CS 229 Final Report: Artistic Style Transfer for Face Portraits

Daniel Hsu, Marcus Pan, Chen Zhu
{dwhsu, mpanj, chen0908}@stanford.edu
Dec 16, 2016

1 Introduction

The goal of our project is to learn the *content* and *style* representations of face portraits, and then to combine them to produce new pictures. The content features of a face are the features that identify a face, such as the outline shape. The stylistic features are the artistic characteristics of a certain portrait or painting, such as brush strokes, or background color. We forward-pass a content image, and several style images through a CNN to extract the desired content and style features. Then we initialize a white noise image, and perform gradient descent on its pixels until it matches the desired style and content features.

2 Related Work

In a recent paper, Gatys et. al. [1] performed style transfer using CNN. His algorithm combined one content with one style image well, but was computationally expensive, as it required a gradient descent loop for every image generated through a pre-trained CNN. Johnson et. al [2] developed an algorithm to perform style transfer using only one forward-pass through a CNN. This approach involved training a separate style-transfer network for each desired style. Dumoulin et. al [3] then extended this approach by generalizing the style transfer network to work with multiple styles.

In view of our limited time and computational resources, we decided to base our work on Gatys' algorithm, but with modifications to work with faces specifically. We used ideas from the eigenface algorithm [4] to generate average style representations

3 Model and Platform

We use a pre-trained CNN for face classification, VGG-Face [5]. This CNN was very deep (27 layers) which is useful for extracting content features, as described below. For style input, we pulled several images from various art sources such as Picasso, Van Gough and anime from the Internet.

As for our platform, we chose MatConvNet, a MATLAB toolbox implementing convolutional neural networks. It is efficient, simple and supports many pre-trained network models. It also provides GPU acceleration functions. We noticed that there were existing implementations of artistic style transfer algorithm based on Torch, Caffe and other deep learning toolboxes. But no one has implemented style transfer using MatConvNet. We hope our project can be a supplement to existing implementations.

4 Gradient Descent Loss Functions

We start with a white noise image, x, and gradient descent on its pixels until it jointly minimizes several loss functions:

$$L_{total} = aL_{content} + bL_{style} + cL_{regularization}$$
$$x := x - \nabla_x L_{total}$$

The loss functions are described below:

Content Loss

As the content image progresses down into deeper and deeper layers of a CNN, it gets down-sampled into smaller and smaller images. This down-sampling essentially captures the details of a section of the image, and generates a summary of that section into a new down-sampled image. So at the deepest levels of the CNN, we capture the most significant features of the image, or its content.

The content loss function for content is just the sum squared difference of activations between the input white noise image, X, and the input content image, P at a certain desired deep layer l:

$$L = \frac{1}{2}\sum_{i,j,k}(X^{(l)}_{ijk} - P^{(l)}_{ijk})^2$$
$$\frac{dL}{dX_{l,ijk}} = \begin{cases} (X^{(l)} - P^{(l)})_{ijk}, P^{(l)})_{ijk} > 0 \\ 0 \text{ otherwise} \end{cases}$$

We can then back propagate the derivative to the original input image.

Figure 1 shows the generated content representation from different deep layers of the network. The representation from the deeper layer is more desirable for our purposes, as it preserves the general structure and lines, but does not enforce a strict similarity with the original image.

(a) Layer 13

(b) Layer 27

Figure 1: Content representations from different layers

Style Loss

In order to compare style, we first transform the output of a layer l in the CNN (3D) into a 2D matrix, F^l. Each row in F represents one filter output of the layer in the CNN. From F, we form a gram matrix G:

$$G^l_{ij} = \sum_k F^l_{ij} F^l_{jk}$$

G_{ij} is the dot product of row i and row j in F. This measures the correlations between layers, but does not capture the global arrangement. We compare the Gram matrices of 2 images to calculate the error between their styles. We define G as the Gram matrix of an input white noise image, passed through the CNN.
We define A as the Gram matrix of a desired style image, passed through the CNN.
The loss from a desired layer l is defined as

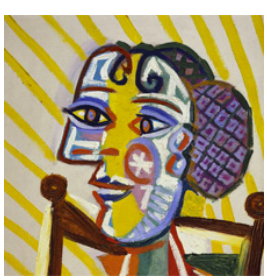
$$L = \frac{1}{2} \sum_{i,j} (G^{(l)}_{ij} - A^{(l)}_{ij})^2$$

From this loss L, we can calculate the gradient of the loss with respect to $F^{(l)}$, a layer in the CNN:

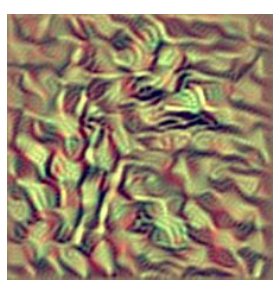
$$\frac{\partial L}{\partial F^{(l)}_{ij}} = \begin{cases} ((F^{(l)})^T (G^{(l)} - A^{(l)}_{ji} \text{ if } F^{(l)}_{ij} > 0 \\ 0 \text{ otherwise} \end{cases}$$

Then we use standard back propagation to calculate the loss derivative with respect to the original input style image.

The results are shown in figure 2. The style representations capture the color and aesthetics of the image, but not its content.



(a) Original Image



(b) Style Representation

Figure 2: Style Representations

Regularization Loss

In order to prevent the algorithm by overfitting the pixels by making them out of the bounds: (0, 255), we define the following per-pixel loss functions as described by Mahendran et. al [6]:

$$L(x_{i,j})_{size} = x^2_{i,j}$$
$$L(x_{i,j})_{var} = (x_{i,j} - x_{i+1,j})^2 + (x_{i,j} - x_{i,j+1})^2$$

Figure 3 shows the effects of regularization. The picture without regularization is noisy, because many of its pixels are out of range and have been clipped to (0,255).

(a) With reg.

(b) Without reg.

Figure 3: Regularization

5 Additional Methods

Multiple Style Transfer

We developed the following algorithms to transfer multiple styles onto a content image:

1. Mixed Gradient Descent
A very direct method to combine multiple styles is to gradient descent on the their collective errors. In this approach, we first note that in our gradient descent, part of the error term comes from the style loss. The style loss corresponds to how much our generated image currently differs from the activated style image we are trying to match. So then to expand this method to multiple images, we simple define our style loss as the sum of the style losses, each corresponding to a single style image. With this approach we do obtain a more accurate total style loss for gradient descent that takes into account multiple style images of a particular artist, but at the same time we increase the running time with each additional style image we include.

2. Eigengram
Taking ideas from the eigenface algorithm, we generate an *eigengram* matrix, the principal eigenvector of all the gram matrices as follows:

Let g_i be the vectorized gram matrix of style image i.
Define G = $\begin{bmatrix} g_1 & \cdots & g_N \end{bmatrix}$.

Calculate u, the principal eigenvector of GG^T.
We regard u as the average style image, and perform gradient descent on it only.

Figure 4 compares the results of the above methods. The results are similar, but method 2 is much faster to run.

(a) Style Image 1 (b) Style Image 2

(c) Avg Img (eigengram) (d) Avg Img (mixed g.d.)

Figure 4: Comparison of Average Style

Gradient Descent

We found standard gradient descent update rules such as momentum, or ADAM to be relatively slow to converge. Newton's method, which uses gradient and inverse Hessian matrix to steer search directions, gives higher convergence speed. However, computing a Hessian can be both time and space consuming. Some functions do not have second derivatives. L-BFGS is a good alternative to gradient descent and Newton's method. Unlike Newton's method, L-BFGS uses an approximation of the inverse of Hessian matrix. To do this, it stores last m updates of x and its gradient $\nabla(f(x))$, which represent the approximation implicitly. This method can also be used when problems are large and memory resource is limited. The way it approximates the product of inverse Hessian and gradient matrix is described as follows.

At iteration k,
$q = g_k$;
for *i=k-1, k-2,..., k-m* **do**
  $\alpha_i = \rho_i s_i^T q$;
  $q = q - \alpha_i y_i$;
end
$(H_k^0)^{-1} = s_{k-1}^T y_{k-1} / y_{k-1}^T y_{k-1}$;
$z = (H_k^0)^{-1} q$;
for *i = k-m, k-m+1,..., k-1* **do**
  $\beta_i = \rho + y_i^T z$;
  $z = z + s_i(\alpha_i - \beta_i)$;
end
$(H_k)^{-1} g_k = z$

Algorithm 1: L-BFGS update

where $g_k = \nabla f(x)$,$s_k = x_{k+1} - x_k$,$y_k = g_{k+1} - g_k$, and $\rho_k = 1/(y_k^T s_k)$.

Once we have the approximation of $(H_k)^{-1} g_k$, we can update our input x as follows:

$$x_{k+1} = x_k - \lambda (H_k)^{-1} g_k$$

where λ is the learning rate

A comparison of the convergence rate of ADAM and L-FBGS can be seen in Figure 5.

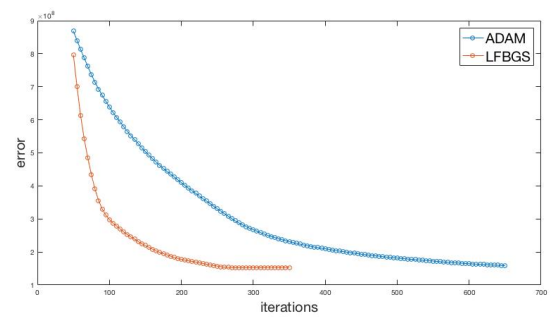


Figure 5: Comparison of optimization methods

6 Results

Some pretty visualizations of our results are in the appendix at the end.

To quantify the performance of our algorithm, we pass our generated image through the face classification network, and compare the classification score with that of the original content image. The results are below:

Image	Class. Score	Correct Class.
Original	0.961	Yes
Warhol Style	0.093	Yes
Anime Style	0.804	Yes
Van Gough Style	0.852	Yes
Picasso Style	0.697	Yes

All the style images were still classified correctly, even though their score was lower than the original.

Quantifying the performance is rather subjective, but we showed that our generated image still resembled the original image well for most styles.

7 Discussion

As we have seen, this project has demonstrated that it is possible to convert any portrait to a desired style. We started with the equations which we used to extract the content and style from the portrait and style images, and from there we wrote code to back propagate the error to the original image. Gradient descent allowed us to generate an image that as closely matched both content and style as we wanted. Furthermore, we added the ability to not only base the error on style and content differences, but also handled overfitting through regularization using per-pixel loss functions. Then given that the whole calculation was taking quite a while for the solution to converge, we implemented L-BFGS since it was able to beat out gradient descent and Newton's method in rate of convergence. The last feature we added was the ability to combine multiple style images together. Previous work only took one style image, and transformed a portrait to that image's style. However in order to more accurately represent an artist's style, we needed to use more than a single sample of the artist's work. This led us to developing mixed gradient descent and eigengram multiple style transfer techniques to accomplish a more holistic artistic style transfer. The collective set of these new features allowed us to explore artistic style transfer more deeply than before.

8 Future Work

Potential future projects include different style transfers on a single image, such as transferring the face to Picasso, while changing the background to Warhol. One approach to accomplish this would involve separating a portrait image into background and facial regions, and running our gradient descent algorithm on those two regions separately. The combination of the two would result in an image of two separate styles on separate regions. We would also like to explore real time style transfer. Having a real time algorithm would open up a whole new set of possibilities, such as transforming a live action film in real time to anime, or even converting a virtual reality enhanced display of the environment to Picasso's artistic style.

References

[1] L. A. Gatys, A. S. Ecker, and M. Bethge. Image style transfer using convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Jun 2016.

[2] Justin Johnson, Alexandre Alahi, and Fei-Fei Li. Perceptual losses for real-time style transfer and super-resolution. *CoRR*, abs/1603.08155, 2016.

[3] Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016.

[4] Matthew Turk and Alex Pentland. Eigenfaces for recognition. *J. Cognitive Neuroscience*, 3(1):71–86, January 1991.

[5] O. M. Parkhi, A. Vedaldi, and A. Zisserman. Deep face recognition. In *British Machine Vision Conference*, 2015.

[6] Aravindh Mahendran and Andrea Vedaldi. Understanding deep image representations by inverting them. *CoRR*, abs/1412.0035, 2014.

Appendix: Visualizations

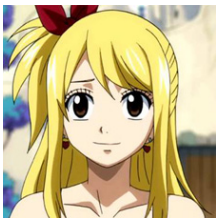
Original Content Image: Chow Yun Fat

(a) Original

(b) Content Rep.

Anime Style:



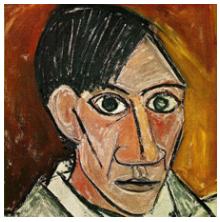




Andy Warhol Style:





Picasso Style:





Van Gough Style: