Plants Classification

Artificial Neural Networks and Deep Learning – First Challenge

Politecnico di Milano, a.y. 2022/2023

Group name: ANNLeCun

Paolo Marzolo
M.Sc. CS and Engineering
ID: 10668259
Username: pollomarzo

Mark Federico Zampedroni
M.Sc. CS and Engineering
ID: 10608097
Username: PhatCat

Marco Zanoni
M.Sc. CS and Engineering
ID: 10627017
Username: Howny

1. Introduction

In the following report we explain the choices we made for solving the first challenge proposed during the Artificial Neural Networks and Deep Learning course at Politecnico di Milano, a.y. 2022/2023.

The task assigned consists in the development of a model for the classification of plant images into 8 disjoint classes. We used the most common design in image analysis: a Convolutional Neural Network.

2. Data

The provided dataset contains 3542 labeled images (size 96x96) of plants for a total of 8 classes.

Metrics in the following sections were calculated using as baseline model the architecture explained in paragraph 3. We start by showing some of the approaches tried for handling the data and their results.

2.1. Split

As preliminary step we divided the data this way.

Set	Percentage	Cardinality
Training	65.0%	2299
Validation	15.0%	530
Test	20.0%	713

2.2. Data augmentation

Overall, few samples were provided. To increase the generality of the patterns learnt by the model, we augmented our training dataset: changing the images brightness, rotating them, flipping, and more.

Here follows a comparison of the performance on the validation set with augmented and non-augmented images. Fill for rotation and shift uses reflection.

Type	Accuracy	F1
None	89.34%	87.67%
Flips	90.46%	89.86%
Brightness	87.80%	86.31%
Rotation + shift	90.41%	89.27%
All	**92.85%**	**91.85%**

2.3. Classes imbalance

The dataset has imbalanced classes: Species1 and Species6 consist only of 5.25% and 6.27% of the samples. We decided to do stratified sampling in order to generate the training, validation and test sets in a way that reflects the distribution of the given data.

During the development we noticed that it is easy to classify with high accuracy most classes except for Species1, which often gets confused with Species8; so our efforts were focused in trying to improve the model capability in correctly identifying Species1.

To allow the model to compensate for such imbalance we tried two approaches: weighing the loss

function and oversampling. In the following, S1 is Species1.

2.3.1. Weights. Comparison on some weights.

Weights	Accuracy	S1 Precision	S1 Recall
Balanced	91.87%	80.00%	73.68%
Uniform	93.27%	96.43%	71.05%
Tuned	**92.71%**	**89.21%**	**84.75%**

The tuned weights are the best weights found after an automatic calibration with relation to the validation set.

2.3.2. Oversampling. Oversampling in order to uniform the samples quantity by class versus no oversampling. It uses uniform weights for the classes.

Used	Accuracy	S1 Precision	S1 Recall
Yes	92.71%	86.67%	68.42%
No	**93.27%**	**96.43%**	**71.05%**

3. Model

Initially, we tried building the whole model from scratch but realized our baseline started from very low accuracy, so we moved to Transfer Learning. In all trainings we used the Adam optimizer and controlled the learning rate through callbacks provided by Keras, in particular ReduceLROnPlateau. The loss function is the categorical crossentropy.

3.1. Input pre-processing

The images pre-processing logic is handled by the architecture chosen as feature extractor. It transforms the format of our images to the format required by the model without the need of defining anything ourselves.

3.2. Transfer Learning

The models available on Keras, if pre-trained on the ImageNet dataset, may have already learnt useful patterns to categorize both low level shapes and high level images of plants or crops. Hence, we can download one of such a models, keep only the feature extractor part and connect a simple classifier; then in order to do Transfer Learning we keep the bottom frozen and train the classifier weights.

3.3. Feature extractor

The models comparison can be visualized by plotting their training history. All training runs use early stopping and a maximum of 100 epochs.

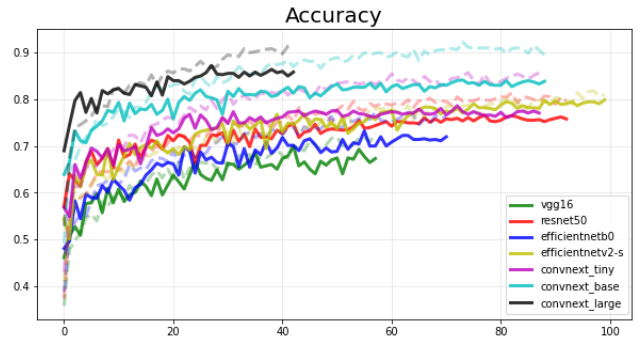

Figure 1. Validation accuracy comparison.

The architecture identifying the best patterns for our problem is ConvNeXt. The results of the same analysis on the different available ConvNeXt versions revealed that **ConvNeXtLarge** heavily outperforms the others. Its bottom has 295 layers and reaches 87.27% accuracy, versus the 84.72% of ConvNeXtBase.

3.4. Classifier

From the simple classifier used during the selection of the feature extractor we added new layers. We tried to regularize the dense layers with lasso, ridge and dropout; only dropout increased the performance.

3.4.1. Pooling layer. The Global Average Pooling layer (GAP) was tested against the Global Max Pooling (GMP) and the results were slightly favourable towards the GAP.

3.4.2. FC layer. After some testing we report only the best results with one, two and three extra dense layers. With no extra layers or very few neurons it performed terribly. We used dropout and batch normalization. The activation function is ReLU.

First	Second	Third	Accuracy
1024	**0**	**0**	**92.45%**
1024	512	0	92.07%
512	256	512	91.69%

3.4.3. Dropout. Dropout is a mask that nullifies the contribution of some neurons towards the next layer. As intended, it was essential to prevent overfitting.

Dropout	Accuracy	S1 Precision	S1 Recall
10%	90.46%	93.33%	73.68%
20%	92.99%	80.56%	76.32%
30%	**93.13%**	**81.58%**	**81.58%**

3.4.4. Activation Function layer. We also tested different activation functions.

Type	Accuracy	S1 Precision	S1 Recall
ReLU	**92.71%**	**89.21%**	**84.75%**
LeakyReLU	91.73%	76.92%	78.95%
GeLU	92.01%	87.88%	76.32%

3.5. Fine-tuning

After having trained the classifier we unfroze some of the last layers of the feature extractor and retrained the CNN with a lower learning rate. The more the layers unfrozen, the higher the generality of the patterns overwritten by the training; in order not to lose the capabilities gained through Transfer Learning we tested various levels of unfreezing. Here we report only the best ones.

Frozen	Accuracy	S1 Precision	S1 Recall
55	92.71%	82.35%	73.68%
90	**94.67%**	**90.00%**	**71.05%**
160	90.18%	80.65%	65.79%

From the results it seems that in order to allow the network to learn useful patterns to classify correctly Species1 we need to unfreeze most of the feature extractor.

3.6. Final Model

The CNN built uses as feature extractor the bottom of ConvNeXtLarge, which feeds the result of its last filters into a GAP (with dropout, to force the classifier to use all the features) followed by a dense layer of 1024 neurons. This FC layer uses dropout, batch normalization and ReLU as activation function. It is then connected to another dense layer of 8 neurons representing the output, processed by a softmax.

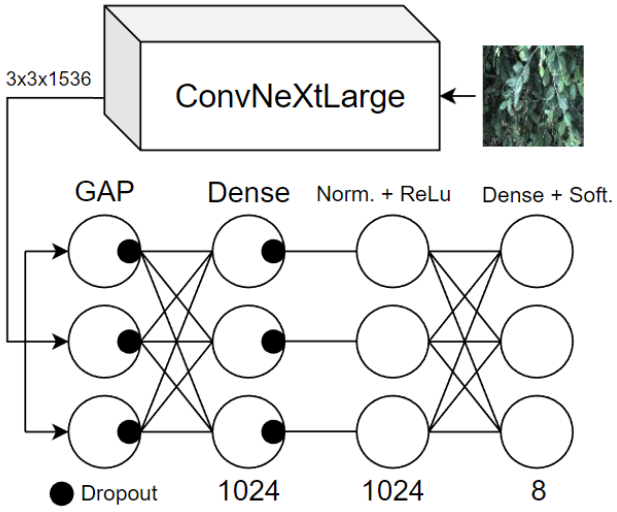

Figure 2. Final CNN structure.

A tabular representation of the network and its training history are shown in the notebook.

4. Testing

Testing is conducted on the testing set mentioned in section 2.1. Since it was created by doing stratified sampling the results reflect the imbalanced distribution of classes samples.

- Accuracy : 94.67%
- Precision : 94.78%
- Recall : 93.17%
- F1 : 93.76%

The confusion matrix is shown in the notebook.

5. Conclusions

After the testing, before submitting the solution, we retrained the CNN following the same process but by resizing the training and validation (for early stopping) sets to 90% and 10% of the total available data.

Eventually we got an accuracy of 93.25% on the first phase of the evaluation and 93.13% on the second, with relatively very good results on the classification of Species1; which, in our opinion, is the most problematic point to handle if aiming for an overall high accuracy.