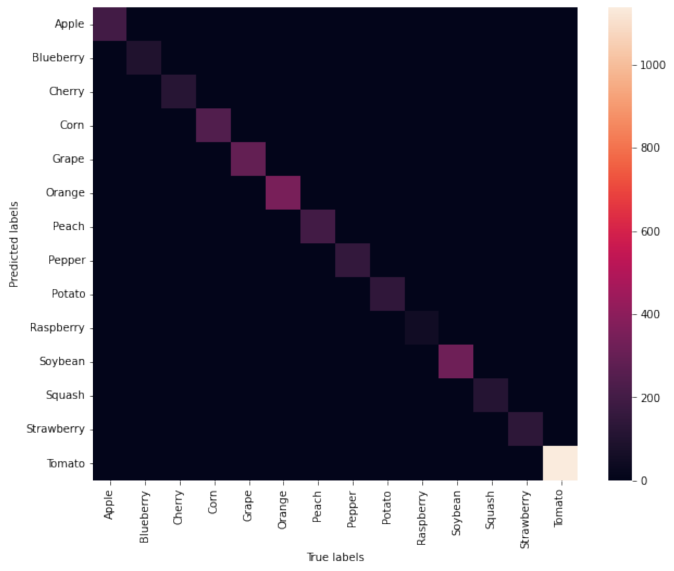

Figure 7. Confusion Matrix on the Testing set.

4.4. Leadboard Evaluation

- Development phase accuracy : **94.91%**
- Final phase accuracy: **94.53%**