An Automated Approach to Artist Classification Using DL With the Help of Classical Principles. AKA Tim.

Alireza Alizadeh

Selected path

Computer Vision (Image classification).

Problem statement and objective

We are aiming to train 1 or more models to predict the artist form a given piece of art with following metrics: Recall≥0.70, Accuracy≥0.70. In addition, all other standard metrics such as F1-Score and Precision will be reported as well.

A secondary objective of this project is to reach the goals mentioned above using consumer grade mid to low tier GPUs (6 GB VRAM or less).

Dataset

For this task there are 2 famous datasets. The WikiArt dataset and the Best Artworks of All Time dataset.

Since we aim to train and use our model locally we choose the 2.4 GB Best Artworks of All Time dataset since the alternative WikiArt dataset is a heavy 33.8 GB dataset.

Dataset information and introduction

The Best Artworks of All Time dataset is a collection of artworks from the 50 most influential artists of all time. The dataset features roughly 8,000 Images both in original resolution and resized to half of the original resolution.

This dataset contains three files/directories:

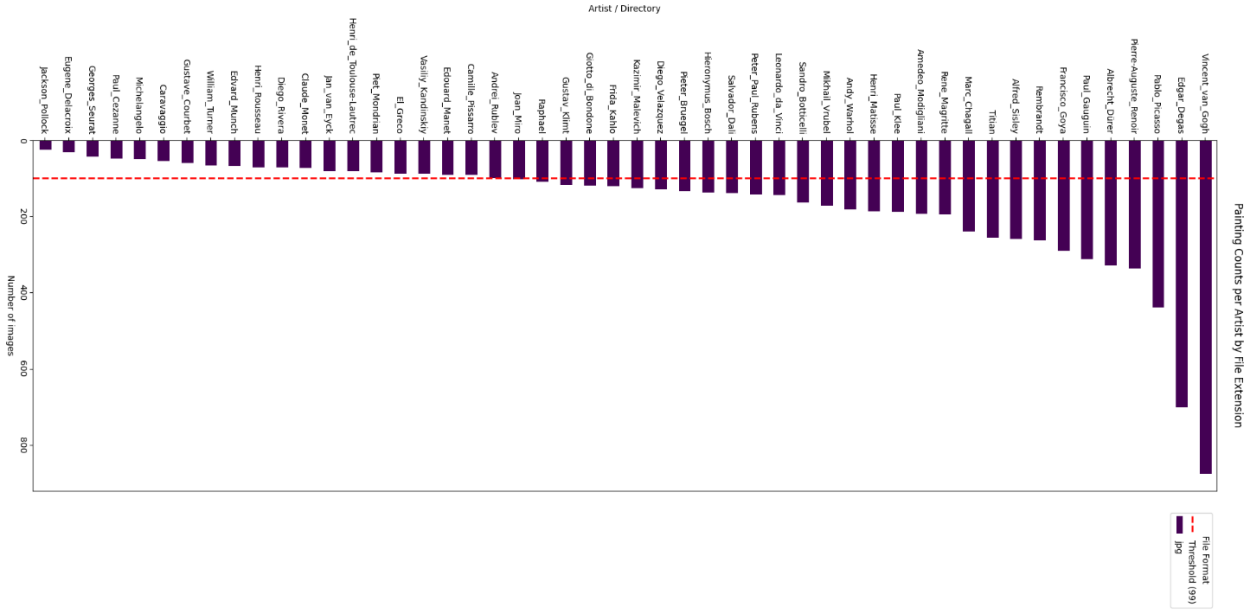
- **artists.csv**: dataset of information for each artist
- **images**: collection of images (full size), divided in folders and sequentially numbered
- **resized**: same collection but images have been resized and extracted from folder structure

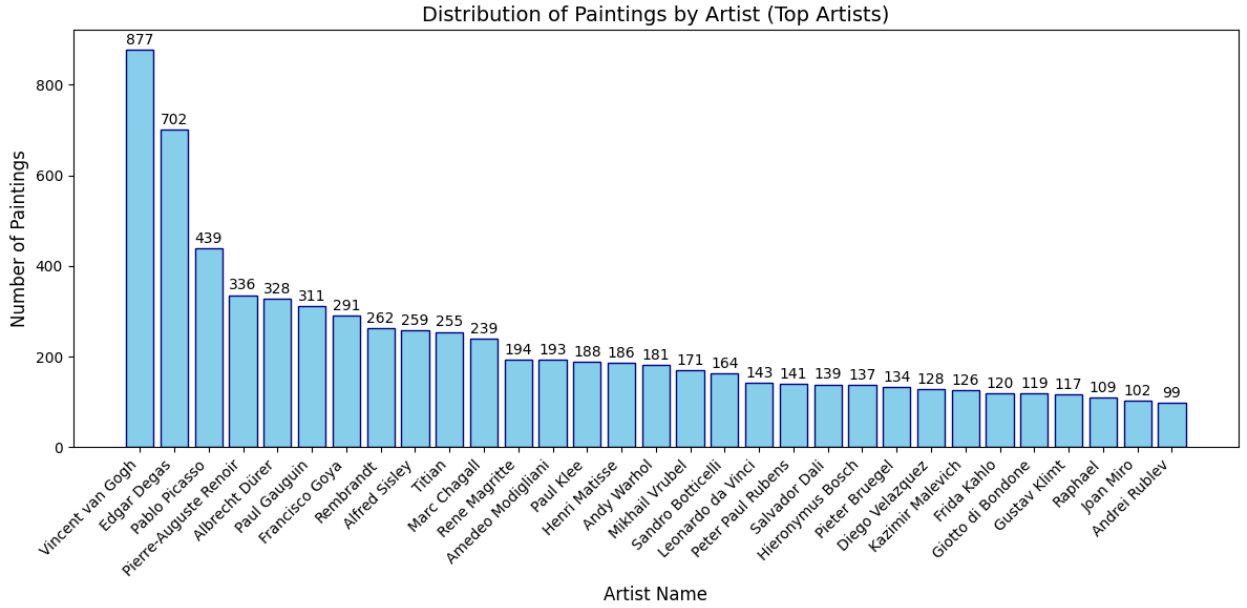
The data was scraped from artchallenge.ru during the end of February 2019.

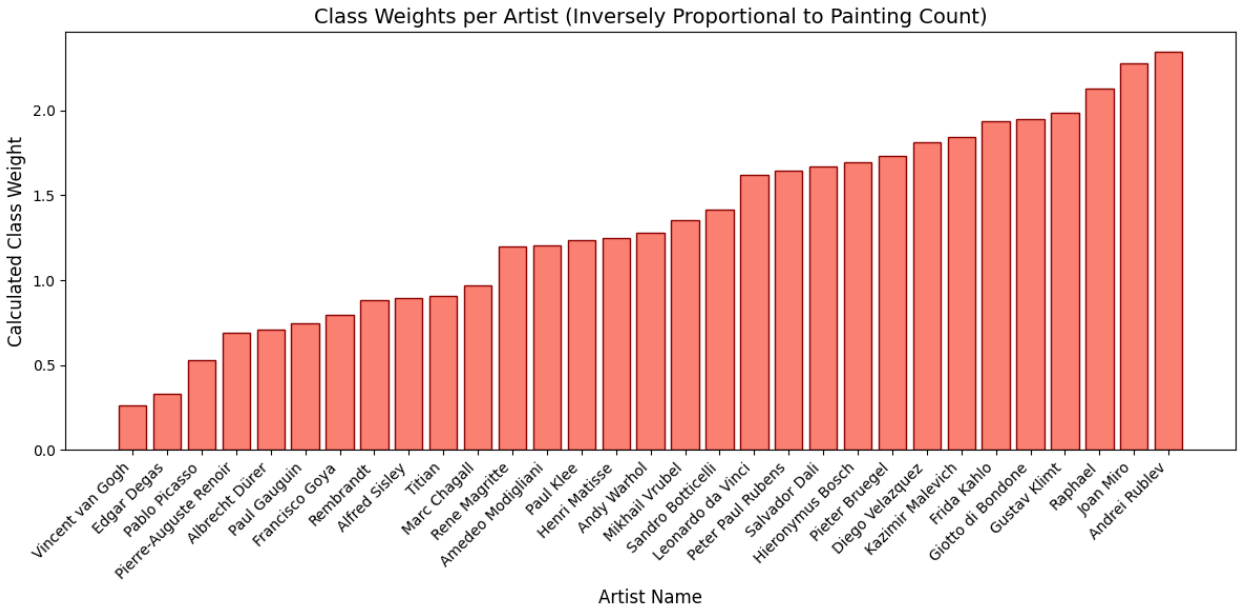
EDA

Before beginning our work in EDA, we used the 'ls' command in our raw and resized directories which uncovered an encoding issue which had led to Albrecht Dürer works being duplicated and represented with "Albrecht_DuΓòá├¬rer" and "Albrecht_Du╠êrer". This led to us doing a small preprocessing step before exploration which we will explain in detail in the Preprocessing section.

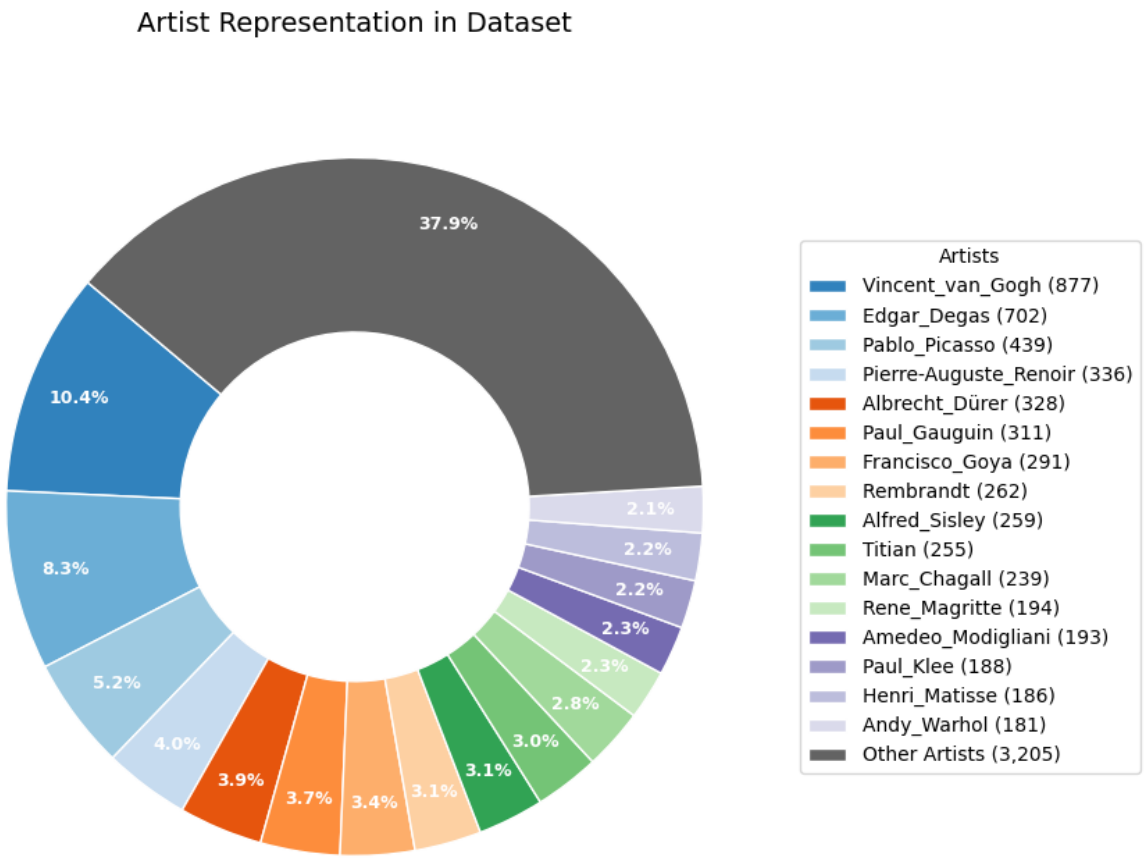
Even though the dataset advertises having 50 artists a quick dive into our data shows that not all artists have the same amount artwork which will to imbalance in training. This characteristic is visible in the following charts. For the purpose of our task, we will only be working with artists that have 99 or more artworks present in the database.


Artist / Directory
Number of images
Painting Counts per Artist by File Extension
File Format
Threshold (99)
jpg

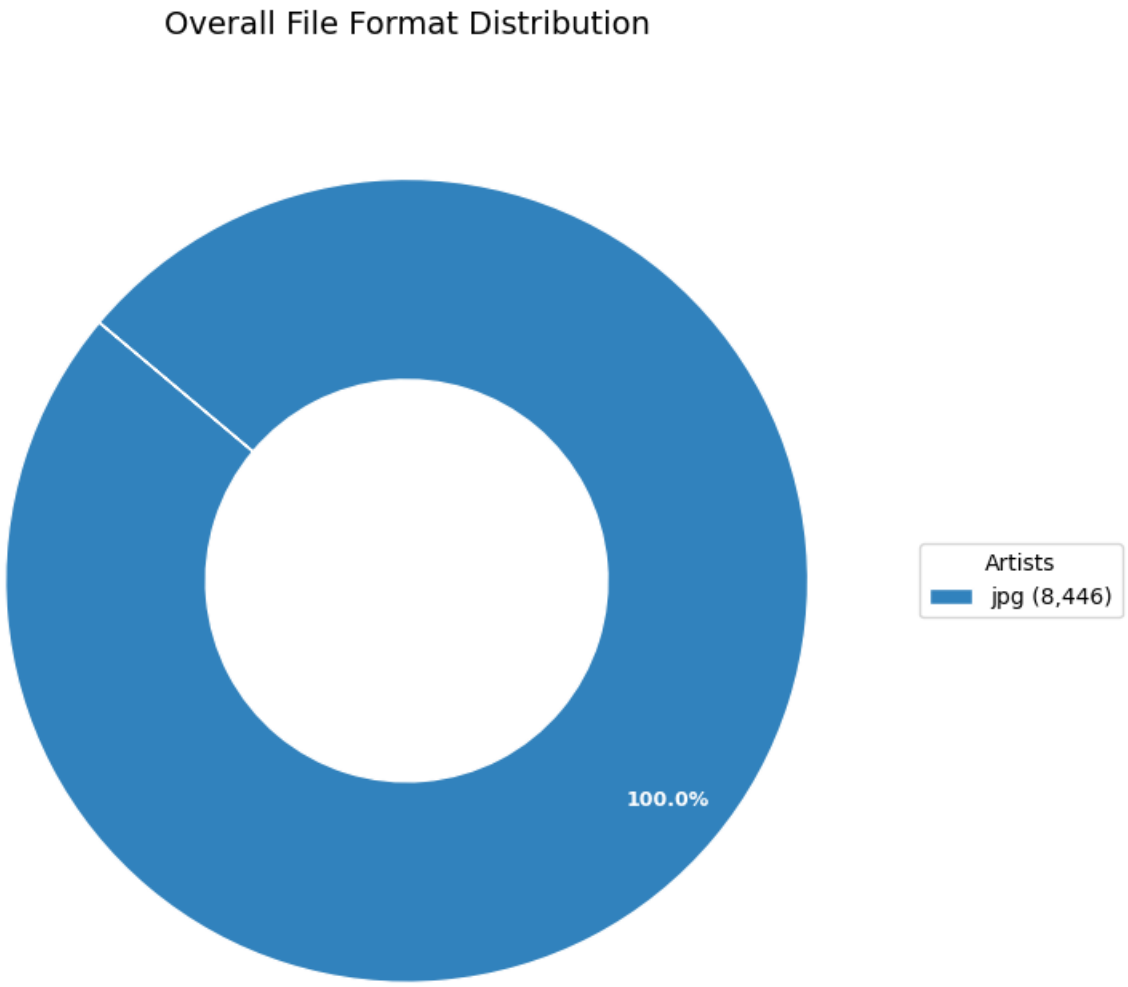

Distribution of Paintings by Artist (Top Artists)
Number of Paintings
Artist Name
877
702
439
336
328
311
291
262
259
255
239
194
193
188
186
181
171
164
143
141
139
137
134
128
126
120
119
117
109
102
99
Vincent van Gogh
Edgar Degas
Pablo Picasso
Pierre-Auguste Renoir
Albrecht Dürer
Paul Gauguin
Francisco Goya
Rembrandt
Alfred Sisley
Titian
Marc Chagall
Rene Magritte
Amedeo Modigliani
Paul Klee
Henri Matisse
Andy Warhol
Mikhail Vrubel
Sandro Botticelli
Leonardo da Vinci
Peter Paul Rubens
Salvador Dali
Hieronymus Bosch
Pieter Bruegel
Diego Velazquez
Kazimir Malevich
Frida Kahlo
Giotto di Bondone
Gustav Klimt
Raphael
Joan Miro
Andrei Rublev



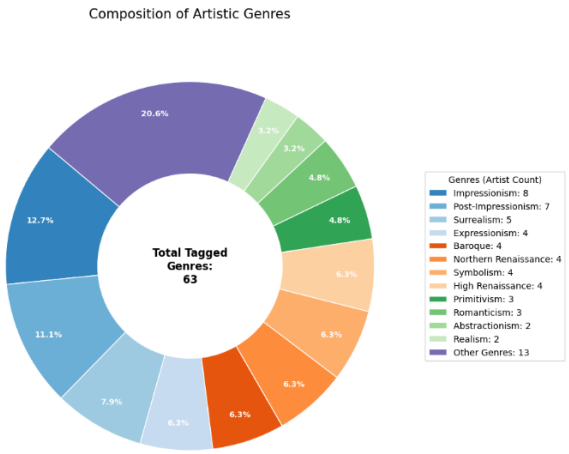
The following pie chart shows the artists representation in dataset.

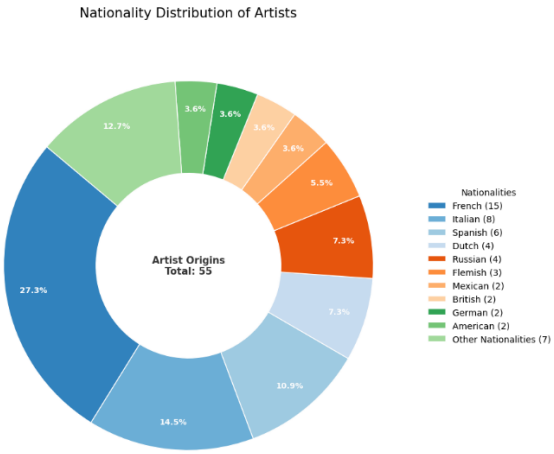


A quick look at file types shows that thankfully all images are uniformly in jpg format.



We can also analyze the nationality and genre distributions in our dataset. These two somewhat correlate with our artist representation charts.





Preprocessing

In this section we discuss the preprocessing tasks done so far. These tasks are only done on artists with 99 or more artworks present in our dataset.

Extraction (prep.py)

First, we extract the zip archive and clean up our data directory to get rid of unnecessarily nested folders.

Name correction and removing duplicate (name_correction.py)

As mentioned in the EDA section Albrecht Dürer had duplicate artwork and an encoding error regarding file names. Here we recursively remove duplicates and rename the artworks to the appropriate name.

Resizing

Here we aim to somewhat standardize and normalize our data by resizing to a model friendly 224×224 size.

Scaling method (resize.py)

Here we create a copy of the eligible data by going through the artists with 99 or more sample and scaling all of them to our desired 224×224 size. This method does not retain aspect ratio nor does it retain relativity of the painting's concepts. But it's beneficial to smaller models since all the data is useful data.

Padding method (resize_padding.py)

Here we create a copy of the eligible data by going through the artists with 99 or more sample and scaling all of them to our desired 224×224 size by applying padding to fill the empty space and retain aspect ratio and relativity of concepts. This could be problematic for smaller models since padding data is useless and model has to learn to not work the black bars around the image. However, it could improve accuracy in bigger models that have the capacity to learn and workaround paddings because of the retained proportions.

Base-model

We will use the ResNet-50 for this purpose for its relatively small size and acceptable performance. This will provide us with a good benchmark for our other models.

Experiments

We plan to experiment with at least two parameters that can be applied to every model, padding and batch size.

Additionally, we will experiment using augmentation to see how that will affect our models.

If any other interesting experiments come up in development they will be added to this section retroactively.

References

https://docs.pytorch.org/vision/main/models/generated/torchvision.models.resnet50.html#torchvision.models.resnet50

https://stackoverflow.com/a/49882055/24042168

https://www.ultralytics.com/blog/what-is-resnet-50-and-what-is-its-relevance-in-computer-vision

https://www.datacamp.com/tutorial/complete-guide-data-augmentation

https://www.kaggle.com/datasets/ikarus777/best-artworks-of-all-time

https://www.kaggle.com/datasets/steubk/wikiart