Convolutional Neural Network: Plants Classification

Artificial Neural Networks and Deep Learning - a.y. 2022/2023

Marco Bendinelli - 10673478 - *M.Sc. Computer Science and Engineering* -
Marco Cayuela - 10859184 - *M.Sc. Mathematical Engineering* -
Pietro Andrea Cirino - 10628055 - *M.Sc. Mathematical Engineering* -
at Politecnico di Milano

Zero Neurons Networks CodaLab group

Abstract

In this report, we explore and compare the development process we followed to design a Convolutional Neural Network able to classify 8 different plants species through their images as input.

1. Introduction

The characteristic of the classification problem suggested 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.

2. Data Pipeline

2.1. Data Augmentation

Due to the reduced dimension of the proposed dataset compared to the parameters complexity of the underlying architectures, Data Augmentation purpose consists in solving the issue by generating at run time several new transformed images which allow a better generalized performance of the model. We tried to improve the quality of classification by introducing the following transformations: Shifting, Flipping and Brightness changing. This was in fact the set of transformation which most of all improved validation accuracy; in particular, we tested the RegNetX040 model with simply GAP and a Softmax layer on top both with and without data augmentation and after 20 epochs we obtained the following results:

No Augmentation	78.75%
Augmentation	**80.08%**

We also tried to furtherly improve the quality of classification by introducing some Advanced Transformations (CutMix, MixUp, CutOut), but these techniques not only did not increased the overall performance of the model, but they also made the training process slower, so we decided to discard them.

2.2. Pre-Processing

Since for the Features Exctractor (section 3.2) 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. First approach

At first, we tried an handcrafted CNN model consisting of 5 convolutional blocks (Convolution layer + MaxPooling) followed by a Flatten layer and with on top two Dense layers with dropout (3.932.488 parameters), as shown in Figure 1. However, with this simple model we obtained an accuracy of just 62.03% on the hidden test dataset, and since after testing other hand-crafted architectures we did not improve this result we decided to switch to Transfer Learning, a more powerful technique.

3.2. Features Extractor

In order to decide which pre-trained model to use as features extractor, we started comparing the most famous architectures showed during the course lectures; to do so, we tested the models through Transfer Learning and simply GAP + Softmax layer as baseline classifier. As shown in Figure 2, we plotted accuracy history for both the validation and training set (dashed line) within 20 epochs. Thus, we discovered that the best model for our purpose was RegNet. Given the existence of many different RegNet

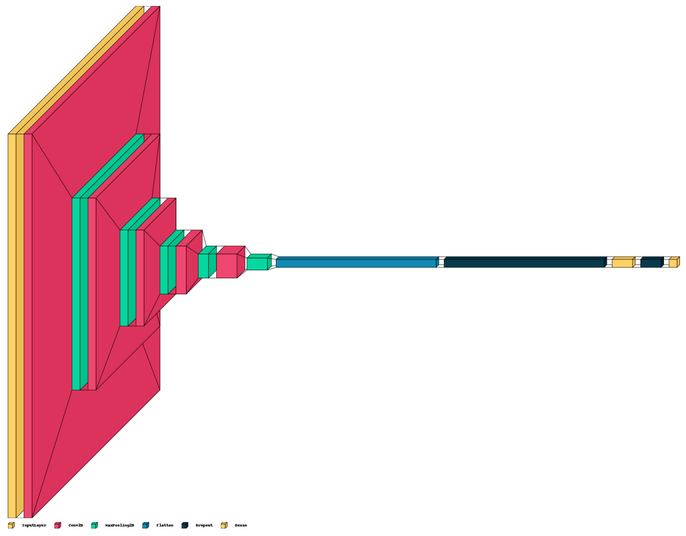


Fig. 1: Architecture of the hand-crafted model

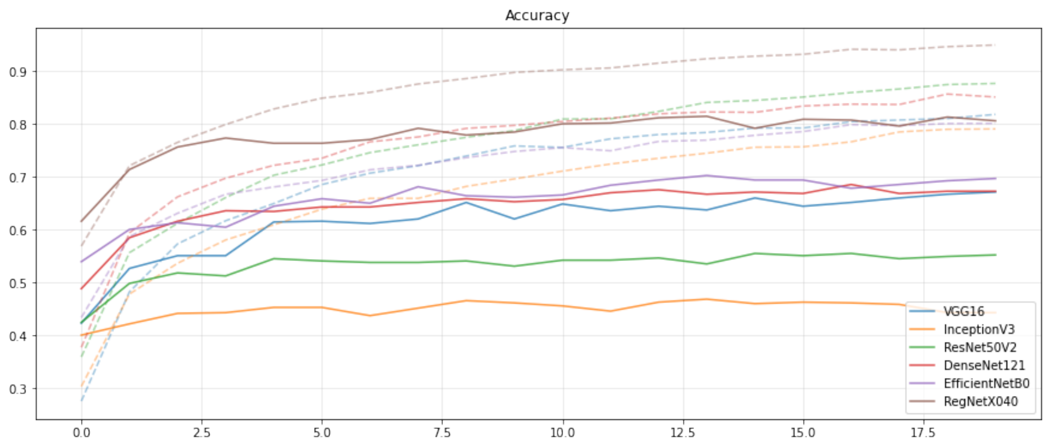


Fig. 2: State-of-Art architectures comparison.

architectures, we also had to decide which one to select between them. As previously, we compared validation accuracy of the different models, obtaining the following results:

RegNetX320	**80.22%**
RegNetX040	80.08%
RegNetY320	79.38%
RegNetX080	79.10%

Hence, we decided to use RegNet X320 as feature exactor.

3.3. Fine tuning Once we had chosen the model, we had to decide the number of layers of the pre-trained model to freeze and the number to train. We observed that the best results were obtained by keeping non trainable the 80 first layers over the total number of 243.

3.4. Global Average Pooling Layer This layer is introduced on top of the model in order to avoid the usage of the Flatten layer, being the latter 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, hence gaining a more powerful generalization performance. We also tried a Global Max Pooling Layer which provided less good results.

3.5. Classifier Starting from a simple GAP + Softmax layer as classifier, we tried to add a Dense layer in order to improve the predictive abilities of our model. However, even with a small number of neurons (16, 32, 64) we observed that the Dense layer caused the model to overfit, lowering the accuracy. Since even adding a Dropout layer did not prevent the overfitting, we decided to keep the classifier as simple as possible, with just a Softmax layer after GAP.

4. Conclusion

4.1. Final Model The final model submitted has a total number of parameters equal to 105,472,744, of which 93,858,248 trainable and 11,614,496 non-trainable.
The main characteristics of the model are the following:

- **batch_size** : 32
- **loss** : Categorical Crossentropy
- **optimizer** : Adam(5e-4)
- **fine_tuning_layer** : 80

and the general structure is:

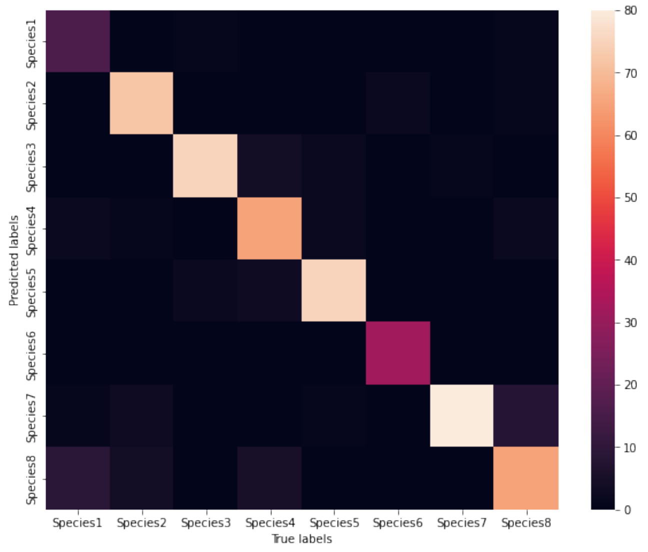
Layer (type)	Output Shape	Param
`input`	[(None, 96, 96, 3)]	0
`regnetx320`	(None, 2520)	105452576
`global_avg_pool`	(None, 2520)	0
`softmax`	(None, 8)	20168

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 accuracy, reached the peak of performance at epoch 23 with the following indexes:

Train. loss	Train. acc.	Val. loss	Val. acc.
0.0601	97.88%	0.3908	90.79%

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 70% for training and 15% each for validation and testing. Testing set performance indexes:

- **Accuracy** : 89.72%
- **Precision** : 90.59%
- **Recall** : 87.35%
- **F1-score** : 88.42%



4.4. Leadboard Evaluation

- Development phase accuracy : **90.48%**
- Final phase accuracy: **88.92%**