Università degli studi di
Milano-Bicocca

Advanced Machine Learning

Final Project

Flowers Recognition

Authors:
Gianmaria Balducci - 807141- g.balducci1@campus.unimib.it
Alessandro Guidi - 808865 - a.guidi@campus.unimib.it

January 20, 2021

Abstract

The aim of this project is to compare different techniques of deep learning in order to obtain satisfying results over a classification task. The task is to classify 102 category of flowers, taking into account a dataset composed by images of flowers with different colors and shapes. In this report we will describe four different approaches that make use of Convolutional Neural Networks and we will evaluate this differents approaches in order to identify and discuss on which of this models is more suitable for this task. The approaches are: Training our CNN from scratch, transfer-learning using DenseNet121 and linear svc classifier and fc layers as classifier and and fine-tuning with DenseNet121. At the end, we will apply a model compression technique to the best model being compared. This differents models are compared for determine which is the best for this type of task that have some troubles because there are many categories that are very similar in colours and in the shapes.

1 Introduction

When considering the image of a flower, it is not possible for a non-professional or non-expert to verify its species due to their lack of botanic knowledge; additionally, for scientists who are not specialized in plants, when only flower images are given, it is not possible for them to obtain the necessary additional information. In natural environments there are many important object to classify and to distinguish; flowers have very important role for the survival of an ecosystem. In this project we propose three methods to resolve the classification problem for 102 category of flowers, the images have large scale, pose and light variations. In addition, there are categories that have large variations within the category and several very similar categories, colors and shapes are very important. So, this project propose four approaches to resolve this problem that use CNN that are very suitable for image classification. CNN are widely use in computer vision and let us to obtain satisfying results in this task but we would like to focus on how important the tranferlearning approach. As we see in the following sections, the result obtained with this approach are better than the approach of training a CNN from scratch because pretrained architecture over IMAGENET dataset give an important boost over performances because this is a task with a dataset very smaller than IMAGENET and different from IMAGENET.

2 Datasets

For this project dataset used is 102 Category Flower Dataset from *Visual Geometry Group*. This is a 102 category dataset that contains 8189 images of flowers belonging to 102 different categories. The flowers chosen to be flower commonly occuring in the United Kingdom. Each class consists of between 40 and 258 images. First, we start by downloading and extracting the data set files :

- "102flowers.tgz" : Flower Pictures (.jpg format with a name of the form image_00001 ¿ image_08189)
- "imagelabels.mat" : Flower Labels (From 1-102)
- "setid.mat" : Flower Ids for data splitting (Train, Validation, Test)
- "names.csv" : Flowers names ordered by their labels (starting from 1)

Then we labelled all data and split dataset in train, validation and test sets. Fortunately, keras has some build-in functions that take care of these steps, particularly the ImageDataGenerator function that helps quickly setting up Python generators that can automatically turn image files on disk into batches of preprocessed tensors, which avoids loading of the whole data in memory. With ImageDataGenerator we normalize all pixels dividing by 255 and apply preprocess function used for IMAGENET dataset imported by keras. With an instance of ImageDataGenerator we can call the function *flow_data_from_directory* that offer many parameters to preprocess images and takes the dataframe and the path to a directory and generates batches. The generated batches contain augmented/normalized data. For train and validation directory we set a batch size of 8 while for testset batch size is 16. Traget size is the dimensions to which all images found will be resized and is set (227, 227), input image is (227, 227,3) because the images are colored obviously, so color_mode is set to rgb. This process helps quickly setting up Python generators that can automatically turn image files on disk into batches of preprocessed tensors. An example of the image of the dataset:

Click *here* to see statistics of datasets number of image per category

Figure 1: image of flower dataset

3 The Methodological Approach

3.1 Start

At first, when we started working, implementing Transfert-Learning, we used VGG16 on IMAGENET for pretrained model. It has been adapted for our input images, in fact the *input_shape* has been modified to shape images. VGG16 did not work badly, the performances were acceptable but no excellent, this motivated us to improve them, at the beginning with little changhes of the hyperparameter as the optimizator,both typology and paramaters.

After execute preprocessing phase over images with the same function that use DenseNet121 on IMAGENET, we do 4 experiments that will be the topics in this section where we describe their characteristics and their implementations. The experiments carried out in this project are:

- train our CNN from scratch
- Transfer-Learning as feature extractor with DenseNet121 using feed forward network classifier
- Transfer-Learning as feature extractor with DenseNet121 using a Linear SVC classifier
- Tranfer learning as fine-tuning with DenseNet121

3.2 Our CNN

ALE QUA METTIAMO CHE CI SIAMO ISPIRATI AD ALEXNET E REFERENZIAMO IL PAPER(di alexnet?

As first experiments we choose to build a CNN model that takes in input the dimension of our images. We was inspired by the architecture of **AlexNet CIT PAPER DA METTERE** and our CNN's architecture is very similar with small difference in the number of units in the fully-connected layers and number of convolution layer that are three.

Layer (type)	Output Shape	Param #
conv2d_3 (Conv2D)	(None, 55, 55, 96)	34944
activation_6 (Activation)	(None, 55, 55, 96)	0
conv2d_4 (Conv2D)	(None, 55, 55, 256)	614656
activation_7 (Activation)	(None, 55, 55, 256)	0
max_pooling2d_2 (MaxPooling2	(None, 27, 27, 256)	0
conv2d_5 (Conv2D)	(None, 27, 27, 352)	811360
activation_8 (Activation)	(None, 27, 27, 352)	0
max_pooling2d_3 (MaxPooling2	(None, 13, 13, 352)	0
flatten_1 (Flatten)	(None, 59488)	0
dense_3 (Dense)	(None, 2048)	121833472
dropout_2 (Dropout)	(None, 2048)	0
activation_9 (Activation)	(None, 2048)	0
batch_normalization_2 (Batch	(None, 2048)	8192
dense_4 (Dense)	(None, 1024)	2098176
dropout_3 (Dropout)	(None, 1024)	0
activation_10 (Activation)	(None, 1024)	0
batch_normalization_3 (Batch	(None, 1024)	4096
dense_5 (Dense)	(None, 102)	104550
activation_11 (Activation)	(None, 102)	0

Total params: 125,509,446
Trainable params: 125,503,302
Non-trainable params: 6,144

Figure 2: Architecture of our CNN

The first convolutional layer has 96 filters with size 11x11, second convolutional layer has 256 filters 5x5 and third convolutional layers has 352 fileters 3x3. Every convolutional layer contain max pooling with pool size 3x3 and activation function used is *Relu*. After these 3 convolutional layers, there are three fully connected layers including output layer. Fisrt Dense layer has 2048 units, over these units is applied Dropout regularization with 0.4 and relu is used as activation function, second layer has 1024 units, we apply dropout also in this layer but with 0.2 and activation function is tanh, the the output layer with 102 units for categories and output function is sofmtax. In all this fully connected layers is apllied batch notrmalization. This kind of architecture is compless and requires a lot of time to complete training phase. Starting from AlexNet we try to simplify this architecture obtaining worse results. We do not expect to solve the classification problem with this architecture but this is a good starting point to understand wich path to take. Furthermore we do not have enough avaiable hardware to build and train a CNN that can obtain satisfayng results. Many approaches in many papers suggest to use transfer learning techniques, so we choose to implement transfer-learning with following three different approaches

DenseNet121

DenseNet was developed specifically to improve the declined accuracy caused by the vanishing gradient in high-level neural networks. In simpler terms, due to the longer path between the input layer and the output layer, the information vanishes before reaching its destination. DenseNet is one of the new discoveries in neural networks for visual object recognition. DenseNet is quite similar to ResNet with some fundamental differences. ResNet uses an additive method that merges the previous layer (identity) with the future layer, whereas DenseNet concatenates the output of the previous layer with the future layer.

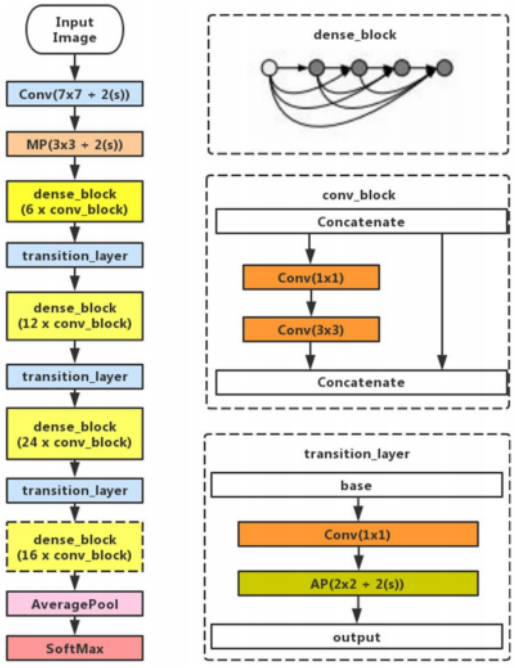

Figure 3: (Left) DenseNet121 architecture. (Right) Dense_block, conv_block and transition_layer

3.3 Features extract using DenseNet121 with feed forward network classifier

3.4 Features extract using DenseNet121 with Linear SVC classifier

3.5 Fine-Tuning using DenseNet121

3.6 Model Compression

4 Results and Evaluation

The Results section is dedicated to presenting the actual results (i.e. measured and calculated quantities), not to discussing their meaning or interpretation. The results should be summarized using appropriate Tables and

Figures (graphs or schematics). Every Figure and Table should have a legend that describes concisely what is contained or shown. Figure legends go below the figure, table legends above the table. Throughout the report, but especially in this section, pay attention to reporting numbers with an appropriate number of significant figures.

5 Discussion

The discussion section aims at interpreting the results in light of the project's objectives. The most important goal of this section is to interpret the results so that the reader is informed of the insight or answers that the results provide. This section should also present an evaluation of the particular approach taken by the group. For example: Based on the results, how could the experimental procedure be improved? What additional, future work may be warranted? What recommendations can be drawn?

6 Conclusions

Conclusions should summarize the central points made in the Discussion section, reinforcing for the reader the value and implications of the work. If the results were not definitive, specific future work that may be needed can be (briefly) described. The conclusions should never contain "surprises". Therefore, any conclusions should be based on observations and data already discussed. It is considered extremely bad form to introduce new data in the conclusions.

References

The references section should contain complete citations following standard form. The references should be numbered and listed in the order they were cited in the body of the report. In the text of the report, a particular reference can be cited by using a numerical number in brackets as [1] that corresponds to its number in the reference list. LaTeXprovides several styles to format the references

References

[1] J. Lee, “Smart Factory Systems,” *Informatik-Spektrum*, vol. 38, no. 3, pp. 230–235, 2015.