Art Movement Classification

From Machine Learning to Deep Learning to identify the artworks

Francesco Pinto, Simone Chieppa, Alessio Galimi, Eugenio Baldo
Statistical Learning project
La Sapienza - University of Rome
Master's Degree in Data Science
Academic year 2021-2022

***Abstract*—**
The aim of this project is to classify correctly artworks belonging to different art movements by using Machine Learning and Deep Learning techniques.

I. Description

Our project has been developed in the following main phases:

- Data collection
- feature extraction
- exploratory data analysis
- Models definition, implementation and evaluation.
- Performance comparison

II. Data Collection

In order to have a suitable dataset for this task we decided to scrape a famous Art Portal named wikiart. We have scraped the following categories from the website: pointillism, tonalism, popart, neoclassicism, high renessaince, futurism, baroque, minimalism and neo-expressionism. However, we decided to use only seven categories among them in the effective classification for memory reasons.

III. Feature Extraction

From the images we extracted a total of 37 features regarding four main characteristics of the artworks:
-**colors:** hue, saturation, brightness, most used colors, red, blue and green values
-**objects:** number of faces and number of objects
-**texture:** contrast, dissimilarity, correlation, homogeneity, energy, ASM
-**variances:** variance of the colors and variance of the Laplacian

IV. Exploratory Data Analysis

After extracting the features, we have observed the distribution of the latter, discovering important information regarding the art currents: the features regarding the **colors** are fundamental to recognise an artistic current, for example in the plot below we can observe the distribution of the average hue in the different currents It's clear that neoclassicism has a totally different distribution, especially from naturalism and high renaissance: it's given by the fact these two use less colors and shades than neoclassicism.

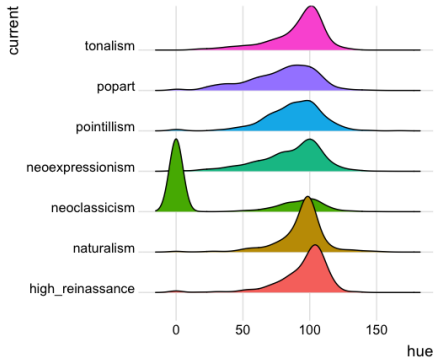

Fig. 1. Distribution of the hue for the different currents

Instead, by looking at the distribution of the **texture** features, for example the *homogeneity*, we can notice that the pointillism is not homogeneous at all.

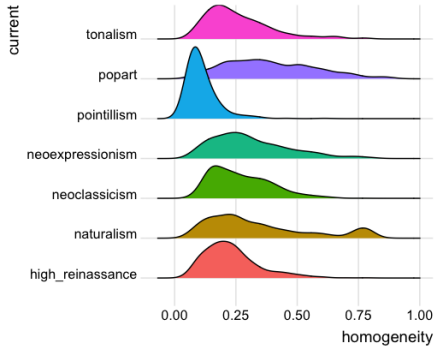

Fig. 2. Distribution of the homogeneity for the different currents

In fact, by looking at any pointillism picture, it's clear that there is no homogeneity in the image.

Some features, like *number of faces* are fundamental for the classification of currents like neoclassicism or high renaissance that have more faces into the pictures respect to pointillism, that has not.

V. Models Definition and Implementation

Of course the task to solve is a classification problem: the goal is to achieve the highest performance in forecasting what artistic current is a given picture.
Before starting the analysis, the standardization of the data is

necessary to optimize our algorithms.

Before discovering the best model we tried four different algorithms that we considered suitable for our task: Support Vector Classifier, Adaboost Classifier, XGBoost Classifier, K-Nearest Neighbour.
Although, these models didn't give us high performances. Instead the Hist Gradient Boosting Classifier turned out to fit very well with our data.

The HistGB is an ensemble method belonging to boosting algorithms. It uses weak learners (decision trees) which are added sequentially.
The model, by using the new added learners, tries to classify the previously misclassified samples.
Given that Gradient Boosting can be slow to train, a solution is to discretize the continuous feature values into bins: that's how HistGB works.
This discretization does not compromise the performance if the number of bins is sufficient.

The **parameters** used are:
-learning rate: 0.15
-Minimum sample leaf: 10
-Maximum iterations: 310
-Loss function: categorical entropy

VI. Convolutional Neural Networks

An alternative approach is to use directly the pixel values as features. This can be done by using Convolutional Neural Networks. We decided to define a custom architecture able to effectively balance bias and variance. The main facts about the architecture are the following:

- 3 convolutional layers with 128 filters and 3x3 kernel size
- max pooling with 2x2 pool size
- RELU as activation function
- dropout probability of 0.2

The images collected by wikiart, being of different shapes among them, have been resized in the shape (160,160,3), where the first dimension is the height, the second the width and the third the number of channels. In addition we have also implemented data augmentation via saturation adjustment.

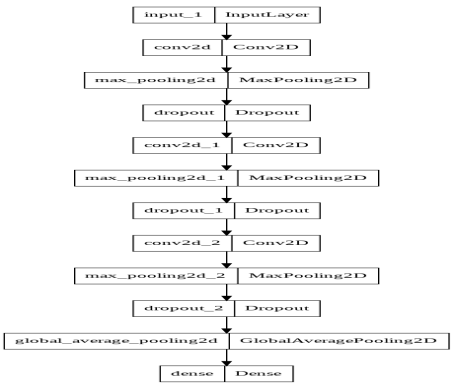

Fig. 3. Network structure

VII. Convolutional Neural Networks and SVM

The Convolutional layers in the network aim to extract features from images, so that, after the global average pooling, we end up with an embedding in a certain dimension, in our case 128d. Usually it is added a block of dense layers with a final softmax layer or simply a softmax layer which performs classification. It is possible to substitute the softmax layer with SVM in image classification as pointed out by Abrien Fred Agarap. To implement this, we have created a new model (feature extractor model) which is identical to the previously trained CNN model apart from the last layer which has been removed. The weights of the CNN model have been respectively copied in the new feature extractor model. Afterwards, this model has been used to encode all the images in 128d. With these new encoded embeddings we have performed a grid search to find the best hyperparameters of the SVM classifier. In our case we found as best parameters: C=100, kernel='rbf',gamma='auto'. Then we applied SVM with the found hyperparameters to the encoded images.

VIII. Performance

In the following table we will show the performance of the developed solutions considering different metrics. We display also the confusion matrix for the best solution we found.

Algorithm	Accuracy	Precision	Recall	F1-score
HistGradientBoosting	0.695	0.670	0.649	0.656
CNN+softmax	0.629	0.640	0.630	0.620
CNN+SVM	0.701	0.705	0.697	0.699

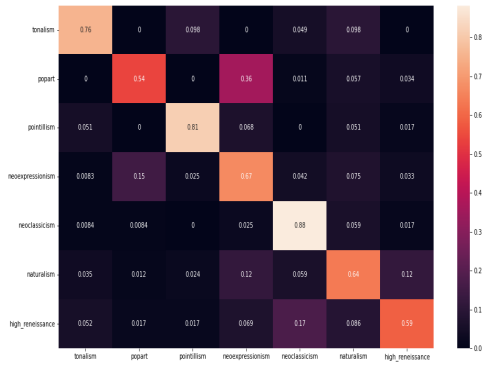

Fig. 4. Confusion matrix of the best algorithm(CNN+SVM)

IX. Conclusions

In this project we have tried different combinations of feature engineering techniques and ML-DL methods. It can be observed that the best working solution for this kind of image data are the ones which implement both a good feature extraction procedure and a powerful classifier (Hist gradient Boosting and SVM). In the future we hope to improve the results we obtained also by increasing the number of classes taken into account.