Flower Classification

Federica Russo FEDERICA.RUSSO283@GMAIL.COM

1. Model Description

I describe here a deep model that consists of a 3 Convolutional Layers, where filters are applied to the original image to extract different features, followed by two Fully connected layers and a Classification Layer, that will perform a multiclass classification (see Fig. 1).

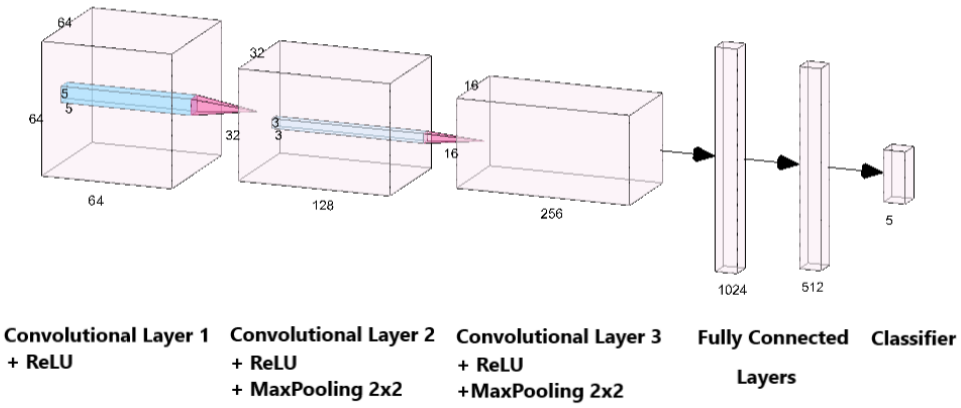

Figure 1: A simple representation of the proposed model.

The model presents three convolutional layers to process the input volume. These three layers have respectively 64, 128 and 256 channels. The first layer has a kernel of size 5, applied with a stride of 1 and followed by a ReLU activation function. The second and third layers have a kernel of size 3 applied with a stride of 1, followed by a ReLU activation function and moreover by a Max Pooling with a kernel size of 2 and a stride of 2. Max pooling will be usefull to provide translation invariance: when a model is translation invariant, it means that it doesn't matter where an object is present in a picture because it will be recognized anyway.
The output of these layers will be transmitted to Fully connected layers which are placed before the classification output and are used to flatten the results before classification. The first layer has input of 57600 channels and output of 1024. The second will take as input 1024 and will have as output 512. The last one, which is the classifier, will have as input 512 channels and as output 5 (i.e. the class of the dataset). As Loss Function I used the Cross Entropy, and as Optimaizer I used the Gradient Descent Optimizer (SGD).

2. Dataset

The dataset was provided by Kaggle.com. It consists of 4323 images that are collected based on the Flickr , Google Images, Yandex Images. The pictures are divided into five classes: daisy, dandelion, rose, sunflower, and tulip. For each class there are about 800 photos. Photos are not high resolution, about 320x240 pixels and are not reduced to a single size but they have different proportions. Some photos contain no backgrounds while others may have disturbing elements, such as animals or people.

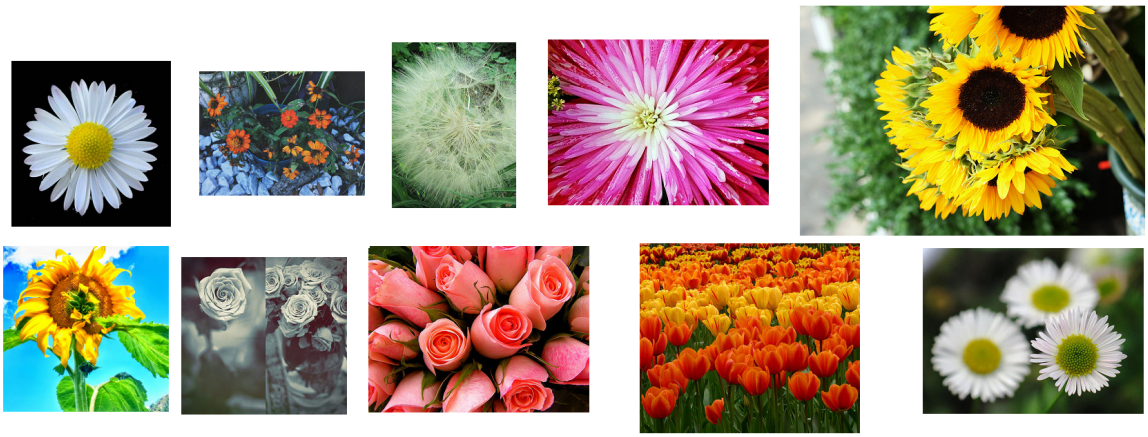

Figure 2: Sample images of the dataset

The images were left in color, while a resizing was carried out to obtain the same measurements in all of them, in particular 64x64. I also applied a Random Horizontal and Vertical Flip (both the standard value of 0.5), a Random Rotation (50) and a Normalization with value for mean and std used for the Channel-wise data. Data are then randomly splitted in train (2523), validation (900) and test set (900). Thanks to ImageFolder class all the images can be associated to their label, that corrisponde to the name of the folders. To each label I also assigned a number to performe the classification.

3. Training procedure

As I already said, I employ several data augmentation mechanisms, namely, random input rotation, translation and mirroring, to reduce overfitting issues.
Beside the FlowerModel, I tested other three models, respectively based on a sequence of 2, 4 and 5 convolutional layers and 3 dense layers.

As training loss I used the CrossEntropy function which measures the performance of a classification model and whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label so a perfect model would have a log loss of 0.
Cross entropy for n classes is defined as:

$$L_{CE} = -\Sigma t_i log(p_i) \tag{1}$$

where T_i is the truth label p_i is the Softmax probability for the i^{th} class.

All models were trained from scratch over 50 epochs on Google Colab. Batch size is set to 32 for all the models. SGD is chosen as optimizer algorithm using a learning rate of 0.01.

4. Experimental Results

Several experiments were conducted to test the effectiveness of the proposed FlowerModel architecture. Models were trained as described in the previous section. Table 1 shows test results for the proposed architectures as well as of ablation studies (i.e., different variants of the final architecture when adding or removing layers).

Model	Validation Accuracy	Test Accuracy
Baseline Net (2 conv layer)	64.9%	63.4%
– + Layer 3	65.3%	67.8%
– + Layer 4	65.2%	66.7%
– + Layer 5	61.5%	64%
Final model	65.3%	67.8%

Table 1: Test performance of the models