Identifying the Most Creative Paintings

Reed Mershon

April 24, 2018

1 Introduction

Much of the research done into computational creativity has been about generating poetry, stories, jokes, games, music, art etc., and creative problem solving. A major characteristic of understanding the creativity of one artifact is understanding the creativity of other comparable artifacts. In this paper I will focus on developing a computational framework for assessing the creativity of different artifacts, specifically paintings and artwork. In order to compare the creativity, I must define the most common definition which emphasizes the originality of the piece, and its influence over other pieces. My framework for this analysis will be based on a network between the different pieces of art. This network will then be used to determine qualities of originality and influence.

2 Background

There is a long and ongoing debate on how exactly to define creativity [1]. This background on creativity will not be focused on being a comprehensive guide to creativity. Instead it will discuss how creativity will relate to the computational framework that I am designing. Defining creativity is inherently subjective, and one strength to a computational framework is that it has the potential of being completely objective in regards to how humans view art. This could be a major step in defining the creativity of artwork.

2.1 Previous Research

The previous research in this field looked at the same dataset that I will be using, and created a similar computational framework for analyzing the creativity. Mine will be far simpler with less computation in different unique aspects about each work. Elgammal and Saleh used complex matrices and mathematical formulas to complete their research and generated some impressive results. While this paper is aimed at emulating their paper, I will not have the exact results as them. They do provide good documentation for the equations that they use. However, the exact similarity comparisons that they computed are unknown to me. They discuss using image analysis tools such as Classeme and

GIST features [1]. I do not have direct access to an easy Python port for the Classeme features, so I will use another kind of feature detection discussed later in the paper. Figure 1 is an example of what they're results returned. Looking at the previous research will be key to understanding my scoring of paintings.

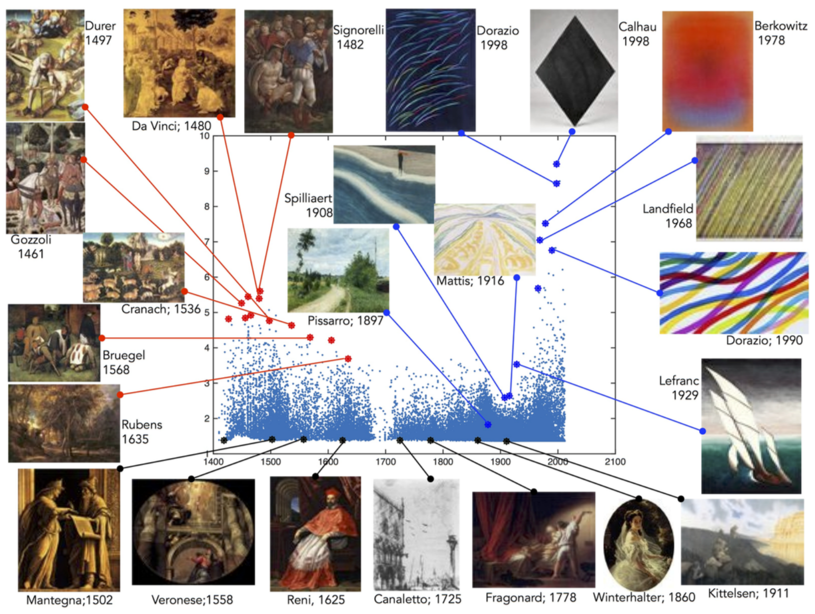


Figure 1: This is the results from Elgammal and Saleh. The further up on the Y-axis the better the creativity is. Each dot represents an individual painting. This is from their corpus of nearly 63,000 paintings.

2.2 Identifying Creativity

Elgammal and Saleh discuss how other research has defined two different notions of creativity; P-creativity, which is a pieces creativity with respect to the creator, and H-creativity which assesses the piece based on the whole of human history [1]. For this project I will be focused on this idea of H-creativity by comparing artwork from many different time periods to determine the most creative. The creativity will be defined through the a computational framework that will assess a piece of art against other, and then determine if the score should be increased or decreased.

3 Computational Framework

Previous research has used different techniques to create the creativity scores discussed before. There are three proper functions discussed by this research [1], a Similarity function, a Balancing function, and scoring the creativity. These three functions combined gave the artwork their final creativity scores.

3.1 Similarity

In order to begin this framework we need to create a graph containing all of the paintings. Let us assume we have a set of paintings $P = \{p_i, i = 1 \cdots N\}$. As discussed above, the goal is to assign some creativity score, $C(p_i)$ for each painting p_i. Each painting in this data must have a time period it was created denoted by $t(p_i)$. This will allow us to create a directed graph where each assigned weight is a positive weight, and where each node i is only connected to j if i was created prior to j.

Now we have similarity functions that are used to compare each of the paintings and created the weighted edges. Each similarity function can be defined simply as $S^a(p_i, p_j)$, where a represents some attribute being evaluated. Below is the full definition of how the weights are added to the adjacency matrix.

$$w_{ij} = \begin{cases} S(p_i, p_j) & \text{if } t(p_i) < t(p_j) \\ 0 & \text{otherwise} \end{cases}$$

The higher the similarity score, the higher the similarity between the two paintings. Since in theory there could be many different similarity functions, again denoted by a, we could have multiple different creativity scores C^a. However, in this paper I will be focusing on a singular similarity which is entirely from the relationship from a feature descriptor.

3.2 Balancing

Now that we have defined this notion of getting some similarity between two paintings with a weighted edge, we must define some way of determining what is a high weight and a low weight. This will be done through a balancing variable called $m(i)$. Elgammal and Saleh describe a way to define this variable as either a global variable, this is how I will define it, and a local sliding variable, only focusing on 'near by' paintings [1]. So any weight above $m(i)$ will be considered high, and any weight below will be considered low. This concept is defined below.

$$B_i(w) = \begin{cases} w - m(i) & \text{if } w > 0 \\ 0 & \text{otherwise} \end{cases}$$

This balancing variable will convert any weights lower than $m(i)$ to a negative value. Essentially what this negative value means is that the node at j

is more creative than the original starting node at i. The possibility of negative weights in the graph would throw off any useful analysis using the Markov chains later on. Thus we must reverse the edges if they have a negative weight after the balancing variable is applied. This is defined below.

$$\begin{aligned}
&B(w_{ij}) > 0 \to \tilde{w}_{ij} = B(w_{ij})\\
&B(w_{ij}) = 0 \to \tilde{w}_{ij} = 0\\
&B(w_{ij}) < 0 \to \tilde{w}_{ji} = -B(w_{ij})
\end{aligned}$$

This will give me a new matrix that will have the balanced weights in it. What this means is that the higher the weight the lower the creativity in the node j is, and the higher creativity is in i. Because of this balancing function all weights will now be positive, which will make the analysis easier.

3.3 Normalizing

The normalizing function is not included in Elgammal and Saleh's paper. It does seem necessary to compute the Markov chain. So the goal of this function is to retrieve the sum of each row in the matrix. Then to divide that total by

$$N(p_i) = \begin{cases} \dfrac{w_{ij}}{\sum\limits_j \tilde{w}_{ij}} & \text{if } \sum\limits_j \tilde{w}_{ij} > 0 \\ \dfrac{\text{length } (w_i)}{\sum\limits_j \tilde{w}_{ij}} & \text{if } \sum\limits_j \tilde{w}_{ij} = 0 \end{cases}$$

In essence this function will make the entire row's weights equal to one. They become proportional to 1 and more like probabilities that the Markov chain needs. The one error I had to handle was for the rows whose totals were just equal to 0. Because the Markov chain requires probabilities that equal 1. So the best way to keep an unbiased result was to just create equal weights across all the different paintings connected to that individual painting. This does include its self, and others in the same year. This was the best way I could think of to fix this issue.

3.4 Markov Chain

Once again, previous research has defined this a the method for determining the creativity scores for a specific attribute. Thus, Elgammal and Saleh have defined the following function.

$$C^* = \frac{(1-\alpha}{N}(I - \alpha\tilde{\tilde{W}})^{-1}1$$

As described by Elgammal and Saleh [1], this equation is essentially equivalent to a random walk in a Markov chain. Essentially the random walk will output a probability for each of the paintings, based on the input probabilities from the normalization function. These output probabilities will be the creativity score for the specific attribute. This will be the key to representing the individual paintings with creativity scores.

3.5 Limiting Connections

Because of the possibility of the sheer number of paintings that could be evaluated. The more recent a painting was created, the more incoming edges this painting will have to it. In the framework this would spell trouble for its general creativity. Because of this I also need a function that will limit the number of connections each painting can have. I will define this as the k-nearest neighbors of the painting p_i. Where the output of this function will be a list of the indices of the k-nearest neighbors, and the inputs will be all painting data, desired k value, and the index of the painting to evaluate. Also since the paintings cannot connect right away to paintings created prior, this function will only look into paintings p_j where $t(p_i) < t(p_j)$

4 Data

The primary source will be from *Wikiart.org*, a public online art archive that has over 250,000 individual artworks, by some 3000 artists [4]. While this is a substantial amount I cannot possibly process all of the artwork on this site. Therefore I will make some restrictions to the number of artworks that I look at. While previous research has looked at nearly 63,000 paintings [1], I will choose from a smaller corpus of just a singular genre or art style. This will allow me to analyze the images and not just focus on the collection of the data.

4.1 Data Collection

For the collection of my data I needed to develop a small script for parsing through a website containing a multitude of paintings. The specific website [4] was used in previous research [1], and will be the site from which all of my artwork comes from. This will allow me to develop a singular scraper with minimal changes needed per page. Wikiart contains over 250,000 unique paintings. While analysis on all of these would be great, I do not want to download any unwanted artwork. This means that I need to also make the web-scraper that can just focus on a singular genre or artist. There is a bit of a web from the script and all of the websites. Figure 2 shows this web.

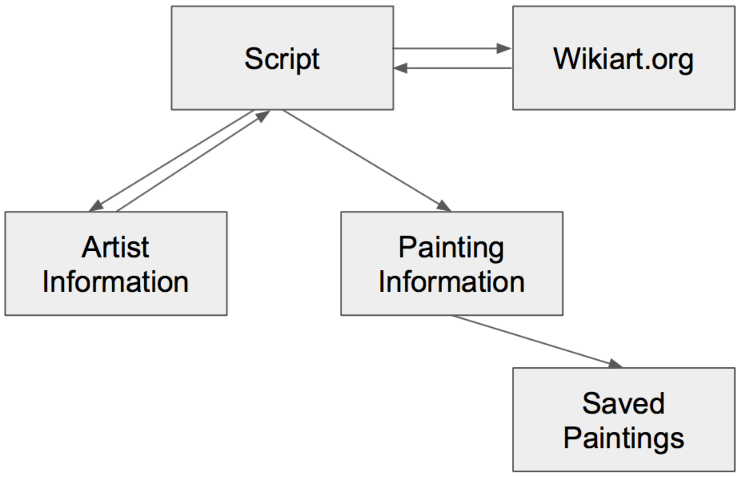


Figure 2: Data Collection Process

The basics for this web-scraper begin with initializing access to the website. This is done with *urllib* and *BeautifulSoup* packages for Python. *urllib* is used for opening the webpage and downloading its source code (HTML). *BeautifulSoup* is then used to parse through the HTML tags. This allows me to quickly parse through large pages and save the information I want. So I designed a function that will open the home page of Wikiart, then save each artist name and the link to their personal page. The link is then used to retrieve links to their paintings as well as other information about them. These painting links also contain valuable information for later analysis. Once at an individual painting link, more scraping must be done. I have to retrieve the actual link to the painting that will allow me to download it as the page that I would be currently looking at is only referencing that link. And as previously mentioned, I must gather a few more bits of information to complete the analysis for this paper. The paintings are then downloaded into a folder, named for the artist they are by, and saved onto a local drive on my computer. This is also another issue with the amount of data that I am able to store and analyze. I don't have enough space to download and look at the entire corpus. Once downloaded, the file path is saved along with the other information about the painting.

Currently, there is an issue with the scraping of the artist pages. While I am able to find the page links to each artist, once on that page I am unable to locate the image links. This has to do with the fact that the website has been updated to use AngularJS to populate its information. As of writing there is no good way to get the raw HTML from these files. Angular dynamically generates the information on pages, and because I am not using a proper web browser in my requests to the page I cannot activate these scripts to generate the needed information. This means that, for now, I am unable to retrieve the information that I need to continue the analysis.

Fortunately I managed to do a small test before the change, and downloaded the first 5 religious artists, and their first 5 paintings. This means that I can continue, at a small scale, with the analysis part of my project.

4.2 New Data Set

Because of the issues I faced with the web scraping aspect of this project, I had to find a new source for this data. I was able to locate a website which held a competition using data from Wikiart.org [3]. This competition had files that were available to download. These files contain works of art collected from Wikiart.org, and also have .csv files that contain important information about the individual artworks. This information included the style, genre, date and title of the piece. The specific image file I downloaded was, $'train_1.csv'$. After also downloading the training file information, I came to the conclusion that the number of paintings in my dataset was around 8,000.

5 Analysis

To begin analysis of the paintings for their creativity, I must first analyze the key features and matches between each pair of paintings. The comparison between two paintings is the key to my similarity function as described before. I used OpenCV [2] for its extensive image analysis libraries. After determining the comparison function I have the rest of the framework coded into python.

5.1 Cleaning & Choosing Data

Before the true analysis of the images could begin, I needed to clean up the specific data. Since the information I recieved was in a *.csv* file, and it contained all training sets (I only had one of ten). I needed to begin slimming the size of the data down to a more reasonable amount. I began by using a Python package called pandas, which can read *.csv* files into an easy to use table format. Pandas has a function that will drop all null values at once. Once this had been applied there was still over 58,000 rows in the table. This is far more than the number of images that were downloaded on to my computer. Next I moved to looking for if the images in the table were in the downloaded source file for the images. If they were, I kept their information. If not, I dropped them from the table. This then narrowed my table down to about 8000 images. Again this is still too large for any analysis done in a reasonable time. Now I decided to choose a singular genre. Luckily pandas allows me to just select all rows with a certain value. I chose *Symbolic Paintings* as it seemed to be a small enough corpus of images that I could compute a reasonable analysis. The entire *Symbolic Paintings* in my data set totaled around 180 paintings. Below I will talk about the process by which the data and information must go through to come out with the final creativity score.

5.2 Similarity Function

I used the built in SURF (Speeded-up Robust Features) to detect keypoints and descriptors in each image. These keypoints and descriptors show what the feature detector SURF found important in the image. After retrieving these values, OpenCV allows the user to compare different keypoints from two images and see if there are any matches. Figure 3 is a visual example of how the algorithm in OpenCV matches the keypoints from two images. If there were matches, I recorded them and then for each weighted edge w_{ij} I took the minimum number of keypoints found from both image i and j. Then I divided the number of matches by the minimum number of keypoints. This is important because one painting couldn't match more points than it already has. So this will give me some fractional value.

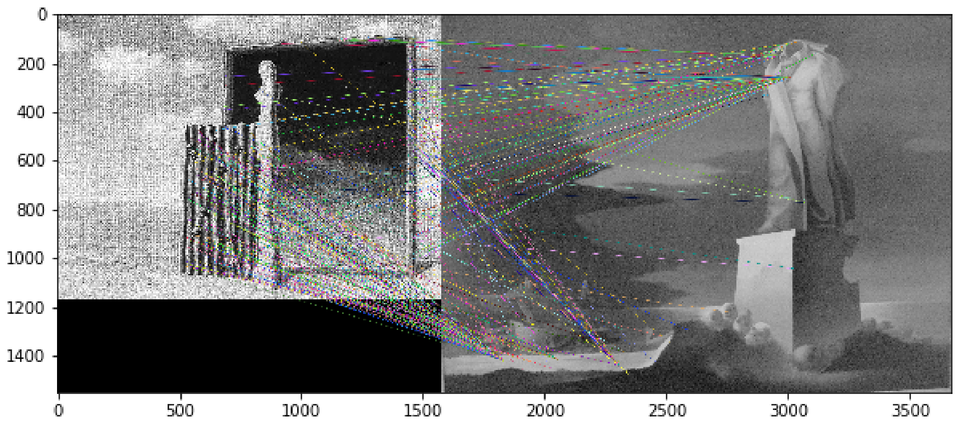
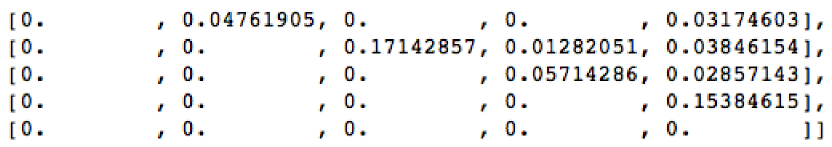


Figure 3: Example of Image Comparison

After evaluating each image, and ensuring that the images being compared are in the temporal neighborhood of one another (k-nearest neighbors in regards to time), the similarity function will return a matrix similar to that in Figure 4. As you can see, if the paintings are organized by the year they were created it makes the analysis much easier.

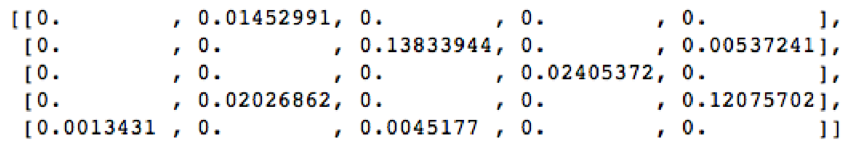


```
[0.        , 0.04761905, 0.        , 0.        , 0.03174603],
[0.        , 0.        , 0.17142857, 0.01282051, 0.03846154],
[0.        , 0.        , 0.        , 0.05714286, 0.02857143],
[0.        , 0.        , 0.        , 0.        , 0.15384615],
[0.        , 0.        , 0.        , 0.        , 0.        ]]
```

Figure 4: Example of what similarity matrix would look like after the function is applied to the images

5.3 Balancing Results

Now that we have the similarities of the different paintings, we must apply a balancing variable to each of the weighted edges w_{ij}. Again looking at how this process is described above the matrix is then put through this balancing function. Any edges that are flipped had a low weight, and any that stayed were still from the original edge w_{ij}. Figure 5 shows how this function would be applied to the original similarity matrix defined in Figure 4. We can see how 3 of the weights were changed from their original values. Also note how there are no negative weights. This was the purpose of switching the edges around. These results will be used when normalizing for the Markov chain.

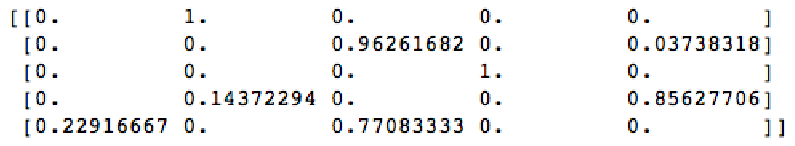


```
[[0.        , 0.01452991, 0.        , 0.        , 0.        ],
 [0.        , 0.        , 0.13833944, 0.        , 0.00537241],
 [0.        , 0.        , 0.        , 0.02405372, 0.        ],
 [0.        , 0.02026862, 0.        , 0.        , 0.12075702],
 [0.0013431 , 0.        , 0.0045177 , 0.        , 0.        ]]
```

Figure 5: Example of what the balancing function would do to the similarity matrix in Figure 4

5.4 Normalizing Results

When normalizing, I take the total of the row, then divide each edge by that total, again this is described above. When doing so, this will give us probabilities for the Markov chain to walk through and give us the creativity scores needed. Note how the first row of Figure 6 has a value of just 1, that is because it was the only value in the balanced matrix for that row. Again this normalized data will be crucial for the Markov walk.

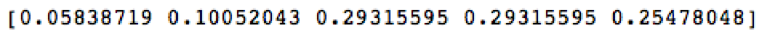


```
[[0.         1.         0.         0.         0.        ]
 [0.         0.         0.96261682 0.         0.03738318]
 [0.         0.         0.         1.         0.        ]
 [0.         0.14372294 0.         0.         0.85627706]
 [0.22916667 0.         0.77083333 0.         0.        ]]
```

Figure 6: Example of the normalizing function applied to the balanced matrix in Figure 5

5.5 Scoring Creativity

Finally we have reached the part the whole paper is about. What exactly is the most creative painting? Using a Python package *Discrete Markov Chain*, a function can be used to output a list of length n, from out 2D array of size $n \cdot n$. Figure 7 is an example of what would be output if the normalized array from Figure 5 was the input. If you were to take the total of all five of the values, they would total 1. This is important to note when looking at the final results from the actual data. Scores will be very near 0, but that is because of the high volume of data points.



```
[0.05838719 0.10052043 0.29315595 0.29315595 0.25478048]
```

Figure 7: Example of Creativity Scores given after all functions have been applied

5.6 Final Results

For the analysis of the 182 *Symbolic Paintings*, I ran them through each of the above functions. Because of the large data size of the matrices, I did not want to include them in this paper. Instead I included the example matrices, which should show what the analysis portion is doing. During the analysis of the paintings I tweaked with the number of neighbors that each painting should have, but found that the higher that k value was, the more it skewed the data towards making newer paintings more creative. I didn't think this was an accurate representation, so I settled at a k value of 10. Further more, there is a threshold to which the SURF algorithm decides to keep the descriptors or keypoints found. I was initially doing it at a threshold of 1000. This threshold is what produced the results below. Finally after analyzing the similarity, balancing, and normalizing the paintings I achieved probabilities that were ready to be run through my Markov chain function.

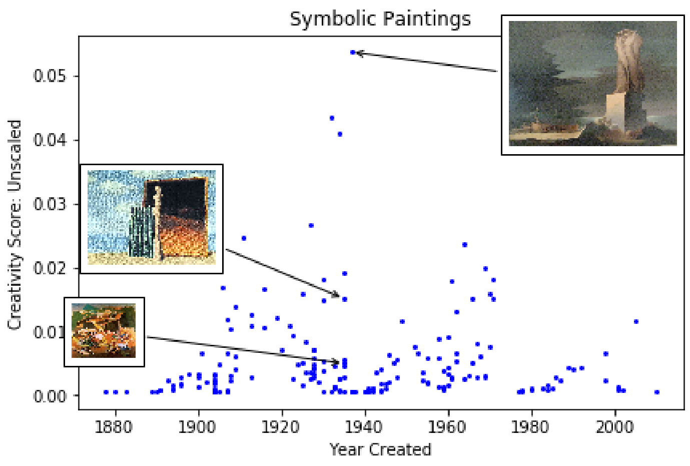


Figure 8: Creativity Scores of the 182 Symbolic paintings, with the highest scored painting and its closest paintings shown.

Above, in Figure 8, are the results from the Markov chain scoring of all the images in this particular set. There were 182 different paintings evaluated, all of which were evaluated with the functions that I created above. This produced the graph below, which shows the unscaled Creativity Scores for the 182 Symbolic paintings that were analyzed. There is clearly a large difference between this painting and all other paintings in the graph. The title is ”Ecce Homo” and was painted in 1937, it achieved a score of 0.0537.

While the goal of this paper was to obviously look into the ”most creative” painting, I also wanted to look into what the least creative painting was. Below, in Figure 9 we can see the lowest scoring painting. This is because is didn't have any relation to any paintings. I believe that this is caused by the threshold set for the SURF algorithm. Since the threshold was so high, it was unable to find enough keypoints in this painting, and presumably others. This means that its matching score will inherently be very low. In my program I found that this painting had exactly 4 keypoints found. Where as the highest scoring image had 299. This significant difference puts the lowest painting at a disadvantage.

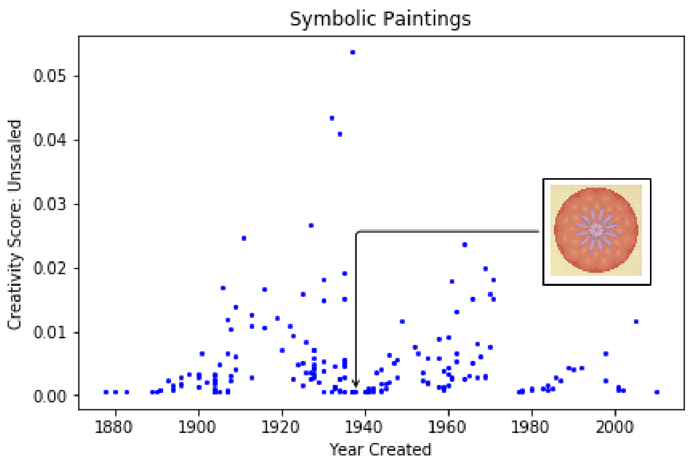


Figure 9: Example of Image Comparison

To check to see if my theory on the keypoints being an issue was correct, I decreased the threshold to just 500. This increased the total keypoints for the originally lowest scoring painting in Figure 9 from 4 to 75. The previously lowest scoring painting was now the highest. This was all due to lowering the threshold for identifying keypoints.

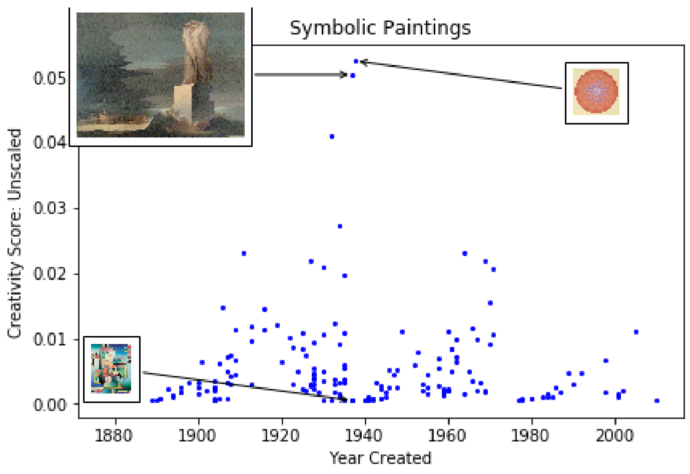


Figure 10: Example of Image Comparison

6 Conclusion

In conclusion, I did find the most creative artwork. Specifically for the genre of symbolic paintings. The title is ”Ecce Homo” and was painted in 1937, it achieved a score of 0.0537. While this was a successful result, I have questions about the accuracy of my system. With such a discrepancy in the keypoints between the different thresholds, I wish I had more time to tweak the parameters to come up with some different results. However, the base for this system has been laid down, and more work could be done to expand upon it. I believe that my work in this final project has shown what this framework is capable of, and what could be achieved by it. I showed the most creative symbolic painting, and the paintings most closely related to it.

7 Future Work

Future work in this field would include more similarity functions, and looking at more genres. This also includes the tweaking of the functions already created for this computational framework. I think that this would be an interesting research project to complete over the course of a year. Much more studying into different image analysis tools would be very beneficial. Also more information about the paintings would help achieve more similarity functions. I know that there could have been functions dealing with the histograms of the colors used in each painting, or just looking at the grey scaled images. These could have added additional categories that would have increased how successful this framework is.

References

[1] Ahmed Elgammal and Babak Saleh. Quantifying creativity in art networks. *arXiv preprint arXiv:1506.00711*, 2015.

[2] Itseez. Open source computer vision library. `https://github.com/itseez/opencv`, 2015.

[3] Kaggle. Painter by numbers. `https://www.kaggle.com/c/painter-by-numbers/data`, 2016.

[4] Wikiart. Wikiart visual art encyclopedia. `https://www.wikiart.org`, 2018.