Neural Style Transfer - Reimplementation

Ankit Mathur and Anaga Rajan

December 2017

Abstract

Neural style transfer is a method to create images that are stylized to match certain artistic styles, while still maintaining the content of the original image. This project uses the outputs of convolutional filters from the VGG 19 network and defines novel losses to create an optimization problem that creates these images.

1 Introduction

What does it mean to be an human in the visual world? It does not just mean just seeing. Countless blind artists could not see, yet they produced some of the most outstanding artwork of all time. These people do not just use sight to classify an object, they use touch. There's something deeper here - texture and color are different parts of art. As we drive forward in the world of computational photography, we can see that the latest developments have focused on teaching machines how to actually create visual beauty. Just like a child, we want to teach them not only how to classify these images but also how to construct textures and objects. We want to teach them to be artists. This is the ambitious goal of this research.

2 Algorithm

The simplest step in this endeavor is to find a way that we can show an algorithm a style of art and a normal image and have the algorithm paint that image in the style of the first. The Gatys et al '14 paper "Image Style Transfer Using Convolutional Neural Networks" describes an algorithm to do this [1].

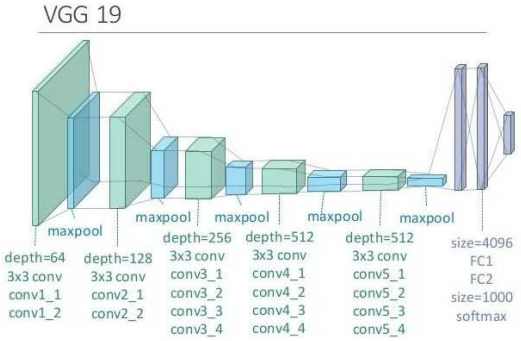
The algorithm described by the Gatys et al '14 paper seeks to optimize for a content and style loss [1]. This is a reasonable abstraction for a way to create an image that has the same style as the original image but still looks like the prior image.

2.1 Convolutional Neural Networks

The method prescribed by the paper asserts that the output of performing certain convolutions on the different parts of the image can be used as signal

about what exists in those regions. This makes sense, since the convolution operator essentially gives you some high level information about the regions it is convolving over.

Then, it must be decided what kind of convolutions we want output from. Here, we bring in the VGG19 neural network. Originally designed for the broad scale image classification case, we are actually more interested in the structure of the network. Here is what it looks like:



Note how the first layers of the convolution have the image as a larger component - this implies that the convolutions you perform will give you more fine-grained details.

What the paper asks us to do is to get the convolutional filter responses from a certain set of convolutional layers. Based on which layer one gets the filter responses from, one would get different kinds of information about the image.

2.2 Content Loss

The paper defines the content loss as the sum of the L2 losses between the corresponding filter responses at the layers of the target image and the content image. Let N_l represent the number of distinct filters in a layer and let M_l be the height times the width of the feature maps in that layer. F^l is a matrix where F^l_{ij} is the activation of the $i'th$ filter at position j in layer l for the target image being optimized, $\vec{x}$. Furthermore, let P^l be this matrix for the content image. Then, the content loss is:

$$\mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{ij} - P^l_{ij})^2 \tag{1}$$

The paper suggests that we use the `conv4-2` layer's filter outputs. This is because `conv4-2` preserves the content while losing the high frequency details (as a result of being from a later layer of the convolutional network).

2.3 Style Loss

The style loss is much more difficult to intuitively describe. In essence, what the paper asserts is that we want the low and high level style features. Thus,

we should be using layers throughout the convolutional depth for that (i.e. `conv1-1`, `conv2-1`, `conv3-1`, `conv4-1`, and `conv5-1`)

To get a representation of style given a certain layer, the Gram matrix is composed via the following mathematical operation.

$$G_{ij}^l = \sum_k F_{ik}^l F_{jk}^l \tag{2}$$

This is is an auto-correlation of the different convolutional filter responses with every other filter response of the image from the same layer. In words, one might describe this as being a representation of style because it represents how different components of the image relate to each other, defining, in a way, the style of the image. Furthermore, it removes much of the strict spatial components since we evaluate the response correlation with every other filter - not just the nearby ones.

To compute the loss, one computes the Gram matrix for the target and the style image and considers the L2 loss between them. Let A^l represent the Gram matrix for the style image, $\vec{a}$. Similarly, let G^l be the gram matrix for the target image being optimized, $\vec{x}$. The style loss for a given layer is then:

$$E_l(\vec{a}, \vec{x}) = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{ij}^l - A_{ij}^l)^2 \tag{3}$$

Finally, since there are multiple layers of filters that we consider for the style loss, we need to compute a weighted average. We found the setting these weights to be the same led to the best result.

$$\mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l(\vec{a}, \vec{x}) \tag{4}$$

2.4 Optimization

The loss is then weighted with constants alpha and beta to describe how much style and how much content we want. This puts our final loss term at:

$$\mathcal{L}(\vec{a}, \vec{p}, \vec{x}) = \alpha \cdot \mathcal{L}_{\text{content}}(\vec{p}, \vec{x}, l) + \beta \cdot \mathcal{L}_{\text{style}}(\vec{a}, \vec{x}) \tag{5}$$

We use the L-BFGS optimizer with the learning rate $\eta = 1$. We start from the content image for the most quality results. At every step of the optimization, we add an extra constraint which keeps the result between the values 0 and 1. We wrote an implementation in Pytorch that is hosted on Github at this link: []Github link].

3 Our Discoveries

3.1 Layer Selection

We found several things that created better results than the Gatys paper. For example, to obtain aesthetically pleasing results, rather than varying the style weights, we found using different layers for matching the style and content to be a better tunable hyperparameter. This was inspired by other work critical of the Gatys work's findings [2]. The Nikulin work names this Partial Style Transfer, but they explore only a specific alternative layer configuration of content using (`conv1-1, conv2-1, conv3-1, conv4-1, conv5-1`) and style using (`conv3-1, conv5-1`). We explore a more custom approach based on the style image - if the style image contains more of the high level features (like cubism), then we will use the higher layers for style and not the lower ones (see the Cubism 4 and the Afremov examples 1):

Furthermore, if the content image is extremely detailed, we will use only `conv5-2` as the matching layer for content. This retains even less of the sharp detail as compared with `conv4-2`. We feel that this led to significantly more pleasing results on average than the standard parameters (see the Mediterranean Garden 2 and Self-Portrait 6).

3.2 Initialization

We found that, while the Gatys paper contends that the initialization does not matter, it makes a significant difference. Starting with the content image leads to aesthetically pleasing results that converge faster. The Nikulin et al work explores this as well, in direct contrast with the original Gatys et al paper which explicitly contends that the initialization does not matter [2] [1].

3.3 Losses and Backpropagation

Finally, as a technical detail, we found that the losses as defined did not converge quite as quickly as defined. Rather than using the sum of the content losses and the style losses, we used the mean squared error. Our results were extremely similar with both kinds of losses, but MSE converged faster somehow.

Moreover, we found that images were of significantly higher quality (and converged faster) if performing gradient updates on the style and content losses *individually*. Rather than maintaining a single gradient graph, we used backward propagation on both individually. Then, we passed the weighted, combined loss to the L-BFGS optimizer, so that we were still running the same optimization problem as the paper.

4 Appendix: Results and References

Figure 1: Expressionist Flower Market
Style: `conv3-1, conv4-1, conv5-1`
Content: `conv4-2`

Figure 2: Picasso's Mediterranean Garden
Style: `conv1-1, conv2-1, conv3-1, conv4-1, conv5-1`
Content: `conv5-2`

Figure 3: Mountain in the style of Kandinsky
Style: `conv1-1, conv2-1, conv3-1, conv4-1, conv5-1`
Content: `conv5-2`

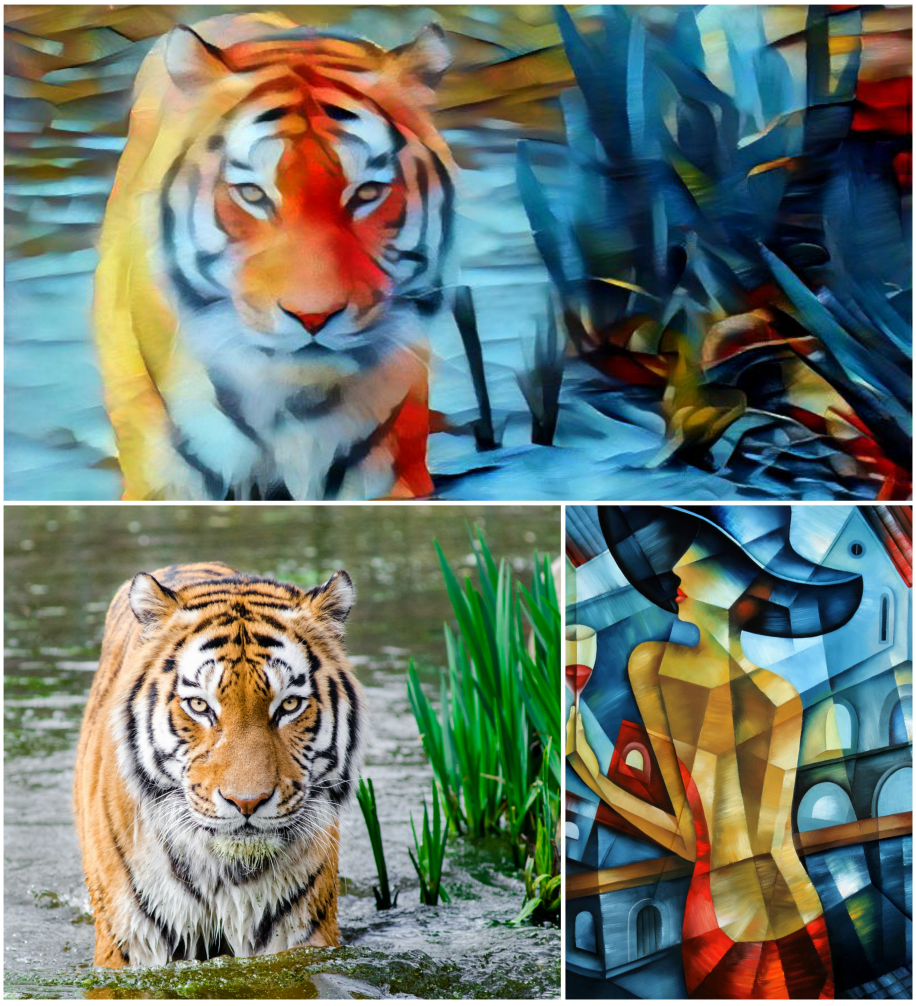


Figure 4: Cubist Tiger
Style: `conv1-1, conv2-1, conv3-1, conv4-1, conv5-1`
Content: `conv4-2`

Figure 5: Season Transfer
Style: `conv3-1, conv4-1, conv5-1`
Content: `conv4-2`

Figure 6: Arya Stark Self Portraits
Style: `conv1-1, conv2-1, conv3-1, conv4-1, conv5-1`
Content: `conv5-2`

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "Image Style Transfer Using Convolutional Neural Networks". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 2016, pp. 2414–2423. DOI: 10.1109/CVPR.2016.265. URL: https://doi.org/10.1109/CVPR.2016.265.

[2] Yaroslav Nikulin and Roman Novak. "Exploring the Neural Algorithm of Artistic Style". In: *CoRR* abs/1602.07188 (2016). arXiv: 1602.07188. URL: http://arxiv.org/abs/1602.07188.