CIS 519 Final Project - Photo Paint

Lucy Kuo YUJUKO@SEAS.UPENN.EDU
Lingyi You LYYOU@SEAS.UPENN.EDU
Keqin Zhou KEQINZH@SEAS.UPENN.EDU

Abstract

In this project, we study Neural-Style algorithm developed by Leon A. Gatys, Alexander S. Ecker and Matthias Bethge to reproduce images with new artistic styles. We train a model for a set of paintings by a chosen artist. The algorithm takes two inputs, a content image and a set of style images, and changes the input to resemble the content of the content image and the artistic style of the style images.

1. Introduction

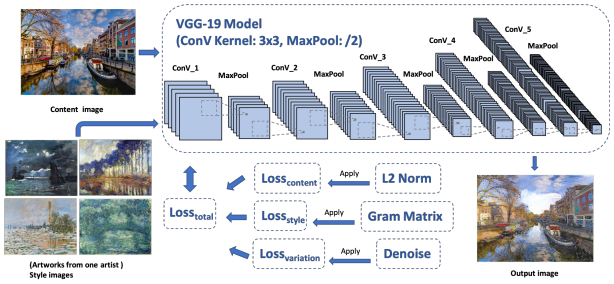


Figure 1. Major Component of Project

Motivated by photo editing apps, our goal is to modify the input photograph style and generate a new picture to resemble an artist's style and develop a machine learning algorithm that maximizes the performance of the style transfer. There are 50 artists (e.g. Pablo Picasso, Vincent van Gogh) in Kaggle dataset, we will pick all artworks from each of them and reproduce the input image with a new artistic style based on the chosen artworks. It is not simply the transfer of color or element, but mimics the way on how the artist would express his/her vision if he/she paints. This application is essential with the increasing aesthetic needs of people and triggers their interest in fine art.

University of Pennsylvania, CIS 419/519 Course Project.

2. Related Work

In the paper "Image Style Transfer Using Convolutional Neural Networks"(Gatys, 2016), Leon A. Gatys, Alexander S. Ecker and Matthias Bethge developed Neural-Style algorithm. They use the high-level features extracted by a trained classifier VGG to represent the content of an image, and use Gram matrices to represent the style. After an iterative optimization process, they reproduced an origin image into an stylized image.

From some preliminary research, a lot of work has been done based on the Neural-Style algorithm (Gatys, 2016). In the paper "Deep Photo Style Transfer" (Fujun L, 2017), the authors solved the problem of style texture's disruption by preventing spatial distortion, constraining the transfer operation to happen only in color space and using a photorealism regularization term in the objective function during optimization.

While there were lots of researches on image style transfer, most of them take in two images, one content image and one style image, and output one generated image. In this project, we will take in around 50 style images from an artist and reproduce the content image based on all of them.

3. Data and Features

We get style images from Kaggle and use some photos as real data input. After imported content and style image, they are transferred into torch sensors, and the values are converted from 0-255 to 0-1. The images are resized to have the same dimension and the max dimension is defined.

4. Approach

Since Neural style transfer can be implemented using any pre-trained convnet, we manually initialize pre-trained vgg19 convolutional neural network just for clarity.

After setting up a network that will compute layer activations for the style image, the content image, and the generated image, we need to define the loss function. Neural style transfer consists in applying the "style" of a style image to a content image, while conserving the "content" of

the content image, so we know what we want to achieve: conserve the "content" of the content image, while adopting the "style" of the style image. Therefore, we can define the loss function as :

content loss = difference(content image, generated image)

style loss = difference(style image, generated image)

4.1. Content Loss

Different layers of convolutional neural network decompose the content of an image over different spatial scales, so we want the content of an image to be presented by the results of a top layer of network. We use L2 norm to calculate the difference of content to ensure that the generated image preserves the content of the original image.

4.2. Style Loss

As Gatys's paper (Gatys, 2016) defined : "We aim at capturing the appearance of the style reference image at all spatial scales extracted by the convnet, not just any single scale", we calculate the style loss in multiple layers of convolutional neural network. In order to catch the patterns, Gatys's paper(2016) introduced "Gram matrix":

Gram matrix = matmul(transpose(Feature), Feature)

It's simply using the inner product of feature maps to calculate the correlations between layers. We adopt the method for style loss to ensure that the results of different layers preserve the patterns in the style image.

4.3. Variation Loss

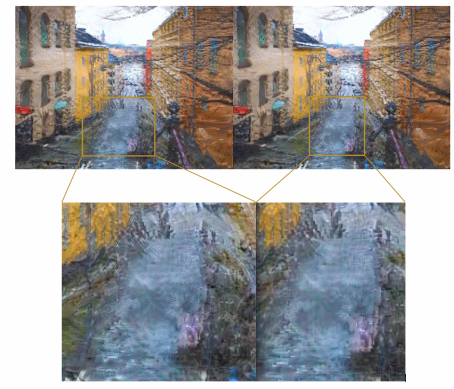


Figure 2. Variation Loss Comparison for Weights of 0 and 10

In order to remove the rough texture of the image and denoise the generated disrupting pixels, we add in total variation loss. By summing up the absolute differences for neighboring pixel-values in the input images horizontally and vertically, we can measure how much noise is in the images.

As shown in Figure 2, the left one has no variation loss while the right one has variation loss with weight of 10. Zooming in and comparing them, without denoising, dark color dots appears discretely on the blue background while after denoising, the dots disappears and pixels become continuously.

4.4. Gradient Descent

By setting weights to content loss, style loss and variation loss, we are able to adjust how artistic we want the generated image to be. Therefore, the total loss becomes:

total loss= (content loss $\times$ *content weight) + (style loss* $\times$ *style weight)+ (variation loss* $\times$ *variation weight)*

Finally, we set up the gradient descent process to minimize this loss function. At the first time, we picked Adam algorithm to optimize the model. The generated image was not as good as we expected, so we tried L-BFGS algorithm and got pretty good results.

4.5. Multiple Style Images

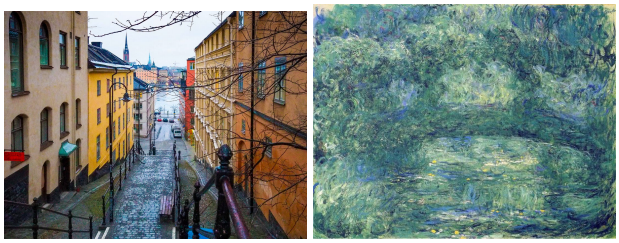


Figure 3. Content and Style Image

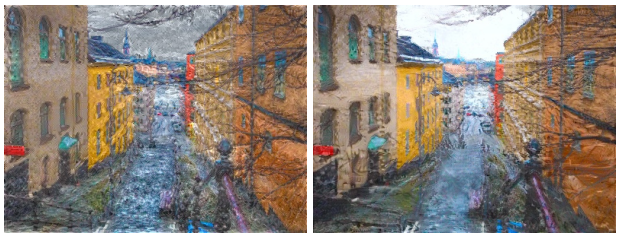


Figure 4. One vs. Multiple (50) Style Image Result Comparison

We use 50 per artist for the style images instead of only one as described in Gatys's paper (Gatys, 2016). With one image, even after applying color preserving, the result is not desirable as the left one shown in Figure 4. The water lilies shape is still clear in the stylized image. This is the "content" of the style image, not the style we defines. With 50 style images as input, the right one eliminates the undesirable effect and achieves better visual experience.

In our project proposal, we mentioned that there are two

candidate approaches for multiple style images. The first approach is to gather all artist's artworks and train a model, while the second one is to train different models for each style image and combine the generated images. After trying both ways, we realized that training multiple models does not result in a better performance, but only consumes more time and storage than simply training one model with all style images at the same time. Therefore, we choose the first approach to put all the artist's style images in a set, add up each of their loss times their weight, and use gradient descent to minimize the total loss.

4.6. Color Preserving

Figure 6. without vs. with Color Preserving

We use luminance-only transfer to preserve the original color of the content image to avoid the possibility of altering the appearance of the scene or bringing in new colors. In Figure 6, the left one introduces the greenish color from the Monet's paintings while the right one eliminates the noise color by applying original color preserving.

As we only want to transfer the luminance, we need to extract the luminance channels from the style and content images, which we can get from the Y channel in our YUV color space. After that, we can get an output luminance image by applying neural style transfer algorithm on the extracted channels. To preserve the color, we combine the color information of the content image(channels U and V) with the luminance image to get a final color preserved output image.

5. Results and Conclusions

5.1. Performance Evaluation and Improvement

As mentioned in Gradient Descent, weighted loss is minimized and used as a way to evaluate the performance of a model. However, the weighted loss depends on the weights of content, style and total variation. Acknowledging the

Figure 5. Inputs and Style Transfer Results

facts that there is no arbitrary and perfect trade-off between content, style and total variation, and appreciating art is a visual experience for people, we tune and choose weights based on the visual effect of the output image, which fulfills the style transfer while minimizing the noise.

Two network combinations are explored with chosen loss weights. Their losses on training different set of images are recorded in Table 1 and 2. To minimize the total loss and iteration number for convergence, network 2 is chosen for our algorithm.

Table 1. Performance of Network 1 on Three Artistic Styles

NAME	TOTAL LOSS	ITER. NUMBER
CLAUDE MONET	(102832.9±1.2)E+7	902.6± 46.2
VINCENT VAN GOGH	(224662.9±0.9)E+7	675.4± 18.2
PABLO PICASSO	(349618.8±3.4)E+7	788.0± 36.6

Table 2. Performance of Network 2 on Three Artistic Styles

NAME	TOTAL LOSS	ITER. NUMBER
CLAUDE MONET	(755213.4±6.6)E+6	516.6± 43.2
VINCENT VAN GOGH	(128899.2±1.4)E+7	421.6± 35.6
PABLO PICASSO	(235110.2±1.8)E+7	477.6± 26.1

5.2. Discussion and Summary

With the optimized network and parameters (as defined in Appendix 6.3), we train and produce a set of paintings using different content images and artist's styles as shown in Figure 5. The result is fascinating and our goal is achieved.

In this project, we learn and develop the algorithm for style transfer machine learning algorithm which transfer a content image to a stylized one with a set of artist's paintings. Multiple style images, denoising loss, color preserving and neural network are designed to maximize the performance of model and visual effect of the outputs. This application attracts people's attention to fine arts and inspires them to create and enjoy their arts.

6. Appendix

We use PyTorch and Colab for this project. The architectures and naming convention are from git repository by (Smith, 2016).

6.1. Training Data Sets

We train our networks with both one style image input and multi style image input. The style images are artworks from Claude Monet, Vincent van Gogh and Pablo Picasso. We download 50 images per artist from Kaggle: Best Artworks of All time. We use a total of 4 sample content images. These are landscape images from Ocean Beach in San Francisco, Amsterdam in Netherland, Yosemite, and Stockholm.

6.2. One Style Image Input

We use VGG-19 as model weight and choose L-BFGS as optimizer. The max width and height for input images are 512x512. The image for initializing the networks is content image. The weight for content loss is 5e0 and weight for style loss is 1e4. We train one style image input model without total variation loss. We use 'yuv' for color converting. The learning rate is 1 with max 4000 iterations, but this model cannot converge after 4000 iterations.

Network for one style image input (Network 1):

content layer: ['conv4_2'], weight: [1.0]

style layers: ['relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu5_1'], weight: [0.2, 0.2, 0.2, 0.2, 0.2]

6.3. Multi Style Image Inputs

For multi style image inputs, we use 50 images per artist for training. Most parameters for multi style images input model are the same as one style image input. All different parameters are listed as follows. The weight for content loss is 5e0 and weight for style loss is 1e7. The learning rate is 1e-1 and max iteration number is 1000.

Network for multi style image inputs (Network 2):

content layer: ['relu4_2'], weight: [1.0]

style layers: ['relu1_2', 'relu2_2', 'relu3_3', 'relu4_3', 'relu5_3'], weight: [0.2, 0.2, 0.2, 0.2, 0.2]

We train multi style image inputs model with and without total variation loss. We use 0, 1, 10 for total variation loss weight. For all these weight, the model can converge within 1000 iterations.

6.4. Network Comparison

All parameters are the same as 6.3 except network layers. Both network 1 and 2 are examined for performance comparison. Variation loss weight of 10 is chosen. For Table 1 and 2, it takes 50 pictures for style images and 1 content image with sample size of 10 and degree of freedom of 9. For 95 % confidence interval, t is 2.626 obtained from t-table.

References

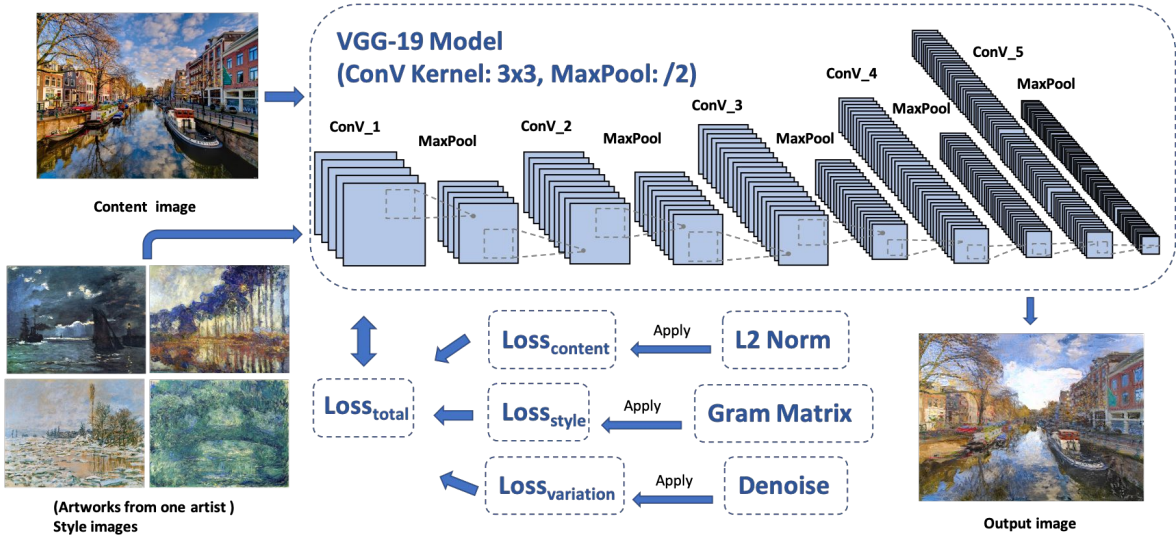
Fujun L, Sylvain P, Eli S Kavita B. Image style transfer using deep photo style transfer, 2017.

Gatys, Leon A. Image style transfer using convolutional neural networks, 2016.

Smith, Cameron. neural-style-tf. https://github.com/cysmith/neural-style-tf, 2016.

INTRODUCTION

A convolutional neural network that styles an image to resemble an artist's paintings with original color preserved.

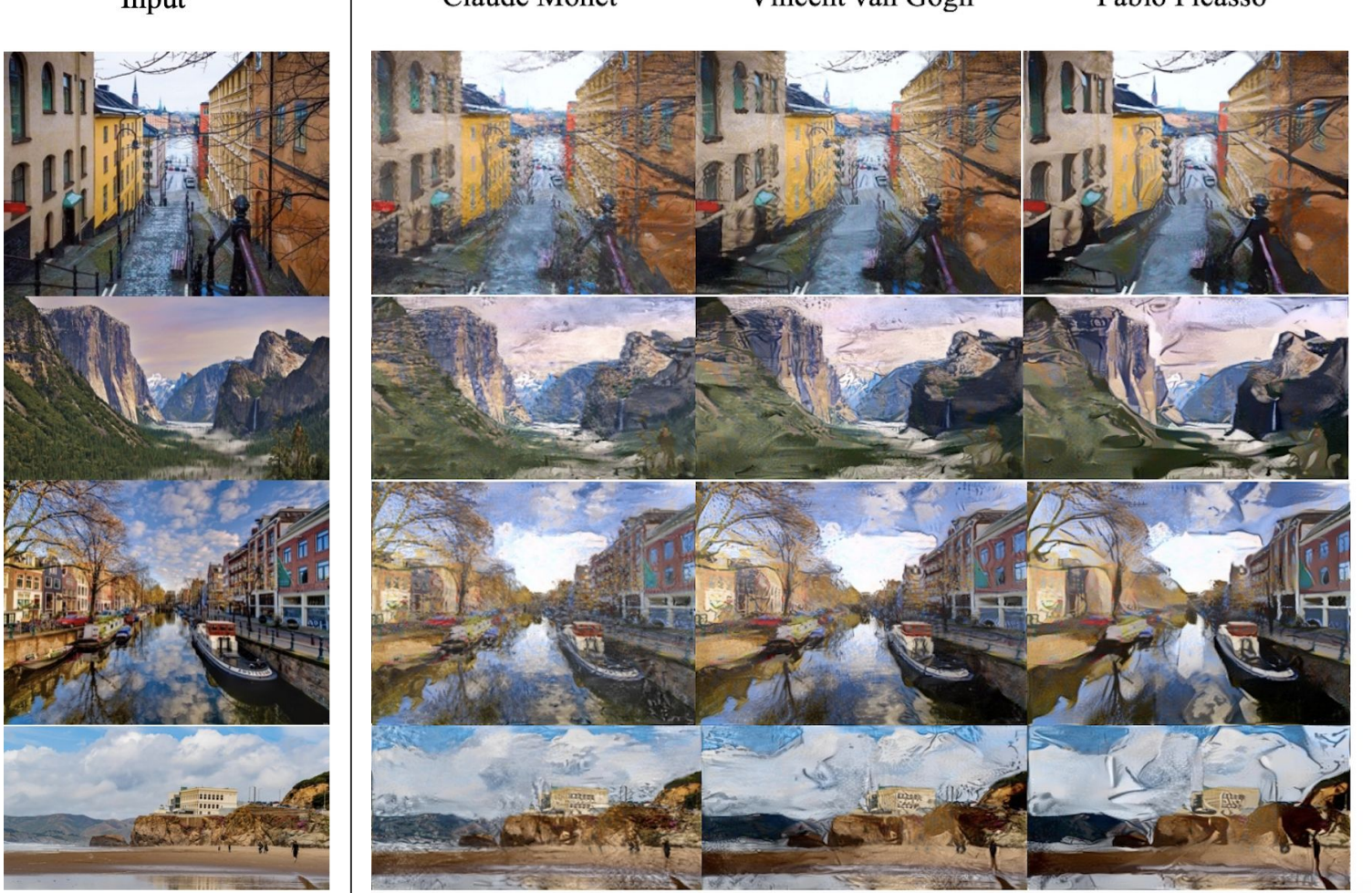


APPROACH

- Initialize VGG19 convolutional neural network
- Define loss functions
  - Content Loss :
    use L2 norm to calculate content difference
  - Style Loss :
    use inner product of feature maps to calculate the correlations between layers.
  - Total Variation Loss :
    sum up the absolute differences for neighboring pixel-values
- Set up gradient descent process (L-BFGS)
- Train the model with multiple style images
- Apply color preserving method : Luminance-only transfer

PHOTO PAINT

RESULT



Vimeo: https://vimeo.com/415706303