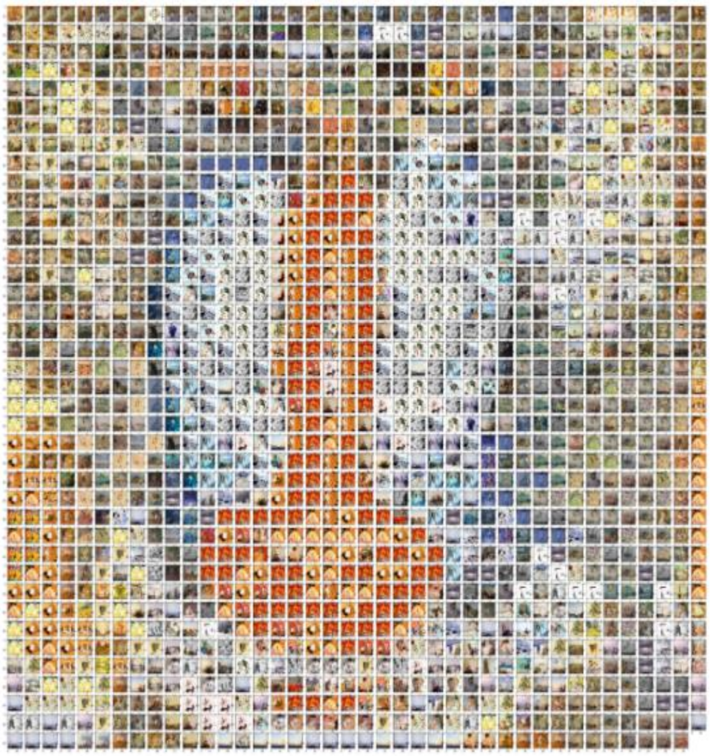


BUILDING A PHOTMOSAIC USING ML

INTRODUCTION

A photomosaic consists of breaking up an image (the target) into square tiles and replacing each tile with another image that has similar features to replicate the target. Since the early 2000s’ computer science researchers have developed and patented commercial tools as well as published papers on different techniques to generate and optimize image mosaics. Most of the research has been done on feature selection and quantification of image similarity, while some research has been done on using classification techniques to reduce the processing time. This final project is focused on the latter. The initial idea of the capstone project was to use machine learning to build a photomosaic. However, the process of building a photomosaic does not involve machine learning techniques. By simply computing the similarity between vectors representing tiles and images, the best fit is found, and the target image is recreated by concatenating all the matched images. This process is known to be computationally intensive and is dependent on multiple factors such as database size and image size, among others. This is where machine learning can come into play. By dividing a large data set into clusters and classifying each tile from the target, the time to find the most similar image can be greatly reduced. This project's main goal is to create a photomosaic and use machine learning techniques to reduce the time to find the best match between tiles from the target image and images from the database.

DATA COLLECTION

The database used for this project is from an article submitted in 2016 entitled “Pandora: Description of a Painting Database for Art Movement Recognition with Baseline and Perspectives”. The database is publicly available and consists of 7700 images of paintings from 12 different art movements. This database was originally used to “investigate how local and global features and classification systems are able to recognize the art movement”. All the images of the database are in JPG format and the original database is divided in the following way:

Art Movement	# Paintings	Art Movement	# Paintings
Old Greek pottery	349	Impressionism	984
Iconoclasm	665	Realism	307
High renaissance	812	Cubism	920
Baroque	960	Abstract expressionism	340
Rococo	844	Fauvism	398
Romanticism	874	Surrealism	241

Table1. Original Pandora database distribution.

For this final project, we are not interested in the art movement that each image belongs to. Thus, all the images of the database were compiled into one folder. Also, 6 of the images were corrupted and our final capstone database only included 7694 images.

MODELING

EDA

As mentioned before one of the key elements to building a photomosaic is defining the feature vector to calculate the similarity score between images and tiles. Before defining the features to include, some exploratory work was done to get familiar with the database. All the work so far has been in jupyter notebooks and mainly using the following python packages: NumPy, pandas, matplotlib, OpenCV, scipy and sklearn.

The EDA work consisted of loading two images to see in what format they were uploaded, switching the format to R, G, B, checking the size of the images, resizing them, plotting them using OpenCV and Matplotlib, getting the colour histograms for each channel, compute the mean, variance and skewness for each histogram.

After the exploratory data analysis, the conclusion was to divide the modelling into three phases:

1. Build a photomosaic using only the flattened image to get a preliminary result.
2. Build the same photomosaic using the flattened image and basic texture features.
3. Build the same photomosaic with the flattened image, texture features and using unsupervised machine learning techniques.

Furthermore, it was defined that the image size for the three phases of the modelling was going to be 16X16 to shorten the computation time.

Texture Features

An image texture is a set of metrics calculated in image processing designed to quantify the texture of an image. Image texture gives us information about the spatial arrangement of colour or intensities in an image or selected region of an image. First-order feature extraction is a method of retrieval based on characteristics of the image histogram. The histogram shows the probability of occurrence of the value of the degree of colour pixels in each channel. From the histogram, several parameters of the first order can be computed such as mean, variance, skewness, kurtosis, and entropy. The mean shows the size of the dispersion of an image and the variance shows the variations of the element on the histogram of an image. Due to time constraints, only mean and variance for each colour channel have been computed so far and included as the texture features for this project.

Mean $= \frac{1}{m} \sum_{i=0}^{n} i * h(i)$

Variance = $\frac{1}{m}\sum_{i=0}^{n}(i - mean)^2 * h(i)$

Where *i* is a level, *n* is the number of levels, *m* is the image size and *h* is the quantized histogram.

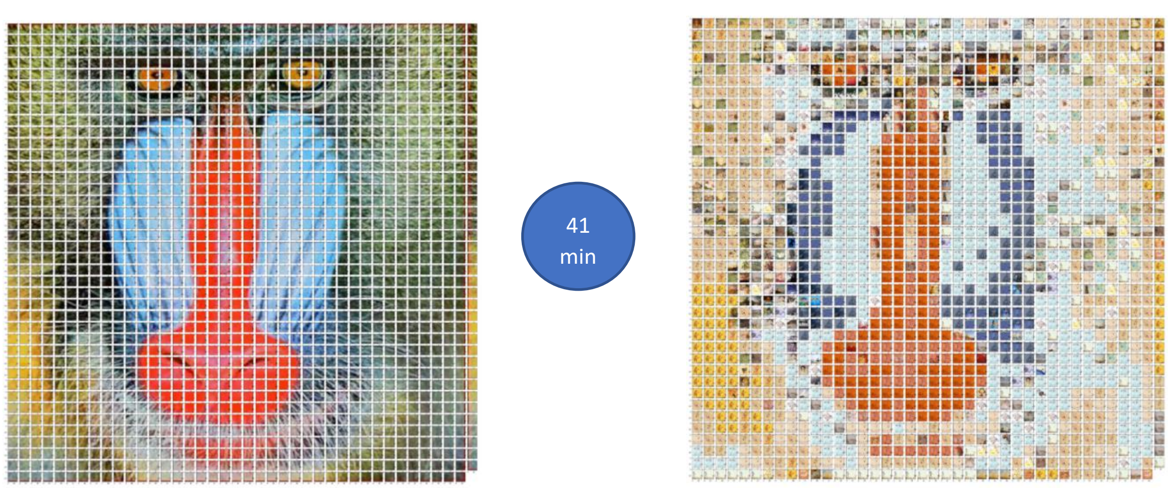
Model selection

We are looking for a classification where we get even distributed classes at the end. This will allow the model to be more efficient and not take longer times in case one of the classes has a considerably bigger portion of the images. For this reason, the unsupervised clustering algorithm that was used was k-means. After multiple iterations for different values of k, the lowest inertia score was found to be for 5 clusters. A silhouette score was also calculated for multiple values of k and the obtained graph only showed a negative slope. After applying the model for 5 clusters we obtained an even histogram and decided to keep the model with 5 clusters.

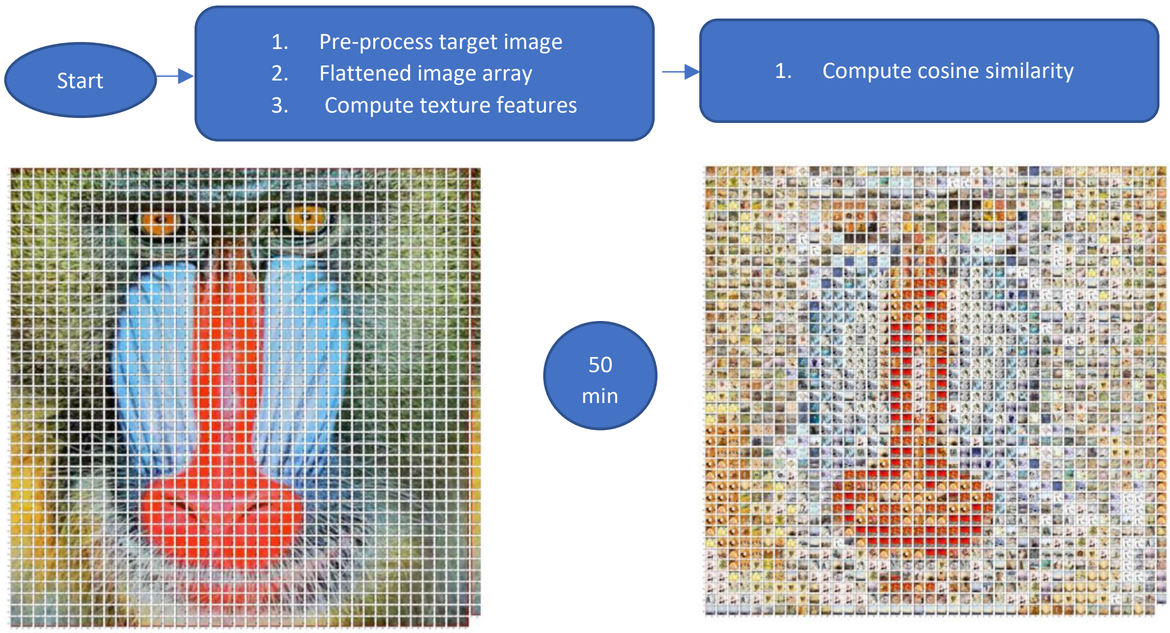
The second technique to be used in the future to enhance the quality of the output will be an autoencoder. The autoencoder will allow the model to extract the most important features for image classification on its own without having to choose and manually compute the texture features.

MODEL FRAMEWORK AND RESULTS TO DATE

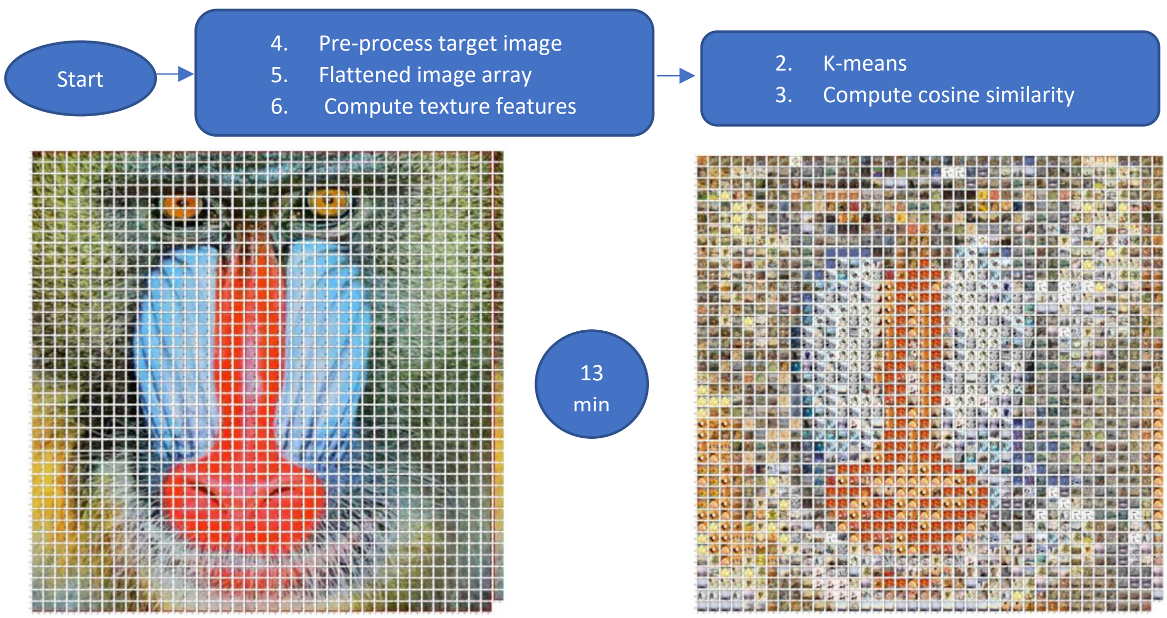
Cosine Similarity



Texture features + Cosine Similarity



Texture features + K-means + Cosine Similarity



From the results above it is clear that by adding the texture features the image quality was greatly enhanced. The baboon that we obtained at first is plain and looks like a cartoon. By adding the texture features the final image became more alive. By adding a clustering technique, the match between tiles images was almost 5 times faster and the result was not compromised.

The next steps to enhance que quality of the photomosaic are:

1. Compare the color histograms of the target image and the result to have a scientific evaluation of the photomosaic
2. Add more texture features into the model
3. Resize tiles and images to 64X64 to have a better visual result
4. Implement an autoencoder to classify the pull of images

REFERENCES

1. Pandora Database, Pandora 7K, images distributed among 12 classes. http://imag.pub.ro/pandora/pandora_download.html

2. Image Similarity Measure using Color Histogram, Color Coherence Vector, and Sobel Method, Kalyan Roy1 , Joydeep Mukherjee2, Jadavpur University, School of Education Technology. https://www.ijsr.net/archive/v2i1/IJSRON2013311.pdf

3. Generating Mosaic Images Based On Texture Analysis Alaa Yaseen Taqa, Mosul University-Education College, https://edusj.mosuljournals.com/article_159299_a6a97c61084df506bb94ce29d075d509.pdf

4. https://www.pyimagesearch.com/2020/03/30/autoencoders-for-content-based-image-retrieval-with-keras-and-tensorflow/

5. https://stackabuse.com/autoencoders-for-image-reconstruction-in-python-and-keras/

6. https://www.pyimagesearch.com/2020/03/30/autoencoders-for-content-based-image-retrieval-with-keras-and-tensorflow/

7. https://www.pyimagesearch.com/2014/01/22/clever-girl-a-guide-to-utilizing-color-histograms-for-computer-vision-and-image-search-engines/

8. https://www.pyimagesearch.com/2014/05/26/opencv-python-k-means-color-clustering/

9. https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/