Multi-Objective Neural Style Transfer

Allison Tam
actam@mit.edu

Andy Kuang
akuang@mit.edu

Abstract

Neural style transfer is the process of recreating one image in the style of a given reference image, through the use of a deep convolutional neural network (DCNN). In this paper, we build upon the work done by Gatys et al. *[1] to accomplish neural style transfer for additional objectives: 1. transferring multiple styles at different proportions, 2. transferring styles at different scales, and 3. color preservation of content image. We describe the changes necessary to achieve these objectives and hope that our results motivate possibilities for continued research in this exciting realm.*

1. Introduction

Style transfer is the process of recreating one image in the style of a given reference image. Traditionally, style has referred to the texture of a given image, such as a large scale pattern (e.g. the swirls/strokes in Picasso's "Starry Night"). In more general terms, style can also refer to the color palette of an image. While transferring different elements of style can be seen as a multi-objective problem, the main constraint is to do this while preserving the semantic content or structure of the original image.

A common paradigm is to separate style from structure. A generative model is then used to create the final output image. Literature differs in their approach in how style is quantified, varying from search-based approaches to inference to neural representations.

In this paper, we explore how deep convolution neural networks (DCNN) extract representations of images across different scales. We use these representations to build output images. We first re-implement the neural style transfer procedure presented by Gatys *et al*. [1]. We then implement extensions to accomplish additional objectives:

1. **Mix two styles:** Goal is to construct an image that mixes two styles. The first extension is a transfer of two styles at different proportions. The second is a transfer of two styles at different scales - one at a global level and one at a local level.

2. **Color-preserving style transfer:** goal is to construct an image with the style of the style image and the content of the content image, while preserving the color of the content image.

1.1. Related Work

There are many approaches to style transfer. The traditional school of thought is to transfer style as if you are transferring textures. Frigo *et al*. uses an adaptive quadtree to divide the images into patches, matche content image patches to those from the style image, and merge the style patches through belief propagation [2]. Elad *et al*. extends the patch approach by matching across different resolutions and aggregating results. They also incorporate other techniques (e.g. color transfer, segmentation) for more intelligent transfer [3].

While patch-based approaches have yielded competitive results, the success of DCNNs can be leveraged in style transfer. DCNNs find features in images that mimic the human visual system and thus may generate more compelling images. We investigate this further in the paper.

2. Neural Style Transfer

We explore a neural approach to style transfer as proposed by Gatys *et al*. [1]. The overarching idea is to leverage VGG's representation of an image to quantify its content and style. The representations are the activations of the first few layers of VGG. A generative model combines two notions of loss (content loss, style loss) to create an output image with both components. The goal is to generate an output image that achieves similar activations in VGG.

2.1. Layers and Representations

Because of its superior performance in ImageNet Challenge, VGG is used to generate features maps. Each layer applies a filter bank on the inputs, highlighting specific features. A layer with N_l distinct filters has N_l flattened feature maps each of size $M_l = hw$, where h, w are the height and width of the output activation. The feature map extracted from the activations of layer l is referred to as $F^l \in \mathbb{R}^{N_l \times M_l}$.

Let the output image be X and content image be C. Con-

tent loss for layer l is the squared error, as defined below.

$$L_{l,\text{content}}(F_C^l, F_X^l) = \frac{1}{2}(F_C^l - F_X^l)^T(F_C^l - F_X^l) \quad (1)$$

The style loss for layer l is the MSE of Gram matrices of the feature maps. Let $G_I^l = F_I^l(F_I^l)^T$.

$$L_{l,\text{style}} = \frac{1}{4N_l^2M_l^2}(G_C^l - G_X^l)^T(G_C^l - G_X^l) \quad (2)$$

The overall loss to minimize is then a weighted average of the above losses. Weights can be moved around to reflect different priorities. α, β says how close the output image will be to the style image or the content image. w_l determines if the output image will mimic small-scale or large-scale features of the style image.

$$\mathcal{L}_{\text{total}} = \alpha L_{l,\text{content}} + \beta \sum_l w_l L_{l,\text{style}} \quad (3)$$

2.2. Implementation Details

Our content loss is based on the feature map from conv2_2. Our style loss is based on feature maps from the first five layers. We used uniform weighting internally between style losses with $w_l = \frac{1}{5}$.

The content and style losses are captured as their own loss modules, inserted after the convolutional layers specified above. An initial image with a gradient parameter is passed through the model. Total loss is computed. Backpropagation then takes place on the initial image itself– not on the weights of the VGG or loss modules. With each iteration, the initial image is transformed into an output image that minimizes the content and style loss, thus incorporating elements from both reference images. We terminate typically after 300 iterations (fig. 1).

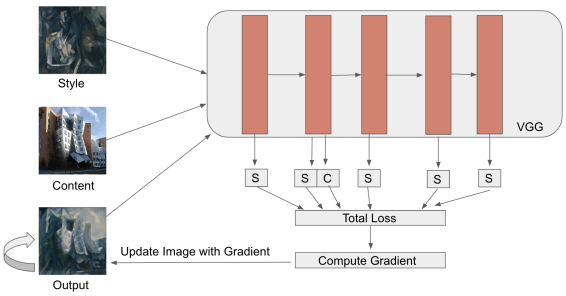

Figure 1. Basic workflow of style transfer process

As suggested by Jacq's tutorial, both the content and style losses are calculated from layers that occur earlier in the VGG, compared to the layers the original authors used [4,1]. With our current configuration, we think that our images look better with more details intact, since using loss from later layers (as in original paper) penalizes less for deviations at the local level.

3. Experimental Results

3.1. Initial Results

We worked with content images of Stata and a dancer. We wanted to transfer styles from *Starry Night* and a dark cubism painting by Picasso. Because content loss is naturally larger, we used $\frac{\beta}{\alpha} = 10^6$. The starting point for the generative model is the original content image. Combined with a high weight on style, style transferred well (table 1).

Table 1. First column contains original style images. First row contains original content images. Grid contains outputs of each style/content input pairing.

3.2. Changing Initial Image

Whereas our initial results initialized the generative model with the content image, Gatys *et al*. initializes with a white noise image [1]. When we initialized with white noise, the style transfer quality was worse. For both the Stata and dancer content images, the output didn't have much content in it. Style was over-represented (fig. 2). To address this problem, we weighted content loss more by

Figure 2. Style transfer of dark Picasso onto dancer with $\frac{\beta}{\alpha} = 10^6$ (left) or 10^5 (right). Output initialized with white noise.

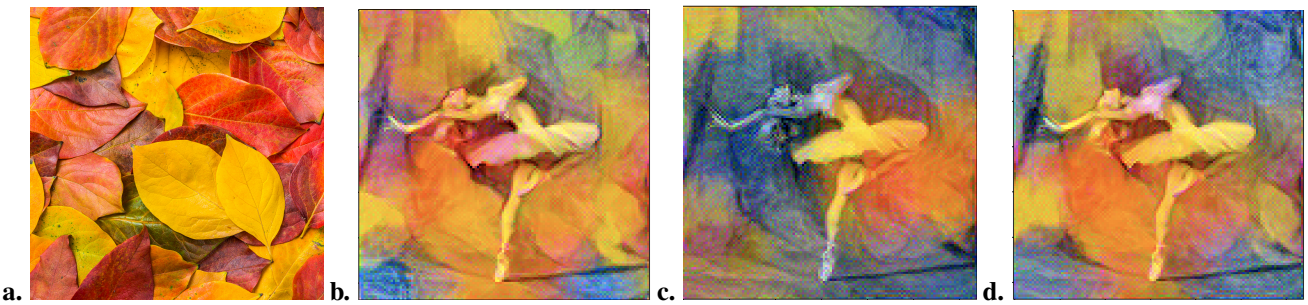

Figure 3. Style transfer of *Starry Night* and autumn fall leaves (a). From left to right, $\beta/\gamma = 0.7, 0.5, 0.3$ (b-d).

slightly increasing α, such that $\frac{\beta}{\alpha} = 10^5$. We were surprised that α only needed to change so little and expected something more 1:1. However, this may show that the content loss is inherently larger and doesn't need a parameter to inflate its importance. However, even with the parameter adjustment, the output still didn't look as good as before when we used the content image as the initialization image. We even let the model run for longer. This suggests that there exists several minima for total loss. By optimizing for two objectives (content and style) at the same time, we get more blurriness and pixelation.

We wanted to see how quality will deteriorate if the initial image was chosen to be purposefully different from both the content and style image. We had to slightly adjust the parameters to emphasize content more, as expected, but style transfer was still successful.

3.3. Extension: Multiple Styles

We extended the model to transfer multiple styles onto the same content image. This involved defining a new style loss module. We redefined total loss to be a weighted average of content loss and the two style losses. We introduce a new parameter γ, which represents how we prioritize style 2.

$$\mathcal{L}_{\text{total}} = \alpha L_{l,\text{content}} + \beta \sum_l w_l L_{l,\text{style1}} + \gamma \sum_l w'_l L_{l,\text{style2}} \quad (4)$$

We were able to see both styles rendered clearly in the content. We used bright autumn leaves as the 2nd style image. By changing the relative weighting of β, γ, we get different weightings of the style. α is increased slightly to maintain the content image, because the style losses are now naturally greater in magnitude. The output image is beautiful, containing patches of each style (fig. 3). Between the patches, there is a nice gradient transitioning between the two styles.

3.4. Extension: Styles Across Scales

To achieve an integration of styles, we might expect one style to be rendered at a large scale and another to be rendered at the local level. We tried achieving this effect by calculating style loss for layers 3-5 for the global style and layers 1-3 for the local style. We attempted this with a dancer (fig. 4). However, pixelation made the local style hard to detect. Further work is needed to sharpen up the resolution.

3.5. Extension: Color-Preserved Style Transfer

In above sections, the color palette of the output image is adapted from the style image. However, we want color preservation– when the output image maintains the same color palette as the content image, while reflecting the style of the style image. Gatys *et al.* achieves this goal by performing style transfer on black/white images and reconstructing color in YIQ colorspace after the fact[5].

Figure 4. Transfer of autumn leaves at global level and starry night at local level to dancer.

We first convert all images from RGB to the YIQ colorspace. We perform neural style transfer on the luminance channels (Y) only. The output image is $h \times w \times 1$ instead of $h \times w \times 3$. Then, we add in color by concatenating the output image with the IQ channels of the content image. Finally, we convert back to the RGB space to get a color-preserved output image.

Fig. 5 shows a comparison of style transfer with and without color preservation. The results are more compelling with color preservation.

The color preservation works well for the most part. However, it fails in cases where the luminosity histograms have distinctly different distributions (fig. 6). We can try to make improvements by applying a linear transformation to the black/white content image to achieve a histogram that is more similar to the style image's. For each pixel p_{ij} of the content image, we normalize with the following formula, where μ_C, σ_C is the mean and standard deviation of the

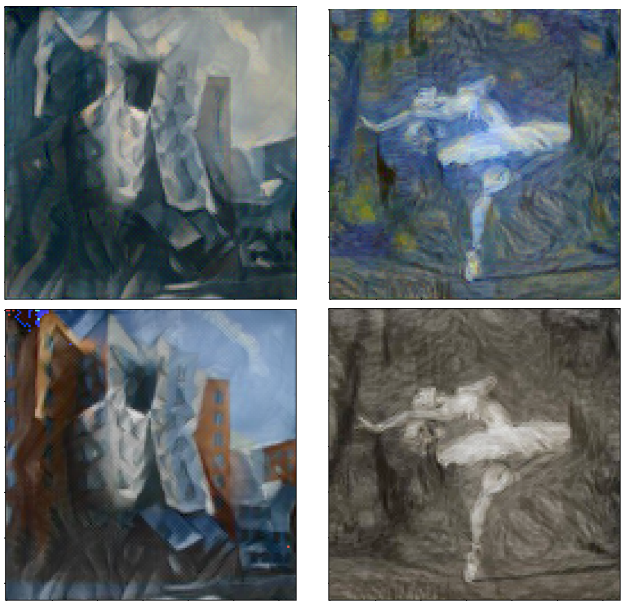

Figure 5. Top row is style transfer without color preservation (Stata with dark Picasso, dancer with *Starry Night*). The bottom row corresponds to the color-preserved version of the above.

content's luminosity histogram and μ_S, σ_S of the style's.

$$p'_{ij} = \frac{\sigma_S}{\sigma_C}(p_{ij} - \mu_C) + \mu_S \tag{5}$$

We applied this technique when doing a style transfer of bright paint strokes onto a waterfall (fig. 6). Despite normalization, there are still some artifacts in the output image. This is due to the fact that color-preservation used a simple concatenation between the style-affected L channels and the original IQ channels.

Fixing artifacts potentially requires colorization techniques. One fix is to use a mapping from $(p_{ij}, i + \delta i, j + \delta)$ to IQ, instead of a strict concatenation. Another is to impose a penalty for quick color changes in the RGB space.

4. Conclusion

We built upon the work of Gatys *et al.* [1] to explore neural style transfer for multiple objectives. We re-implemented the paper and explored the initialization and loss parameters. Then, we extended style transfer for additional objectives: transferring multiple styles at different proportions, transferring styles at different scales, and color preservation of content image. Furthermore, we applied normalization to luminance intensities to achieve better color preservation results.

With the increased research into DCNNs, we expect that there will be continued improvements to the quality of style transfer due to better image representations. Further work will be needed to smooth out pixelation or artifacts seen in

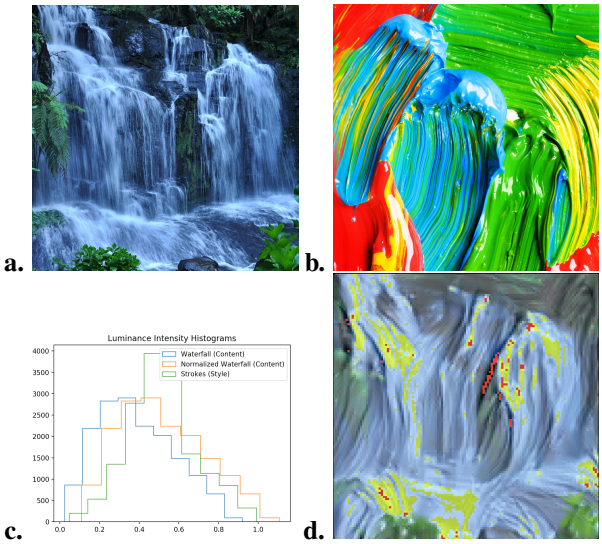

Figure 6. Output image (d) is a result of style transfer of paint strokes (b) onto a waterfall (a). Normalization was applied to to the intensity histograms (c).

generated output images. It may help to change the procedure to generate the image using a multi-scale pyramid [3].

We foresee many applications for further exploration. One is style transfer for videos. Another could be to work out the details of style transfer of cartoons, instead of artistic styles. Together, they can open the opportunity to render an animated film based on live-action movies or vice versa.

5. Individual Contribution

We pair-programmed together to recreate the initial paper and the color-preserved style transfer extension. While I was mainly responsible for the multiple styles extension and Allison for the styles across scales extension, we debugged as a team.

6. 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, pages 2414 − 2423, 2016.

[2] O. Frigo, N. Sabater, J. Delon, and P. Hellier, Split and Match: Example-Based Adaptive Patch Sampling for Unsupervised Style Transfer, CVPR, 2016.

[3] M. Elad, P. Milanfar: Style-transfer via texture-synthesis. CoRR abs/1609.03057 (2016). http://arxiv.org/abs/1609.03057

[4] A. Jacq. Neural Transfer Using Pytorch, 2016. https://pytorch.org/tutorials/advanced/neural_style_tutorial.html

[5] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman. Preserving color in neural artistic style transfer. arXiv preprint arXiv:1606.05897, 2016.