Comparison of Machine Learning Techniques for Artist Identification

Jennie Chen
Department of Computer Science
Stanford University
jchen437@stanford.edu

Andrew Deng
Department of Electrical Engineering
Stanford University
andrewde@stanford.edu

1 Introduction

Artist identification is the task of identifying the artist of a work given only the image with no other metadata. Many paintings have unknown or highly contested artists, and experts in the field need a long time to learn the styles of various artists before being able to analyze paintings. Additionally, even with significant expertise, artist identification can be difficult because one artist's style can vary wildly from painting to painting. With machine learning, we can provide experts with baseline estimate to reduce necessary time and effort, as well as making artist identification more accessible to those with less experience.

Additionally, this task was chosen because it is a fairly straightforward image classification task, and we want to compare conventional machine learning techniques for image classification to more recent deep learning techniques. Specifically, in this report we will compare the use of feature extraction with an SVM to the use of a CNN. Our dataset contains 256 x 256 x 3 color images of paintings; for the SVM, we extract features from these images as the input for the classifier, while we feed in the raw image to the CNN. For both models, the output is a predicted artist. The metrics we chose to compare the two methods are accuracy, training and inference times, and ease of implementation.

2 Related work

As previously stated, artist identification is often a task done by human experts, such as curators in museums, art historians, and other collectors. However, recent times has lead to a marked increase in computational methods for artist identification.

Much of previous work on this task involves exploration into feature extraction and the subsequent application of a classifier like an SVM. Blessing and Wen uses features including Dense SIFT, HOG2x2, SSIM, and texton histograms to classify works using an SVM with 85.13% accuracy [1]. Similarly, color moments and Tamura textural features are used in addition to SIFT by Liu and Jiang in classifying Chinese paintings [2], and GIST features, Classemes, and PiCoDes are used by Saleh and Elgammal [3]. Work on similar tasks such as art style classification and aesthetic evaluation are also generally feature-based; Misumi et al. utilize SIFT features when classifying works in 3 different styles [4], and Sahu et al. utilize Local Binary Patterns, color histograms, HOG, and GIST in their aesthetic evaluation of artwork [5].

Approaches using deep learning have also been very successful on this task. Viswanathan explores the use of three different CNN models, demonstrating that features from ImageNet are generally applicable to artist identification and thus showing that transfer learning can be very useful for this task [6]. Balakrishnan, Rosston, and Tang similarly explore the value of transfer learning on fine-tuned ResNet and VGG models, achieving 90.75% accuracy when classifying works from the Rijksmuseum between 10 artists and 87.7% when classifying between 20 artists [7].

Most prior work has worked with datasets involving relatively few classes; Blessing and Wen only used work from 7 artists [1], while Liu and Jiang used work from 4 artists [2]. Jou and Agrawal similarly only use paintings from 5 artists [8]. Balakrishnan et al. build classifiers for both 10 and 20 artists [7], while Saleh and Elgammal classify with 23 artists [3], and Viswanathan uses by far the most general dataset by classifying between 57 artists [6]. In similar tasks, Misumi et al. only classify between three different styles [4]. While we do not have the largest label space, our dataset has more variety than a large portion of related work. Additionally, most prior work focuses only on either a feature-based approach or a deep-learning approach. Our work differs in that we aim to explore both approaches and explicitly compare their performance.

3 Dataset and Features

3.1 Obtaining the Data

Our data is obtained from the dataset for the Kaggle competition "Painters by Numbers" [9], which focuses on the task of taking pairs of paintings and identifying if they have the same artist. The dataset is largely obtained from WikiArt, with additional contributions by other artists; it contains roughly 103,000 paintings from around 2300 artists. For our work, we narrowed down the data by only selecting artists with at least 450 paintings, resulting in a final dataset of 7462 paintings from 15 artists. Each artist had between 450 and 499 paintings, so classes were roughly balanced. Each color image was resized to 256 by 256 pixels; the data was then randomly divided into training, validation, and test with a 80%/10%/10% split independently among each class. As a final count, we had 5982 training images, 737 validation images, and 743 test images.

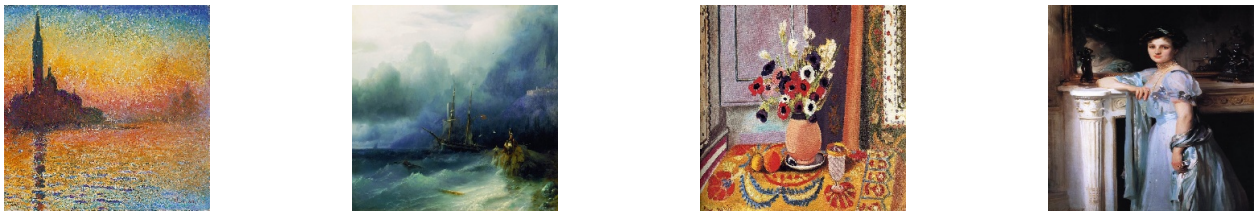
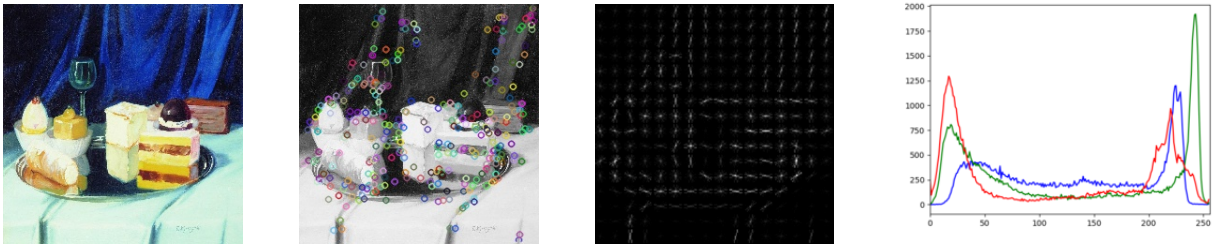


Figure 1: Example input paintings by Monet, Matisse, Aivazovsky, and Sargent

For the CNN, augmentation was performed on the training dataset with rotation, zooming, flipping, and shearing operations. This involved creating transformed duplicates of existing training examples, increasing the number of our training examples by a factor of 4 up to a total of 23928.

3.2 Feature Extraction

A set of 6 image descriptors were explored as features for the SVM: GIST descriptors, Hu moments, color histograms, SIFT keypoints, histogram of oriented gradients, and Haralick textures. Each of the 64 possible feature combinations was explored by concatenating feature vectors obtained from each method horizontally. After exploring the combinations thoroughly, the 3 most useful features were found to be the GIST descriptors, Hu moments, and color histograms.



(a) Example painting (b) SIFT keypoints (c) HOG gradients (d) Color histogram

Figure 2: Examples of feature extraction

GIST descriptors are a set of 5 attributes that represent intuitive properties of a scene: naturalness, openness, roughness, expansion, and ruggedness. These attributes were determined by Aude Oliva and Antonio Torralba [10] to be a good representation of a scene. Each attribute is calculated for an image by taking the energy spectrum of the image and performing the inner product with empirically determined Discriminant Spectral Templates (DSTs), made of a linear combination of Karhunen-Loeve basis functions for the energy spectrum with coefficients determined by the authors of the paper.

Hu moments are a set of 7 polynomial combinations of image moments that are defined as to be scale, shift, and rotation invariant. Scale and shift invariance are ensured by using the central moments to account for shift, and scaling by the 0th moment to account for scale. Rotation invariance is obtained by the specific polynomial combinations of moments from Ming-Huei Hu [11].

The color histogram is a simple set of 3 histograms, representing the distribution of color appearances in the image. A histogram for each color was computed by quantizing the color values for each channel into 8 bins and counting appearances of each quantized color. The histogram has no representation of the spatial distribution of the colors, only their appearance.

Each feature combination was scaled using min/max scaling so that each element was in the rage $[0, 1)$. For input to the SVM, PCA was applied with 100 components on the training features to determine the top 100 principal components. Test data was projected onto the same principal components prior to prediction. Examples of feature extraction can be seen in Figure 2.

4 Methods

4.1 SVM

The SVM used was a multiclass (15 classes) SVM with the RBF kernel. The input to the SVM was the set of feature vectors decomposed to the first 100 principal components, vectors of length 100.
The SVM was trained to minimize the objective

$$\min_{w,b,\xi} \|w\|_2^2 + C \sum_{i=1}^{n} \xi_i$$

$$\text{sub. to } y_i(w^T \phi(x_i) + b) \geq 1 - \xi_i$$
$$\xi_i \geq 0,\ i = 1, \ldots, n$$

for each class, where a one-vs-rest classification scheme was used so a classifier for each class was trained to identify that particular class. For each class, a distinct SVM finds the margin that best separates the data in one class from the data in the rest after the RBF kernel is applied, while minimizing the allowable misclassification terms ξ_i.

4.2 CNN

As a deep learning approach to image classification, the CNN is a neural network that consists of several layers of different types, including convolutional, max pooling, ReLU activation, dropout, dense/fully connected, and softmax. Each convolutional layer is a set of learnable 2D filters, which are applied to input data by the 2D cross-correlation operation. In max pooling layers, data is downsampled by a set factor n by choosing only the max value in each $n \times n$ square to propagate forward. The ReLU activation function is a unit ramp function $f(x) = \max(x, 0)$ that allows for nonlinearlity in the network. In dense layers, input data is flattened into 1D vectors, multiplied by a matrix of learnable weights, and added with a learnable bias. Dropout removes a percentage of activations to help prevent overfitting. Finally, the softmax layer computes the class probabilities for the data.

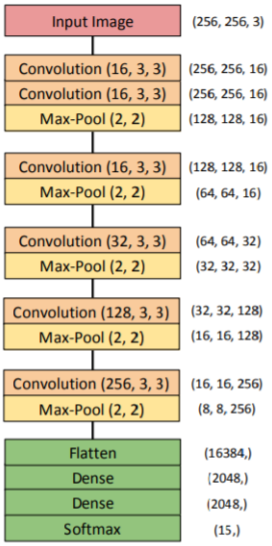


Figure 3: Architecture for our CNN

For our CNN, we base our architecture on the winning submission to the Kaggle "Painters by Numbers" competition [12], as well as on work by Viswanathan [6]. Taking in raw 256 by 256 color images as input, the CNN has 6 convolutional layers using 3x3 filters as well as 5 max-pooling layers to reduce the dimensionality of each feature map by a factor of 2. The filters were chosen to have window size 3, as multiple 3x3 filters can mimic the effects of filters of other sizes, such as 5x5 or 7x7 filters. Each convolutional layer uses a ReLU activation function and additionally uses batch normalization to reduce the reliance between layers. These convolutional and max-pooling layers are followed by a single dropout layer, 2 dense layers, and another dropout layer, with both dropout layers using a keep-probability of 0.5. Finally, cross-entropy loss is computed on the outputs of the softmax layer:

$$L_i = -\log(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}})$$

where L_i indicates the loss for example i, y_i is the correct class for example i, j is one of the potential classes, and f_k indicates the score for class k according to the CNN. The architecture is shown in Figure 3 without the dropout or batch normalization layers for readability.

5 Experiments

Our SVM model is implemented with the scikit-learn package [13], while our CNN is implemented using Keras [14]. Feature extraction relied largely on the scikit-learn package as well as the OpenCV package [15]. The SVM model was trained using the entire training set as one batch. Learning rate was not adjusted beyond the default value because we did not notice any problems with slow convergence/divergence.

The CNN was trained using an Adam optimizer with a learning rate of 0.000074 and the default parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$; weights were initialized with He normal initialization. It was found that increasing the learning rate beyond 0.0001 would cause the training to diverge. The CNN model was trained using batches of size 96.

To choose the best hyperparameters (features, γ, and C) for our SVM, we ran a grid search and 3-fold cross validation across $\gamma \in (10^{-5}, 10^5)$ and $C \in (10^{-5}, 10^5)$ using the training set for each feature combination. The model for each feature combination with best γ and C was then tested on our validation data, and the best was chosen as our best model. The CNN was also trained multiple times using different architectures (adding and removing convolutional and max-pooling layers), and the best model was chosen using the final validation accuracy after training.

6 Results and Discussion

6.1 Performance

The metric we used for judging our models was accuracy on the test set. Our dataset was overall reasonably balanced, so this should be a good measure for how our models perform on these classes. We found that the 3 most useful features in classifying artists were the GIST descriptors, Hu moments, and color histograms.

Model	Training Accuracy (%)	Test Accuracy (%)	Train Time (s)	Inference Time (ms)	Features Used
LR*	67.4	61.2	1.86	179.8	GIST, Color Hist
SVM	**97.9**	**68.1**	**6.78**	**180.2**	**GIST, Hu Moments**
SVM	97.8	67.3	6.69	181.0	GIST, Color Hist
SVM	97.1	61.2	5.12	178.4	GIST, Color Hist, Hu Moments
CNN	**81.3**	**74.7**	**28921**	**12.2**	**NA**

* Baseline model - logistic regression with best combination of features

Figure 4: Accuracy across models

Artist	Precision	Recall
Albrecht Durer	0.7	0.64
Boris Kustodiev	0.58	0.52
Camille Corot	0.53	0.63
Camille Pissarro	0.52	0.54
Childe Hassam	0.56	0.69
Claude Monet	0.55	0.58
Edgar Degas	0.59	0.59
Eugene Boudin	0.68	0.72
Giovanni Battista Piranesi	0.98	1
Gustave Dore	0.88	0.86
Henri Matisse	0.74	0.7
Ilya Repin	0.62	0.46
Ivan Aivazovsky	0.84	0.86
Ivan Shishkin	0.78	0.64
John Singer Sargent	0.59	0.64

(a) SVM

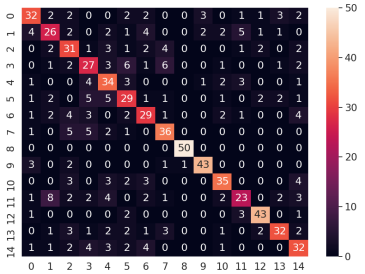
Artist	Precision	Recall
Albrecht Durer	0.82	0.65
Boris Kustodiev	0.57	0.71
Camille Corot	0.86	0.63
Camille Pissarro	0.51	0.44
Childe Hassam	0.52	0.59
Claude Monet	0.82	0.4
Edgar Degas	0.78	0.78
Eugene Boudin	0.93	0.67
Giovanni Battista Piranesi	0.75	1
Gustave Dore	0.68	0.84
Henri Matisse	0.63	0.87
Ilya Repin	0.69	0.49
Ivan Aivazovsky	0.89	0.87
Ivan Shishkin	0.62	0.78
John Singer Sargent	0.7	0.66

(b) CNN

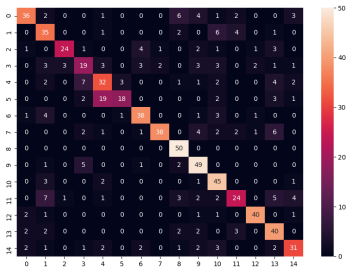
Figure 5: Precision and Recall by artist

From the results tabulated in figure 4, we immediately notice that our SVM has a severe overfitting issue. The training accuracy is almost 100%, while the test accuracy is much lower. This is in spite of tuning our regularization parameter C across a wide range of values, which indicates that other methods for addressing overfitting should be found and applied, such as using early stopping in the training. A possible source of the overfitting is the dimension of the feature vectors we use as input; using fewer principal components might reduce the ability of the svm to overfit.

We can see that compared to even the best SVM results, the CNN is superior. It achieves higher test accuracy with less noticeable overfitting.



(a) SVM



(b) CNN

Figure 6: Confusion Matrices (x-axis = predicted class, y-axis = true class)

Looking at Figure 6, we can see by the bright colors in the diagonal entries that most artists were classified relatively well. The CNN has some difficulties differentiating between works by Claude Monet (label 5) and works by

Childe Hassam (label 4); this is not surprising, as both artists were Impressionist painters using similar color palettes, and indeed the two were in contact throughout the years. We can see in Figure 7 that works by the two artists look very similar.

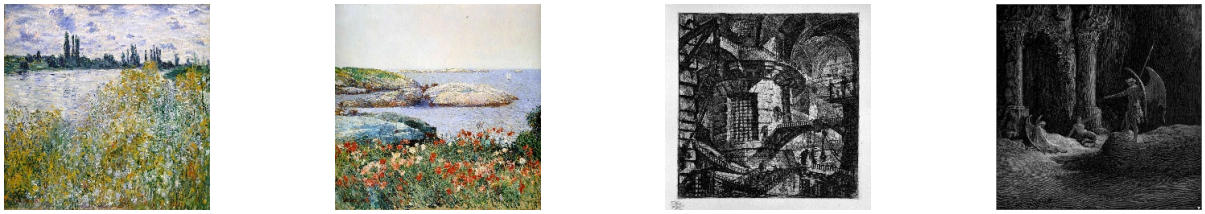


Figure 7: Sample paintings for Claude Monet (far left), Childe Hassam (left), Giovanni Battista Piranesi (right) and Gustave Dore (far right)

We can also see that some of the artists (Giovanni Battista Piranesi (label 8) and Gustave Dore (label 9)) have much better precision and recall than the others. This is likely because out of the entire dataset, these two artists prefer to paint mostly in black and white (as seen in Figure 7). This would allow features like the color histogram to distinguish them easily.

6.2 Training and Inference Time

The CNN was trained on a commercial Nvidia GTX 1070 GPU, while the SVM was trained on an Amazon c5.4xlarge EC2 instance with 16 vCPUs. Even though the CNN's training was accelerated by a GPU, it took on average 8 hours to train the full network. Compared to this, training the SVM took about 45 minutes total, almost all of which was spent extracting the features. The feature extraction would have likely taken even less time if the extraction procedures were accelerated by a GPU as well, since most of the techniques are highly parallelizable.

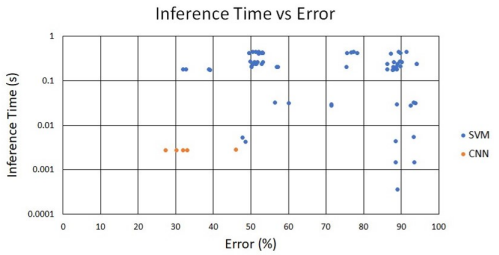


Figure 8: Plot of inference time vs. test error

To compare inference time fairly, inference for both models were performed on the EC2 instance. The results show that CNN took an order of magnitude less time to run inference while achieving higher accuracy. The bulk of the inference time for the SVM can also be accounted for by the feature extraction of the test data, however, so if feature extraction were accelerated the SVM might achieve similar time results. Judging from our current results, however, the CNN is superior in inference.

Each blue data point in Figure 8 is an instance of the SVM model with a different combination of feature vectors, and each orange data point is an instance of the CNN model with a different architecture. Comparing models across both test error and inference time, points closer to the origin are better, because we would like to minimize both.

6.3 Ease of Implementation

In terms of ease of implementation, the CNN was superior; all it required was setting up the initial architecture and feeding in the raw images. In contrast, implementing the SVM required a lot of overhead in researching potential features, correctly extracting these features from the images, and then tuning different combinations to be used with the model.

7 Conclusion and Future Work

We explored the task of artist identification using a dataset of 7462 paintings from 15 artists and compare the performance, training and inference time, and ease of implementation for both the classical method of feature extraction with an SVM classifier and the deep learning method of a CNN. Our best result came from the CNN, with an accuracy of 74.7% in comparison to our best SVM (using GIST features and Hu Moments) with an accuracy of 68.1%.

For future work, we would like to address the overfitting in our SVM models. Despite the tuning of the regularization parameter, our SVMs are overfitting heavily to the training data; this may be mitigated by other regularization techniques such as early stopping.

We would also like to investigate additional features such as Classemes [16] and PiCoDes [17]. From our work, we see that features often used for object recognition (such as Hu moments) were useful; other work [3] has shown Classemes and PiCoDes can be helpful in image classification tasks, so it may be worthwhile to explore this avenue for our own models.

Given our rather small dataset, we believe that transfer learning would have a lot of success on this task. Training our CNN with a large dataset such as the ImageNet dataset and then fine-tuning for artist classification with our smaller dataset can help our model gain knowledge about object recognition, which has already proven useful in features for our SVM.

Contributions

The initial dataset and implementations of the CNN [12] and the feature extraction for the CNN [18] were found by Jennie. Andrew worked to preprocess and organize the data; both of us worked together to modify the initial implementations to read in our data in batches in the desired formats. Andrew took the lead on iterating over various models to tune hyperparameters, while Jennie worked to explore the literature and research potential additional features for the SVM. The actual implementation of the found features was also done by both of us.

Code

Our code can be found at `www.github.com/jchen437/artist-classification`.

References

[1] A. Blessing and K. Wen. Using machine learning for identification of art paintings. Technical report, Stanford University, 2010.

[2] C. Liu and H. Jiang. Classification of traditional Chinese paintings based on supervised learning methods. *IEEE International Conference on Communications and Computing*, pp. 411-412. 2014.

[3] B. Saleh and A. Elgammal. Large-scale classification of fine-art paintings: Learning the right metric on the right feature. *arXiv preprint arXiv:1505:00855*, 2015.

[4] M. Misumi, H. Orii, T. Sharmin, K. Mishima, and T. Tsuruoka. Image classification for the painting style with SVM. *Proceedings of the 4th IIAE International Conference on Industrial Application Engineering.* 2016.

[5] T. Sahu, A. Tyagi, S. Kumar, and A. Mittal. Classification and aesthetic evaluation of paintings and artworks. *13th International Conference on Signal-Image Technology and Internet-Based Systems (SITIS).* 2017.

[6] N. Viswanathan. Artist identification with convolutional neural networks. Technical report, Stanford University, 2017.

[7] T. Balakrishnan, S. Rosston, and E. Tang. Using CNN to classify and understand artists from the Rijksmuseum. Technical report, Stanford University, 2017.

[8] J. Jou and S. Agrawal. Artist identification for Renaissance paintings. Technical report, 2011.

[9] Kaggle. Painter by Numbers. Retrieved from https://www.kaggle.com/c/painter-by-numbers. 2018.

[10] A. Oliva and A. Torralba. Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision* **42**(3):145-175, 2001.

[11] M.K. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory* **8**, pp. 179-187, 1962.

[12] N. Ilenic. Winning solution for the Painters by Numbers competition on Kaggle. Retrieved from http://github.com/inejc/painters. *GitHub repository.* GitHub. 2018.

[13] Pedregosa et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* **12**, pp. 2825-2830, 2011.

[14] F. Chollet. Keras. Retrieved from http://github.com/fchollet/keras. *GitHub repository.* GitHub. 2015.

[15] G. Bradski. The OpenCV library. Retrieved from http://github.com/opencv/opencv. *GitHub repository.* GitHub. 2000.

[16] L. Torresani, M. Szummer, and A. Fitzgibbon. Efficient object category recognition using Classemes. *European Conference on COmputer Vision*, 2010.

[17] A. Bergamo, L. Torresani, and A. Fitzgibbon. PiCoDes: Learning a compact code for novel-category recognition. *Neural Information Processing Systems (NIPS)*, 2011.

[18] G. Ilango. Using global feature descriptors and machine learning to perform image classification. Retrieved from http://github.com/gogul09/image-classification-python. *GitHub repository.* GitHub. 2017.

[19] A. Collette. HDF5 for Python. Retrieved from http://h5py.alfven.org. 2008.

[20] T.E. Oliphant. A guide to NumPy. USA: Trelgol Publishing. 2006.

[21] J.D. Hunter. Matplotlib: A 2D graphics environment. *Computing in Science and Engineering* **9**, pp. 90-95, 2007.

[22] S. van der Walt et al. Scikit-image: Image processing in Python. PeerJ 2:e453. Retrieved from http://dx.doi.org/10.7717/peerj.453. 2014.

[23] M. Abadi et al. Tensorflow: Large-scale machine learning on heterogeneous systems. Retrieved from http://www.tensorflow.org. 2015.

[24] M. Waskom. Seaborn. Retrieved from http://github.com/mwaskom/seaborn. *GitHub repository.* GitHub. 2014.

[25] W. McKinney. Data structures for statistical computing in Python. *Proceedings of the 9th Python in Science Conference*, pp. 51-56, 2010.

[26] Y. Tsuchiya. A python wrapper for Lear's GIST implementation working with NumPy. Retrieved from http://github.com/tuttieee/lear-gist-python. *GitHub repository.* GitHub. 2018.

[27] L.P. Coelho. Mahotas: Open source software for scriptable computer vision. *Journal of Open Research Software* **1**(1), 2013.

[28] R.S. Arora and A. Elgammal. Towards automated classification of fine-art painting style: A comparative study. *Proceedings of the International Conference on Pattern Recognition (ICPR)*, 2012.

[29] A. Lecoutre, B. Negrevergne, and F. Yger. Recognizing art style automatically in painting with deep learning. *Proceedings of Machine Learning Research* **77**, pp. 327-342, 2017.

[30] E.J. Crowley. Visual recognition in art using machine learning. PhD thesis. University of Oxford, Oxford, United Kingdom. 2016.

[31] Y. Hong and J. Kim. Art painting identification using convolutional neural network. *International Journal of Applied Engineering Research* **12**(4), pp. 532-539, 2017.

[32] S. Hicsonmez, N. Samet, F. Sener, and P. Duygulu. Draw: Deep networks for recognizing styles of artists who illustrate children's books. *Proceedings of the 2017 ACM on International Conference on Multimedia Retrieval*, pp. 338-346, 2017.