Classification and Reproduction of Famous Artists’ Work

Chih-Ching Chang, Junwei Su

Abstract

Art style gives rich information about the painting and it allows the grouping of works into related categories. Many factors could attribute to the style of an painting, including the choice of tools, the materials and medium used by the artists and so on. In this project, we want to study the main features that are affecting the art style of famous paintings from a computer vision point of view, and try to find out what we could do to make use of this art style information. We start by investigating the task of automatically classifying the underlying artists and art movements of paintings based on image characteristics. After that, we make use of features extracted from artworks to generate artistic style images.

Introduction

The research of visual perception of object and face recognition in computer vision area has huge progress these years. Several successful products about face or object recognition are widely used now because of the high accuracy to identify objects. However, compared with content of an image, there is no dramatic success about recognizing the style of an image due to the fact that “style” is a high level abstract concept and it’s hard to be defined. Moreover, there isn’t an exciting improvement on producing content with arbitrary kind of artistic style while humans or artists have mastered the skill. Therefore, we are interested in solving problems about style.

We break our projects into two parts about style. First, we tried to solve the problem of recognizing style of paintings from several artists. Second, we tried to utilize deep learning algorithm to solve the problem of generating a plausible stylish image.

Related Work

The analysis of paintings is one of the most challenging task in computer vision due to the complex nature of paintings. A comprehensive survey of the painting classification is given by Lombard [14], where he explored the key relationship between painting styles and discussed the capacity of different machine learning techniques in identifying painting styles. A more recent study was done by Karayev et al. [17]. In this paper, the author used Deep Convolutional Neural Network for feature extraction and achieved a fairly great performance. In the meantime, they also compared the different feature extraction methods such as GIST, Lab Histogram, Graph-based visual saliency and Meta-class binary features.

Convolutional Neural Networks is a type of artificial neural networks where it consists of layers of small computational units that process visual information hierarchically in a feed-forward manner. Recent studies have shown that this machine learning technique can produce promising results in image and video recognition applications [10,15,16].

In addition to object recognition, Gatys et al. [3] had demonstrated that the feature of style and content of an image are separable in convolutional neural network (CNN). They utilized the CNN model trained by Simonyan et al. to extract the features of the input images. Basically, given images A and B, their algorithm was able to produce image C which matches the content of A and the style of B.

Another popular implementations of this neural style algorithm can be found in Justin Johnson's open-source Torch implementation [6]. The program was written in Lua, and it achieves similar results to the paper. Meanwhile it also allows users to blend multiple styles into one image.

Approach

Part 1: Style Classification

We want to classify the paintings by artists based on image styles. In order to finish this task, we have to first extract the image features, and then apply machine learning algorithms to conduct the classification. In the following, we will discuss these two steps in detail.

Feature Extraction

Based on previous studies on classifying images for different tasks, a large number of features extraction techniques are proposed. Most of these techniques involve acquiring the color, texture, light and line information of the image. Here are the features we experimented for our application:

Color: Color based features can be subdivided into two categories, namely, global color features and local color features. Global color features referred to averaging per pixel feature values over the entire image plane, while local color features means the average of pixel feature values within a segmented image region.

Texture: In general, texture feature are extracted by computing the pixel statistics or finding the local pixel structures in original image domain. Some existing texture feature extraction methods include but not limited to Gabor filter, energy filters and edgeness factor, and wavelet transform. However, because our dataset has relatively low resolution, the texture features did not work very well.

Saliency map: Saliency map is a gray-scale image that offers a measure of the most important regions of a given scene. We utilize the "simpsal" matlab package devised by Jonathan Hare [12].

Edges: Edge information could be useful when it comes to recognizing images from the cubism class, where the images are very distinctly defined by the layout of lines and quality of edges. Hough transform were used to capture the different styles of lines in different class of paintings.

SIFT: We learned from the class that SIFT is mainly used for content recognition, so it could be useful in detecting repeated image content. The SIFT feature extraction were implemented based on [11].

After experimenting with different combination of the feature extraction techniques mentioned above, we decided to choose the color, saliency map and edge as our feature attributes.

Machine Learning Methods

In our tasks, we have multiple classes with unequal number of samples, so accuracy is not a good measure in such case. As a result, we used F-1 score along with 5 fold cross validation in our experiments to compare different classifiers. Five different machine learning methods were selected: random forest, naive bayes, logistic regression, decision tree and 5 nearest neighbour.

Part 2: Generate Artistic Style Images

In this part, we will utilize the convolutional neural network model to transform an image to artistic style. The following is the algorithm to generate a stylish image:

```
- Generate(style_img, content_img, canvas_img, epoch):
- FOR EACH epoch DO
    a. Input three Images into the CNN model
    b. Get feature maps generated from different layers of CNN model
    c. Compute loss for both of:
       i. style feature maps and canvas feature maps
       ii. content feature maps and canvas feature maps
    d. Do backpropagation to get the gradient of loss
    e. Use gradient to update the canvas image
```

The CNN model is used to extract the feature maps of the input images. There are several layers in CNN model. Each layer will extract different features from the input images. The first few layers will extract low level features, such as edge; the latter layers will extract high level features, such as an edge is actually a corner or a boundary. Since style is an abstract high level concept, we extract feature maps from all 5 layers. However, we only extract feature maps from one of the last few layers for the content image.

After extracting the feature maps, we need to compute the loss of them. We used mean square error to compute it. However, there is difference between the calculation of content loss and style loss.

For content image, we use a normal way to compute the loss. Let X be the feature map of canvas image and C be the feature map of content image. Let $M \cdot N$ be the size of the feature map and n be the number of filters in each layer of CNN model. The content loss will be the following:

$$Loss_{content} = \frac{(X - C)^2}{(M \cdot N \cdot n)} \tag{1}$$

For style image, since style should not be dependent on its position in the image, we would like to use gram matrix to calculate the loss. Let X be the feature map of canvas image and A be the feature map of style image. Define the gram matrix for X be $G_X = X \cdot X^T$ and the gram matrix for A be $G_A = A \cdot A^T$. Again, let $M \cdot N$ be the size of the feature map and n be the number of filters in each layer of CNN model. The number i denotes which layer we are looking at. The number I denotes the total number of layers we use. The loss equation is the following:

$$Loss_{style} = \frac{1}{I} \sum_i \frac{(G_{X_i} - G_{A_i})^2}{(M_i^2 \cdot N_i^2 \cdot 4 \cdot n_i^2)} \tag{2}$$

We then compute the total loss as the following:

$$Loss_{total} = \alpha \cdot Loss_{content} + Loss_{style} \tag{3}$$

where α is the ratio that will be adjusted to balance the loss of content and the loss of style.

After computing the loss, we use standard back-propagation to compute the gradient of the loss, and then update the canvas image by the gradient. The following formula is for updating the canvas image. Let X be the canvas image.

$$X \leftarrow X + \frac{\partial Loss_{total}}{\partial X} \tag{4}$$

After we get a plausible result image, we apply color histogram mapping to adjust the biased color of the result image. We use the standard way to get the color histogram of style image and result image, and then do color matching for them [8,9]. Finally, transform the color of the result image based on the color of the style image.

Experiment

Part 1: Style Classification

The data was acquired from the website WikiArt [19]. This website contains paintings sort by artists and art movements [17]. A total of 7 artists from 4 major art movements were selected. The table below shows the detail of our data.

Table 1: Details of image data

Art Movements	Artists	Number of paintings
Impressionism	Monet	108
Post-impressionism	Van Gogh	134
	Gauguin	136
Cubism	Braque	113

	Gris	117
Renaissance	Raphael	67
	Titian	92

After that, we implemented the 5 machine learning algorithms mentioned above to distinguish paintings between different artists. The results are shown in the Table 2.

Table 2: Results of seven way classification

Classifier	**F1-Score**
Random Forests	0.54 (+/- 0.02)
Naive Bayes	0.47 (+/- 0.02)
Logistic Regression	0.42 (+/- 0.02)
Decision Tree	0.40 (+/- 0.01)
6-NN	0.35 (+/- 0.01)

We noticed that the random forest gave us the best accuracy among all the classifier. Based on this, we also conduct a two-way classification using random forest between paintings in different art movements. The results are shown below:

Table 3: Two-way classification using random forests

Art Movement	**Artists**	**F1-Score**
Renaissance vs. Renaissance	Raphael - Titian	0.69 (+/- 0.03)
Post-impressionism vs. Post-impressionism	Vangogh - Gauguin	0.75 (+/- 0.05)
Post-impressionism vs. Impressionism	Vangogh - Monet	0.83 (+/- 0.02)
Post-impressionism vs. Cubism	Vangogh - Gris	0.90 (+/- 0.01)
Post-impressionism vs. Renaissance	Vangogh - Titian	0.92 (+/- 0.02)

Part 2: Generate Artistic Style Images

We did several experiments by varying different parameters, modifying equations, or changing part of the implementation. The following are the results and evaluations for each experiment.

Different initial image

We figure out for some of the input images, the result is not good even if we run more epoch. Therefore, we want to try to use different initial canvas image. Fig. 1 is the result of changing initial canvas image. There are two kinds of content image, one is lake mendota and the other is an owl. We figured out that the result of owl picture is bad when we use noise image as the initial image but is much better when we use content image as the initial image. However, the lake mendota image has better result for noise image. In the case of setting style image as the initial image, the results are bad that keeps the location information of style image, which requires to be removed.

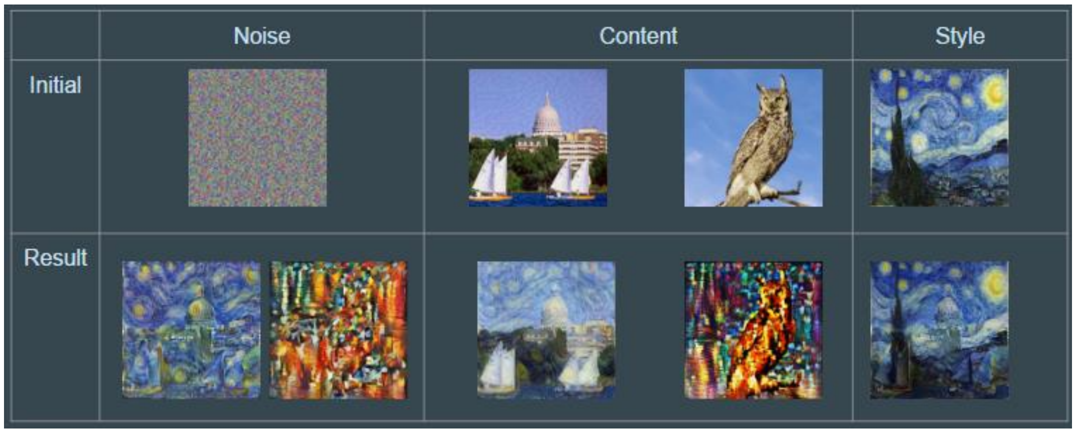

Fig 1. Result with different initial images.

Different loss function and weight ratio

In order to improve the result, we try to use different formula to compute the style loss and content loss. We still used mean square error to compute the loss but changed their parameters. In the original implementation, the style loss for each layer is divided by the square of the size of the feature maps. We compared it with a new loss function as the following:

$$Loss_{style} = \frac{1}{I} \sum_i \frac{(G_{X_i} - G_{A_i})^2}{(M_i \cdot N_i \cdot 4 \cdot n_i)} \tag{5}$$

From the formula itself, it is obvious that the loss will become larger since we use smaller values in the denominator, but the value of the loss is not a good standard to evaluate the result. Nevertheless, when we run the same number of epoch, the result image has worse performance that it is biased to style image (Fig. 2).

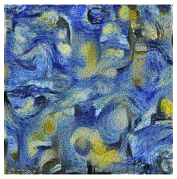

Fig 2. 1000 epoch with the new loss formula

We also change the value of α in equation (3). We use different values: 0.001, 0.005, and 0.01 to adjust the weight between content loss and style loss. In our hypothesis, we expected when the value is smaller, the image will be close to content image and when the value gets larger, the result image will look more stylish. However, in figure 3, the results do not look to have a positive correlation between the ratio α and the degree of style.

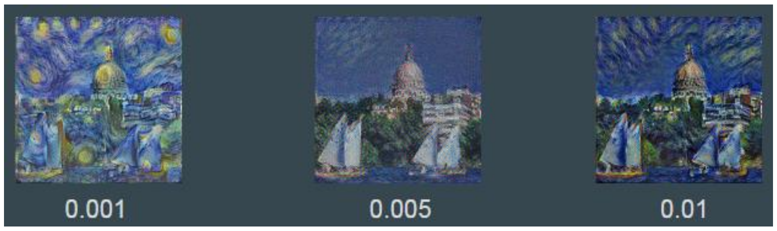

Fig 3. The result with different ratio α.

Different optimizer / learning rate

In order to increase the “time performance” of the result, we tried different parameters and optimization methods in our approach. Since it is hard to define the “plausible final result”, our time performance here actually denotes how well the result image is with the same number of epochs. We tried two different optimization methods built in chainer [4]: Adam and standard stochastic gradient descent. From the results, Adam has better performance.

We also tried different learning rate for the optimization. Although it is obvious that loss converges faster with larger learning rate, we would like to check whether we can get the global optimal result with slightly smaller learning rate. The number we tried are 4.0, 3.0 and 2.0. Unfortunately, we cannot see the effect of “global optimal” but only concludes that when learning rate equals 4.0, the result looks plausible when all of them have the same number of epochs.

Adjust the color

When doing the original approach, we figured out that the color of the result image is dimmer. The possible reason might be that the author modified the CNN model to get the better result in the original paper. However, we would like to treat the CNN model as a black box and not modify it.

We tried two different ways to adjust the color. For the first method, we tried to normalize the style image before using it as an input of the CNN model and denormalized it after generating the result image. We expected to have a result with better color with this approach. However, the loss diverges after we normalize the style image and therefore we discarded the result for it. For the second method, we tried to do post-editing of the result image. We chose to use color histogram mapping as stated in the approach section.

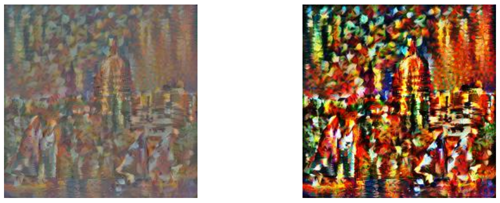

Fig 4. The result before and after color histogram mapping.

Conclusion

Based on the results of painting classification, we demonstrate that the extracted features such as color and edge are able to classify paintings from seven different artists from four art movements. A maximum accuracy of 54% is reached by using random forests algorithm. Further analysis show that paintings within the same move art movement tend to have a lower classification accuracy compared to paintings within different art movement. This indicates artistic styles of painting can be generally determined by its belongs to a certain art movements. However, we can also notice that paintings within different art movements can have implicit influence between one and other. For example, because post-impressionism derived from impressionism, we have a relatively low accuracy in distinguishing paintings from these two art movements. Many current research works suggest deep learning can have a better performance when it comes to image classification, so our future work will be focusing more on this part such as implementing convolutional neural network.

For the part of generating an artistic image, the algorithm for it is highly dependent on input images. In order to get a better result, one might need to use similar content and style image (to some degree). For instance, if the color in style image and the color in content image is nearly complementary, then one would expect to get a worse result. Moreover, complicated input images might also lead to worse result. It is better to try different parameters to get the best result.

For the future work, we would like to try GPU programming to increase the time performance for each epoch. (Our code only used CPU because of lacking access to a workable GPU.) Furthermore, we figured that even if we use color mapping to adjust the result image, there are small patches with awkward color, which did not show in the original code implemented by Johnson [6]. We've tried to use some denoising technique. It removed the strange patches, but also made the result image losing the texture of the style image. Therefore, we would like to try different method to get a better detail of the image. Last, after listening to the presentation of other groups, it is worth trying to build a CNN model with less layers but similar result of feature extraction as VGG-19 model. Neural network with less layers will hugely improve the runtime.

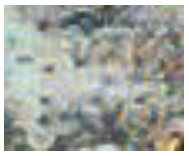

Fig 5. There are weird yellow contours in the image.

Contribution

Chih-Ching Chang: 1. Literature review, implementation and write-up for "generating artistic style image". 2. Project web page. 3. Project github framework and maintenance.
Junwei Su: 1. Literature review, implementation and write-up for "style classification".

References

1. Nikulin, Y., and Novak, R. 2016. Exploring the neural algorithm of artistic style. CoRR abs/1602.07188.
2. Kathryn Siegel. Picasso's Marilyn Monroe and Other Blends: Neural Style in TensorFlow.
3. L. A. Gatys, A. S. Ecker, and M. Bethge. A neural algorithm of artistic style. CoRR, abs/1508.06576, 2015
4. Chainer: http://chainer.org/
5. Berkeley Vision and Learning Center: https://github.com/BVLC/caffe/wiki/Model-Zoo
6. https://github.com/jcjohnson/neural-style
7. https://github.com/mattya/chainer-gogh
8. Color Mapping: https://en.wikipedia.org/wiki/Color_mapping
9. https://github.com/stefanv
10. K. Simonyan, A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556
11. Jianxiong Xiao, James Hays, Krista A. Ehinger, Aude Oliva, and Antonio Torralba. SUN Database: Large-scale Scene Recognition from Abbey to Zoo. IEEE, 2010
12. J. Harel, A Saliency Implementation in MATLAB: http://www.klab.caltech.edu/~harel/share/gbvs.php
13. Ying Wang and Masahiro Takatsuka. A Framework towards Quantified Artistic Influences Analysis. IEEE, 2012
14. Lombardi, T. E. (2005) The classification of style in fine-art painting, Ph.D. thesis, Citeseer.
15. Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Advances in neural information processing systems, 1097–1105
16. Pierre Sermanet, David Eigen, Xiang Zhang, Michael Mathieu, Rob Fergus, Yann LeCun (2014). OverFeat: Integrated Recognition, Localization and Detection using Convolutional Networks. In ICLR, 2014.
17. S. Karayev, M. Trentacoste, H. Han, A. Agarwala, T. Darrell, A. Hertzmann, and H. Winnemoeller. Recognizing image style. arXiv preprint 1311.3715, 2013
18. Ying Wang and Masahiro Takatsuka. SOM based Artistic Style Visualization. IEEE International Conference on Multimedia & Expo (ICME'13). San Jose. USA, July, 2013.
19. WikiArt: http://www.wikiart.org