UNIVERSITÀ DI PISA

Artificial Intelligence and Data Engineering

Computational Intelligence and Deep Learning

Artist Identification with Convolutional Neural Networks

Project Documentation

TEAM MEMBERS:
Edoardo Fazzari
Mirco Ramo

Academic Year: 2020/2021

Contents

1 — Introduction

Artist identification is traditionally performed by *art historians* and *curators* who have expertise and familiarity with different artists and styles of art. This is a complex and interesting problem for computers because identifying an artist does not just require object or face detection; artists can paint a wide variety of objects and scenes. Additionally, many artists from the same time period will have similar styles, and some such as **Pablo Picasso** (see figure 1) have painted in multiple styles and changed their style over time.

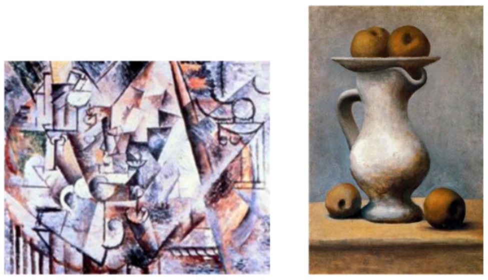

Figure 1: Both of these paintings were created by Pablo Picasso, but they have vastly different styles and content since they correspond to two different periods.

The aim of this project is to use Convolutional Neural Networks for the identification of an artist given a painting. In particular, the CNN networks will be modeled using multiple techniques: from scratch; via pretrained network; networks based on comparison with baseline input and using an ensemble network made of by the best classifiers found.

1.1 State of the Art

As mentioned, artist identification has primarily been tackled by humans. An example of that is the Artsy's Art Genome Project [1], which is led by experts who manually classify art. This strategy is not very scalable even if it is highly precise in the classification (the site is a marketplace of fine-arts for collects, you can find Pisarro, Bansky and other famous artists).

Most prior attempts to apply machine learning to this problem have been feature-based, aiming to identify what qualities most effectively distinguish artists and styles. Many generic image features have been used, including scale-invariant feature transforms (SIFT), histograms of oriented gradients (HOG), and more, but with the focus on discriminating different style in Fine-Art Painting[2].

The first time the problem of artist identification was really tackled was with J. Jou and S. Agrawal[3] in 2011, they applied several multi-class classification techniques like Naïve Bayes, Linear Discriminant Analysis, Logistic Regression, K-Means and SVMs and achieve a maximum classification accuracy of 65% for an unknown painting across 5 artists. Later on, the problem of identifying artists was retackled by the *Rijksmuseum Challenge*[4]. The objective of the challenge was to predict the artist, type, material and creation year of the 112,039 photographic reproductions of the artworks exhibited in the Rijksmuseum in Amsterdam (the Netherlands). The year later, Saleh and Elgammal's paper[5] was the first attempt to identify artists with a large and varied dataset, but still using generic features.

[1]https://www.artsy.net/categories

[2]T. E. Lombardi. The classification of style in fine-art paint- ing. ETD Collection for Pace University, 2005

[3]J. Jou and S. Agrawal. Artist identification for renaissance paintings.

[4]T. Mensink and J. van Gemert. The rijksmuseum challenge: Museum-centered visual recognition. 2014

[5]B. Saleh and A. M. Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. CoRR, abs/1505.00855, 2015

More recent attempts are related to the *Painter by Numbers*, a **Playground Prediction Competition** by *Kaggle*[6]. This competition used a pairwise comparison scheme: participants had to create an algorithm which needs to examine two images and predict whether the two images are by the same artist or not. Thus, it is not our same objective, however it can be consider the first application of Deep Learning to the problem. The real deal was taken by Nitin Viswanathan[7] in 2017. Viswanathan, using the same dataset of the mentioned *Kaggle Challenge*, proposed the use of ResNet with transfer learning (he first held the weights of the base ResNet constant and updated only the fully-connected layer for a few epochs). This trained network reached a train accuracy of 0.973 and a test accuracy of 0.898.

1.2 Dataset

Unfortunately, the dataset provided by the *Kaggle Challange* is too huge to be used in Colab, in fact it is about 60GB unbearable on the free version of Colab, which provides only about 30GB of disk. Stated that, we decided to use a different dataset[8] with only 2GB of data and about 8k unique images.
The data downloaded from Kaggle has the following directories and csv file:

```
/
├── images
│   └── images
│       ├── Albrecht_Durer
│       ├── Alfred_Sisley
│       ├── Amedeo_Modigliani
│       ├── Andrei_Rublev
│       ├── Andy_Warhol
│       ├── Camille_Pissarro
│       ├── Caravaggio
│       ├── Claude_Monet
│       ├── Diego_Rivera
│       ├── Diego_Velazquez
│       ├── Edgar_Degas
│       ├── Edouard_Manet
│       ├── Edvard_Munch
│       └── an many others (total of 50 different artists)
├── resized
└── artists.csv
```

The *resized* directory and the *artists.csv* are not useful for our studies, hence we deleted them to save space on the disk. Then, we modified the structure of the *images/images* directory in order to create two directories, **train** and **test**, containing 85% and 15% of the images from each different artist's directory respectively. The newly created directories have the same structured of *images/images*. This was done in *python* in this way:

```python
import os
import numpy as np
import shutil

rootdir= '/content/images/images' #path of the original folder
classes = os.listdir(rootdir)

for i in classes:
  if not os.path.exists(rootdir + '/train/' + i):
```

[6]https://www.kaggle.com/c/painter-by-numbers/data
[7]Nitin Viswanathan, Artist Identification with Convolutional Neural Networks
[8]https://www.kaggle.com/ikarus777/best-artworks-of-all-time

```python
    os.makedirs(rootdir + '/train/' + i)
  if not os.path.exists(rootdir + '/test/' + i):
    os.makedirs(rootdir + '/test/' + i)

  source = os.path.join(rootdir, i)
  allFileNames = os.listdir(source)

  np.random.shuffle(allFileNames)

  test_ratio = 0.15
  train_FileNames, test_FileNames = np.split(np.array(allFileNames),
                                                          [int(len(allFileNames)*
                                          (1 - test_ratio))])

  train_FileNames = [source+'/'+ name for name in train_FileNames.tolist()]
  test_FileNames = [source+'/' + name for name in test_FileNames.tolist()]

  for name in train_FileNames:
    shutil.copy(name, rootdir +'/train/' + i)

  for name in test_FileNames:
    shutil.copy(name, rootdir +'/test/' + i
```

After that we created the train/validation/test-set using the *image_dataset_from_directory* function provided by **Keras** in the following way:

```python
import tensorflow as tf

training_images = tf.keras.preprocessing.image_dataset_from_directory(
    TRAIN_DIR, labels='inferred', label_mode='categorical',
    class_names=None, color_mode='rgb', batch_size=BATCH_SIZE,
    image_size=(IMAGE_HEIGHT,  IMAGE_WIDTH), shuffle=True, seed=RANDOM_SEED,
    validation_split=VALIDATION_SPLIT, subset='training',
    interpolation='bilinear', follow_links=False
)

val_images = tf.keras.preprocessing.image_dataset_from_directory(
    TRAIN_DIR, labels='inferred', label_mode='categorical',
    class_names=None, color_mode='rgb', batch_size=BATCH_SIZE,
    image_size=(IMAGE_HEIGHT, IMAGE_WIDTH), shuffle=True, seed=RANDOM_SEED,
    validation_split=VALIDATION_SPLIT, subset='validation',
    interpolation='bilinear', follow_links=False
)

test_images = tf.keras.preprocessing.image_dataset_from_directory(
    TEST_DIR, labels='inferred', label_mode='categorical',
    class_names=None, color_mode='rgb', batch_size=BATCH_SIZE,
    image_size=(IMAGE_HEIGHT, IMAGE_WIDTH), shuffle=True, seed=RANDOM_SEED,
    interpolation='bilinear', follow_links=False
)
```

Obtaining:

- 6081 files for training (belonging to 50 classes).
- 1073 files for validation (belonging to 50 classes).
- 1292 files for testing (belonging to 50 classes).

2 — CNN from Scratch

3 — Pre-Trained Models

3.1 ResNet50V2

3.1.1 Test 1: Classical ResNet50V2 with 50 classes

On a small dataset, overfitting will be the main issue. Data augmentation is a powerful way to fight overfitting when you're working with image data.

3.1.2 Test 2: Completely Newly Output Layers Architecture

3.1.3 Test 3: 1 and 2 with Data Augmentation

3.1.4 Test 4: Fine Tuning with One Layer

3.1.5 Test 5: Fine Tuning with Two Layers

3.1.6 Test 6: Genetic Algorithm for Hyper-parameters and Architecture Optimization

3.2 ResNet101V2

3.2.1 Test 1: Classical ResNet101V2 with 50 classes

3.2.2 Test 2: Completely Newly Output Layers Architecture

3.2.3 Test 3: Fine Tuning with One Layer

3.2.4 Test 4: Fine Tuning with Two Layers

3.3 InceptionV3

3.3.1 Test 1: Classical ResNet101V2 with 50 classes

3.3.2 Test 2: Completely Newly Output Layers Architecture

3.3.3 Test 3: Fine Tuning with One Layer

3.3.4 Test 4: Fine Tuning with Two Layers