Photo Style Transfer in Tensor Flow

Aly Kane
Stanford University
alykane@stanford.edu

Amelia Lemionet
Stanford University
lemionet@stanford.edu

Fjori Shemaj
Stanford University
fshemaj@stanford.edu

Abstract

Style transfer between a photograph and artistic image is a common and well-studied subfield in computer vision. These models do not generalize well to style transfer between two photographs, as photographs tend to have very localized style. However, transfer between two images could potentially be useful for image filtering in apps or image enhancement techniques. Recent work by Luan et al has set forth updates to traditional style transfer models to further enhance such transfer. This paper seeks to recreate and improve upon the framework in Tensorflow.

1. Introduction

Style transfer is a popularly studied subfield in computer vision. Using convolutional neural networks, an image can be recreated in the artistic style of a painting while maintaining the key contents of the reference photograph. For example, style transfer can be used to transform a photograph of a bridge into a "painting" of the same bridge in the style of Van Gogh's Starry Night. Applications of such transfer are mostly for recreational and artistic purposes.

Style transfer between two photographs is a less studied problem. Given both a content image and a style image, photographic style transfer will recreate the content image in the style of the second image. This can be used to transfer effects such as time of day, season, and illumination. As an example, we could take a photograph of a bridge at daytime and modify the image to look as though it was taken at sunset. Photographic style transfer could have vast applications such as photographic filtering, home improvement visualizations, and photo enhancing.

Applying traditional style transfer methodology to two photographs has severe limitations. Paintings tend to have a very generalized style - an entire painting commonly has a consistent style. Photographs tend to have a more localized style, where the foreground of an image may be quite different stylistically than the background. Because of this, transfer methods may be confused as to how or where to transfer style. Style transfer tends to leave output with distored edges. This doesn't matter with an artistic style because paintings generally have an element of surrealism. However this distortion in transfer between two photos can create an unrealistic output.

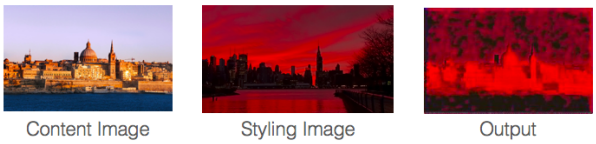

Figure 1. Photographic style transfer using traditional methods

In the paper *Deep Photo Style Transfer*, Luan et al [10] propose two updates to traditional style transfer methods for better transfer between two photographs. The first update is a semantic segmentation of both the content and style images. Using this segmentation, style can be transfered only between specific layers to avoid style spilling and content-mismatch problems. The second update is a photorealism regularization term, added to the loss function, which penalizes output for image distortion. In this work, we will focus on implementing local style transfer by using segmented images.

2. Related Work

Our work seeks to recreate the paper *Deep Photo Style Transfer* by Luan et al [10]. This paper builds upon the well known work Neural Style Transfer by Gatys et al. [3].

Throughout this paper we will give an in-depth explanation of neural style transfer as described in [3] and discuss specifics of how [10] improves the original algorithm to generalize well to two photographs.

The largest improvements in this method are gained through semantic segmentation of images. The authors of [10] use

a pre-trained DilatedNet [2]. Such network provides a very granular output in terms of segmentation, but as [10] suggests, such granularity is not necessary for style transfer. For example, when transferring style, we are not interested in the difference between an apple and an orange. Rather, we are interested that this object sits in the foreground of the image and wish to transfer the style of one fruit to the other. After semantic segmentation, Luan et al collapse like categories to improve performance.

3. Approach

3.1. Style transfer

The purpose of style transfer, similar to what was explained in the introduction, is to transfer the style or texture of a given image to some other image of interest. This idea was first introduced in [3], which we will use as a starting point for the current work.

Formally, we have two images: the content image $\vec{p}$ and the style image $\vec{a}$. The goal is to generate a new image $\vec{x}$ such that it contains the content from $\vec{p}$ with the style from $\vec{a}$. To do so, we first need to extract content and style from any given image, to then combine them in the output image.

Using a pre-trained CNN, the output of each ConvLayer reveals differently filtered versions of the input image. Generally, lower level layers reproduce exact pixel values, whereas higher level images capture high-level content. Style is not as easily extracted from an image, but we will do so by using the same trained CNN.

The following sections explain more formally the details of content and style extraction. Later on we will explain how to generate an image that matches both content and style.

3.1.1 Content extraction

As stated above, after each ConvLayer we have a feature map (also called activation map) of the input content image $\vec{p}$. Let:

- N_ℓ be the number of filters in the $\ell - th$ ConvLayer
- M_ℓ be the size of each resulting activation map ($M_\ell = H \times W$)

Thus far we have represented the output of a layer ℓ as a three dimensional image, of dimensions (W, H, N_ℓ). A different way of representing this would be by storing the responses from layer ℓ in a two dimensional matrix $F^\ell \in \mathbb{R}^{N_\ell \times M_\ell}$, where $F^\ell_{i,j}$ is the activation of the i-th filter, at position j in layer ℓ.

Such matrix is called the feature representation matrix in layer ℓ and will serve as the main building block for content transfer.

3.1.2 Generation of image matching input content

To create an image that recreates content from the input image $\vec{p}$ we start by generating a random white noise image $\vec{x}$. Let then P^ℓ and F^ℓ be the feature representations in layer ℓ of $\vec{p}$ and $\vec{x}$ respectively. We can thus compute the content loss between the two feature representations as:

$$\mathscr{L}_{\text{content}}(\vec{p}, \vec{x}, \ell) = \sum_{i,j} (F^\ell_{i,j} - P^\ell_{i,j})^2 \tag{1}$$

If we use ReLU after each ConvLayer, the derivative of the content loss is:

$$\frac{\partial \mathscr{L}_{\text{content}}}{\partial F^\ell_{i,j}} = \begin{cases} 2(F^\ell_{i,j} - P^\ell_{i,j})^2 & \text{, if } F^\ell_{i,j} \geq 0 \\ 0 & \text{, otherwise} \end{cases}$$

With this, we can compute $\nabla_{\vec{x}} \mathscr{L}_{\text{content}}$ and update $\vec{x}$ using back propagation until a response is generated that is a close approximation to that of the original $\vec{p}$ in some layer ℓ.
Thus, we have described a way to generate an image $\vec{x}$ that replicates the content of our original content image $\vec{p}$. We will now explain how to replicate the desired style.

3.1.3 Style extraction

To extract style from an image,on top of each layer of the CNN, we compute the correlations between each activation map. This information is encoded in a matrix $G^\ell \in \mathbb{R}^{N_\ell \times N_\ell}$ called the Gram matrix. Formally, each entry of G^ℓ can be computed as:

$$G^\ell_{i,j} = \sum_k F^\ell_{i,k} F^\ell_{j,k} \tag{2}$$

which is the dot product of the vectorized version of each activation map.

Similarly as with content extraction, lower level layers reveal local structures of style whereas higher level layers reveal a more general sense of styling.

3.1.4 Generation of image matching input style

Same as we did for content, we start by generating a random white noise image $\vec{x}$. Let then A^ℓ and G^ℓ be the Gram matrices after layer ℓ of $\vec{a}$ and $\vec{x}$ respectively, where $\vec{a}$ is the input styling image we want to copy.

We can thus compute the contribution of layer ℓ to the total loss as:

$$E^\ell = \sum_{i,j} (G^\ell_{i,j} - A^\ell_{i,j})^2 \tag{3}$$

By considering several layers for styling, we can write the overall style loss as:

$$\mathscr{L}_{\text{style}}(\vec{a}, \vec{x}, \ell) = \sum_{\ell} w_\ell E^\ell \tag{4}$$

We can again do backpropagation to optimize $\vec{x}$ so that its style is very close to that of the original styling image.

3.2. Generation of image mixing content and style

Now that we have a way to generate an image that separately matches content of one image and style of another image, we want to bring these two components together to create an output image that combines both content and style. We achieve this by jointly minimizing the distance of the random image $\vec{x}$ to the styling image $\vec{a}$ and the content image $\vec{p}$.

Finally, in addition to content and style loss, we include an additional total variation loss, also called TV loss. This loss allows to denoise the outcome image by ensuring a smooth variation between adjacent pixels. The expression for the TV loss is:

$$\mathscr{L}_{\text{tv}}(\vec{x}) = \sum_{c=1}^{3} \sum_{i=1}^{H-1} \sum_{j=1}^{W-1} (x_{i,j+1,c} - x_{i,j,c})^2 + (x_{i+1,j,c} - x_{i,j,c})^2$$

Thus, the overall loss we aim to minimize is:

$$\mathscr{L}_{\text{total}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathscr{L}_{\text{c}}(\vec{p}, \vec{x}) + \mathscr{L}_{\text{s}}(\vec{a}, \vec{x}) + \gamma \mathscr{L}_{\text{tv}}(\vec{x}) \tag{5}$$

where α and γ are tunable hyperparameters.

Through back propagation, after several iterations this yields an output image $\vec{x}$ that combines content from $\vec{p}$ and style from $\vec{a}$[1].

The following image summarizes how to generate an image mixing content and style.

3.3. Deep photo style transfer

Deep photo style transfer builds on Neural Style Transfer while in addition it attempts to preserve the photorealism of images and generalize to a variety of content and style images.

The authors from [10] propose a method that improves the Style Transfer algorithm via two main ideas. First, they include a photorealism regularization term in the objective function. Second, they perform semantic segmentation of the inputs to avoid style spilling and content-mismatch problems. This way, the sky from the input image, for instance, will be styled based on the sky in the styling image, without any noise from other parts of the styling image. Throughout this paper we will focus on the implementation of style transfer with segmented images to avoid style spillovers.

[1] In [3] this is done by choosing content representation from one layer and style representation from any number of layers.

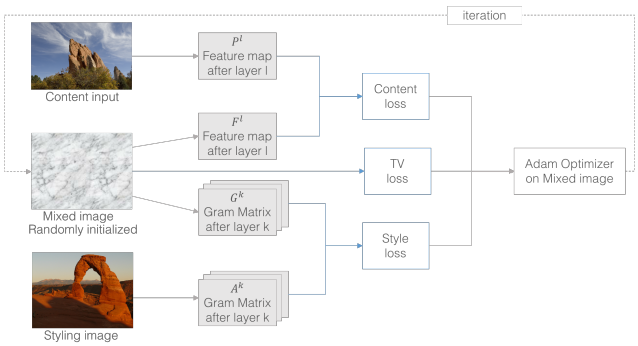

Figure 2. Image generation mixing content and style

We opted for using pre-segmented images so that we could focus our efforts on optimizing the style transfer part of the problem. The authors from [10] have generously shared a set of 120 segmented content and style images on their Github repository [7].

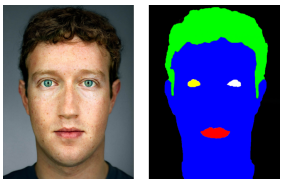

Figure 3. Example of segmented image

Formally, to ensure that style is transferred to related content, we generated a mask based on the segmented version of an image. Let k be the number of segments in an image of height H and width W. Then we generate k masks of height H and width W where the i-th mask contains the value `True` in every pixel corresponding to the i-th segment of the image and `False` everywhere else. This way, we have generated a mask M of shape (H, W, k). Recall from Section 3.1.4 that in order to generate an image that matches the styling image we obtain Gram matrices from the style representation of different layers ℓ. Call J^ℓ such feature representations. By the means of `imresize`, we resize the mask M to match the shape of the style representation J^ℓ. Finally, each channel of J^ℓ is multiplied by each one of the mask layers. This way, we have k

'masked' versions of the feature representation J^ℓ, which will be used to compute k segment-wise Gram Matrices G_k^ℓ and A_k^ℓ. for the style and mixed image respectively

This is done with the styling image as well as the mixed image (for the mixed image, the mask from the content image is used) and then the style loss is computed on the resulting Gram matrices.

The style loss is redefined as follows:

$$\mathscr{L}_{\text{style}}^{+} = \sum_{k \in segments} \sum_{l \in layers} \sum_{i,j} (G_k^\ell - A_k^\ell)_{ij}^2 \quad (6)$$

Therefore, the total loss function in Deep photo style transfer can then be written as:

$$\mathscr{L}_{\text{tot}}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathscr{L}_{\text{c}}(\vec{p}, \vec{x}) + \mathscr{L}_{\text{s}}^{+}(\vec{a}, \vec{x}) + \gamma \mathscr{L}_{\text{tv}}(\vec{x}) \quad (7)$$

The following image summarizes how we used masked images to transfer style locally from image to image.

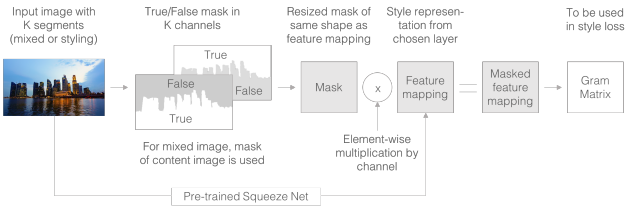

Figure 4. Using segmented images for localized style transfer

4. Experiments and Results

This section outlines the details of the networks and datasets we used as well as the experiments run and corresponding results throughout the course of the project.

4.1. Evaluation

There is no metric of objective evaluation for how photo realistic an image is, thus we will subjectively evaluate results. Using a random sample of external observers, we will show output for a variety of hyper-parameters and ask observers to rate the output.

4.2. Datasets

4.2.1 Images

The methodology we have described could potentially use any combination of content and style based photographs. In this paper, we will use photos provided by Luan et al. to ensure we are achieving at least an existing level of success. This database of photos consists of 60 content and 60 style photos. These photos cover a broad range of content and style; content consists of landscapes, objects, rooms, people and style contains a variety of weather, lighting, etc. For the most part, we used content and styling images with similar content so that the localized style transfer made sense in the context of both images.

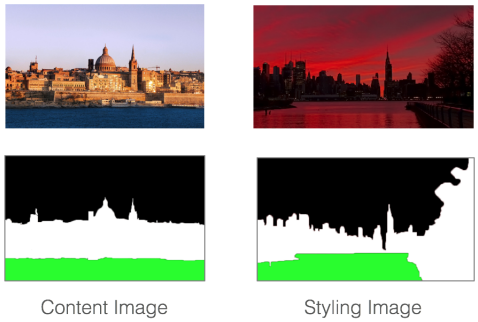

Figure 5. Content and style images were chosen so that they had similar sub-segments

4.2.2 Feature Extractor

In the paper, a pre-trained VGG-19 net is used as the feature extractor for both the content and style layers. VGGNets [11] are deep neural networks which use small filters - only 3x3 convolutional layers and 2x2 max pooling layers. These deep networks have more non-linearities and use less parameters as compared to networks with similar depth but larger filters.

In an attempt to recreate [10], we initially attempted to build our model using such network [8], however, VGGNets require a large amount of memory. For that reason, we experimented using a pre-trained Squeezenet model [1] [5] for feature extraction. Squeezenet is a recent network which uses significantly less parameters while maintaining accuracy. It does so by using even smaller filters and 1x1 convolutional layers as compared to VGGNet's 3x3 convolutional layers. Squeezenet calls each layer of the neural network a "fire module". Each "fire module" squeezes the network using 1x1 conv filters and then expands the network using a combination of 1x1 and 3x3 conv filters.

Both models have been trained on Imagenet. Squeezenet uses 421,098 parameters. In contrast, VGG-19 has over 140 million parameters. We hypothesized that no information would be lost by using a more efficient, shallow network. Upon investigation, we found that traditional neural style transfer using Squeezenet has been performed [4] successfully.

After several attempts with VGG networks, we built our final implementation on top of the Squeezenet model. Many of the images that were initially tuned using this

model can be found in [6].

4.3. Preliminary attempts

4.3.1 Photorealism loss

In [10], the authors add an additional loss to the objective function to preserve edges and enforce photorealism. Photorealism loss is defined as:

$$\mathscr{L}_{\mathrm{m}} = \sum_{c=1}^{3} \mathrm{V}_{\mathrm{c}}[O]^{\mathrm{T}} \mathscr{M}_{\mathrm{I}} \mathrm{V}_{\mathrm{c}}[O] \tag{8}$$

where $\mathrm{V}_{\mathrm{c}}[O]$ is a vectorized version of the current output image and $\mathscr{M}_{\mathrm{I}}$ is the Matting Laplacian matrix of the original content image [9].

A Matting Laplacian matrix is an NxN (where N = number of pixels in the content image) representation of natural image matting, which separates the foreground from the background of an image. This loss function ensures that RGB colors are being transfered linearly by pixel from the input to the output image.

Following the methods outlined in [9], we successfully created a Matting Laplacian matrix for the input content images as a sparse matrix in Python language. However, we ran into issues when converting this sparse matrix into a tensor. When this matrix was converted to a dense tensor, we quickly ran into storage issues. Thus, we were restricted in the number of pixels we could use to represent a photo. Capped out around l20 pixels, quality of output image was severely compromised. Our next approach was to convert to sparse matrix directly to a sparse tensor. However, when calculating loss we had issues running mathematic operations between a dense and sparse tensor.

Due to time constraints, we were unable to implement photorealism loss. In the future, we hope to do so to continually improve upon our methods.

4.3.2 Hyperparameter tuning

As we mentioned in the Approach section, several parameters had to be tuned in order to make sure we obtained the optimal result. While this model has many hyperparameters, we will mainly focus on the following:

- α: Weight of content loss
- $\vec{w}$: Vector of weights for each styling layer
- γ: Weight of total variation loss
- $\vec{l}$: Set of layers to be used for content

Surprisingly, the weight of each loss did not have as much impact as other hyperparameters. Our initial belief was that various loss functions in the final objective function would control transfer and distortion levels of the image.

The most important hyperparameters in this method were the layers chosen for the style loss, and their respective weights. We learned that by choosing and giving a high weight to earlier style layers, the outcome has a style that visually is closer to that of the styling image. In contrast, later layers transfer a very distorted and almost psychedelic style.

After review of several images, we learned that parameters must be tuned specifically for each pair of content-style images to be combined. That is, one combination of hyperparameters that yielded good results for one set of images is very likely to give poor results with a different pair of images. For the images we present in the Results section, we specified two to four styling layers, with weights going from `1e8` for earlier layer to `1e-2` for later layer. In general, the weights decreased as the layer number increased.

While style layers proved to be a crucial selection for transfer, content layers were relatively stable across photos. As long as we chose early layers for content, changing the specific layer did not entail significant differences in the results. This is likely because more definitive and precise pixels are needed to maintain a photorealistic output. For the images shown in results section, the third content layer was chosen.

4.4. Final results

Figure 6 shows the results on a sample of images we trained. The third column shows the result after transferring style using traditional style transfer, while the last column shows the results after using the improvements discussed throughout this paper. From these images, it is clear that several of the issues encountered with traditional style transfer are solved when using segmentation.

In the first row, we can see that using traditional style transfer results in a distorted outcome. While the result captures the overall style of the styling image, we are losing any type of localized style. We see the vibrant red sunset across most of the image with black at the very bottom. Traditional methods lack the use of content information to transfer sunset effects only to the sky, and so on. This problem is solved, as seen in the last column, when using segmentation.

In the second and third row, traditional style transfer is causing a spillover effect. That is, the sky is slightly styled

with colors from the main content, in this case the rock or mountain. Using segmentation limits the spillover, forcing sky to be re-styled solely on the sky of the styling image. This same reasoning holds for the ground of these images.

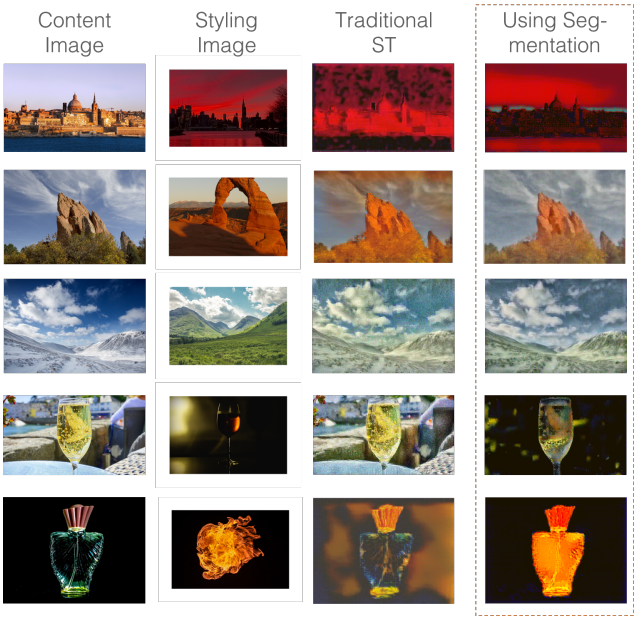

Figure 6. Results

Some results also show less successful examples of style transfer. A current limitation is that the number of segments in the content image must be equal to the number of segments in the styling image. Because of this, it is currently not possible to transfer style from an image with a sky and an object to an image with a sky, object, and ground.

Further, through experimentation we learned that style is transferred best between like images. As can be seen in Figure 7, results were not as satisfying when content and style images did not have similar semantic segmentation. That is, transferring style from an image of an apple to an image of a room shows poor results. In our example, we see that style from a desert landscape does not transfer well to an inanimate object, like a perfume bottle. It looks like these settings 'confuse' the model and don't operate as well. Style is best transferred between two similar images, like two mountains, taken under different effects.

5. Conclusion and Next Steps

In this paper, we implemented an extension of neural style transfer to style transfer between two images. Through updating traditional style transfer by using semantic segmentation, we avoid style "spillover". A second improvement which could further improve current results is

Figure 7. Results from transferring style between images with different content

to add a photorealism loss to the objective function to help avoid distortion.

We achieved results that are roughly equivalent to those published in [10], however we are interested in further refining our methods. First, we aim to improve upon the tensorflow implementation of the photorealism loss. Next, hyperparameter tuning is currently done on a photo by photo basis. If some type of hyperparameter streamlining was created, style transfer effors would be much easier and more ubiquitous. Finally, no standard method of evaluation exists for style transfer. In the future, it would be useful to have such a metric to measure success and tune parameters more easily.

References

[1] C. 231n. Assignment 3. `http://cs231n.github.io/assignments2017/assignment3/`.

[2] L. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *CoRR*, abs/1606.00915, 2016.

[3] L. Gatys, A. Ecker, and M. Bethge. A neural algorithm of artistic style. *arXiv preprint arXiv:1508.06576*, 2015.

[4] P. Gonchar. neural-art-mini. `https://github.com/pavelgonchar/neural-art-mini`, 2016.

[5] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and <0.5mb model size. *arXiv:1602.07360*, 2016.

[6] J. Johnson. neural-style. `https://github.com/jcjohnson/neural-style`, 2015.

[7] F. Juan. deep-photo-styletransfer. `https://github.com/luanfujun/deep-photo-styletransfer/tree/master/examples`.

[8] T. Lee. Keras: Deep learning for python. `https://github.com/fchollet/keras/blob/master/keras/applications/vgg19.py`, 2017.

[9] A. Levin, D. Lischinski, and Y. Weiss. A closed form solution to natural image matting. *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 1 (CVPR06)*.

[10] F. Luan, S. Paris, E. Shechtman, and K. Bala. Deep photo style transfer. *arXiv preprint arXiv:1703.07511*, 2017.

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