CS 229 Project Final Report: Neural Style Transfer

Fangze Liu [fangzel@stanford.edu] [1] **Wen Wang** [wenwang@stanford.edu] [2]
Jixun Ding [xunger08@stanford.edu] [2]

1. Introduction

Given a pair of images, the process of combining the "style" of one image with the "content" of the other image to create a piece of synthetic artwork is known as *style transfer*. Style transfer is a popular topic in both academia and industry, due to its inherent attraction and wide applicability. Gatys et al. (2015) demonstrated a generalized style transfer technique by exploiting feature responses from a pre-trained CNN, opening up the field of *neural style transfer*, or the process of using neural networks to render a given content image in different artistic styles. Since then, many improvements, alternatives, and extensions to the algorithm of Gatys et al. has been proposed.

In this work, we implement Gatys' algorithm – with some minor additions and modifications – to produce some synthetic artwork. We explore how different hyperparameter settings produce different output images. We also implement a spatial control extension to Gatys' algorithm based on Champandard (2016) and Gatys et al. (2016), and check if and how the spatial control extension improves output image quality.

Our code can be found in this GitHub repo.

2. Related Work

A comprehensive overview of the field of neural style transfer can be found in Jing et al. (2017).

Our work closely follows Gatys et al. (2015) and Gatys et al. (2016). Gatys' method and its extensions outlined in these works is still considered the state-of-the-art method for neural style transfer. It extracts feature responses from passing images through a pre-trained CNN and optimizes the loss function constructed from these feature responses. It produces visually impressive images, and is flexible in the sense that it can transfer the style of an arbitrary style image onto an arbitrary content image. It suffers from the drawback that its iterative optimization process, which involves repeatedly passing a target image forward and backward through a CNN, is very time consuming. Our implementation (CPU only) for a 600×800 image typically takes about 1-2 hours to converge [depending on how the convergence criteria is defined, this takes 300-500 iterations].

A different class of neural style transfer methods is *real-time style transfer*, demonstrated by Johnson et al. (2016); Ulyanov et al. (2016). This class of methods uses the same loss function as Gatys et al., but instead of direct optimization, approximates the solution by training one "image transformation network" for every target style. At test time, the user supplies a content image and the stylized output is obtained by one single forward pass through the image transformation network. This type of method is very efficient at test time, but suffers from the drawbacks of slightly lower image quality and the need to train one neural network for every target style, which is time consuming and inflexible. Efforts to achieve real-time style transfer that can transfer multiple or even arbitrary styles include Li et al. (2017; 2018); Gu et al. (2018). We do not attempt to tackle real-time style transfer in our work.

3. Dataset and Features

In our implementation of Gatys' method we use weights of VGG-19 (Simonyan & Zisserman, 2014), a 19-layer deep convolutional neural network which has been pre-trained on the ImageNet dataset.

In artwork generation and algorithm evaluation, we use a variety of content and style images. As content images, we use personal photos, stock images, as well as a few images in the benchmark dataset *NPRgeneral* proposed by Mould & Rosin (2016). As style images, we use well known artworks of Van Gogh, Claude Monet, Wassily Kandinsky, Pablo Picasso, etc., all of which are in the public domain.

Before the algorithm begins, pairs of style and content images are cropped and resized to be of matching aspect ratios and sizes. Before images are fed through VGG-19, they are normalized by subtracting the RGB average of the ImageNet dataset.

[1]Department of Physics, Stanford University [2]Department of Applied Physics, Stanford University.

4. Methods

Many implementations of neural style transfer based on Gatys' method are available online, and we have used primarily Zhao (2018); Athalye (2015); Wang (2017) as reference when writing our implementations.

We first implement the vanilla version of Gatys' method, following Gatys et al. (2015). This method makes use of the fact that a CNN trained on sufficient labeled data for a specific task can learn "generic feature representations" that generalize to other datasets and computer vision tasks.

Passing an image $\vec{x}$ through the CNN produces feature maps $\{F^{[l]}(\vec{x})\}$. The *style representation* of the image at layer l is represented by the Gram matrix:

$$\mathcal{G}\left(F^{[l]}\left(\vec{x}\right)\right) = \left[F^{[l]}\left(\vec{x}\right)\right]\left[F^{[l]}\left(\vec{x}\right)\right]^{T} \tag{1}$$

To transfer the style of an artwork $\vec{a}$ onto a photograph $\vec{p}$ and produce a synthesized image $\vec{x}$, we initialize a random image $\vec{x}$, and minimize the total loss function (Eq. 2) defined as a linear combination of content loss (Eq. 3), style loss (Eq. 4) and total variational loss (Eq. 5). In this way, the output image $\vec{x}$ strives to match $\vec{p}$ in content and match $\vec{a}$ in style. Gatys' paper only specified content loss and style loss, but total variation regularization (Eq. 5) is a standard addition to Gatys' method, and helps reduce the random noise in the output image $\vec{x}$.

$$\mathcal{L}_{\text{total}}(\vec{p},\vec{a},\vec{x}) = \alpha\mathcal{L}_{\text{c}}(\vec{p},\vec{x}) + \beta\mathcal{L}_{\text{s}}(\vec{a},\vec{x}) + \gamma\mathcal{L}_{\text{v}}(\vec{x}) \tag{2}$$

$$\mathcal{L}_{\text{c}} = \sum_{l\in\mathcal{C}} w_{\text{c}}^{[l]} \left\| F^{[l]}\left(\vec{p}\right) - F^{[l]}(\vec{x}) \right\|_2^2 \tag{3}$$

$$\mathcal{L}_{\text{s}} = \sum_{l\in\mathcal{S}} w_{\text{s}}^{[l]} \left\| \mathcal{G}\left(F^{[l]}\left(\vec{a}\right)\right) - \mathcal{G}\left(F^{[l]}(\vec{x})\right) \right\|_2^2 \tag{4}$$

$$\mathcal{L}_{\text{v}} = \sum_{i,j} \left[\left|\vec{x}_{(i,j)} - \vec{x}_{(i+1,j)}\right| + \left|\vec{x}_{(i,j)} - \vec{x}_{(i,j+1)}\right| \right] \tag{5}$$

In equations above, $\mathcal{C}$ is the set of content representation layers, $\mathcal{S}$ is the set of style representation layers, the weights $\left\{w_{\text{s}}^{[l]}\right\}$ and $\left\{w_{\text{c}}^{[l]}\right\}$ determine the relative weights between loss obtained at multiple style and content representation layers in the CNN. The weights α, β, γ determine the relative weights between content, style, and total variational loss. All these parameters are hyperparameters that need to be manually chosen by the user to obtain the most visually pleasing result.

We also implement a spatial control extension to Gatys, following (Gatys et al., 2016). Many artworks and photos contain spatially separable style and content regions within the same image. By splitting the style and content images into segments, and applying the style in each specific segment of the style image to the corresponding segment of the content image, we expect to obtain interesting results and achieve more flexibility in the style transfer procedure.

For this extension, we create mask images for the content and style images as additional inputs. The mask images are drawn by hand on an iPad. Matching regions in the style and content images are drawn in the same color. The number of regions/colors, K, is a hyper-parameter chosen by the user. Then, K-means is used to extract the matching pixels within the content and style images. Between every pair of matching segments, a normalized style loss is calculated and summed together to produce a normalized style loss. Compared to the vanilla method, the style loss (Eq. 4) is replaced by the normalized style loss (Eq. 6), while content and total variational loss are defined as before.

$$\mathcal{L}_{\text{s}} = \sum_{l\in\mathcal{S}} \sum_{k=1}^{K} \frac{w_{\text{s}}^{[l]}}{K} \left\| \frac{1}{N_{a_k^{[l]}}} \mathcal{G}\left(F^{[l]}\left(\vec{a_k}\right)\right) - \frac{1}{N_{x_k^{[l]}}} \mathcal{G}\left(F^{[l]}(\vec{x_k})\right) \right\|_2^2 \tag{6}$$

In Eq. 6, K is the number of segments/colors we separate the style and content images into. x_k is the pixels classified as the k-th segment in the synthesized image and a_k is the pixels classified as the k-th segment in the style image; $N_{x_k^{[l]}}$ is the number of neurons classified as the k-th segment in the feature map $F^{[l]}(\vec{x_k})$ and $N_{a_k^{[l]}}$ is the number of neurons classified as the k-th segment in the feature map $F^{[l]}(\vec{a_k})$.

The total loss function (Eq. 2) for both the vanilla and spatially controlled version of Gatys method can be minimized numerically over $\vec{x}$ via any optimizer of the user's choice. We use the package TensorFlow to build a CNN that uses the same architecture, activation functions, and pre-trained weights as VGG-19, except max pooling layers are replaced by average pooling, as specified by Gatys et al. Loss function values are obtained by passing $\vec{x}, \vec{p}, \vec{a}$ forward through the network, and the gradient $\partial\mathcal{L}_{\text{total}}/\partial\vec{x}$ can be found via back propagation. Tensorflow does not support the L-BFGS method used by Gatys et al., so we use Adam optimizer. The optimizer's learning rate r is also a hyperparameter.

5. Results and Discussion

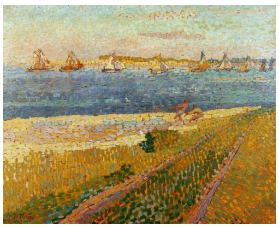
5.1. Vanilla method

When not specified, hyperparameters default to settings listed in Table. 1. The choice pf $\mathcal{C}$ and $\mathcal{S}$ agrees with Gatys et al. The weights $\left\{w_{\text{s}}^{[l]}\right\}$, $\left\{w_{\text{c}}^{[l]}\right\}$, α, β, γ are chosen so that content, style, and variational loss have roughly the same order of magnitude in most of our attempts. The learning rate r is chosen so that convergence occurs in a few hundred iterations and loss curve is smoothly decreasing in most of our attempts. However, optimal hyper-parameter choice

Table 1. Default hyperparameter settings

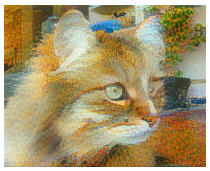
$\mathcal{C}$	'conv4_2'
$\mathcal{S}$	'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1', 'conv5_1'
$\left\{w_c^{[l]}\right\}$	1
$\left\{w_s^{[l]}\right\}$	$1, 1, 1, 1, 1$
(α, β, γ)	(1e-4, 1, 1e-5)
learning rate	start at $r = 10$, linear decay over 1000 iterations to $r = 1$

(a) Content: one image from *NPRgeneral*

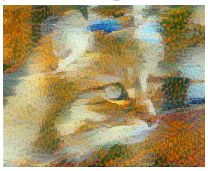


(b) Style: *The Schelde near Veere* by Jan Toorop

(c) α =1e-2



(d) α =1e-3



(e) α =1e-8

Figure 1. Changing content weight. Default hyperparameter settings except changing α. Image resolution `400x500`. Termination at 400 iterations.

generally depends on the specific pair of content and style images and require a tedious and subjective tuning process.

We use some examples to showcase the properties of Gatys algorithm. Changing α while keeping all other weights constant allows us to change the mix of content and style in the synthesized result. This is shown in Figure 1.

The algorithm is capable of transferring arbitrary styles to arbitrary content images. Figure. 2 shows applying styles of (left) *Nature Morte* by Gino Severini, (middle) *Tables for Ladies* by Edward Hopper, (right) *Frenières* by Hubertine Heijermans to the same image of berries in the *NPRgeneral* dataset. Even only using default hyperparameter settings, we obtain acceptable results.

5.2. Spatial control extension

A successful example of spatially controlled neural style transfer is shown in Figure. 3. We transferred the style of *Landscape from Saint-Rémy* by Vincent van Gogh to a photo of the Stanford Dish, first without any spatial control, then with separate landscape/sky regions. Qualitatively, we

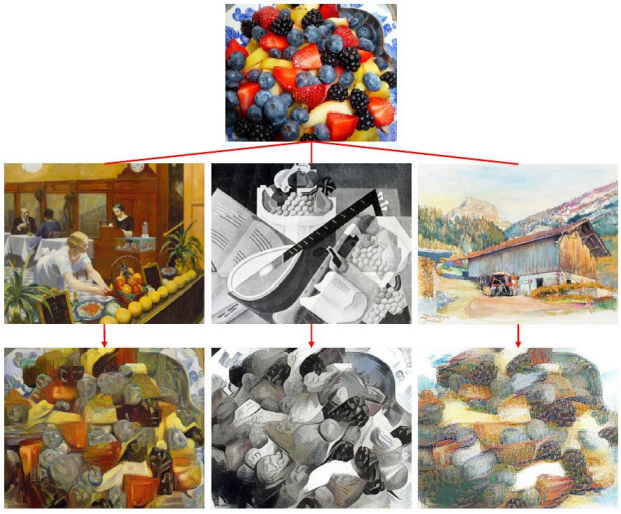
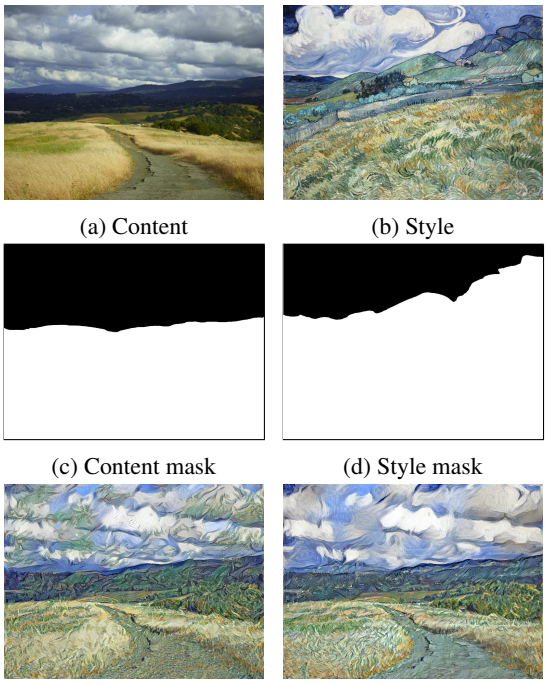


Figure 2. Applying different styles to the same image. Default hyperparameters, termination at 400 iterations, resolution `400x500`.

see that spatial control extension produces a better result. The green/yellow tone of the grass and road and blue/white tone of the cloudy sky is better preserved in the spatially controlled output. The vanilla method applies style globally to the output, and as a result we have some green patches mixed into the sky.

An unsuccessful example of spatially controlled neural style transfer is shown in Figure. 4. We attempt to transfer the style of *Portrait of Sally Fairchild* by John Singer Sargent to a photo of Jixun Ding. First, we ran Gatys' method without spatial control, and got Fig. 4g, a portrait with no face. Gatys' method is well known to not work well for portraits (Selim et al., 2016), so this result is not surprising. Next, we try applying simple masks that separate the foreground and background of input images. This experiment produces slightly better results (Fig. 4h), but still produces distorted facial features. Finally, we try separating the style and content image into $K = 5$ segments, containing mouth, face, hair, clothing, and background, respectively. We expected this experiment to produce the best result, but instead it produced a Nightmare image (Fig. 4i)!

The application of detailed spatial control may have failed because our target style is more photorealistic than typical style images. This "style" may be harder to capture than that of *Starry Night*, for example. Another possibility is that because spatial control introduces distortions at region boundaries, and our images are limited in resolution, more detailed control introduces distortions that end up dominating the image. It would be interesting to try spatially controlled neural style transfer with a different style, or larger image resolution.

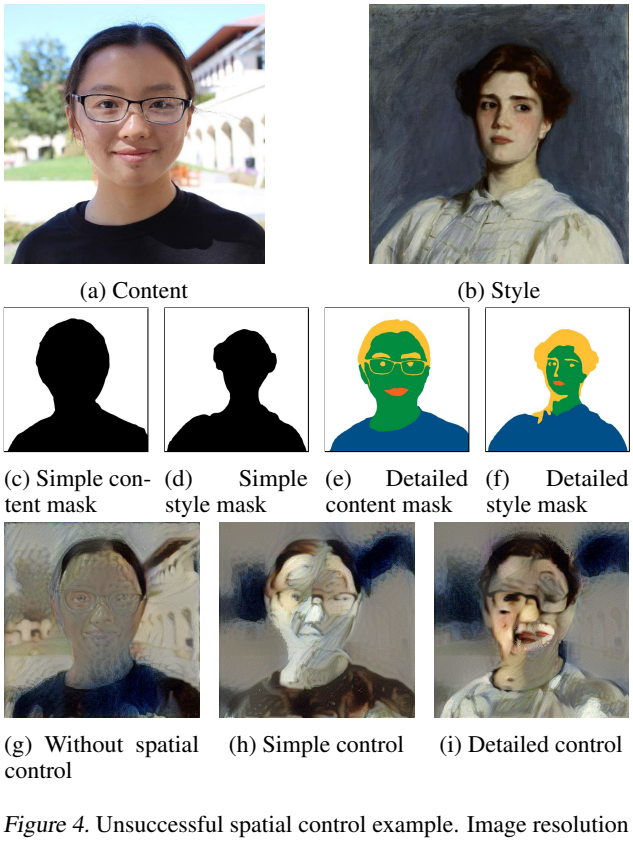


(a) Content (b) Style

(c) Content mask (d) Style mask

(e) Without spatial control (f) With spatial control

Figure 3. Successful spatial control example. Image resolution `600x800`.



(a) Content (b) Style

(c) Simple content mask (d) Simple style mask (e) Detailed content mask (f) Detailed style mask

(g) Without spatial control (h) Simple control (i) Detailed control

Figure 4. Unsuccessful spatial control example. Image resolution `500x500`. Termination at 400 iterations. Default hyperparameters.

6. Conclusion and Future Work

In this work, we have implemented Gatys' neural style transfer algorithm and explored the impact of hyperparameter setting on the output. We implemented a spatial control extension to Gatys' algorithm and showcased successful and unsuccessful examples of adding spatial control to neural style transfer.

Gatys' method is capable of arbitrary style transfer, and produces high quality output, but has high computational cost ($>$ 1 hour CPU runtime for 500x500 image, approximately 300 iterations). All our output images are limited in resolution and contains residual noise from random image initialization. Additionally, each pair of style, content images requires custom hyperparameter tuning in order to obtain the most visually impressive result.

Our spatial control method requires the user to generate two additional input masks. It enables us to obtain higher quality output for some examples, but introduces distortion at the mask boundaries, and may fail spectacularly if the target style is too photorealistic, or if image resolution is too low.

Potentially interesting future work of this project includes:

1. Implement other direct extensions to Gatys' method, such as color control, feature size control, and interpolation between multiple styles.

2. Implement a real-time style transfer algorithm such as Gu et al. (2018).

3. Explore options for automatic hyperparameter tuning to avoid the tedious hyperparameter tuning process.

4. Explore more quantitative ways to evaluate style transfer results, instead of subjective judgment.

7. Contributions

Everyone worked on the basic implementation of Gatys' algorithm. Fangze and Wen implemented the spatial control extension based on K-means. Jixun wrote most of the final report and made the poster, with suggestions from Fangze and Wen.

References

Athalye, A. Neural style. `https://github.com/anishathalye/neural-style`, 2015.

Champandard, A. J. Semantic style transfer and turning two-bit doodles into fine artworks. *CoRR*, abs/1603.01768, 2016. URL `http://arxiv.org/abs/1603.01768`.

Gatys, L. A., Ecker, A. S., and Bethge, M. A neural algorithm of artistic style. *CoRR*, abs/1508.06576, 2015. URL http://arxiv.org/abs/1508.06576.

Gatys, L. A., Ecker, A. S., Bethge, M., Hertzmann, A., and Shechtman, E. Controlling perceptual factors in neural style transfer. *CoRR*, abs/1611.07865, 2016. URL http://arxiv.org/abs/1611.07865.

Gu, S., Chen, C., Liao, J., and Yuan, L. Arbitrary style transfer with deep feature reshuffle. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

Jing, Y., Yang, Y., Feng, Z., Ye, J., and Song, M. Neural style transfer: A review. *CoRR*, abs/1705.04058, 2017. URL http://arxiv.org/abs/1705.04058.

Johnson, J., Alahi, A., and Fei-Fei, L. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*, 2016.

Li, X., Liu, S., Kautz, J., and Yang, M. Learning linear transformations for fast arbitrary style transfer. *CoRR*, abs/1808.04537, 2018. URL http://arxiv.org/abs/1808.04537.

Li, Y., Fang, C., Yang, J., Wang, Z., Lu, X., and Yang, M. Universal style transfer via feature transforms. *CoRR*, abs/1705.08086, 2017. URL http://arxiv.org/abs/1705.08086.

Mould, D. and Rosin, P. L. A benchmark image set for evaluating stylization. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, Expresive '16, pp. 11–20, Aire-la-Ville, Switzerland, Switzerland, 2016. Eurographics Association. URL http://dl.acm.org/citation.cfm?id=2981324.2981327.

Selim, A., Elgharib, M., and Doyle, L. Painting style transfer for head portraits using convolutional neural networks. *ACM Trans. Graph.*, 35(4):129:1–129:18, July 2016. ISSN 0730-0301. doi: 10.1145/2897824.2925968. URL http://doi.acm.org/10.1145/2897824.2925968.

Simonyan, K. and Zisserman, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

Ulyanov, D., Lebedev, V., Vedaldi, A., and Lempitsky, V. S. Texture networks: Feed-forward synthesis of textures and stylized images. *CoRR*, abs/1603.03417, 2016. URL http://arxiv.org/abs/1603.03417.

Wang, Z. guided-neural-style. https://github.com/wzirui/guided-neural-style, 2017.

Zhao, N. Image style transfer. https://github.com/NELSONZHAO/zhihu/tree/master/image_style_transfer, 2018.