Escola d'Enginyeria de Telecomunicació i Aeroespacial de Castelldefels
UNIVERSITAT POLITÈCNICA DE CATALUNYA

MASTER THESIS

TITLE: Relate that image: A tool for finding related cultural heritage images

MASTER DEGREE: Master's degree in Applied Telecommunications and Engineering Management (MASTEAM)

AUTHOR: Byron Chacón Palacios

ADVISOR: Maria Cristina Marinescu

DATE: July, 22th 2019

Title: Relate that image: A tool for finding related cultural heritage images

Author: Byron Chacón Palacios

Advisor: Maria Cristina Marinescu

Date: July, 22th 2019

Abstract

Museums, galleries, art centers, etc. are increasingly seeing the benefits of digitalizing their art work collections – and acting on it. The more visible benefits usually have to do with advertising, involving the citizens, or creating interactive tools that get people interested in coming to museums or buying art. With the availability of these increasingly large collections, analysis of art images has gained attention from researchers.

This master thesis proposes a tool to recommend paintings that are similar to a given image of an artwork. We define different similarity measures that include criteria existent in the metadata associated with the digitized pictures (e.g. style, genre, artist, etc.), but also image content similarity. The work is more closely related to existing approaches on automatic classification of paintings, but also shares techniques with other areas such as image clustering.

Our goal is to offer a tool that can enable creative uses, support the work of gallery / museum curators, help create interesting and interactive educational content, or create clusters of images as training sets for further learning and analysis algorithms.

Acknowledgements

I would like to express my gratitude to my professor Francesc Tarrés, which was very helpful with his knowledge and experience in neural networks and image processing fields.

To Juaquim More for his selfless support and comments, being of great help especially in the area of natural language processing.

Finally, I would like to express my deepest appreciation to my adviser Cristina Marinescu, who introduced me to machine learning. Thanks for your patience, advice, and support. Without her guidance and persistent help this thesis would not have been possible.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

INTRODUCTION

Visual arts are a powerful form of representing the world around us throughout time. Images are in fact the most common form of preserving an invaluable part of our history and culture. In today's digital age there is a huge amount of artwork collections that have been digitized by museums, art centers, libraries, cultural foundations, etc. The motivation is double: on the one hand, cultural heritage is one of the best sources for understanding the history, culture, and societal influences of the past; on the other hand, it is also a tool to approach the future, find inspiration, and innovate.

Motivation

This master thesis proposes a tool to recommend paintings that are similar to a given image of an artwork. We define different similarity measures that include criteria existent in the metadata associated with the digitized pictures (e.g. style, genre, artist, etc.), but also image content similarity. The work is more closely related to existing approaches on automatic classification of paintings, but also shares techniques with other areas such as image clustering.

Our goal is to offer a tool that can enable creative uses, support the work of gallery and museum curators, help create interesting and interactive educational content, or create clusters of images as training sets for further learning and analysis algorithms. At a higher level, our project is aligned to the European Commission policy on Culture and Media, whose goal is to "protect cultural heritage and diversity across countries and harness the cultural and creative industries' contribution to jobs and growth".

Brief technical context

Many solutions are currently available for automatic classification of paintings in categories such as artist, genre, style, date, etc. [1][2][3] The classification criteria of these solutions are based on the existing metadata. The models are trained over labeled data to predict the label of a new painting. For instance, Zujovic et al. [1] train a model able to classify a painting between 5 genres.

The work we present in this thesis uses classification algorithms to offer an interactive recommendation tool for paintings. We use different machine learning and data mining techniques to retrieve paintings related to an initial painting based not only on the metadata available (author, genre, etc.), but also in the visual content and semantic information of a painting.

Objectives

To develop a good image recommendation system, we set the following objectives:

1. Based on the available data and metadata, analyze and decide which are the similarity criteria that are both desirable and achievable.

2. Create a database of image / label set pairs that in itself is a valuable resource that can be consulted on demand.

3. Design and develop an implementation of the recommendation system.

4. Analysis of the results.

Summary of results

Two different approaches are used to find similarities between paintings. The first approach uses a pre-trained convolutional neural network as feature extractor. Based on these features similar paintings are clustered into 16 different groups. The second approach uses semantic features that we extract to cluster similar paintings. This approach uses Latent Dirichlet allocation to group similar paintings into 14 different topics.

Finally, an Android application was created. This application combines both approaches to retrieve similar paintings to an input image. The implementation finds similar paintings belonging to different styles, genres, dates and authors. That is, although the paintings do not share the same metadata, our system finds groups that in most cases have clear visual similarities, even though sometimes it isn`t clear what topic they could be assigned to. In a minority of cases we do get a lot of noise, which may be minimized with more sophisticated natural language processing and information retrieval techniques that could potentially find better semantic labels. This work is outside the scope of this thesis.

Structure of the document

Chapter 1 of this document presents the state of the art related to this thesis; the principal problems and the current solutions proposed by different researchers are covered in this chapter.

Chapter 2 covers the data and the architecture of the system. It describes the different approaches and the phases we implement to develop the recommendation system.

Chapter 3 presents the analysis of the results with the different approaches. This chapter also describes the details of the implementation and of the results we obtain.

The final part covers the conclusions of the work, sustainability considerations and ethical considerations. At the end of this work, annexes contain the different algorithms used by the recommendation system.

CHAPTER 1. STATE OF THE ART

1.1. Introduction

This chapter presents a review of the state of the art related to this thesis and is divided in three sections. The first section shows the solutions for automatic classification of paintings into different categories based on existing metadata. The reviewed state of the art classifies paintings in categories such as style, genre, author, etc. The second section describes the methods used for restoration of paintings. especially in ancient paintings that suffer more damage over time. The last section shows the current methods used to find similarities between images.

Museums, galleries, art centers, etc. are increasingly seeing the benefits of digitizing their artwork collections – and acting on it. The more visible benefits usually have to do with advertising, involving the citizens, or creating interactive tools that get people interested in coming to museums or buying art. Behind the scenes, digitization is a first step to building tools that can help curating exhibits, organizing thematic cultural tours, etc. Finally, the European Commission on Culture and Media is providing increased support to those sectors whose activities are based on cultural values or other artistic and cultural institutions that work towards creating a quality database of cultural heritage such as the Europeana Foundation.

Many papers currently apply machine learning and deep learning techniques to automatically classify paintings into categories like genre, style, etc. To be able to make this classification, manually defined feature extractors and deep learning feature extractors are defined. This thesis uses the deep learning feature extraction technique.

In the area of painting restoration, paintings suffer from color degradation and cracks caused by aging. Crack restoration is based on detection of similar content of nearest zones. This approach to finding similar content could also be applied in this thesis.

We are not aware of a lot of work on finding similarities in paintings, although there exist techniques to find similarities between everyday images. Analyzing these techniques is very relevant for this thesis. Our challenge is a bit different in that to obtain good results we need to train on very large data sets; the existing models are trained on everyday images, of which there exist huge databases like ImageNet [4] or the Microsoft COCO Dataset. We, in change, used about 80000 paintings from Wikiart, so other techniques must also be used to generate good results.

1.2. Painting classification

There are different approaches for automatically classifying digital pictures of paintings. Most models are created to predict the metadata (e.g. artist, style, genre, etc). All these methods use feature extraction before the classification task. In part of the work we present here, we also use automatic feature extraction to cluster similar paintings based on these features. This section thus presents the different approaches to feature extraction used for painting classification.

There are different approaches for automatic classifying digital pictures of paintings [1][2][3]. Whether the classification is by genre, artist, style, etc., all these methods must first use feature extraction before the classification step. The automatic extraction of useful information called feature engineering is a difficult task that depends on the type of input information. Manually defined image feature extractors such as Scale-Invariant Feature Transform (SIFT) and Histogram of Oriented Gradients (HOG) or variation of those were the standards in previous years. Current deep learning models, which implement automatic feature extraction in the base layers, replaced these manually defined feature extractors[6].

Khan et al[7] uses manually defined image feature extractors such as local binary patterns (LBP) and SIFT. They perform artist and style classification applying the support vector machine algorithm (SVM). The results show 50% accuracy predicting the painter, and 60% in artistic style prediction.

A similar work is presented by Blessing and Wen [8], who classify the paintings predicting the artist's name among seven different ones. In their work they combine different manually defined image feature extractors (e.g. HOG, SIFT, LBP, etc). The results show an accuracy pf around 82% when predicting the author.

More recent research addresses the problem of image classification through the use of deep learning approaches. In this approach the classification is accomplished by training convolutional neural network (CNN) models. Deep CNN have shown to outperform previous state of the art approaches for image classification [9]. Their success must in part be attributed to the availability of large labeled training sets such as those provided by the ImageNet benchmarking initiative.

The use of CNN for paintings classification is problematic for learning descriptive features when training data is scarce. Hentschel en al. compare three approaches for paintings classification. They compare linear classifier with CNN trained from scratch and with the use of pre-trained models and they demonstrate that the use of pre-trained models outperforms different approaches [9]. An example is the work presented by Rodríguez et al. [3], where they use the transfer learning approach to obtain a low computational and data costs model. This model is able to distinguish the style of painting.

Pre-trained models are also used as feature extractors. When using the CNN as feature extractor, the last layers are removed. Then feature values can be

extracted as raw data. Paul et al use this approach to predict survival among patients with lung cancer [10]. In this thesis a similar approach is used for feature extraction from digital pictures of paintings.

1.3. Digital painting restoration

A common problem with old artwork is damage over time, for instance cracks, color degradation, etc. A way to solve this problem is digital restoration. This technique allows a visual estimation of the original appearance of the paintings based on pattern recognition, image processing, and machine learning.

Machine learning algorithms are used to first detect damaged zones and then digitally reconstruct them. The most used methods include nearest neighbors [11] and CNN [12]. The reconstruction of damaged zones is based in similar pixels around the damaged zone. This approach of finding similarities could not be applied in this thesis because is only based on some pixels, and not in the full content of the painting.

1.4. Images similarity

The most common application for retrieving similar images is in Web search engines. The images retrieved are images clustered based in low-level visual features or by metadata information.

An example of a method to detect images similarities based in low-level visual features is the work presented by Dueck and Frey. They present in their work a method to find the non-metric similarity between image pairs. This method is based in translation-invariance and the number of matching SIFT features [13].

Gao et al. propose an interesting method for clustering Web images. This method is based on the consistent fusion of the information contained in both low-level features and surrounding texts. Experiments on a real-world Web image collection showed that their proposed method outperformed the methods only based on low-level features or surrounding texts [14].

A similar approach to Gao et al. is used in this thesis. This thesis presents a method that retrieves similar paintings based on the painting’s semantic information, and the features obtained from the pre-trained CNN.

1.5. Summary

A lot of research is involved in the area of arts since collections are been digitalized a lot of paintings are available. These huge amounts of collections raise the attention of researchers. Machine learning has been used in different tasks such as classification or digital restoration of paintings.

Classification of paintings by deep learning approaches have outperform classification methods with manually defined feature extractors. These deep learning approaches use pre-trained models to offer a low computational and data costs model with higher accuracy results. These pre-train CNN models can also be used as feature extractors in the base layers. This approach is the one used in this thesis.

The work done in restoration of painting is only based on the pixels close to the damaged zone. For that reason, the state of the art of paintings digital restoration is not useful to this thesis.

The approach used to detect similarities between images is based in low-level features. The work presented by Gao et al. demonstrates that better results are obtained when working with features and surrounding texts. In this thesis the features are obtained by a pre-trained CNN, so the method to find similarities differ from the state of the art. Also, the semantic information is also used to find similarities between paintings.

CHAPTER 2. SYSTEM ARCHITECTURE

2.1. Introduction

In this thesis we use two different types of approaches to find similarities in the paintings. On the one hand, we focus on similarities based on painting content, which we approach via training a deep neural network. On the other hand, we look at more semantic features, which we tackle via labels and web entities related to the painting, as discovered by Google`s Cloud Vision API. We also take into account existing metadata from WikiArt. Both approaches follow a 4-phase process. This chapter describes the different algorithms and methods used in each process.

2.2. System architecture overview

Due to the fact that enough collections of paintings are publicly available nowadays, it was possible to obtain a dataset with about 80 thousand paintings from WikiArt, an online visual art encyclopedia. This large collection of paintings allows to have variability in terms of style, genre, period, etc. The collection data used is explained in section 2.3.

Starting from this collection of paintings, two different approaches are used to find similarities between paintings. The first approach is based on deep learning and uses a pre-trained convolutional neural network as feature extractor. The second approach - the semantic approach - uses the metadata and semantic features. Each approach follows four phases:

- **Image preprocessing:** The painting collection required preprocessing of the image files which includes normalization and resizing. The preprocessing is different in the two approaches.
- **Feature extraction:** Our first approach uses a pre-trained convolutional neural network (CNN) as a feature extractor. The second approach obtains semantic information from the paintings by the use of an existing API.
- **Feature selection:** This is one of the most difficult processes in machine learning. Its purpose is to reduces the number of features to the most significant ones. As a result, feature selection reduces the algorithm complexity, computational cost, and improves the results. Feature selection is fundamentally different for each of the two approaches.
- **Clustering:** In order to find similarities based on the semantic approach, statistical methods are used. In this case, the Latent Dirichlet Allocation (LDA) model is used to cluster paintings in topics. When working with deep features, algorithms as k-means clustering and Euclidean distance are used to find similar paintings.

Figure 2.1 shows the complete architecture of the system. At the top of the figure we illustrate the process for the deep learning approach; the semantic approach process is at the bottom of the figure.

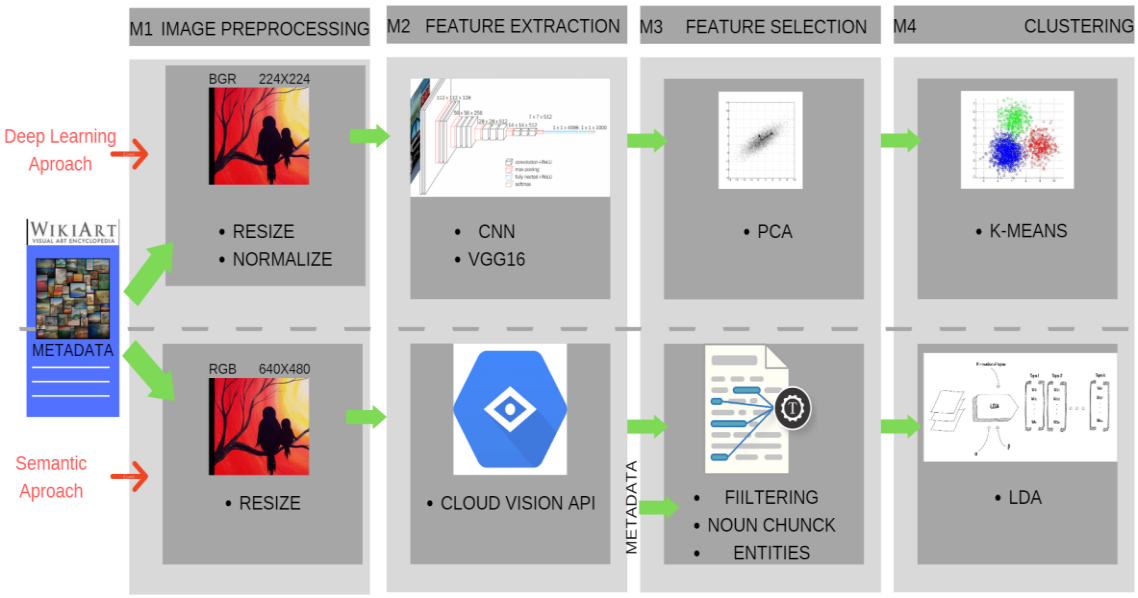


Figure 2.1 System architecture

2.3. Data

Our collection contains 79400 publicly available paintings from the Wikiart website, and a file with the metadata for each of the paintings. This collection has paintings from over 2000 artists since the eleventh century. The paintings are categorized into 140 styles and 46 genres. The metadata contains the following information for each painting:

- **Artist:** Artist's name.
- **Date:** Year painting was created, if available.
- **Genre:** Genre information from Wikiart. (e.g. flower painting, portrait, animal painting, etc.)
- **Pixels X:** Width dimension of the image.
- **Pixels Y:** Height dimension of the image.
- **Size bytes:** Image size in bytes.
- **Style:** Style information from Wikiart. (e.g. impressionism, cubism, realism, etc.)
- **Title:** Title of the painting.
- **New filename:** The image filename.

Section 2.4 describes the deep learning approach to find similarities between paintings; the semantic approach is described in section 2.5.

2.4. Deep Learning Approach

In the field of computer vision a feature is described as an interesting part of an image. When the input data (e.g. the image's pixels) is redundant and too large to be processed, it can be transformed into a set of features. Feature extraction algorithms create new features reducing the memory and computational power, but still describing the data with sufficient accuracy. As shown in the state of the art, there are many methods for features extraction. In this thesis, one of the methods we use is deep learning.

Features learned by deep networks have shown great power in a range of vision tasks, including image classification. Deep learning has been proven to be a powerful technique to extract high-level features from low-level information [15]. In this approach we use a convolutional neural network (CNN) as the tool for feature extraction.

A convolutional neural network is a deep learning algorithm. CNNs can take an image as an input, and over several layers assigns weights to various aspects of the image, being able to distinguish patterns. A CNN has different types of layers: convolutional – which compute the output of neurons that are connected to local regions in the input, max pooling – which does subsampling of the inputs, and fully connected layers – compute the class scores [16]. To train a convolutional neural network the data must be labeled. Labeled data means a group of samples (images) where each sample have been tagged with one or more labels (usually objects present in the image). Since our data is not labeled, we are using CNN as feature extractor and apply transfer learning, rather than using it for classification. This approach does not require labeled data.

The transfer learning method allows the system to apply the knowledge from previous tasks that have already been learned to a new related task [17]. The state of the art shows best results using the transfer learning method. There are many architectures of pre-trained CNNs available. In this approach the VGG16 model is used as feature extractor.

VGG refers to a deep convolutional network developed by Oxford's Visual Geometry Group for object recognition. The model is trained over the large image repository ImageNet[3]. The input of conv1layer is a fixed-size 224 x 244 RGB image. The image passes through a stack of 16 convolutional layers, and small 3x3 filters in all convolutional layers. All hidden layers are equipped with the rectified linear unit ReLU [18].

2.4.1. Image preprocessing

Every painting has to be preprocessed to be used as an input to the CNN. An advantage of CNNs is that the image preprocessing required is much lower if we compare with other classification algorithms. The preprocessing required for paintings includes resizing and normalization.

In this thesis the VGG16 model used accepts input images of size 224x224x3. The paintings had to be resized before being used as an input of the network. The preprocess also includes converting images from RGB to BGR. Finally, images are normalized by subtracting dataset's mean pixel. Namely, the following BGR values should be subtracted: [103.939, 116.779, 123.68].

2.4.2. Feature extraction with convolutional neural networks.

Feature extraction uses the pre-trained VGG16 model. This model is trained for image classification. To use this model as feature extractor, the last layer – specialized for classification - must be removed. The features are then extracted from the previous layer (first fully-connected layer) as raw data, obtaining a total of 4096 deep features per image.

The complete architecture is presented in Table 2.1. In this table, the first column indicates the name of the layer and in parenthesis the type of layer. The second column shows the shape (width, height, and depth) of each layer.

Table 2.1 VGG-16 Architecture

Layer (type)	Output Shape
input (InputLayer)	(224, 224, 3)
block1_conv1 (Conv2D)	(224, 224, 64)
block1_conv2 (Conv2D)	(224, 224, 64)
block1_pool (MaxPooling2D)	(112, 112, 64)
block2_conv1 (Conv2D)	(112, 112, 128)
block2_conv2 (Conv2D)	(112, 112, 128)
block2_pool (MaxPooling2D)	(56, 56, 128)
block3_conv1 (Conv2D)	(56, 56, 256)
block3_conv2 (Conv2D)	(56, 56, 256)
block3_conv3 (Conv2D)	(56, 56, 256)
block3_pool (MaxPooling2D)	(28, 28, 256)
block4_conv1 (Conv2D)	(28, 28, 512)
block4_conv2 (Conv2D)	(28, 28, 512)
block4_conv3 (Conv2D)	(28, 28, 512)
block4_pool (MaxPooling2D)	(14, 14, 512)
block5_conv1 (Conv2D)	(14, 14, 512)
block5_conv2 (Conv2D)	(14, 14, 512)
block5_conv3 (Conv2D)	(14, 14, 512)
block5_pool (MaxPooling2D)	(7, 7, 512)
flatten (Flatten)	(25088)
fc1 (Dense)	(4096)
fc2 (Dense)	(4096)
predictions (Dense)	(1000)

Annex 1 contains the complete python code for feature extraction with the pretrained model VGG-16.

2.4.3. Feature selection

The number of features obtained with the CNN is 4096. The computational power and time required for processing all these features are excessive. We implement Principal Component Analysis (PCA) algorithm to reduce the number of features.

PCA is a statistical technique to convert high to low dimensional data by selecting the most important features that capture maximum information about the dataset. The features are selected on the basis of the variance that they explain in the output. This algorithm does linear dimensionality reduction using Singular Value Decomposition (SVD) of the data to project it in a lower dimensional space [19].

This algorithm requires as an input the number of components desired for the output data. The way to select the correct number of components of the output was based on the covariance of the data. We first select 0.95 as the variance retained parameter, which means that the minimal number of principal components will be chosen such that 95% of the variance is retained. This result in 1171 components. We also tested with other values for the retained variance (90 and 85%). Table 2.2. shows the different number of components obtained.

Table 2.2 Number of Components after PCA with different fractions of Variance

Variance Retained	Number of Components
0.95	1171
0.90	880
0.85	687

In the end we chose 880 as the number of components for the PCA algorithm. With 880 components the computational power required is not excessive, and a 90 % of variance retained results in a good representation of the original features. The data was thus reduced from 4096 to 880 new components per image. In the next step we group similar paintings based on these values. The code for the PCA implementation is available in annex 2.

2.4.4. K-Means Clustering

In this step we cluster the paintings based on their features. Since the features represent image content, each cluster will represent a collection of paintings which are similar in terms of content. There are many algorithms for clustering. We use k-means clustering.

Clustering based on K-means is considered a fast method because it is not based on computing distances between all pairs of data points. K-means finds k centers and works iteratively, assigning each data point to one of the groups such that the sum of the distances from data points to centers is minimized.

To select the appropriate number of cluster we need to run K-means for a range of K values. A useful method to select the correct number of clusters is to represent the mean distance from the data points to the centroids. This represents the compactness of the clustering, while the goal is to minimize this distance. This is also called the elbow method.

Figure 2.2. shows the results of elbow curve for values from 2 to 50 clusters; it is not easy to determine from this plot the appropriate number of clusters. For this reason we also implement the *average silhouette method*. This method determines how similar an object is to its own cluster. Like in the elbow method, here the average silhouette is computed for different values of K. The optimal number of clusters K is the one that maximizes the average score. Figure 2.3. shows the results of the silhouette curve.

From the elbow and silhouette curves we select 16 as the number of clusters K. The method used is Elkan's k-means algorithm [20], which works faster with a high number of dimensions.

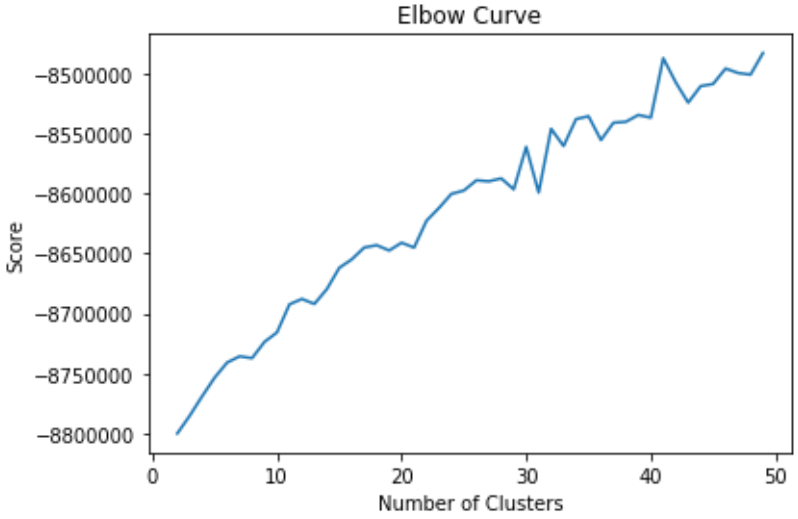


Figure 2.2 Elbow curve result

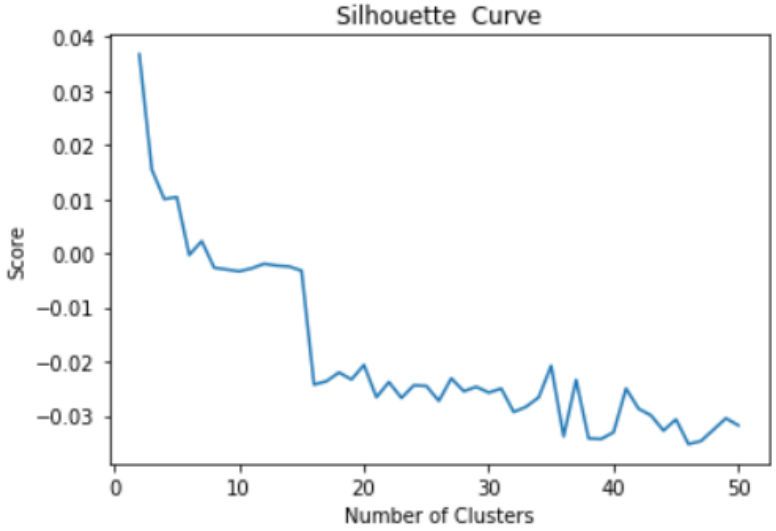


Figure 2.3 Silhouette curve results

2.5. Semantic Approach

The second technique we use to find similarities between paintings is based on semantic information we could associate with a painting. These semantic features describe the visual content of a painting in a more conceptual way, and include metadata that may contain information about the title, author, period, style, etc.

Extracting information at a semantic level refers to the extraction of concepts or descriptions of the image at levels that may be more abstract than the image content itself or more detailed than the extraction of objects in the image. This is the type of information that one would like to have as rich metadata. The semantic features are information that usually capture some form of common sense that is impossible to extract simply from the pixels.

There exist some cloud computing solutions that allows us to build natural language features or to extract more general descriptions from images. Some of the most popular are:

- Amazon Rekognition API[1]: This service can identify the objects, people, text, scenes, and activities, as well as to detect any inappropriate content [21].
- Microsoft Computer Vision API: Microsoft created this API able to process images and return information as brands, colors, pornographic content, faces recognition, object detection, and image description and tagging[22].
- Google Cloud Vision AI API: This API can assign labels, detect objects and text, and build valuable metadata from images[23].

In this thesis, we used Google Cloud Vision AI API to extract the semantic information from images. We also use the existing metadata on the WikiArt website.

2.5.1. Image preprocessing

Similar to the CNN requirements, Cloud Vision requires preprocessing of the paintings. The API allows a minimum image size of 640 x 480 pixels, and a maximum size of the file of 10 Mbytes. The pictures of paintings that do not meet the requirements must be resized.

2.5.2. Semantic features extraction

Cloud Vision API is a cloud computing solution that analyzes the content of an image and returns contextual data such as labels, objects, etc. Cloud vision utilizes machine learning models pre-trained on a large dataset of images (ImageNet). Vision API can perform feature detection on a local image file by

[1] API – Aplication Programing Interface

sending the contents of the image file as a base64 encoded string in the body of your request.

The API has multiple vision detection features available and those are:

- Face detection.
- Landmark detection.
- Logo detection.
- Label detection.
- Text detection.
- Document text detection (dense text / handwriting).
- Image properties.
- Object localization.
- Crop hint detection.
- Web entities and pages.
- Explicit content detection (Safe Search).

We use labels and web entities as semantic labels for the paintings.

2.5.2.1. Label detection.

The Vision API can detect and extract information about entities in an image across a broad group of categories, and provide generalized labels based on the Google Knowledge Graph. Google Knowledge Graph uses the relationships between words and concepts to understand the context of a query and to assign specific meaning to user queries.

Labels can identify general objects, locations, activities, animal species, products, and more. For each label the API returns a textual description, a confidence score, and a topicality rating.

- **Mid:** contains a machine-generated identifier (MID) corresponding to the entity's Google Knowledge Graph entry.
- **Description:** the label description.
- **Score:** the confidence score, which ranges from 0 (no confidence) to 1 (very high confidence).
- **Topicality:** The relevancy of the label to the image. It measures how important/central a label is to the overall context of a image.

Figure 2.4 Sunflower painting by M.C. Escher.

As an example Figure 2.4. represents the painting “Sunflowers” by M.C. Escher. Figure 2.5 shows all corresponding labels and their scores. We can see from the labels the visual characteristics of the image, but also more abstract – and in this case, not so precise - categories such as Fictional Character. Note however that, although the human eye recognizes the drawing to be a flower (and an ambiguous object in the top right corner), the option of a fictional character isn`t so far fetched. One can pretty easily see an eye and a weird flying object with tentacles. Given that learning can only be as good as the training data, producing a generic label such as Fictional Character is not that surprising – although it does introduce noise in the system. The generation of the precise label Fictional Character probably comes from the WordNet category with this name.

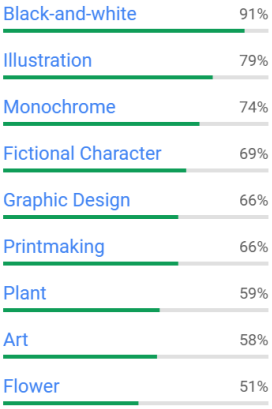


Figure 2.5 API Label detection response for Sunflowers painting.

The response of the API shows other interesting results. We can see the labels: “Black-and-white”, “Monochrome”, “Plant”, and “Flower”. These labels are good features that describe the content of the painting. On the other hand, the API also returns labels like “Illustration”, “Graphic Design”, “Printmaking”, “Art”. These labels are general and are present in most of the images since all the images are about paintings. Section 2.5.3 describes the method used to filter these type of labels out.

2.5.2.2. Web detection.

Web Detection detects Web references to an image. This detection is based on the technology of Google Reverse Image Search to find entities belonging to a certain theme. Web detection has different features available:

- **Web entities:** Inferred entities (labels/descriptions) from similar images on the Web.
- **Best guess label:** A best guess as to the topic of the requested image inferred from similar images on the Internet.
- **Full matching images:** A list of URLs for fully matching images of any size on the Internet.

- **Partial matching images:** A list of URLs for images that share key-point features, such as a cropped version of the original image.
- **Pages with matching images:** A list of Webpages (identified by page URL, page title, matching image URL) with an image that satisfies the conditions described above.
- **Visually similar images:** A list of URLs for images that share some features with the original image.

The response of the last four features are lists of URLs with images somehow related to the input image and they aren`t very useful for us, while “Web entities” and “Best guess label” features are. These features shows descriptive words inferred from similar images on the internet.

Figure 2.6 shows the web entities response from the API when we use the Sunflower painting as the input image. For this painting, the best guess label response is "mc escher sunflowers" which is the name of the artist and the painting. Notice that some of the labels are artwork by the same author.

Figure 2.6 Web detection response for Sunflower painting.

The complete python code for semantic feature extraction ussin Cloud Vision API is available in annex 3.

2.5.3. Semantic features selection

Semantic features make it possible to understand the content of a painting at a more abstract level that may be obvious and intuitive to a human, but impossible to deduce solely by automatic algorithms that don´t embed common sense, such as machine learning and deep learning algorithms.

In our case the set of semantic features associated to a painting are features extracted by Cloud Vision API and the metadata information of the painting from Wikiart. The feature selection algorithm first filters very generic words present in

most of the paintings, which do not bring any information specific to the image, for instance: 'Painting', 'Art', 'Visual arts', 'Artwork', 'Illustration', 'Paint', 'Drawing', 'Artist', 'Painter', 'Work of art', 'Photography', 'Canvas', 'Wikiart', etc.

After filtering these stop-words, noun chunk phrases are extracted. Noun chunk phrases are proper or common nouns, as well as noun phrases composed of noun plus the words describing the noun (adjectives, determiners, numbers and quantifiers). The idea is that the objects described in the image – things that are *seen* - will be nouns or noun phrases, but not verbs or other parts of speech.

For each painting the input features for the feature selection algorithm include the title and semantic information extracted from cloud vision API. The title was included because in many cases it describes the content of the painting. For instance, here we show two paintings. The first painting is “Snowy landscape with a woman brandishing a broom and a man holding an umbrella” by Utagawa Kunisada (Figure 2.7); its title faithfully describes its content. On the other hand, Figure 2.8 shows the painting “Sunshine and shadow” by Hans Heysen where the title is not enough to understand the painting’s content. The style and genre (in the metadata) are not included in the feature selection algorithm because they don`t need any preprocessing.

Figure 2.7 Painting “Snowy landscape with a woman brandishing a broom and a man holding an umbrella" by Utagawa Kunisada

Figure 2.8 Painting "Sunshine and shadow" by Hans Heysen

Table 2.3 shows the inputs to the feature selection algorithm for these two paintings. Table 2.4 shows the results after filtering out proper nouns and extracting chunk nouns. The complete python code for semantic feature selection is available in annex 4.

Table 2.3 Data inputs example for two different paintings

Title	Web Entities	Label entities
Snowy landscape with a woman brandishing a broom and a man holding an umbrella	The Tale of Genji	Tapestry
	Ukiyo-e	Wall
	Japanese art	Textile
	Ukiyo	Mural
	Landscape painting	Watercolor paint
	Snow	World
	Printmaking	
	Woodblock printing	
	Woodcut	
Sunshine and shadow	Tree	National Gallery of Victoria
	Natural environment	Sunshine and shadow
	Woodland	Oil painting reproduction
	Forest	Oil painting
	Trunk	Landscape painting
	Grass family	Brachina Gorge
	Woody plant	Art museum
	Northern hardwood forest	Oil paint
	Plant	

Table 2.4 Example of features processed output for two paintings

Painting	Processed features				
"Snowy landscape with a woman brandishing a broom and a man holding an umbrella"	Woman	Broom	Man	Umbrella	The Tale
	Genji	Ukiyo-e	Japanese art	Ukiyo	Landscape painting
	Snow	Printmaking	Woodblock printing	Woodcut	Tapestry
	Wall	Textile	Mural	Watercolor paint	World
"Sunshine and shadow"	Sunshine	Shadow	National Gallery	Victoria	Sunshine
	Oil painting reproduction	Landscape painting	Brachina Gorge	Art museum	Tree
	Natural environment	Forest	Trunk	Grass family	Woody plant
	Northern hardwood forest	Plant			

The processed features in Table 2.4 also make it easier to filter out words depending on the similarity criteria, for instance type of medium, provenance, museum, artist, etc.

2.5.4. Latent Dirichlet allocation (LDA) clustering

Latent Dirichlet allocation is a generative probabilistic model for collections of discrete data like text, in which each item of a collection (i.e. document) is modeled as a mixture over an underlying set of topics. LDA finds topics of the members of the collection that enable efficient processing of large collections while preserving the essential statistical relationship that are useful for basic tasks such as classification, novelty detection, summarization, and similarity and relevance judgments. [24]. We consider a document to be the set of semantic features obtained as a result of the previous feature selection phase.

For a given number k of latent topics across the whole collection of documents, LDA assigns each document a percentage of contribution to the topics. We preprocess the data before passing it to the LDA algorithm to perform stemming[2], lemmatizing, [3]and filtering of stop-words to improve the performance of the system. Stop-words include the information we are not interested in. For instance, artist does not describe the content of the painting, so the stop-words includes the list of all artists in our collection.

The typical approach to find the optimal number of topics is to build an LDA model for a value of k in a range and compute, for each one of these models, the coherence value. The optimal model is the one with the highest coherence value. Figure 2.9 suggests a number of topics of 14.

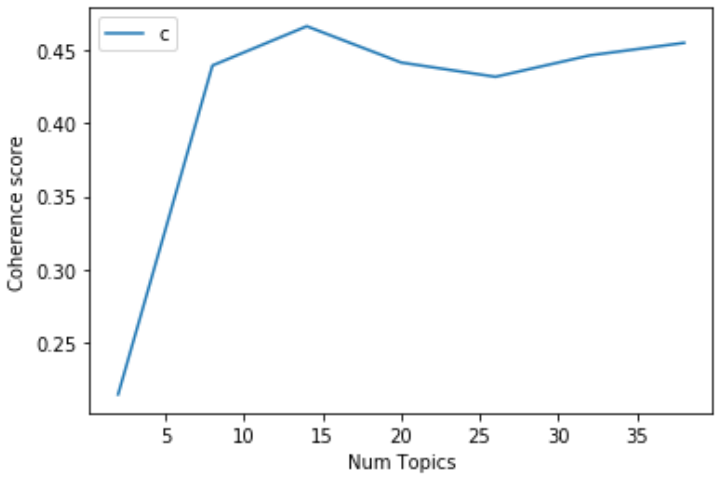


Figure 2.9 LDA coherence results.

[2] Stemming – the process of reducing inflected word to their base or root form.

[3] Lemmatizing – the process of removing inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma.

The final step is to assign each painting to the corresponding dominant topic and calculate the percentage of contribution to that dominant topic. The full LDA algorithm is available in annex 5.

2.6. Summary

This chapter presents the overall architecture of the system, with its two approaches for learning similarities between paintings. The first criteria of similarity are based only on the images themselves and are obtained via deep learning and k-means clustering. Because of the large number of features and paintings, PCA is used to reduce the number of features.

The second criteria are more semantic, as they not only rely on concrete image (pixel) content, but they are text descriptions that reflect other, more common sense information and existing painting metadata. These text descriptors are filtered and processed to finally cluster painting in different topics using LDA.

CHAPTER 3. RESULTS

3.1. Overview

This chapter first presents the results obtained when working separately with deep features and with semantic information. We comment on the clusters we obtain, both those that are very consistent and those that are surprising or present a lot of noise.

We then introduce an application which allows a user to choose and drag a painting in a predefined window and uses the system we detailed in this work to return similar paintings.

3.2. Results with deep learning approach

From the elbow and silhouette curves we selected 16 as the number of clusters in the deep learning approach. To analyze each cluster, we visualize the paintings closest to the center of the cluster. We use Euclidean distance to measure this distance.

Figure 3.1 shows the mosts representatives paintings of cluster 3 on the left, and 12 on the right. Both clusters contain exclusively portraits, and the algorithm groups them differently based on the shape of the frame. This may seem an irrelevant criteria at first, but it may not be so depending on the final user.

Figure 3.1 Most representatives paintings of: (a) cluster 3 (b) cluster 12.

This results are not based on the style, genre, or the metadata information. They are exclusively based on the pixel content. The only picture that is somewhat surprising is the religious painting, which contains more than one person.

Figure 3.2 illustrates the ten paintings closest to the center of the cluster 15. In this cluster it is possible to see visual similarities such as many people and animals present in each painting. The similarities in this cluster are not so obvious as in the previous clusters.

Figure 3.2 Most representative paintings of cluster 15.

Table 3.1 contains the information of these paintings. In this table we can appreciate that the grouped paintings belong to different styles, genres, dates, and authors.

Table 3.1 Metadata of closest paintings in cluster 15

Artist	date	Genre	Style	Title
Konstantin Bogaevsky	1925	landscape	Symbolism	Clock towers of Alupka Palace
Sandro Botticelli	1482	religious painting	Early Renaissance	The Youth of Moses
Peter Paul Rubens	1630	history painting	Baroque	Triumphal Entry of Henry IV into Paris
Vittore Carpaccio	1507	religious painting	High Renaissance	St. George Killing the Dragon
Lucas Cranach the Elder	1530	religious painting	Northern Renaissance	Sinking Of The Pharaoh In The Red Sea
Vasily Perov	1880	sketch and study	Realism	Walkers-seeking
Tintoretto		religious painting	Mannerism (Late Renaissance)	Christ washing the Feet of the Disciples
Henri Martin		genre painting	Divisionism	The Pergola in Marquayrol
Grégoire Michonze		genre painting	Naïve Art (Primitivism)	Village Street
Vasily Surikov	1873	religious painting	Realism	A rich man and Lazarus

In all clusters it is possible to see similar paintings and see the type of paintings in each cluster. As we saw clusters 3 and 12 contains paintings of portraits, cluster 4 contains landscapes with mountains, cluster 9 forests and so on. All the clusters obtained with deep learning approach are available in annex 6.

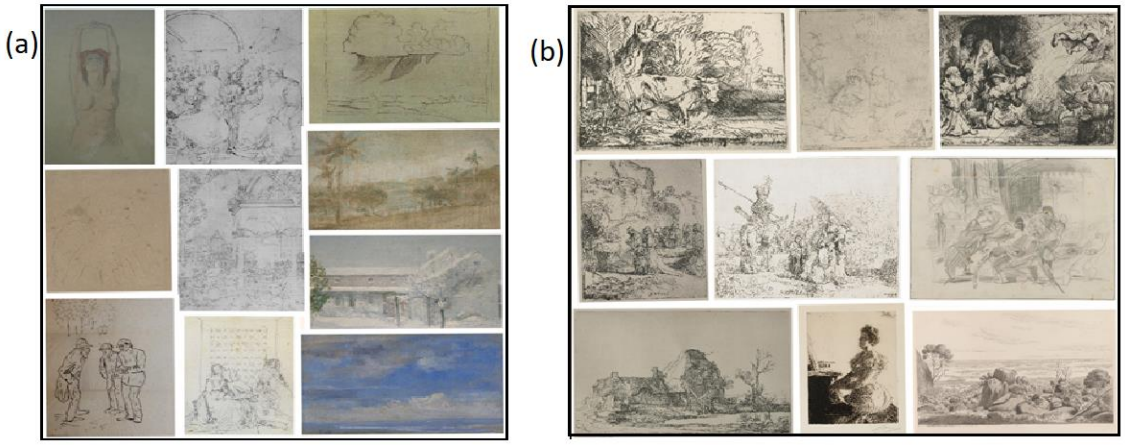


Figure 3.3 Most representative paintings of: (a) cluster 1 (b) cluster 2.

Other interesting result with deep learning approach is appreciated in Figure 3.3. These are the most representative paintings for clusters 1 and 2. We can observe similarities between the paintings, but if we focus on the content of each painting we can see that in the same cluster we have paintings with people and other paintings with landscapes.

The next section describes the results using the semantic approach, which clusters paintings based on the semantic features. This improve the results of the deep learning approach.

3.3. Results with semantic approach

Based on semantic information we cluster the paintings in 14 different topics. LDA does not assign names to the topics. A topic can be represented by the words occurring in that topic and their relative weight. The list of all topics is described in annex 7.

If we look at the words with the highest percentage of contribution to the topic, some of the topics are very well formed. For instance, figure 3.4 (a) shows the paintings with the highest percentage of contribution to topic 0. Analyzing this topic, we see that it is represented by plants and flowers. Topic 11 (Figure 3.4.b) shows paintings of landscapes with trees. At the figure 3.4 you can bee see that the most representative paintings of these topics show high visual similarities. Other well-formed topics are: topic 3 about water and (water) banks or cliffs, topic 7 about landscapes, topic 8 about mythology, and topic 9about architecture.

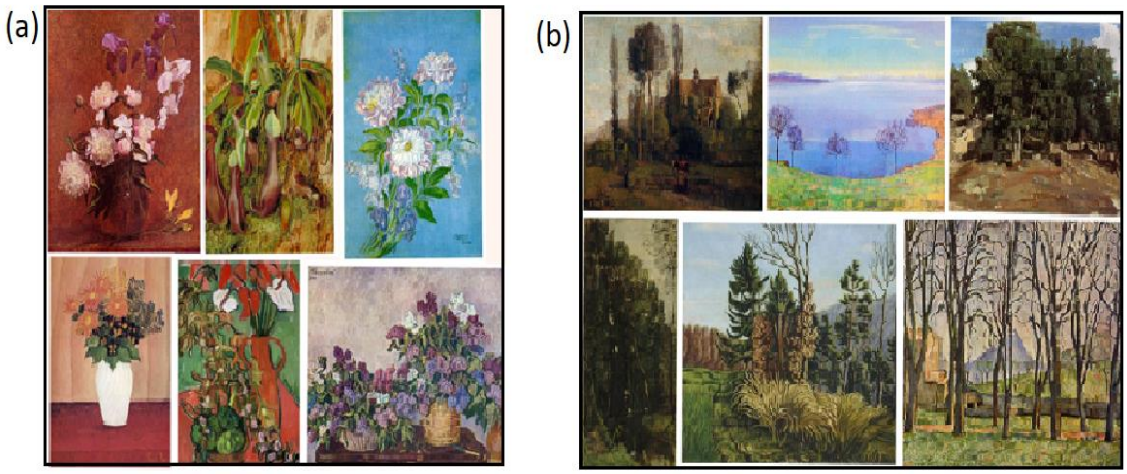


Figure 3.4 Most representative pictures for well-defined clusters.

Some other topics ones are a rather bizarre mix. For instance, figure 3.5(a) contains the most representative paintings of topic 1. Here you can see that the paintings with the greatest contribution in the topic are birds and boats. When analyzing the topic 1 we could see that the content of this topic includes birds and ships. In the figure you can see that there are similarities between the paintings that contain birds. Also, there are similarities between those that contain ships. However, if we compare paintings that contain birds with those that contain ships, they do not share many similarities.

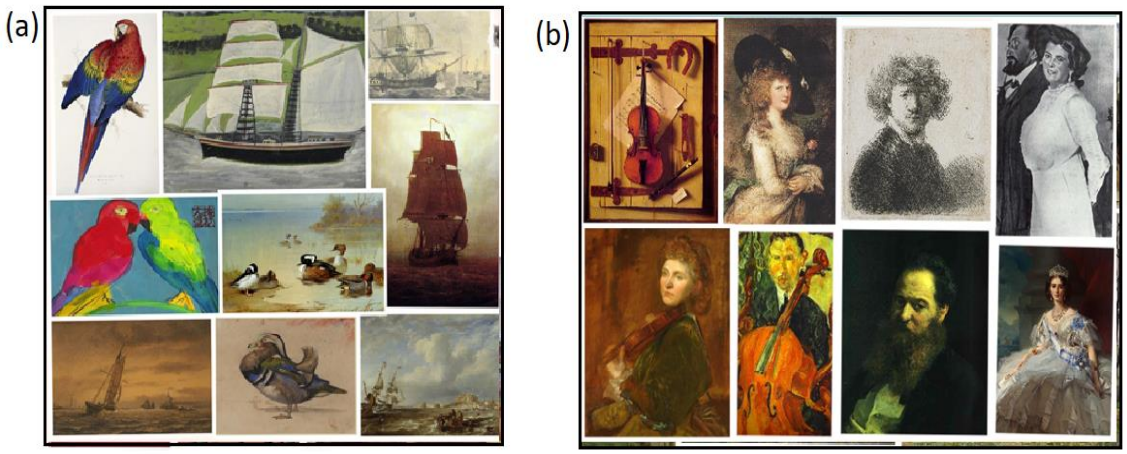


Figure 3.5 Most representative pictures for not well-defined clusters.

Figure 3.5 (b) shows another example of a not well-defined topic. The figure represents the topic 5 that seem to be about portraits, but three of the most representative paintings include violins.

Both the deep learning and the semantic approaches show pretty good results, but there are also problems in some clusters. Combining these approaches improves the results. Next section describes the application created to do that.

3.4. Related paintings application

We created an application that allows us to test the system more easily with users. The application is developed for mobile phones using the Android operating system.

The application consists of two panels. In the main panel the user chooses and visualizes the input painting. The paintings available are from the dataset described in section 2.3. Figure 3.6 shows the appearance of the main panel.

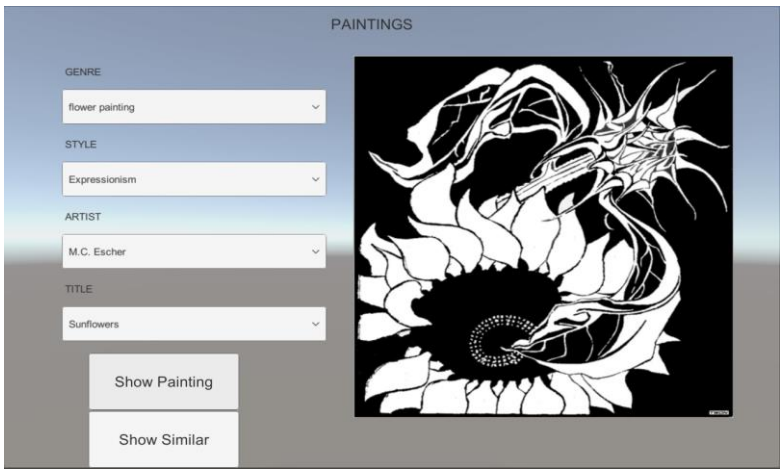


Figure 3.6 App main panel.

Similar paintings are displayed in the second panel, including all their information. By default the application shows similar paintings based on the deep learning approach. The application allows the user to obtain the similar paintings combining both the semantic and the deep learning approach. Other features included are to filter the results by style or genre.

Figure 3.7 shows the results for the painting an the previous figure. On the right are displayed the results obtained with deep learning approach, and on the left the results combined with the semantic approach.

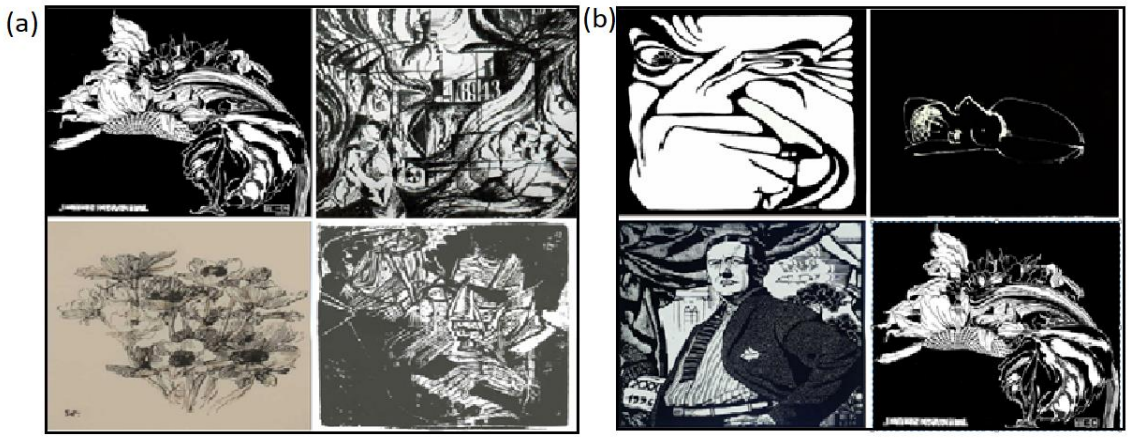


Figure 3.7 a) Similar paintings combining approaches b) Results with deep learning approach

One can see that in both result sets there are paintings similar in certain aspects to the reference painting. In fact one of the similar paintings is the same with both approaches. It can also be observed that when combining the two approaches, better results are obtained. This is because apart from visually being similar, the paintings are required to be the same topic so you can not see the paintings of people that can be observed in the deep learning approach.

An interesting result was obtained by trying one of the paintings that showed problems when using the deep learning approach. The painting “Nude Drawings” by Raphael Kirchner (left of figure 3.8) is used as the reference painting. In Figure 3.8 you can see that the results with deep learning are not so good, given that the system returns landscapes and abstract paintings, while the reference painting is the portrait of a woman. By combining deep learning with the semantic approach the results improve considerably. This can be seen at the right of Figure 3.8. The similar painting obtained with the combined approach is also a portrait of a woman and in a certain way similar to the reference painting. A thing to consider is that when combining both approaches the number of similar paintings obtained by the system is smaller. In the case of “Nude Drawings” only one similar painting was obtained.

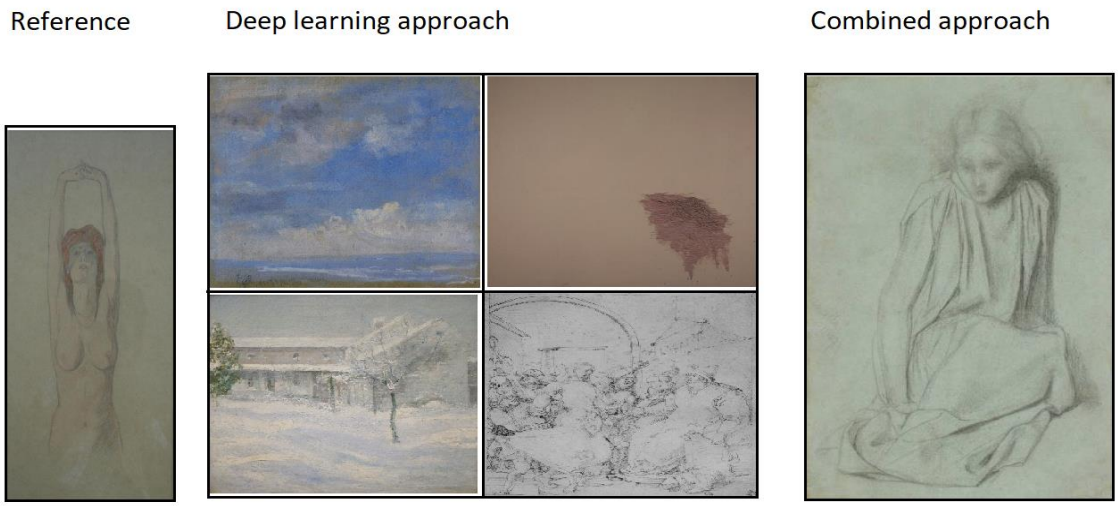


Figure 3.8 Results comparison (combined vs deep learning approach)

Similarly, one of the paintings in topic 1 that presented problems in the semantic approach was tested. The painting “Macrocercus aracanga” by Edward Lear shows the portrait of a parrot standing on a stick (left of Figure 3.9). The results with semantic approach show paintings of birds and ships at the center of Figure 3.9. When combining the two approaches, the results improve since the boat paintings are eliminated and the results obtained are much more similar to the reference painting. This can be clearly seen on the right of Figure 3.9.

Figure 3.9 Results comparison (combined vs semantic approach)

The results show that combining the two approaches returns much more similar outputs. This can be best appreciated when testing with paintings from the clusters that showed poor results individually. One thing to keep in mind is that by combining the two approaches the number of results decreases considerably.

3.5. Dataset

In this master thesis we have compiled a collection of paintings and their enriched metadata. The paintings and some original metadata have been obtained from wikiart.org. This collection contains 79400 paintings from over 2000 artists. The paintings are categorized into 140 styles and 46 genres.

Based on this collection we created a repository to make this information publicly available, along with the metadata we extracted, to be used only for non-commercial research purposes. In addition to the original metadata, we included the features obtained through the deep learning and semantic approaches.

The data and metadata can be downloaded free of charge at the following link: https://drive.google.com/drive/folders/1RLLmLhvfvmvuSVw-9GnQXOoNarO6AfQH?usp=sharing. Here you can find three datasets in CSV format. The files "Metadata.csv", "SEMANTIC.csv" and "DEEP-FEATURES.csv". These files contain the original metadata information, semantic information, and deep features respectively. In each of these datasets there is an identifier column "ID" that allows one to relate the datasets. This ID also corresponds to the name of the image file.

CONCLUSIONS

In this Master Thesis we propose a system able to find similar paintings to an input painting. The system uses two different approaches. The first approach is based on features obtained by a pre-trained convolutional neural network. The second approach uses semantic information of the paintings. Each approach consists of several modules: image preprocessing, feature extraction, feature selection, and clustering.

Our results show how both approaches are useful to group paintings in similar clusters. Although most clusters make intuitive sense, we also obtain some unexpected results and some noisy clusters that would require richer semantic information to be able to perform smarter disambiguation. The best results are obtained by combining the two approaches.

An Android application was created which uses the system proposed in this thesis to retrieve similar paintings to a reference painting. In the application both approaches are combined showing much better results.

In this thesis we created a repository of 79400 paintings from over 2000 artists, collected from Wikiart, and their enriched metadata. The repository includes the features obtained with the two approaches presented in this thesis and is made publicly available for non-comercial use.

Sustainability considerations

This thesis creates an interactive tool that could increase the interest of people in art, help with the selection of paintings in exhibits, organize thematic cultural tours or pamphlets, etc. From an economic point of view, more people interested in art, going to galleries, museums, etc. generates a flow of money that supports the economy. Additionally, we make our dataset available to anyone that may use it, including other application writers.

The social impact of art is high. Art influences society and allows viewers to see the world from a new perspective. It is a tool for understanding the history, culture, and societal influences of the past, but at the same time it is a powerful tool to approach the future, find inspiration, and innovate.

Ethical considerations

In this project we worked with a very large number of paintings. Many artworks online are copyright protected. In this thesis we only worked with the collection from WikiArt. WikiArt presents mostly public domain artworks. In the case of copyright protected artwork, they are enabled for educational purposes and in low resolution unsuitable for commercial use.

ACRONYMS

SIFT	Scale-Invariant Feature Transform
HOG	Histogram of Oriented Gradients
SVM	Support Vector Machine
LBP	Local Binary Patterns
CNN	Convolutional Neural Network
ReLU	Rectified Linear Unit
AI	Artificial Intelligence
ICA	Image Content Annotation

REFERENCES

[1] J. Zujovic, L. Gandy, S. Friedman, B. Pardo, y T. N. Pappas, «Classifying paintings by artistic genre: An analysis of features classifiers», en *2009 IEEE International Workshop on Multimedia Signal Processing*, 2009, pp. 1-5.

[2] B. Saleh y A. Elgammal, «Large-scale Classification of Fine-Art Paintings: Learning The Right Metric on The Right Feature», *ArXiv150500855 Cs*, may 2015.

[3] C. S. Rodriguez, M. Lech, y E. Pirogova, «Classification of Style in Fine-Art Paintings Using Transfer Learning and Weighted Image Patches», en *2018 12th International Conference on Signal Processing and Communication Systems (ICSPCS)*, 2018, pp. 1-7.

[4] J. Deng, W. Dong, R. Socher, L. Li, K. Li, y L. Fei-fei, «Imagenet: A large-scale hierarchical image database», en *In CVPR*, 2009.

[5] T.-Y. Lin *et al.*, «Microsoft COCO: Common Objects in Context», *ArXiv14050312 Cs*, may 2014.

[6] A. Zheng y A. Casari, *Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists*, 1st ed. O'Reilly Media, Inc., 2018.

[7] F. S. Khan, S. Beigpour, J. van de Weijer, y M. Felsberg, «Painting-91: a large scale database for computational painting categorization», *Mach. Vis. Appl.*, vol. 25, n.º 6, pp. 1385-1397, ago. 2014.

[8] A. Blessing, «Using Machine Learning for Identification of Art Paintings», 2010.

[9] C. Hentschel, T. P. Wiradarma, y H. Sack, «Fine tuning CNNS with scarce training data — Adapting imagenet to art epoch classification», en *2016 IEEE International Conference on Image Processing (ICIP)*, 2016, pp. 3693-3697.

[10] R. Paul *et al.*, «Deep Feature Transfer Learning in Combination with Traditional Features Predicts Survival Among Patients with Lung Adenocarcinoma», *Tomography*, vol. 2, n.º 4, pp. 388-395, dic. 2016.

[11] Y. Zeng y Y. Gong, «Nearest Neighbor based Digital Restoration of Damaged Ancient Chinese Paintings», en *2018 IEEE 23rd International Conference on Digital Signal Processing (DSP)*, 2018, pp. 1-5.

[12] Y. Zeng, J. Tang, J. C. A. van der Lubbe, y M. Loog, «Learning Algorithms for Digital Reconstruction of Van Gogh's Drawings», en *Digital Heritage. Progress in Cultural Heritage: Documentation, Preservation, and Protection*, 2016, pp. 322-333.

[13] D. Dueck y B. J. Frey, «Non-metric affinity propagation for unsupervised image categorization», en *2007 IEEE 11th International Conference on Computer Vision*, 2007, pp. 1-8.

[14] B. Gao, T.-Y. Liu, T. Qin, X. Zheng, Q.-S. Cheng, y W.-Y. Ma, «Web image clustering by consistent utilization of visual features and surrounding texts», en *Proceedings of the 13th annual ACM international conference on Multimedia - MULTIMEDIA '05*, Hilton, Singapore, 2005, p. 112.

[15] J. Lei, X. Song, L. Sun, M. Song, N. Li, y C. Chen, «Learning deep classifiers with deep features», en *2016 IEEE International Conference on Multimedia and Expo (ICME)*, 2016, pp. 1-6.

[16] S. Saha, «A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way», *Towards Data Science*, 15-dic-2018. [En línea]. Disponible

en: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. [Accedido: 21-jun-2019].
[17] L. Torrey y J. Shavlik, *Transfer Learning*. .
[18] K. Simonyan y A. Zisserman, «Very Deep Convolutional Networks for Large-Scale Image Recognition», *ArXiv14091556 Cs*, sep. 2014.
[19] «sklearn.decomposition.PCA — scikit-learn 0.21.2 documentation». [En línea]. Disponible en: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. [Accedido: 17-jun-2019].
[20] C. Elkan, «Using the Triangle Inequality to Accelerate -Means», p. 7.
[21] «Amazon Rekognition – Videos e imágenes – AWS», *Amazon Web Services, Inc.* [En línea]. Disponible en: https://aws.amazon.com/es/rekognition/. [Accedido: 04-jul-2019].
[22] «Image Processing with the Computer Vision API | Microsoft Azure». [En línea]. Disponible en: https://azure.microsoft.com/en-us/services/cognitive-services/computer-vision/. [Accedido: 04-jul-2019].
[23] «Vision AI | Derive Image Insights via ML», *Google Cloud*. [En línea]. Disponible en: https://cloud.google.com/vision/. [Accedido: 04-jul-2019].
[24] D. M. Blei, «Latent Dirichlet Allocation», p. 30.

ANNEXES

ANNEX 1

Python code for deep features extraction using CNN.

```
import csv
import os
import numpy as np

from keras.preprocessing import image
from keras.applications.vgg16 import VGG16
from keras.applications.vgg16 import preprocess_input
from keras.models import Sequential

modelVGG116 = VGG16(weights='imagenet', include_top=True)
model = Sequential()
#Exclude last two layer from copying
for layer in modelVGG116.layers[:-2]:
    model.add(layer)

for layer in model.layers:
    layer.trainable = False
model.summary()
vgg16_feature_list = []
names = []
path1 = "D:/Documentos/MASTEAM/TESIS/data/painter-by-numbers/images"

with open("D:/Documentos/MASTEAM/TESIS/data/painter-by-
numbers/all_data_info.csv", "r", encoding="utf8") as csv_file:
    csv_reader = csv.DictReader(csv_file, delimiter=',')
    for lines in csv_reader:
        images = lines['new_filename']
        dirname = os.path.join(path1, images)
        try:
            img = image.load_img(dirname, target_size=(224, 224))
            img_data = image.img_to_array(img)
            img_data = np.expand_dims(img_data, axis=0)
            img_data = preprocess_input(img_data)
            vgg16_feature = model.predict(img_data)
            vgg16_feature_np = np.array(vgg16_feature)
            vgg16_feature_list.append(vgg16_feature_np.flatten())
            names.append(images)
        except:
            print('error image not supported: ', images)
            pass
vgg16_feature_list_np = np.array(vgg16_feature_list)
names_np = np.array(names)
print(vgg16_feature_list_np.shape)
print(names_np.shape)
df = np.append(names_np.reshape(-1,1), vgg16_feature_list_np, axis =
1)
np.savetxt("features_4096.txt", df, delimiter=",", fmt="%s")
```

ANNEX 2

Python code for Principal Component Analysis.

```
import csv
import pandas as pd
import numpy as np
from sklearn.decomposition import PCA
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from matplotlib import pyplot as plt
import statistics

my_data = pd.read_csv("features_4096.txt", header=None)

my_data.shape

x = StandardScaler().fit_transform(my_data)

pca = PCA(n_components=880)
#pca = PCA(.85)

principalComponents = pca.fit_transform(x)
#principalDf = pd.DataFrame(data = principalComponents)

principalComponents.shape

np.savetxt('features_PCA.csv', principalComponents, fmt='%10.5f',
delimiter=",")
```

ANNEX 3

Python code for semantic features extraction using Google Cloud Vision API.

```
from PIL import Image, ImageFile
import os
from os.path import getsize
import csv
from google.cloud import vision
from google.cloud.vision import types
from google.oauth2 import service_account


# Pass the image data to an encoding function.
def encode_image(path, file):
    new_path = "new_images"
    new_file = os.path.join(new_path, file)
    print(new_file)
    with open(path, 'rb') as file:
        img = Image.open(file)
        ImageFile.MAXBLOCK = img.size[0] * img.size[1]
        img.save(new_file, quality=80, optimize=True)
    return new_file


# Pass the image to cloud vision API
def detect(filename):
```

```
    path1 = "D:/Documentos/MASTEAM/TESIS/data/painter-by-
numbers/images"
    credentials = service_account.Credentials.
from_service_account_file('MyApplicatioAPI-fd4bf870d870.json')
    client = vision.ImageAnnotatorClient(credentials=credentials)
    label_detection_feature = {
        'type': vision.enums.Feature.Type.LABEL_DETECTION
    }
    object_detection_feature = {
        'type': vision.enums.Feature.Type.OBJECT_LOCALIZATION
    }
    web_detection_feature = {
        'type': vision.enums.Feature.Type.WEB_DETECTION
    }
    request_feature = [label_detection_feature,
object_detection_feature, web_detection_feature]
    requests = []
    filename1 = os.path.join(path1, filename)
    orig_size = getsize(filename1)
    if orig_size > 10000000:            #Reduce the image_file_size
        filename1 = encode_image(filename1, filename)
        size = getsize(filename1)
        #print(size)
        #print(filename1)
    with open(filename1, 'rb') as image_file:
        content = image_file.read()
        image = vision.types.Image(content=content)
    request = types.AnnotateImageRequest(
        image=image,
        features=request_feature
    )
    requests.append(request)
    response = client.batch_annotate_images(requests)
    return response


header_label = ['new_filename', 'artist', 'date', 'genre', 'style',
'title', 'Label1', 'ScoreL1', 'Label2', 'ScoreL2', 'Label3',
'ScoreL3']
header_obj = ['new_filename', 'artist', 'date', 'genre', 'style',
'title', 'Object1', 'ScoreO1', 'Object2', 'ScoreO2', 'Object3',
'ScoreO3']
header_web = ['new_filename', 'artist', 'date', 'genre', 'style',
'title', 'WebEntity1', 'ScoreW1', 'WebEntity2', 'ScoreW2',
'WebEntity3', 'ScoreW3']

data_label = []
data_obj = []
data_web = []

matrix_label = []
matrix_obj = []
matrix_web = []

matrix_label.append(header_label)
matrix_obj.append(header_obj)
matrix_web.append(header_web)

words = ['Painting', 'Art', 'Visual arts', 'Modern art', 'Artwork',
'Illustration', 'Paint', 'Drawing', 'Artist',
         'Painter', 'Work of art', 'Abstract art', 'Oil painting',
```

```
'Oil paint', 'painting', 'Artist', 'Acrylic paint',
         'Photography', 'Canvas', 'WikiArt', 'Watercolor painting']

with open("CV_input.csv", "r", encoding="utf8") as csv_file:
    csv_reader = csv.DictReader(csv_file, delimiter=',')
    for lines in csv_reader:
        images = lines['new_filename']
        artist = lines['artist']
        date = lines['date']
        genre = lines['genre']
        style = lines['style']
        titles = lines['title']
        print(images)
        try:
            res = detect(images)
            #print(res)

            for annotation_response in res.responses:
                data_label.append(images)
                data_label.append(artist)
                data_label.append(date)
                data_label.append(genre)
                data_label.append(style)
                data_label.append(titles)
                data_obj.append(images)
                data_obj.append(artist)
                data_obj.append(date)
                data_obj.append(genre)
                data_obj.append(style)
                data_obj.append(titles)
                data_web.append(images)
                data_web.append(artist)
                data_web.append(date)
                data_web.append(genre)
                data_web.append(style)
                data_web.append(titles)
                for label_response in
annotation_response.label_annotations:
                    try:
                        if label_response.description not in words:

data_label.append(label_response.description)
                            data_label.append(label_response.score)
                    except:
                        pass
                for object_response in
annotation_response.localized_object_annotations:
                    try:
                        data_obj.append(object_response.name)
                        data_obj.append(object_response.score)
                    except:
                        pass
                for web_response in
annotation_response.web_detection.web_entities:
                    try:
                        if web_response.description not in words:
                            data_web.append(web_response.description)
                            data_web.append(web_response.score)
                    except:
                        pass
            matrix_label.append(data_label)
```

```
            matrix_obj.append(data_obj)
            matrix_web.append(data_web)
            data_label = []
            data_obj = []
            data_web = []
        except:
            pass

with open('Cloud_vision_data/label_features.csv', 'w', newline='',
encoding='utf-8') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(matrix_label)
csvFile.close()
with open('Cloud_vision_data/obj_features.csv', 'w', newline='',
encoding='utf-8') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(matrix_obj)
csvFile.close()
with open('Cloud_vision_data/web_featues.csv', 'w', newline='',
encoding='utf-8') as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(matrix_web)
csvFile.close()
```

ANNEX 4

Python code for processing semantic information

```
import spacy
import csv
import pandas as pd
from nltk.corpus import stopwords

df = pd.read_csv("web.csv")

artist = df.iloc[:, 1]
titles = df.iloc[:, 5]
web_entity1 = df.iloc[:, 6]
web_entity2 = df.iloc[:, 8]
web_entity3 = df.iloc[:, 10]
web_entity4 = df.iloc[:, 12]
web_entity5 = df.iloc[:, 14]
web_entity6 = df.iloc[:, 16]
web_entity7 = df.iloc[:, 18]
web_entity8 = df.iloc[:, 20]
web_entity9 = df.iloc[:, 22]
web_entity10 = df.iloc[:, 24]


nlp = spacy.load(en_core_web_sm-2.0.0')
stopWords = stopwords.words('english')
swords = ['Painting', 'Art', 'Visual arts', 'Modern art', 'Artwork',
'Illustration', 'Paint', 'Drawing', 'Artist', 'Painter', 'Work of
art', 'Abstract art', 'Oil painting', 'Oil paint', 'painting',
'Artist', 'Acrylic paint', 'Photography', 'Canvas', 'WikiArt',
'Watercolor painting', 'Work', 'art', 'Me']
stopWords.extend(swords)
entity = []
response = []
```

```
names = ['artist', 'title', 'entity1', 'entity2', 'entity3',
'entity4']
response.append(names)
counter = 0
web_entities = []
for title in web_entity1:
    web_entities.append(web_entity1[counter])
    web_entities.append(web_entity2[counter])
    web_entities.append(web_entity3[counter])
    web_entities.append(web_entity4[counter])
    web_entities.append(web_entity5[counter])
    web_entities.append(web_entity6[counter])
    web_entities.append(web_entity7[counter])
    web_entities.append(web_entity8[counter])
    web_entities.append(web_entity9[counter])
    web_entities.append(web_entity10[counter])
    web_ent_nan = [x for x in web_entities if x == x]
    #web_ent_nnan = filter(None, web_ent_nan)
    #x = [str for str in web_ent_nan if str]
    #web_ent_nnan = [x for x in web_ent_nan if x]
    entity.append(artist[counter])
    entity.append(titles[counter])
    counter += 1
    for label in web_ent_nan:
        doc = nlp(label)
        for np in doc.noun_chunks:
            if np.text not in stopWords:
                entity.append(np.text)
            for ent in np.ents:
                if ent.label_ is 'PERSON':
                    entity.pop()
        web_entities = []
    response.append(entity)
    entity = []
with open('web_entities_final.csv', 'w', newline='', encoding='utf-8')
as csvFile:
    writer = csv.writer(csvFile)
    writer.writerows(response)
csvFile.close()
```

ANNEX 5

Python code for LDA program

```
import pandas as pd
import os
stopwords1=['untitl','artist','titl','figur','paint','artwork','imag',
'view','painter','madelein','madonna','museum','art','canva','galleri'
,'photographi','illustr','charact','watercolor']
data_web = pd.read_csv('web_entities_final.csv',encoding ='latin1')
data_label=pd.read_csv('label_entities_final.csv',encoding ='latin1')
df_web_lbl_ttl = pd.concat([data_web, data_label],
axis=1).T.drop_duplicates().T.drop(['artist','new_filename'], axis=1)
df_fn=data_label['new_filename']

artist = data_label['artist']
import gensim
from gensim.models import CoherenceModel
```

```
import csv
data1 = df_web_lbl_ttl.fillna('')
data2 = data1.loc[:, data1.columns != 'new_filename'].apply(lambda
row: ' '.join(row.values.astype(str)), axis=1)
from gensim.utils import simple_preprocess
from gensim.parsing.preprocessing import STOPWORDS
from nltk.stem import WordNetLemmatizer, SnowballStemmer
from nltk.stem.porter import *
import numpy as np
np.random.seed(2018)
import nltk
stemmer = SnowballStemmer('english')
def lemmatize_stemming(text):
    return stemmer.stem(WordNetLemmatizer().lemmatize(text, pos='v'))
def preprocess_artist(text):
    result = []
    for token in gensim.utils.simple_preprocess(text):
        if token not in gensim.parsing.preprocessing.STOPWORDS and
len(token) > 3:
                result.append(token)
    return result
def preprocess(text):
    result = []
    for token in gensim.utils.simple_preprocess(text):
        if token not in gensim.parsing.preprocessing.STOPWORDS and
len(token) > 3:
            if token not in artist_names:
                result.append(lemmatize_stemming(token))
    return result
processed_artists = list(map(preprocess_artist, artist))
x = list(map(' '.join, processed_artists))

subList = [elem.split(' ') for elem in x]
artist_names = [l for elem in subList for l in elem]
text_data = data2
words1 = []
for word in text_data:
    word.split(' ')
    words1.append(word)
processed_docs1 = list(map(preprocess, words1))
processed_docs=[]
for review in processed_docs1:
    wr=[]
    for word in review:
        if word not in stopwords1:
            wr.append(word)
    processed_docs.append(wr)
from gensim.utils import simple_preprocess
from gensim import corpora, models
# Create Dictionary
id2word = corpora.Dictionary(processed_docs)
# Create Corpus
texts = processed_docs
# Term Document Frequency
corpus = [id2word.doc2bow(text) for text in texts]
lda_model1 = gensim.models.ldamodel.LdaModel(corpus=corpus,
                                             id2word=id2word,
                                             num_topics=14,
                                             random_state=100,
                                             update_every=1,
                                             chunksize=100,
```

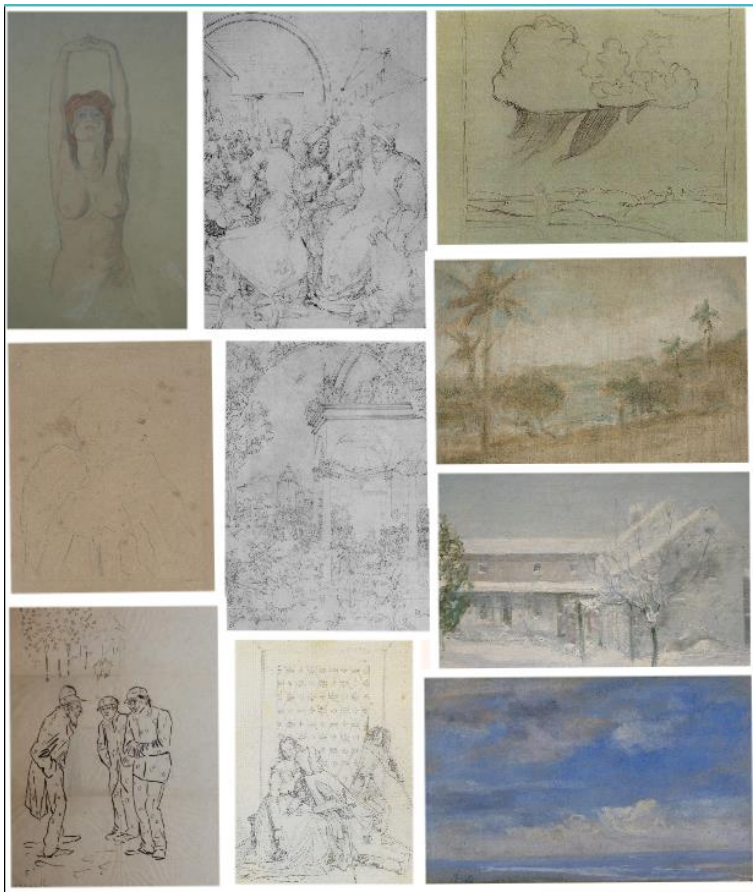
```
                                            passes=10,
                                            alpha='auto',
                                            per_word_topics=True)
cm = CoherenceModel(model=lda_model1, corpus=corpus,
coherence='u_mass')
coherence = cm.get_coherence()  # get coherence value
print('\nCoherence Score: ', coherence)
for idx, topic in lda_model1.print_topics(-1):
    print('Topic: {} \n {}'.format(idx, topic))
# Init output
sent_topics_df = pd.DataFrame()
# Get main topic in each document
for i, row_list in enumerate(lda_model1[corpus]):
    row = row_list # if ldamallet.per_word_topics else row_list
    row = sorted(row[0], key=lambda x:x[1], reverse=True)
# Get the Dominant topic, Perc Contribution and Keywords for each
document
    for j, (topic_num, prop_topic) in enumerate(row):
        if j == 0:  # => dominant topic
            wp = lda_model1.show_topic(topic_num)
            topic_keywords = ", ".join([word for word, prop in wp])
            sent_topics_df =
sent_topics_df.append(pd.Series([int(topic_num),
round(prop_topic,4)]), ignore_index=True)
        else:
            break
sent_topics_df.columns = ['Dominant_Topic', 'Perc_Contribution']
# Add original text to the end of the output
contents = pd.Series(df_fn)
sent_topics_df = pd.concat([sent_topics_df, contents], axis=1)
sent_topics_df.to_csv('LDA_total_GSIM.csv', encoding='utf-8',
index=False)
```

ANNEX 6

Clusters obtained with deep learning approach.

Cluster 0

Cluster 1



Cluster2

Cluster 3

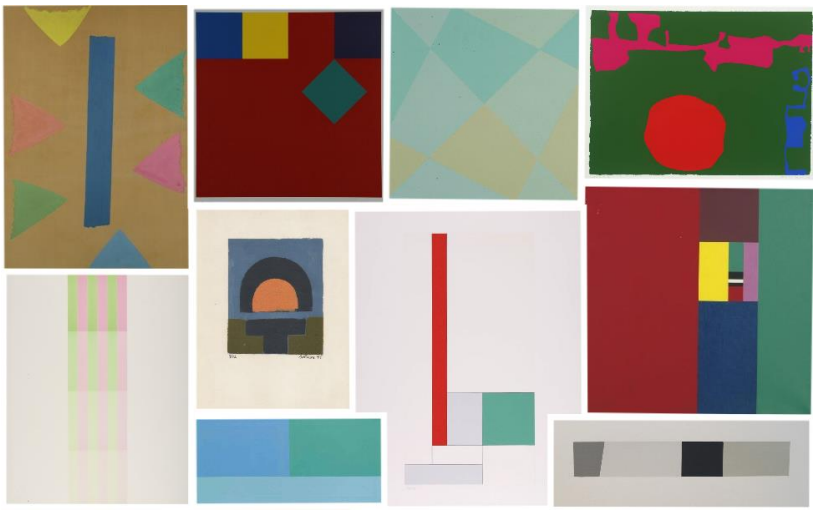
Cluster 4

Cluster 5

Cluster 6

Cluster 7

Cluster 8



Cluster 9

Cluster 10

Cluster 11

Cluster 12

Cluster 13

Cluster 14

Cluster 15

ANNEX 7

Topics obtained with semantic approach.

Topic: 0

0.069*"flower" + 0.063*"plant" + 0.056*"contemporari" + 0.027*"monochrom" + 0.024*"bouquet" + 0.021*"centuri" + 0.017*"venus" + 0.014*"collect" + 0.014*"leaf" + 0.013*"allegori"

Topic: 1

0.066*"head" + 0.045*"bird" + 0.042*"vehicl" + 0.028*"ship" + 0.025*"chin" + 0.023*"face" + 0.023*"branch" + 0.023*"state" + 0.021*"cheek" + 0.020*"fine"

Topic: 2

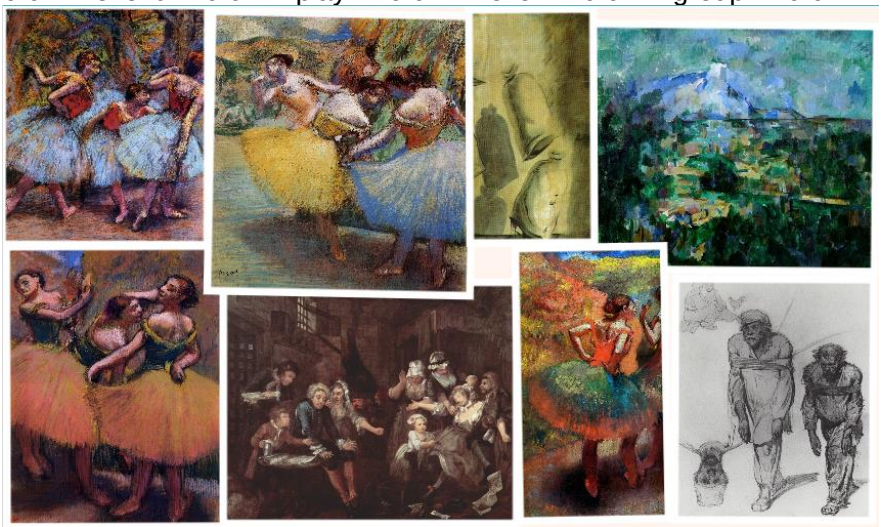
0.171*"impression" + 0.149*"reproduct" + 0.044*"post" + 0.023*"bridg" + 0.023*"pari" + 0.022*"orang" + 0.017*"bank" + 0.016*"peasant" + 0.014*"port" + 0.013*"madam"

Topic: 3

0.074*"river" + 0.019*"school" + 0.018*"coast" + 0.018*"bank" + 0.017*"storm" + 0.017*"italian" + 0.016*"cliff" + 0.016*"unit" + 0.015*"product" + 0.013*"poet"

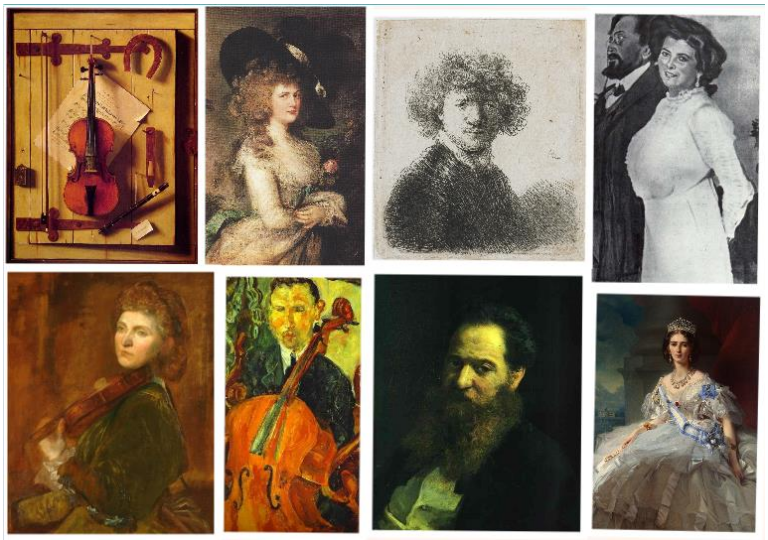
Topic: 4

0.018*"king" + 0.017*"ballet" + 0.016*"pink" + 0.014*"dancer" + 0.013*"academi" + 0.012*"event" + 0.012*"play" + 0.012*"relief" + 0.012*"group" + 0.011*"franc"



Topic: 5

0.165*"self" + 0.092*"ladi" + 0.040*"gentleman" + 0.037*"hair" + 0.029*"mountain" + 0.027*"nation" + 0.021*"fashion" + 0.020*"area" + 0.020*"realism" + 0.018*"rural"



Topic: 6

0.080*"expression" + 0.047*"abstract" + 0.043*"printmak" + 0.041*"surreal" + 0.040*"graphic" + 0.032*"paper" + 0.023*"field" + 0.021*"color" + 0.021*"mural" + 0.020*"botani"

Topic: 7

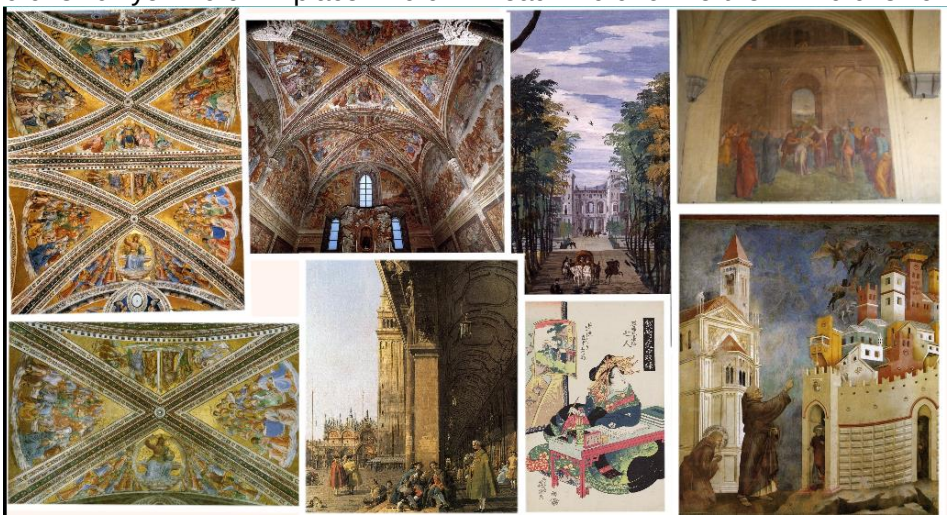
0.055*"boat" + 0.038*"rock" + 0.028*"window" + 0.026*"beach" + 0.019*"world" + 0.019*"snow" + 0.018*"tabl" + 0.017*"geolog" + 0.017*"plate" + 0.016*"fauvism"

Topic: 8

0.124*"mytholog" + 0.099*"fiction" + 0.031*"angel" + 0.030*"poster" + 0.019*"prophet" + 0.017*"bather" + 0.016*"space" + 0.016*"pilgrim" + 0.015*"cupid" + 0.012*"annunci"

Topic: 9

0.073*"architectur" + 0.063*"renaiss" + 0.043*"histori" + 0.034*"build" + 0.032*"hors" + 0.023*"ukiyo" + 0.021*"place" + 0.017*"road" + 0.016*"northern" + 0.015*"chapel"



Topic: 10

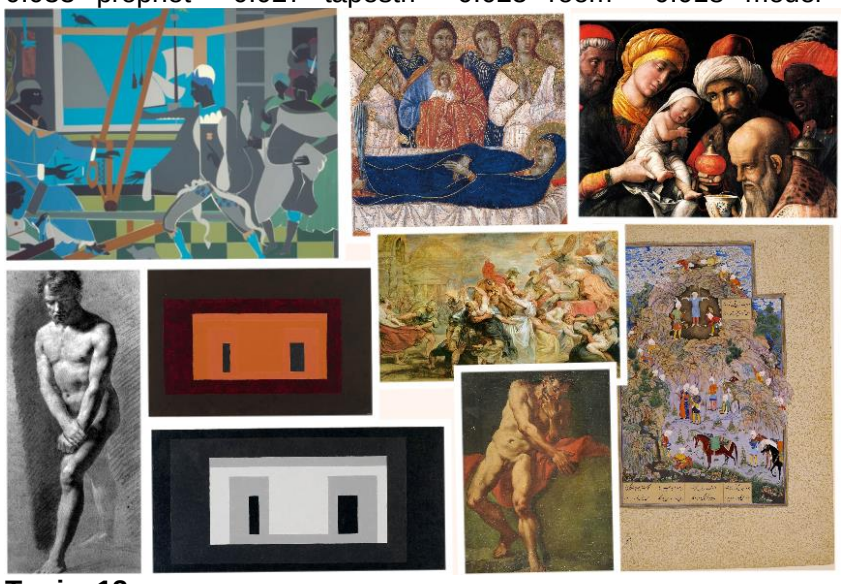
0.067*"sculptur" + 0.051*"saint" + 0.043*"cubism" + 0.038*"night" + 0.034*"wall" + 0.026*"interior" + 0.024*"print" + 0.017*"calligraphi" + 0.014*"tourist" + 0.014*"attract"

Topic: 11

0.149*"tree" + 0.091*"plant" + 0.080*"natur" + 0.057*"realism" + 0.023*"atmospher" + 0.020*"mother" + 0.019*"organ" + 0.018*"woodi" + 0.017*"circl" + 0.016*"photograph"

Topic: 12

0.050*"textil" + 0.041*"human" + 0.040*"age" + 0.038*"middl" + 0.037*"virgin" + 0.033*"prophet" + 0.027*"tapestri" + 0.025*"room" + 0.018*"model" + 0.016*"edg"



Topic: 13

0.132*"stock" + 0.065*"garden" + 0.059*"histori" + 0.039*"symbol" + 0.036*"romantic" + 0.021*"appl" + 0.019*"palac" + 0.018*"rose" + 0.015*"academ" + 0.010*"player"