The goal of this project was to build and validate a predictive model that can identify the artist of a given painting. Below you will find a full description of the dataset and the approaches I used to train my final model.

***Data:***

The data set I used was pulled from a former Kaggle competition entitled “Painter by Number” that ended in 2016. It includes over 100,000 images of paintings as well as a csv file containing information about each painting including artist, year, style, and title. The images were already broken out into two folders – a training folder with 79433 images and a testing folder with 23817 images. The csv file contains all information from both sets and has a feature that identifies if the painting was in the training set or not.

The original dataset contains paintings from 1504 artists. I chose to hone in and focus on 6 unique artists that had the highest number of paintings in the dataset and that represented a variety of artistic styles. The dataset contains approximately 500 paintings from each artist listed below.

**Claude Monet**: *Impressionism*

**Edgar Degas**: *Impressionism*

**Salvador Dali**: *Surrealism*

**Raphael Kirchner**: *Art Noveau (Modern)*

**Giovanni Battista Piranesi**: *Neoclassicism*

**John Singer Sargent**: *Realism*

***Preprocessing:***

Once my target variable labels were selected, I filtered out all other artists from the dataset and created a dataframe with only my selected artists. This cut down my data to 2975 rows which was much more manageable to work with. I then filtered out the testing images leaving me with 2254 rows.

Using PIL, I created copies of the 2254 images and converted the color scale to RGB. In order to run through a CNN, each image needed to be the same size. I chose to crop each image to 64x64 pixels which cut down features and improve speed. Although I lost a degree of image integrity by cropping, this was better than the alternative of adding a black padding to my images to reshape. Adding black borders would add arrays of ‘0’s to my data that would not help my model’s accuracy.

Once my images were cropped, I read each image into my jupyter notebook as a list of arrays (shape 2254 x 64 x 64 x 3). Next, I reshaped this 4D array into a 2D array (shape 2254 x 12288) so that I could convert it into a pandas dataframe and add the jpg column and artist column. Once the jpgs were pulled in, I was able to merge the artist column onto my dataframe.

Next, I identified X and y columns, converted y to categorical data and train-test-split my data. Then I standardized my data by dividing both my X\_train and X\_test by 255. Lastly, I converted each of the dataframes back into arrays, and reshaped them into 4-dimensional arrays so that I could run them through a Convolutional Neural Network.

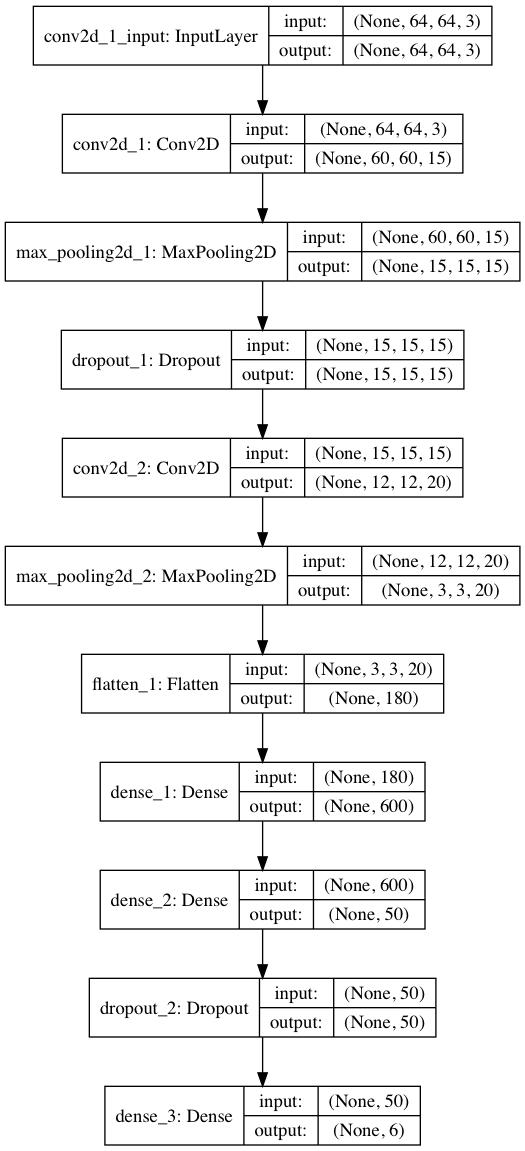
***Modeling:***

My first basic CNN consisted of one hidden Convolutional layer, one hidden Dense layer, and one Pooling later. The activation function was set to rectifier. With 20 epochs, this yielded me an accuracy score of .98 on the training data, but only a cross-validated accuracy of .57 on the testing data. Seemingly very overfit, I started by using Dropout regularization and L2 regularization, and then began to add more hidden layers in order to close the gap. I continued to tune by adding Convolutional layers, Dense layers, Dropout layers, and Pooling layers. I also experimented by tuning the number of epochs and the batch size. Note that the modeling portion of my project was completed on an ec2-user instance due to the size of my data. This sped up the process dramatically.

Once I had the basic structure of my model, I needed to find the optimal hyperparameters. In this case I chose to use RandomizedSearchCV over GridSearchCV, as Gridsearch painstakingly searches each and every parameter combination given. RandomizedSearchCV is often more efficient when running a deep neural network. This cross validated search gave me a good idea of the hyperparameters that were best suited and I was able to manually tune from there on my own.

Ultimately, more than one hidden convolutional layer and more than two hidden dense layers did not add any additional value to my model. The input layer is followed by a pooling layer and a dropout layer. The hidden convolutional layer is followed by a pooling layer, and the last hidden dense layer is followed by a pooling layer. On the right, you can actually see the outline of my final model with the shape of each input and output.

This model scores a cross-validated accuracy in the mid 60s with a training accuracy score in the high 60s. When I feed this model my test set, the accuracy drops to 50%. This suggests my model is still a bit overfit. Incorporating additional regularization techniques down the line should help to fix this. Although lower than I had hoped, the baseline accuracy here is 19% so this is still a significant improvement over the baseline. In the last paragraph, you will find additional improvement techniques that I will try going forward.



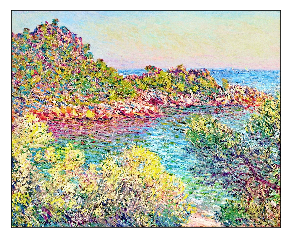
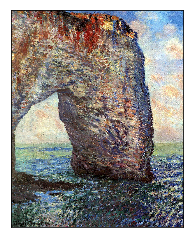
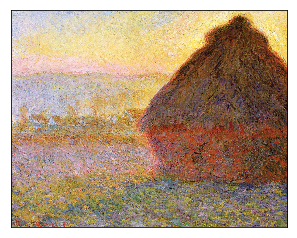
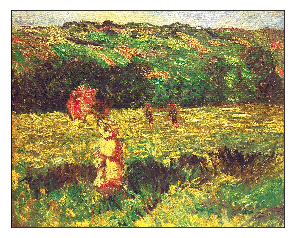
***Additional Analysis:***

Interestingly, although my overall accuracy was only 50%, certain artists were much classified with much more accuracy than others. See below for breakdown of accuracy by artist.

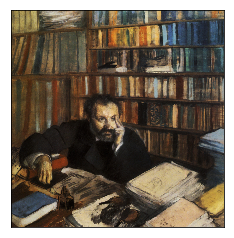
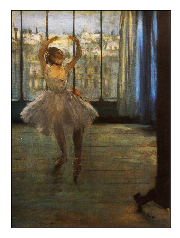
|  |  |
| --- | --- |
| Artist | Accuracy |
| Raphael Kirchner | 95% |
| Giovanni Battista Piranesi | 88% |
| John Singer Sargent | 48% |
| Salvador Dali | 29% |
| Claude Monet | 23% |
| Edgar Degas | 15% |

Below represents a sample of artwork by each painter. Visually it becomes clear why Raphael Kirchner and Giovanni Battista Piranesi were the most easily identifiable, as they have specific characteristics unique from the other artists. The bulk of Piranesi’s work is black and white, painted with a dot-like texture that looks almost fuzzy, and he focuses heavily on light, perspective and architecture. Kirchner also has a distinct style that is much more modern. He was primarily a portrait painter and is best known for his paintings and illustrations in magazines and even post cards in the early 20th century.

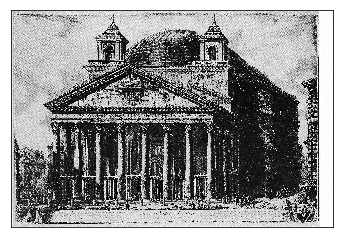
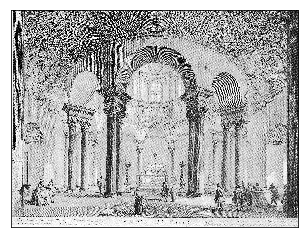
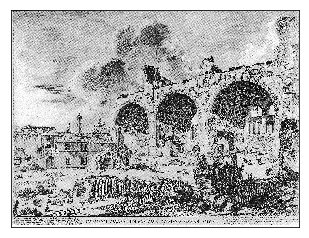
Claude Monet:

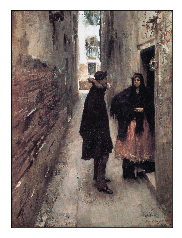
Edgar Degas:

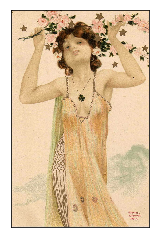
Giovanni Battista Piranesi:

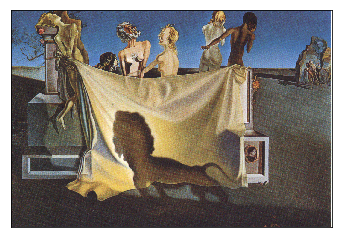
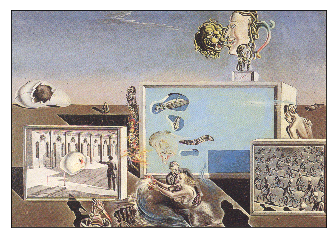
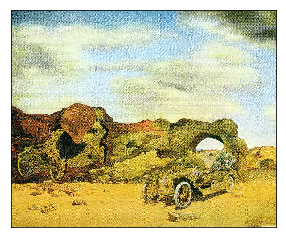
John Singer Sargent:

Raphael Kirchner:

Salvador Dali:

In addition, in the heatmap to the right, you can see the correlation between artists based on their average features. This model needs to be taken with a grain of salt, as it represents the average features and may miss certain characteristics. However, many of the results are not surprising. Piranesi is relatively uncorrelated to the other artists which is could be expected given my model was able to classify his paintings with such accuracy. It is on the other hand strange, that Monet is so uncorrelated to the others given his low score. Monet does have quite a bit of variety in his work so maybe the mean of his features is not a good representation of his work. Lastly, most of Dali’s and Degas’ misclassifications were mistaken as Kirchner’s paintings. This does make sense as the three are highly correlated in this instance.



It is difficult to identify the artist of each painting with complete accuracy for many reasons. The more classes that exist, the more a model generalizes and the harder this task becomes. Going forward I might try cutting down the number of artists further in order to improve the accuracy. I’d also like to spend time learning and trying out more advanced feature extractors. I tried using a SIFT feature extractor and bucketing the results into a histogram, but this gave less accurate results overall. I would need more time to refine this data and to tune this model in order to see improvement. Some other feature extractors I would like to try include SURF, HOG2x2 and SSIM. These have been proven to work well with artwork and they would likely yield a higher accuracy rate.

In the below images, you can see the keypoints that were extracted using SIFT. In the first image, SIFT did a great job identifying the object. The keypoints in second image are a little bit more vague suggesting that this method is will work better on simple portraits and images with less going on.





Regardless, there is so much more for me to explore in this field. I can’t wait to apply these methods and continue to improve my model and its accuracy.