Multimedia Information Retrival and Computer Vision

Alice Nannini
Marco Parola
Stefano Poleggi

A.Y. 2020-2021

Contents

1 Introduction

In this project we implemented a Web Search Engine working on "ArtImages" dataset and "MirFlickr" dataset as distractor. The Web Search Engine is based on the **Locality Sensitive Hashing** (LSH) index and on the pre-trained convolutional neural network ResNet50 with weigths computed on the "Imagenet" dataset.

2 Data Analysis and Preprocessing

The Art dataset is composed of 5 categories of images, corresponding to the 5 classes:

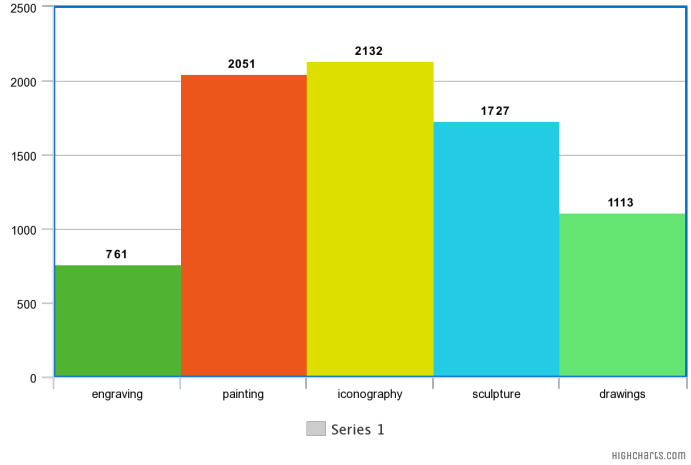
- Engraving
- Sculpture
- Iconography
- Painting
- Drawings

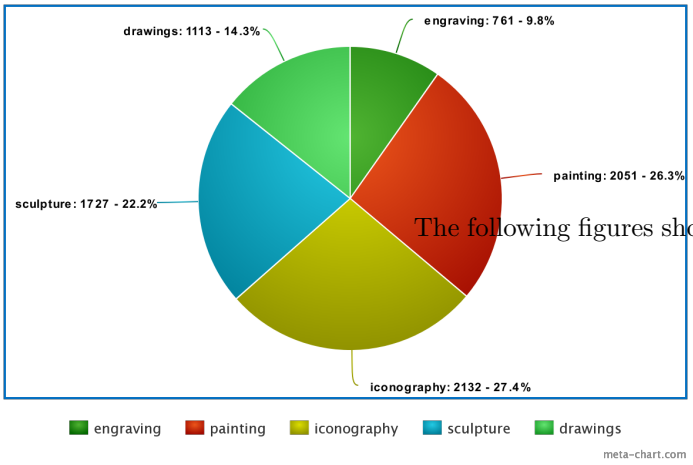
2.1 Data cleaning

After a preliminary analysis of the dataset, we realized that some images are corrupted both on trainingset and validationset, so we detected and deleted them using the following script.

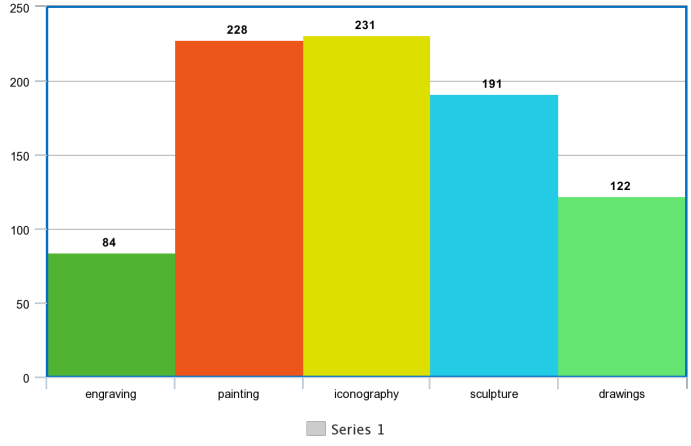
```
from PIL import Image
import glob

for filename in os.listdir(VALIDATION_DATA):
    try:
      image.load_img(VALIDATION_DATA + filename, target_size=(300, 300))

    except :
      print("ERROR" + VALIDATION_DATA + filename)
      os.remove(VALIDATION_DATA + filename)
```

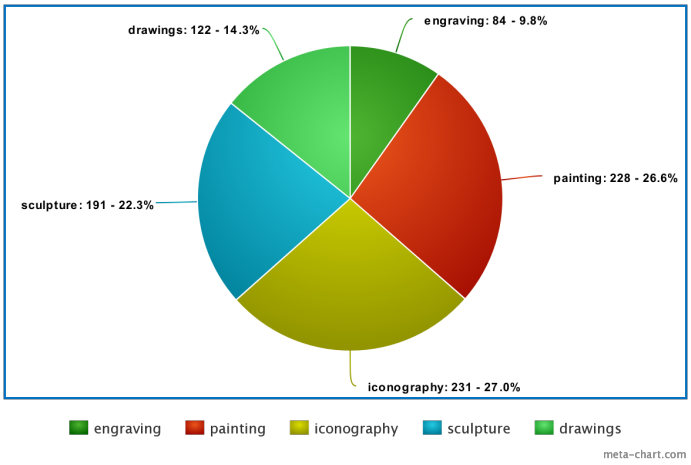
The following figures show the distribution of the classes of the trainigset, after the execution of the snippet of code shown above :





The following figures show the distribution of the classes of the validationset, after the execution of the snippet of code shown above :


250
200
150
100
50
0
84
228
231
191
122
engraving
painting
iconography
sculpture
drawings
Series 1
Highcharts.com


drawings: 122 - 14.3%
engraving: 84 - 9.8%
painting: 228 - 26.6%
sculpture: 191 - 22.3%
iconography: 231 - 27.0%
engraving
painting
iconography
sculpture
drawings
meta-chart.com

3 Extract features from the images

During this phase we extracted the features from the images and we stored them in order to reload them as numpy-array when it is necessary.
The pretrained convolutional neural networks used are:

- ResNet50
- ResNet50v2

3.1 Features Extraction approach

In this phase we extracted the features from the images using the pretrained convolutional neural networks **Resnet50** and **Resnet50v2** trained on *Imagenet* dataset, adopting a fetures extraction approach.
The following snippet of code shows the extraction of the trainingset using Resnet50, the same script is been used also with Resnet50v2 and for the validationset and distractor.

```
def extract_features(extractor, generator, sample_count, dim=2048):
  features = extractor.predict_generator(generator, sample_count)
return features

datagen = image.ImageDataGenerator( rescale = 1./255 )

train_generator_scaled = datagen.flow_from_directory(
        TRAINING_DATA,
        target_size=IMAGE_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='sparse',
        shuffle=False)

conv_base = ResNet50(weights='imagenet',
include_top=False,
input_shape=(300, 300, 3),
pooling='avg')


training_features = extract_features(conv_base,
 train_generator,
TRAINING_SIZE )
np.save(FILE_TRAINING_FEATURES, training_features)
```

3.2 Fine Tuning approach

During this phase we carried out several tests using different hyperparameters for the fully connected part added at the end both of the networks Resnet50 and Resnet50v2.
We compute this phase adopting a trial and error approach, trying different hyperparameter values for the fully connected neural network appended to the pretrained convolutional one and we compared the performance.
We stopped the training exploiting an *early stopping condition*, monitoring the loss function and setting the patience value equal to 10 and restoring the weights to 10 epochs before.
The following subparagraphs show only the most relevant experiments, we didn't report all of them, in order to avoid some redundancies.

3.3 Resnet50

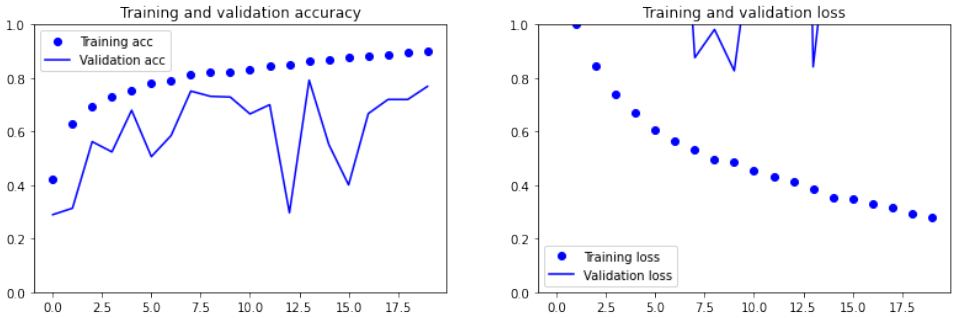
3.3.1 Model 1

In this trial we trained the pretrained network Resnet50, we appended a fully connected part defined by the following hyperparameter:

- Unfreezed layers from conv5_block1_1_conv
- 1 dense layer composed by 256 neurons
- Activation function: softmax
- Learning rate: 1e-3

	precision	recall	f1-score	support
	0.0	0.59	0.39	0.47	122
	1.0	0.63	0.77	0.70	84
	2.0	0.90	0.86	0.88	231
	3.0	0.89	0.78	0.83	228
	4.0	0.65	0.86	0.74	191
	accuracy			0.76	856
	macro avg	0.73	0.73	0.72	856
	weighted avg	0.77	0.76	0.76	856

loss_test : 0.7306886315345764
acc_test : 0.7616822719573975



3.3.2 Model 2

Looking at the accuacy plot of the first experiment, we can observe the presence of overfitting phenomenon; in order to reduce this problem, we added some *Dropout Layer*.
So in the second trial we trained the pretrained network Resnet50, we appended a fully connected part defined by the following hyperparameter:

- Unfreezed layers from conv5_block1_1_conv
- Dropout layer, rate: 0.5
- 1 dense layer composed by 256 neurons
- Dropout layer, rate: 0.5
- 1 dense layer composed by 128 neurons
- Dropout layer, rate: 0.5
- Activation function: softmax
- Learning rate: 1e-4

```
precision    recall  f1-score   support

         0.0       0.47      0.33      0.39       122
         1.0       0.70      0.56      0.62        84
         2.0       0.60      0.94      0.73       231
         3.0       0.98      0.22      0.36       228
         4.0       0.53      0.81      0.64       191

    accuracy                           0.59       856
   macro avg       0.66      0.57      0.55       856
weighted avg       0.68      0.59      0.55       856

loss_test : 0.9764283299446106
acc_test : 0.5934579372406006
```

3.3.3 Model 3

In the third trial we trained the pretrained network Resnet50, we appended a fully connected part defined by the following hyperparameter:

- Unfreezed layers from conv5_block1_1_conv
- 1 dense layer composed by 512 neurons
- 1 dense layer composed by 256 neurons
- 1 dense layer composed by 128 neurons
- 1 dense layer composed by 64 neurons
- Activation function: softmax
- Learning rate: 1e-4

```
precision    recall  f1-score   support

         0.0       0.67      0.02      0.03       122
         1.0       0.69      0.39      0.50        84
         2.0       0.46      0.97      0.62       231
         3.0       0.82      0.82      0.82       228
         4.0       0.80      0.38      0.52       191

    accuracy                           0.61       856
   macro avg       0.69      0.52      0.50       856
weighted avg       0.68      0.61      0.56       856

loss_test : 0.9576621651649475
acc_test : 0.605140209197998
```

3.3.4 Model 4

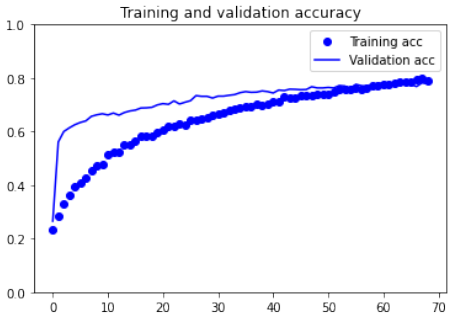
In the fourth trial we trained the pretrained network Resnet50, in order to better generalize the dataset, so we appended a fully connected part defined by the following hyperparameter:

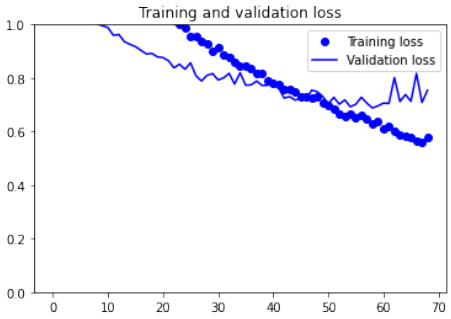
- Unfreezed layers from conv5_block1_1_conv
- Dropout layer, rate: 0.5
- 1 dense layer composed by 256 neurons
- Dropout layer, rate: 0.5
- 1 dense layer composed by 128 neurons
- Dropout layer, rate: 0.5

- 1 dense layer composed by 64 neurons
- Activation function: softmax
- Learning rate: 1e-6

	precision	recall	f1-score	support
	0.0	0.71	0.34	0.46	122
	1.0	0.64	0.68	0.66	84
	2.0	0.80	0.95	0.87	231
	3.0	0.96	0.87	0.91	228
	4.0	0.76	0.92	0.83	191
	accuracy			0.81	856
	macro avg	0.78	0.75	0.75	856
	weighted avg	0.81	0.81	0.79	856

loss_test : 0.6162440776824951
acc_test : 0.8072429895401001



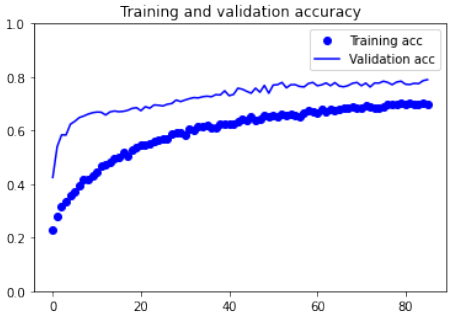


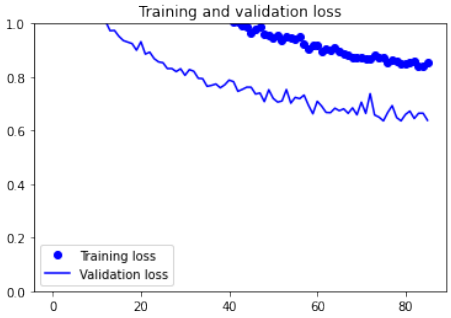
3.3.5 Model 5

In the fifth trial we trained the pretrained network Resnet50, where we appended the same fully connected part of the previous experiment applying a data augmentation technique, in order to better generalize the dataset.

	precision	recall	f1-score	support
	0.0	0.62	0.28	0.38	122
	1.0	0.53	0.74	0.62	84
	2.0	0.85	0.87	0.86	231

```
           3.0       0.91      0.85      0.88       228
           4.0       0.71      0.87      0.79       191

    accuracy                           0.77       856
   macro avg       0.72      0.72      0.70       856
weighted avg       0.77      0.77      0.76       856

loss_test : 0.6571111083030701
acc_test : 0.7675233483314514
```





3.3.6 Conclusion

	Accuracy	**Loss**	**mAP** euclidian	**mAP** cosine	Items
Model 1	0.7617	0.7307			
Model 2	0.5935	0.9764			
Model 3	0.6051	0.9577			
Model 4	0.8072	0.6162			
Model 5	0.7675	0.6571			

3.4 Resnet50v2

3.4.1 Model 1

In this trial we trained the pretrained network Resnet50, we appended a fully connected part defined by the following hyperparameter:

- Unfreezed layers from conv5_block1_1_conv
- 1 dense layer composed by 256 neurons
- Activation function: softmax
- Learning rate: 1e-3

3.4.2 Model 1

3.4.3 Model 2

3.4.4 Model 3

3.4.5 Model 4

3.4.6 Model 5

4 Sequential Scannig

The first implementation of our image search engine is based on a numpy array as data structure and a sequential scanning as search algorithm, using the following similarity distance:

- Euclidian distance
- Cosine similarity
- Hamming distance