COS 424 Final Project: Classifying Famous Artwork

Victor Hua Princeton University vhua@

Jeremy Pulmano Princeton University jpulmano@

Son Do Princeton University sqdo@

Max Land Princeton University mland@

Abstract

In this paper, we perform feature analysis and image classification on different paintings by matching them to their respective artists. That is, we take the full dataset containing thousands of paintings from 50 of the most influential artists of all time, as well as various subsets of this data, and develop several classifiers to identify a given painting's artist. First, we analyze the latent variables in the dataset using PCA and K-Means Clustering. Then, we reduce each painting to a uniform array of RGB values and train several classifiers – Logistic Regression (LR), Support Vector Machines (SVM), Random Forests (RF), and Convolutional Neural Networks (CNN) – to classify the test paintings for all 50 artists. We find that, while SVM and CNN perform the best, each classifier performs relatively poorly, with accuracy and F_1 scores below 0.30. We remedy this poor performance by several means: using Stratified K-Folds to split the data proportionally, converting each painting to grayscale, reducing the number of artists analyzed, and more. We found that SVM and CNN perform the best, reaching F_1 scores of 0.554 and 0.513 respectively. Lastly, as an extension of our work, we performed neural style transfer, taking arbitrary images and applying any painting's style (say, Van Gogh's Starry Night) to it using deep learning. Thus, this project provides an interesting introduction to image classification and its creative applications.

1 Introduction

Image classification has an extremely wide range of applications, from facial recognition within apps such as Google Photos, all the way to recognizing cancer within images of brain scans. In this project, we attempt to extend our coursework and explore one such application of image classification: famous paintings.

One of main goals in this project was to become more familiar with some of the many applications of image/object classification. Computer vision, in particular, is essential to autonomous vehicle engineering, image and face recognition on social networks, and much more [1].

In our case, studying paintings' characteristics and attributing them to their creators can be of considerable interest to art enthusiasts and especially those in the field of art history. We put a unique emphasis upon using machine learning to understand the ways in which famous artists created their paintings, which can be crucial for determining the many features – e.g. texture, line use, colors, subject, and much more – that are indicative of that artist's unique style. Thus, using machine learning models to analyze and classify paintings represents a creative and innovative way to approach the topic of image classification.

2 Related Work

Applications of machine learning to image classification of paintings is a relatively new and unseen field that has only risen within the past decade or so. In one of the very first papers on this topic, a group of researchers analyzed a data set consisting of 16-bit, grayscale, digitized representations of Vincent van Gogh's paintings and their brushstrokes [2]. The goal was to use image processing to

tackle the problem of artist authentication, using Support Vector Machines to classify the images. While the results were interesting and encouraging, the paper also suggested that using different signal analysis tools and richer image representations (such as color) might provide better results. This motivated our project: to apply new machine learning models, including SVM, to analyze, classify, and derive features from a much larger dataset of paintings created by a myriad of different artists.

Furthermore, in a more recent research paper on Art Style Recognition [3], the authors discovered that a bagging technique increased accuracy and stability of their classifier. This motivated us to explore Random Forests, which is an extension of over bagging. Bagging is used with the goal of reducing the variance of a decision tree, where in addition to taking a random subset of data, it also takes the random selection of features rather than using all features to grow trees. Moreover, the paper also centered on deep learning; through retraining and use of a residual neural network, the researchers were able to achieve a high accuracy rate when classifying images by art style. This motivated us to explore the use of Convolutional Neural Networks.

However, before running these classifiers, we needed to process our images. To do so, we used nearest neighbor interpolation after discovering its effectiveness in a paper on image interpolation methods [4]. The paper compared three interpolation methods out of nearest neighbor, bi-linear, and bi-cubic interpolation. Out of the three, nearest neighbor interpolation was the most efficient and fastest image processing method. Despite producing a lower quality and less sharp image than the other two methods, nearest neighbor interpolation is better at maintaining contrasts, which we decided was more important for image classification of paintings.

3 Methods

3.1 Data Processing

We are given a data set of 8354 images with varying dimensions, each one with an author name as a label. We reduce the dimensions of each image to a 40-by-40 image with nearest neighbor interpolation using the Python Image Library (PIL); then, we obtained the RGB values of each pixel. Thus, we generated a csv of dimensions 8354 x 4801, where each row represents a painting and each column represents a red, green, or blue value of a pixel in the image (40 x 40 x 3 = 4800). The last column denoted the artist, encoded as a integer between 1 and 50. We refer to this csv as `color.csv`.

Moreover, we also convert each of the images to grayscale and perform the same data processing as above (referred to hereon as `grayscale.csv`); this is because we hypothesized that the wide variation in color between paintings and artists might have caused our classifiers to perform extremely poorly, as discussed further in the Classification Methods section below.

It is also important to note that, although we originally attempted to classify paintings across all 50 artists, it soon became clear that this task was very difficult. That is, our accuracy and F-1 scores were rather poor for all of our classifiers, for reasons discussed later in the Results section. Thus, we decided to reduce the number of artists to 5 by simply taking the artists who had the most paintings in the dataset. This yielded more promising results.

3.2 Latent Variable Methods

1. **K-Means Clustering** (K-Means): K-Means is an algorithm that is broken into 3 steps (also known as Lloyd's alogrithm). The first is setting each sample to a random cluster (n different clusters). The second is finding the mean of each of the clusters, and the third is assigning each sample to a new cluster that is closest to that cluster. Steps 2 and 3 is repeated until there is no change in cluster assignments.
2. **Principal Components Analysis** (PCA): A classical technique to find low-dimensional representations which are linear projection of the original data. This is done via eigen-decomposition where each principal component are linearly independent eigenvectors with a corresponding eigenvalue.

3.3 Classification Methods

We apply the following classifiers from sklearn [5] and tensorflow [6] to our datasets. Settings are the default unless otherwise specified:

- **Logistic Regression (LR)**: The algorithm fits a logarithmic function to a multiclass variable by minimizing the residual error between the function and true values of the testing data, with multinomial cross-entropy loss and LBFGS optimization.
- **Random Forests (RF)**: This model uses an ensemble of decision trees and uses a randomized subset of features to make decisions, with $N = 10$ trees (where N was tested for $\{1,2,\ldots, 20\}$ with $N = 10$ performing the best).
- **Support Vector Machines (SVM)**: The classification algorithm separates the data into two categories by maximizing the distance between points of either category, with one-vs-one multiclass support and gamma set to "scale." Gamma in this case uses $1/(n_{\text{features}} \cdot X.\text{var}())$ as its value; a lower gamma value prevented overfitting and returned the best results [7].
- **Convolutional Neural Networks (CNN):** Our CNN contained one convolutional layer, was trained for 20 epochs with a batch size of 256, compiled with a categorical cross entropy loss function and Adam optimizer [8].

*Although we had not explored Convolutional Neural Networks in depth, we discovered in our research that neural networks are very effective in classifying images [8] and thus attempted to implement them in our classification. We draw upon a repository that aims to classify fashion images of t-shirts, sweaters, tank tops, etc. using a convolutional neural network [9].

3.4 Evaluation Methods

We measure the error in our models using accuracy, precision, recall, and F_1-score. We build a classification report that shows us these metrics for each of the 50 artists, as well as confusion matrices that show us the number of correctly predicted samples for each artist in the dataset.

To evaluate our latent variable methods we used two separate methods, one for K-Means Clustering and another for PCA. For PCA we used a scree plot – that is, a plot of eigenvalues – to observe if there was an elbow in the curve (see Section 4); such a scree plot helps us understand how much data is being accounted for given the amount of principal components used. For K-Means clustering, we used the objective function, which is essentially the total squared distance from each image with its centroid. We compare the the objective function with various values of cluster amounts (k-values) to uncover the best amount of clusters to use for K-Means Clustering.

3.5 Neural Style Transfer

As an interesting extension of our work, we implemented neural style transfer of some of the paintings included in the original dataset. Neural style transfer refers to algorithms that seek to manipulate images to take the appearance and style of another image. In our case, we utilize and modify the parameters of already existing techniques to apply the style of any given artist to various pictures of Princeton's campus to observe the results. For our methodology, we mostly follow the methods detailed in an article entitled "Neural Style Transfer: Creating Art with Deep Learning" [10], and we also utilize the code within a repository entitled "Neural Style Transfer" [11]. We mainly focus on tweaking the weights of the total loss function of a convolutional neural network and observing the resulting images; to briefly describe the loss function, let us assume that we have our content image C (the image that we would like to style), the style image S (the painting), and the generated image G). The *content* loss is defined as the euclidean distance between the feature map F^l of C and that of Y, where l is any given content layer in the network:

$$\mathcal{L}_{\text{content}} = \frac{1}{2}\sum_{i,j}(F^l_{i,j} - P^l_{i,j})^2$$

The *style* loss is defined similarly as the euclidean distance between the Gram matrix G^l of the feature map of S and the Gram matrix A^l of the feature map of Y:

$$\mathcal{L}_{\text{style}} = \frac{1}{2}\sum_{i,j}(G^l_{i,j} - A^l_{i,j})^2$$

Then, the total loss is simply the weighted sum of both the style and content losses:

$$\mathcal{L}_{\text{total}} = \alpha \mathcal{L}_{\text{content}} + \beta \mathcal{L}_{\text{style}}$$

We hyperparameterize α and β, as well as modify the number of iterations, to find the best results. Our results are shown and described in Section 6.

4 Latent Variable Analysis

4.1 PCA

From the scree plots, we see that the elbow starts to form a little before 100 principal components (PCs) for both the color and grayscale image data. Thus, from the scree plots we can assume that the first few PCs describe the majority of the variance of the data. Reducing the features to 100 PCs would still be a good representation of the data.

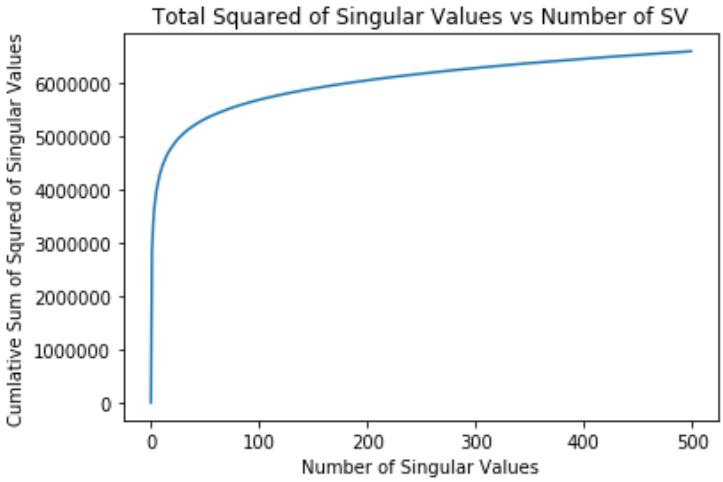

Figure 1: Scree Plot from PCA

Next, graphing the photos projected on the PC1 vs PC2 (Figure 7 and Figure 9 in the Appendix) we see that PC1 really focuses light vs dark backgrounds for both the gray scale and color artworks. PC2 on the other hand focuses on how dark the top vs bottom is for both the gray scale and the color representation of the artwork. PC1 vs. PC2 did not vary much with gray scale or with color.

On the other hand PC3vPC4 was much different depending on the type of photo we used (gray scale vs color). For the gray scale PC3vPC4 (Figure 8) the x-axis (PC3) the left had more photos where the majority of the lighter colors was in the middle where as the right had lighter colors near the edges. Pc4 for gray scale data was representing whether the right or left was darker (the bottom photos had dark colors on right whereas the top photos where the dark grays were on the right). PC3vPC4 for the color artwork data (Figure 10) grouped mostly color together with blue dominated colors on top and orange/lighter colors near the bottom.

Finally, we converted the top 16 eigenvectors of the PCs of both the gray scale and the colored data to see the PCs more profoundly (eigen-images). This can be found in the appendix (Figure 11). Sadly, we weren't able to convert the colored data into PC pictures as the eigenvectors created had negative values due to the high variance of color.

4.2 K-Means Clustering

For KMeans we used the objective function to find the best K number of clusters for gray scale and color artwork data. We found that after more than 10 clusters the objective function did not decrease much, and thus we simply used KMeans clustering with 10 clusters for both color and gray scale. We found that for gray scale data, the different clusters of photos represented different areas that

were lighter. This meant KMeans clustering often grouped photos based on where light and dark colors predominantly exist on the paintings.

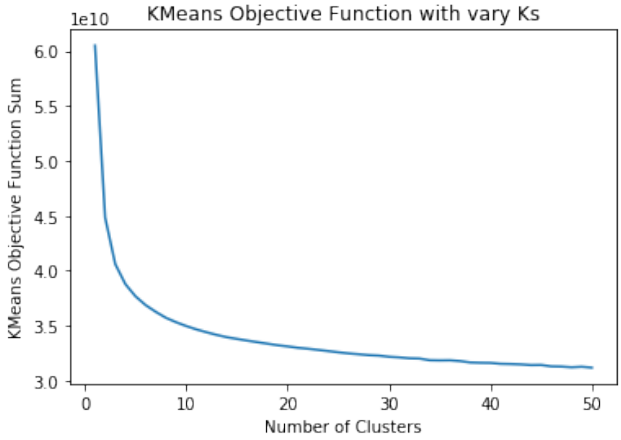

Figure 2: Objective Function Graph for Grayscale Artwork

On the other hand, the clusters for colored data were mostly either gray or brown. This indicated that clusters were not grouped via color, but more by the gradient of dark vs light areas. Furthermore, there were pronounced shapes in the clusters, indicating that shape of the images also played a significant role in the clusters.

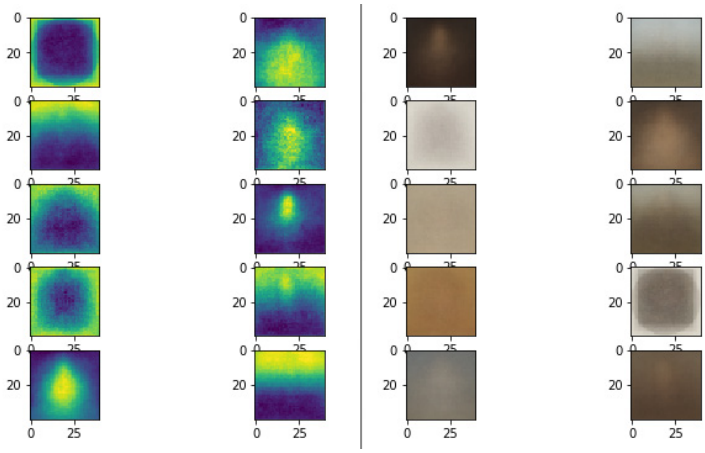

Figure 3: **Left:** K-Means Clusters for Grayscale Artwork. **Right:** K-Means Clusters for Color Artwork

5 Classification Results

5.1 Initial Testing

As mentioned briefly in the Data Processing section, we first attempted to use the entire data set, `color.csv`, which we split into an 80/20 train-test ratio, and to predict the held-out paintings for each of the 50 artists. However, every classifier performed rather poorly – the best classifiers, SVM and CNN, returned F_1 scores of 0.132 and 0.149, respectively (see Figure 6).

As such, we attempted to convert the pictures to grayscale to reduce the variation in color between paintings and artists, so that our classifiers could focus more on other latent variables within the paintings such as: texture and line usage (are lines jagged or smooth?), subject matter (landscape or person?), etc. However, despite our reasoning, this proved to be slightly detrimental to our metrics, with accuracies and F_1 scores dropping slightly, by approximately 0.02-0.05. As a result, we

believed that, even though the data had less variation, there was less information overall for our classifiers to use to make predictions, thus bringing our metrics down. As such, it seemed as if that color plays an important role in classification, leading us to investigate further.

5.1.1 Variance in Color between Artists

Per a suggestion that we received during our poster session, we decided to find the *variance* of color in paintings between artists. To do so, we took the variance of RGB values across every row of `color.csv`. Then, we found the average variance across the paintings of each of the 50 artists. We also found the paintings of minimum variance and maximum variance for each artist. The results were very interesting; artists such as Albrecht Durer, who painted in only black and white, had the lowest average color variance, as one would expect. Other artists such as Andy Warhol, who painted in a wide variety of vivid colors, had the highest average color variance. See Figure 4 for examples. This led us to conclude that, while each artist generally had a large amount of variance in each of their paintings, an artist's use of color (or lack thereof) can still play a very important role in matching the painting to artist.

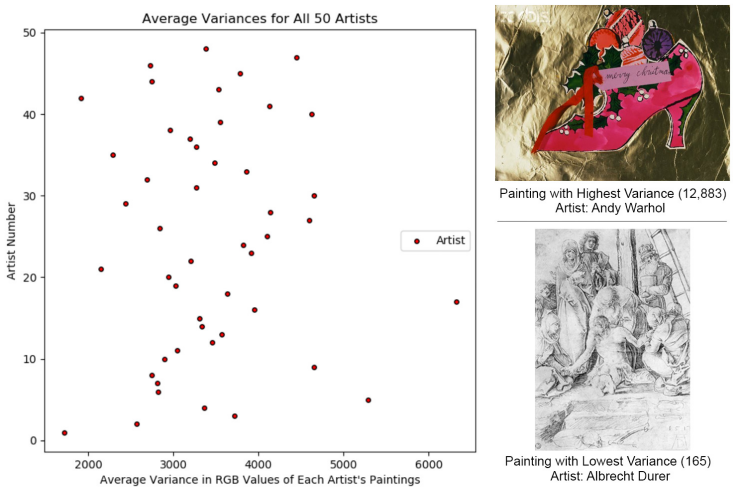

Figure 4: **Scatterplot of average variance per artist**, with paintings of max. and min. variance.

Now, focusing our attention back on our classification methods, we continued to explore new steps and adjust our goals to make our project more feasible. One step we took was utilizing stratified K-Folds to create an 80/20 train/test paintings ratio for each artist, since the number of paintings by each artist varied drastically. Specifically, the maximum number of paintings was 877, the minimum 24, and the mean 167 across the artists. Thus, Stratified K-Folds helped us guarantee that each artist had a similar proportion of training paintings to test paintings.

Next, we reduced the number of artists to 10, which improved our metrics; even so, however, it seemed as if our classifiers were only correctly identifying the paintings whose artists had the *most* paintings in the dataset: in other words, given that Van Gogh (#1) had 877 paintings in the dataset and Titian had 255 (#10), our classifiers often correctly identified Van Gogh's paintings but not Titian's. As such, we decided to reduce the number of artists to 5, which seemed much more feasible for a classification goal.

5.2 Classification on Five Artists

These top five artists, numbered 1-5 and represented as such in the confusion matrices in Figure 5, are the following:

1. Albrecht Durer 328 paintings; **2. Edgar Degas**, 702 paintings; **3. Pablo Picasso**, 439 paintings; **4. Pierre-Auguste Renoir**, 336 paintings; **5. Vincent van Gogh**, 877 paintings.

In Figure 5, we highlight the performance of CNN and SVM, which outperformed our other classifiers; this was mostly expected given our findings in related works on the effectiveness of neural

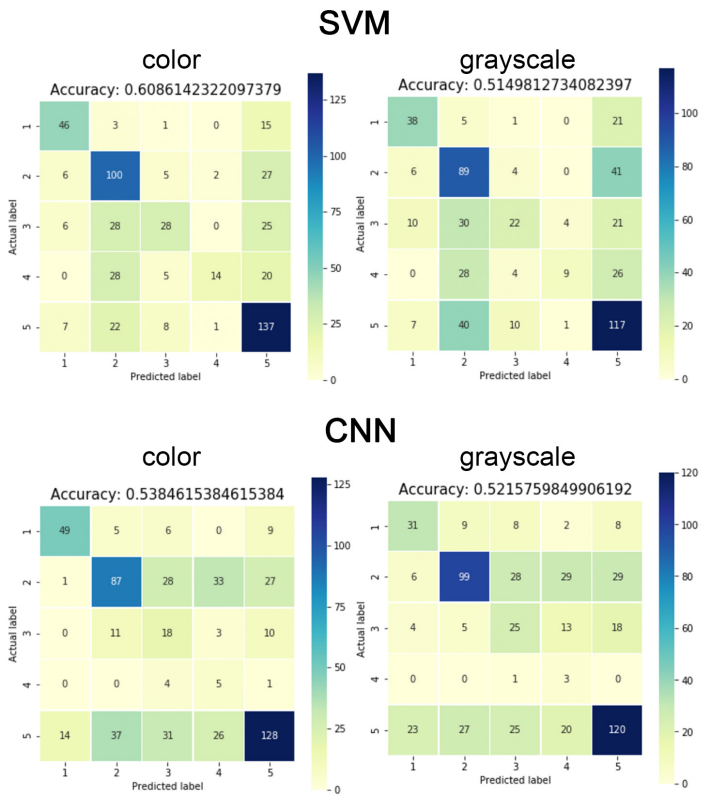

Figure 5: **Confusion Matrices** for the top five artists with the most paintings, featuring Support Vector Machines and Convolutional Neural Networks. On the x-axis is the predicted label and on the y-axis is the actual label. Darker shades represent more matches for that pair of labels. Thus, a dark diagonal from the top left corner to the bottom right corner is most ideal.

networks and support vector machines on image classification. We also observe that classifying with color outperformed classifying with gray scale. As mentioned previously, this was counter-intuitive to us, since we assumed that the huge variance in color would detract from accuracy. However, the results of our classifiers showed that color added value and improved the accuracy of our classifiers.

Now comparing artists, we can see that Artists 2 and 5 – Edgar Degas and Vincent van Gogh – had the highest prediction accuracy. However, this is likely attributed to the fact that those two artists had the largest sample size (more than double the other three), so our classifiers had more data to more accurately train for those two artists in particular. On the other hand, Artists 1 and 4 – Albrecht Durer and Pierre-Auguste Renoir – had the two lowest sample sizes, but Artist 1 had a significantly higher accuracy prediction that Artist 4. Looking into the paintings themselves, this is likely due to the fact that almost all of the paintings of Artist 1 are in black and white (as we saw in Figure 4) and thus have a more consistent style. On the other hand, Artist 4 had a much more diverse set of paintings; In addition to painting with various color schemes, he also painted many different subjects: portraits, landscape, still-lifes, etc.

Looking now at Figure 6 below, we see that our tests with color outperformed grayscale again. Next, SVM and CNN were the top two performers for both color and grayscale. Specifically, SVM performed the best for color and CNN performed the best for grayscale. Again, the reason for the decrease of performance for SVM is most likely contributed to the loss of information when converted to grayscale. Nonetheless, the fact that SVM outperformed CNN for color is significant. We expected CNN to outperform SVM, given the current literature on the effectiveness of CNN. However, the reason for CNN failing to perform the best might be attributed to our small sample size. Due to the complex nature of CNNs, they often require a massive amount of data to be as

effective as possible. However, our dataset is relatively small; in reality, these artists, even being the most famous of all time, might not provide us with enough data to allow us to train effective classifiers. Thus, this finding is interesting because the further use of SVMs and improvement of CNNs is worth considering for those looking to further expand on the potential of using machine learning algorithms to classify paintings.

Color: 50 Artists				
Classifier	Accuracy	Precision	Recall	F1
LR	0.161	0.096	0.092	0.089
RF	0.155	0.027	0.051	0.028
SVM	0.297	0.237	0.145	0.132
CNN	0.270	0.153	0.181	0.149

Grayscale: 50 Artists				
Classifier	Accuracy	Precision	Recall	F-1
LR	0.096	0.104	0.063	0.069
RF	0.137	0.024	0.038	0.018
SVM	0.234	0.175	0.105	0.093
CNN	0.260	0.128	0.199	0.118

Color: 5 Artists				
Classifier	Accuracy	Precision	Recall	F1
LR	0.461	0.431	0.456	0.440
RF	0.404	0.279	0.335	0.298
SVM	0.609	0.658	0.547	0.554
CNN	0.563	0.520	0.545	0.513

Grayscale: 5 Artists				
Classifier	Accuracy	Precision	Recall	F-1
LR	0.335	0.309	0.315	0.309
RF	0.375	0.256	0.292	0.260
SVM	0.515	0.557	0.455	0.458
CNN	0.548	0.496	0.529	0.500

Figure 6: **Metric tables for every classifier** evaluated by accuracy, precision, recall, and F_1 scores across color and grayscale with both 50 artists and 5 artists. Best performance is highlighted in green for each table.

6 Extension: Style Transfer

Our last step in our project was to observe the results from neural style transfer after tweaking the content weight and style weight parameters of the loss function. We use various pictures of Princeton's campus; our top two results are highlighted in the Appendix, Figure 12. In the loss function, we set $\alpha = 1$ and $\beta = 500$, and we set the number of iterations to 10; we did so because higher values of β and lower values of α returned images that more closely mirrored the style of the given painting, with the drawback of greater content loss. That said, this increase in content loss was not worrisome to us because the pictures we provided were already very recognizable. Unfortunately, the algorithm we used took several hours to complete per image – we attempted reducing the number of iterations and the output image size but this ultimately led to distorted, seemingly incomplete images – and so we were not quite able to produce a wide variety of high-quality images, other than the ones shown in the Appendix.

7 Conclusion

In conclusion, this assignment answered many of the questions posed to us after reading past related works on the topic of image classification of paintings using machine learning. For one, our results confirmed the theory suggested by one of the papers that using the original color of the images produced more accurate classification models. Second, reducing the number of artists labels from 50 to 5 dramatically increased accuracy of our models. In terms of the classification models themselves, SVM and CNN both had the best performances across all models, with SVM edging out CNN in terms of color classification.

However, despite these results, much more research needs to be conducted before these classification models can be used for real world applications, such as detecting if a painting is fake. A large part of the setback is the sample sizes of our dataset. Complex models such as CNN require a large amount of data, and we found that the relatively low number of paintings per artist was not enough to accurately classify on 50 artists. Furthermore, our classification models are biased towards artists with more paintings. As seen in Figure 5, artists with more paintings were generally more likely to have paintings classified under their label. Ultimately, the biggest takeaway from this assignment is feeding the classification models a quality data set. SVM and CNN are good models to use, but different methods of image processing should be explored in order to make the those two specific models as accurate as possible.

Furthermore, on the topic of neural style transfer, we hope to further explore convolutional neural networks, tweak more parameters than just content weight, loss weight, and iterations, and thus create output images of even higher quality.

Possible extensions to this project would be to use an autoencoder (specially the one form Keras Blog). "Autoencoding" is a data compression algorithm that is data-specific, lossy, and learned automatically from examples rather than engineered by a human. It is trained via data and applied to the rest of the data. Autoencoders are known to be a great way to do dimesionality reduction for data visualization. Furthermore, one of the faults in our analysis was that we used mostly euclidean distances of pixels to compare images when may not be the best way. Autoencoders help compress the data into more interesting features that may result in better latent modeling and classifications [12].

References

[1] "Rohan & Lenny #2: Convolutional Neural Networks." *Medium*. https://ayearofai.com/rohan-lenny-2-convolutional-neural-networks-5f4cd480a60b

[2] C.R. Johnson, Jr., E. Hendriks, I.J. Berezhnoy, I.J., E. Brevdo, S.M. Hughes, I. Daubechies, J. Li, E. Postma, and J.Z. Wang. Image processing for artist identification. Signal Processing Magazine, IEEE, 25(4):37-48, 2008

[3] Lecoutre, Adrian Negrevergne, Benjamin & Yger, Florian. (2017). Recognizing Art Style Automatically with deep learning.

[4] Han, Dianyuan. (2013). Comparison of Commonly Used Image Interpolation Methods. Proceedings of the 2nd International Conference on Computer Science and Electronics Engineering. 10.2991/iccsee.2013.391.

[5] Sci-Kit Machine Learning Libraries. https://scikit-learn.org/.

[6] TensorFlow Machine Learning Libraries. https://www.tensorflow.org/

[7] Ben Fraj, Mohtadi. "In Depth: Parameter Tuning for SVC." *Medium.* https://medium.com/all-things-ai/in-depth-parameter-tuning-for-svc-758215394769

[8] Le, James. "4 Convolutional Neural Network Models." *Medium.* https://towardsdatascience.com/the-4-convolutional-neural-network-models-that-can-classify-your-fashion-images-9fe7f3e5399d

[9] "CNN: 1 Convolutional Layer." *Github*. Repository created by user "khanhnamle1994." https://github.com/khanhnamle1994/fashion-mnist/blob/master/CNN-1Conv.ipynb

[10] Yuan, Raymond. "Neural Style Transfer: Creating Art with Deep Learning using tf.keras and eager execution." *Medium & TensorFlow*. https://medium.com/tensorflow/neural-style-transfer-creating-art-with-deep-learning-using-tf-keras-and-eager-execution-7d541ac31398

[11] "Neural Style Transfer." *Github.* Repository created by user "titu1994." https://github.com/titu1994/Neural-Style-Transfer

[12] "Building Autoencoders in Keras." *The Keras Blog.*https://blog.keras.io/building-autoencoders-in-keras.html

8 Appendix

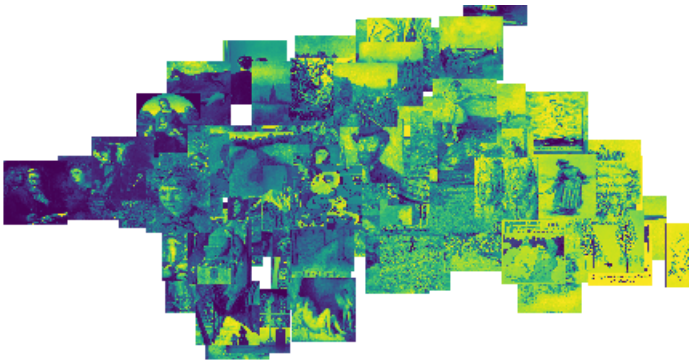

Figure 7: plot of PC1(x-axis) vs PC2(y-axis) (gray)

Figure 8: Plot of PC3 (x-axis) vs PC4 (y-axis) (gray)

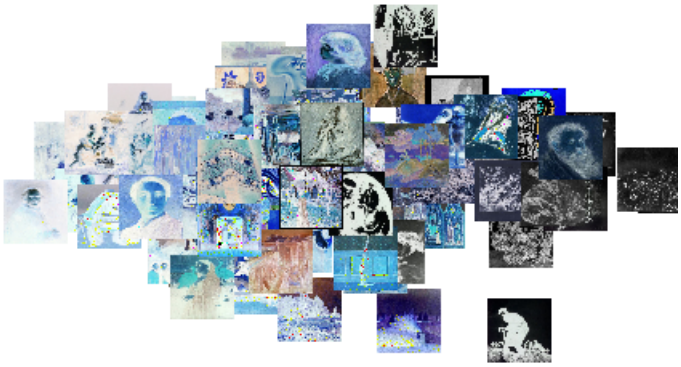

Figure 9: Plot of PC1 (x-axis) vs. PC2 (y-axis)

Figure 10: Plot of PC3 (x-axis) vs PC4 (y-axis)

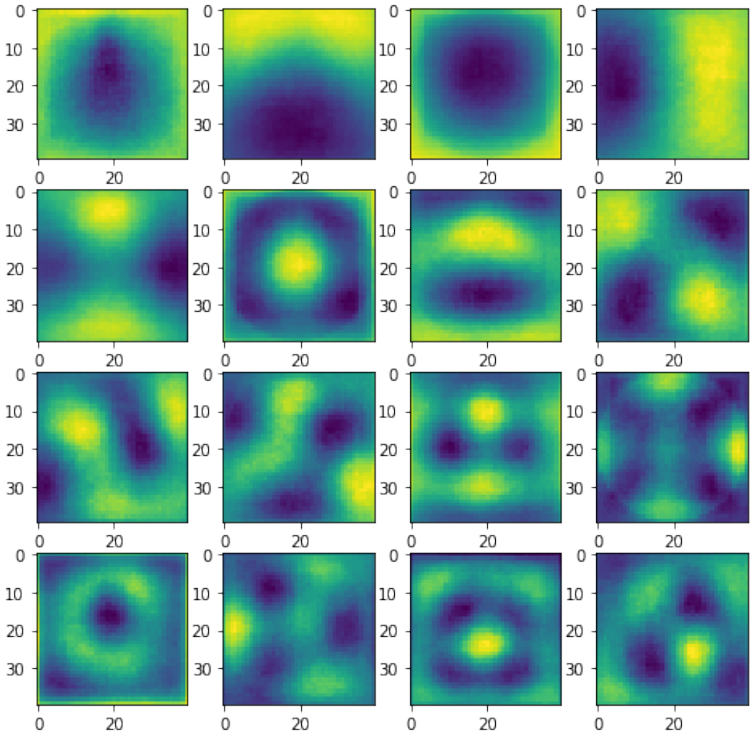

Figure 11: Plot of PC3 (x-axis) vs PC4 (y-axis)

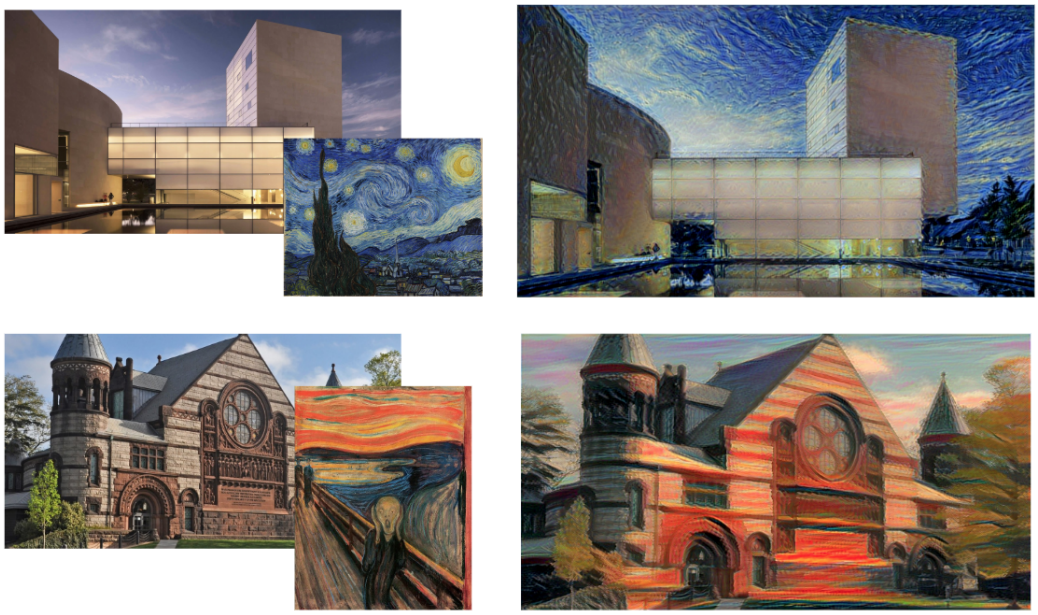

Figure 12: **Sample output from Neural Style Transfer**: The Lewis Center for the Arts stylized with van Gogh's "Starry Night," and Richardson Auditorium stylized with Edvard Munch's "Scream." We placed an emphasis on style weight over content weight, set with $n = 10$ iterations, which provided us with more easily recognizable output.