Artificial neural network and deep learning - First Challenge 2022

Image Classification

Giulia Forasassi, Manuel Montecalvo, and Diego Martin

Politecnico di Milano

November 28, 2022

1 Introduction

Image classification is one of the most common Machine Learning problems. Given a fixed set of classes, image classification is the task of predicting which label to assign to an input image. In this case, the purpose is to classify plants images in order to predict their species using Deep Neural Networks.

In this project we have analyzed the data and experimented different techniques for training the network, including convolution neural networks, transfer learning with fine tuning and ensemble technique.

2 Data analysis

The training dataset that was provided consists in a set of 3542 pictures of different species of plants. They are organized in 8 directories, one for each plant species that is being analyzed (Specie1, Specie2, Specie3, Specie4, Specie5, Specie6, Specie7, Specie8). The images are `96x96` resolution JPEGs (Figure 1) and the distribution of samples in the dataset is unbalanced, since there is a different number of samples for each of the classes. (Figure 2)

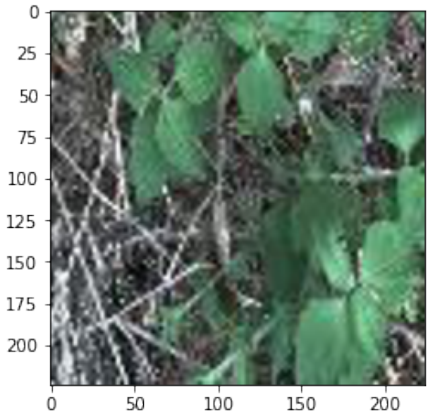

Figure 1: Example of a plant image

Figure 2: Classes distribution

3 Data preparation

3.1 Data splitting

We started by splitting the dataset into training and validation sets using a fixed ratio of 85 and 15 and taking random elements from the initial dataset.

To overcome the classes imbalance problem we used two alternatives methods: oversampling and class weighting. With oversampling, random images in the minority classes are duplicated until each class is of the same size. On the other side, class weights allows to give different importance to prediction errors on a per class basis. By increasing the weight of a class the lower the number of its samples, the class imbalance problem is mitigated.

3.2 Data augmentation

To prevent overfitting and give more robustness to the network, we have considered appropriate to effectively increase the number of training samples provided by introducing data augmentation. Using `ImageDataGenerator` of Keras we have applied random transformations creating new versions of the same image. We used rotation, zoom, vertical and horizontal flipping, vertical and horizontal shifting and shears filling the empty space with the reflect mode.

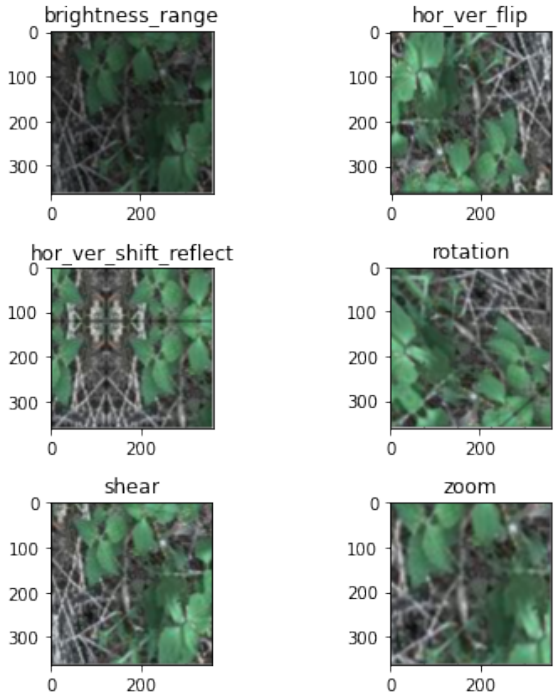

Figure 3: Augmented images

4 Models and techniques

The models selection and construction followed three main approaches: custom deep convolutional neural networks, transfer learning using some known high performance pre-trained networks for image classification, and ensemble technique on previously trained models.

4.1 Base CNN model

The first model we tried is a CNN, made by a sequence of Convolutional and MaxPooling layers with a couple of Dense layers at the end of the network. The network is 14 layers deep with Relu activation function after each convolutional layer and the Softmax activation function in the final layer. We added a Dropout layer between the two Dense layers to reduce overfitting. High levels of accuracy were not achieved with this basic model so we decided to implement new models using the transfer learning technique.

4.2 Transfer learning and Fine tuning

The second approach we tried is the transfer learning technique in order to achieve better generalization performance by taking advantage of other datasets. We used some networks pretrained on the Imagenet dataset. In particular, we chose Densenet201, Resnet152V2 and other networks from the EfficientNet family taken from the `Keras Applications` library.

To adapt these networks to our problem, we replaced the original heads with a Multi Layer Perceptron, and we added the Resize layer at the beginning to make all compatible with the images of our dataset. In the Figure 4 the structure of the network with the best model as pretrained is reported.

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 224, 224, 3)]	0
efficientnetv2-b3 (Functional)	(None, 7, 7, 1536)	12930622
global_average_pooling2d (GlobalAveragePooling2D)	(None, 1536)	0
batch_normalization (BatchNormalization)	(None, 1536)	6144
dense (Dense)	(None, 512)	786944
elu (ELU)	(None, 512)	0
dropout (Dropout)	(None, 512)	0
dense_1 (Dense)	(None, 256)	131328
elu_1 (ELU)	(None, 256)	0
dropout_1 (Dropout)	(None, 256)	0
dense_2 (Dense)	(None, 64)	16448
elu_2 (ELU)	(None, 64)	0
dense_3 (Dense)	(None, 8)	520

Figure 4: network summary

All of these networks have then been fine tuned on the plant dataset. To not exceed the available RAM, we kept the lower layers frozen while training the higher ones.

4.3 Ensemble technique

To improve our performance we used the ensemble technique that combines several base models to produce an optimal predictive one. This has been done by sending the input image to each single model, and then making the average of the output predictions. In the Figure 5 we can see the ensemble model that produced the best results.

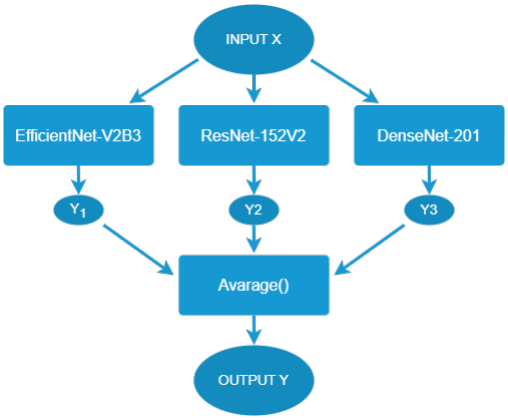

Figure 5: Ensemble model

4.4 Best models

In the end, considering only the transfer learning and fine tuning techniques, the model that achieved the best result was the one utilizing EfficientNetV2B3 as pre-trained network. While, if we consider also the ensemble technique explained above, the best model was the one combining 3 networks respectively based on EfficientNetV2B3, ResNet152V2 and DenseNet201.

5 Training

The training was done entirely on Google Colab and was performed by measuring the accuracy and loss functions on the validation set and tracking the potential overfitting over the training data.

To measure the performance of the classification models was used the Cross Entropy Loss function setting a number of epochs between 100 and 200. Also, we experimented different batch sizes (32 or 256) and a number of patience equal to 20 for the early stopping technique. In some models we optimized using Adam method with the default learning rate, whereas in other models we tested a dynamic learning rate using the Exponential Decay function in which, starting from an initial value, the learning rate was decremented exponentially.

6 Results

This section shows the results of the most successful models used for the classification task, and the accuracy values are reported in the table below.

Model	Accuracy
(1) DenseNet-201	0.9047
(2) Resnet152v2	0.9252
(3) EfficientNetB5	0.9103
(4) EfficientNetB7	0.9327
(5) EfficientNetV2S	0.9252
(6) EfficientNetV2B3	0.9290
(7) Ensemble (2 + 5 + 6)	0.9402
(8) Ensemble (1 + 2 + 6)	0.9383
(9) Ensemble (3 + 5 + 6)	0.9254

Table 1: Model accuracy on local test data

As we can see in the accuracy data (Table 1), the best results were obtained by combining models that exploited both the transfer learning and the fine tuning techniques.

Following, we report additional metrics and results from the best model on local test data. The confusion

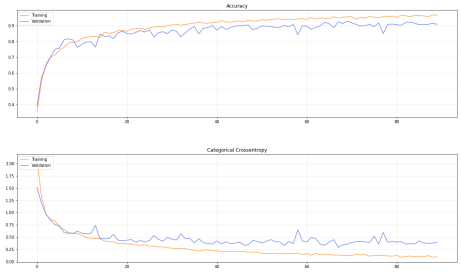

Figure 6: `EfficientNet-V2B3` accuracy and cross-entropy plots

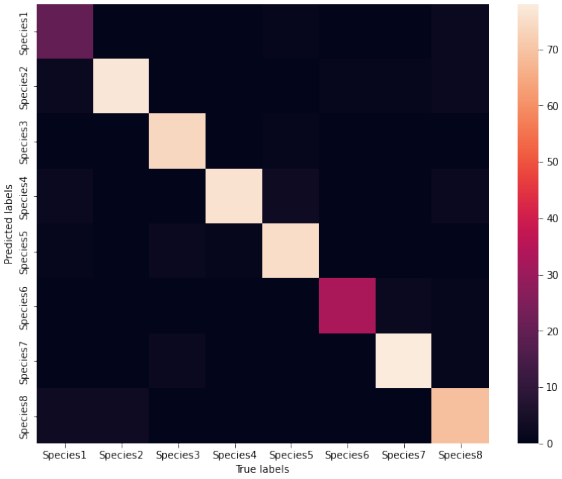

Figure 7: `Ensemble model` confusion matrix

matrix (Figure 7) shows classification performance on the local validation set and the loss and accuracy metrics (Figure 6) show the training speed and metric trends of the best model (EfficientNet-V2B3) without considering the ensemble technique.

7 Conclusions

From the experiments emerges that models using transfer learning with fine tuning obtained the best performance in terms of accuracy. Then, joining best models in an ensemble one, even better results on unseen test set accuracy are achieved.

Tools

- Tensorflow
- Keras
- Scikit-Learn
- Jupyter notebook