Show, Divide and Neural: Weighted Style Transfer

Ethan Chan
Stanford University
ethancys@stanford.edu

Rishabh Bhargava
Stanford University
rish93@stanford.edu

Abstract

The neural style algorithm has been very successful in obtaining the style of famous pieces of art. There have been attempts to overlay this style on images, to varying degrees of success. We believe that we can improve the vanilla Neural Style by performing image segmentation, and applying neural style only on the background, or only on our main object or both. This leads to our object remaining distinct, while the new image still manages to take an artsy look.

1. Introduction

1.1. Motivation

Since the release of the Neural Style paper [4], there have been a number of applications that used it to apply stylistic transfers from one image to another. Despite it achieving great results for certain stylistic transfers, there is a limitation to the current method as the style transfer can only be applied to the image as a whole. Looking at Figure 1, when Sunset in Venice's (Monet) style is transferred onto the Emma Watson's photograph, the image does have the desired style effects, but in the process, the focus of the image, Emma Watson, is lost and now she looks "blended" as part of the image.

Figure 1. Monet's Sunset in Venice transfer to Emma Watson

We hope to address this limitation of style transfer currently that only applies across the whole image evenly, by coming up with a way to distinguish between the "important" object and the "non-important" background in the image. We then apply different stylistic transfers on the different parts to maintain the focus of the original image; in this case, Emma Watson will be less "blended" into the background of the image.

The way our system will work is to begin with a content image (in our example above it was an image of Emma Watson) and a style image (a Monet painting as seen above). We will then go on to find the main object in the image, and create a mask which marks the object and background differently. We then perform Neural Style on just the object, or just on the background, or different neural styles on the object and background. These images produced will be our final result.

2. Review of Previous Work

Our system involves two parts: segmentation on our content image to give us a mask, followed by Neural Style. We will review both sections separately.

2.1. Segmentation

We tried off-the-shelf graph based segmentation algorithms [3] to locate important regions of the image, but the results were as seen in Figure 2 on such a simple photograph with the bottle in the foreground and a nearly plain white background. Such a segmentation does not take into account different parts of an image that correspond to the same object (or the background), and thus are not useful to us.

Figure 2. Original image, and graph segmentation of bottle

We also considered an alternative approach using Saliency Maps [7] as an indicator for which pixels are the largest contributors to the class that the photo belongs to. However, in this case, we often get a very noisy mask for the region of the image which belongs to the object, and

this could lead to issues while performing neural style as seen in Figure 3.

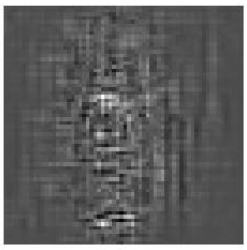

Figure 3. Original image, and graph segmentation of bottle

Finally, we looked at methods that performed semantic segmentation. There has been work done on this using Fully Convolutional Networks [5], different regions and parts of the image [1] and by performing learnable upsampling [6]. There are a number of good options to perform semantic segmentation, but we chose to use the method of interpreting Conditional Random Fields as Recurrent Neural Networks [9].

2.2. Neural Style

Since the original paper on Neural style was published, there have been a number of implementations for it, but all have been applied on the whole image (see Figure **??**), and none have treated different pixels or regions differently. Since this work is very recent, a number of new applications are being discovered. A very recent paper explores the concept of Neural Doodle [2] where we get artistic results from MS Paint style doodles.

Figure 4. Original Image, Vanilla Neural Style Image

3. Methodology

This approach to our problem consists of many subsystems. We will discuss all of them in the order that computation happens. We begin with a content image I.

3.1. Semantic Segmentation

We perform Semantic Segmentation using code from an implementation[1] of CRF-RNN [9]. This model was trained on the PASCAL VOC 2012 dataset, and is trained to segment 20 classes. We have important classes contained in this, such as 'Person', 'Cat', 'Dog' etc. Using this code, we obtain a mask M for image I, examples of which are given in Figure 5. These masks will be used in later sections.

Figure 5. Masks obtained by performing Segmentation on our green bottle image from Figure 2 and on the Emma Watson image from Figure 10

3.2. Vanilla Neural Style

This section gives a background of what Neural Style is, and how it works. We have a content image I and a style image S. The objective is to build a new image X, such that X has the content of I and the stylistic features of S.

We use a pre-trained deep convolutional network. It needs to deep enough, such that relevant features (as described later) can be extracted from multiple layers to give varying global sized features. In our case, we use a 19 layer VGG model [8] trained on ImageNet. We now use this model to obtain features for content and style.

Content features are obtained by doing a forward pass using the model, and stopping at a particular layer, and extracting the features at that layer. In the original paper, they suggested using the features from just the 'conv4_2' layer, since those seem to have enough information about the global content of I. Let's represent these features as i_C.

For the style features, we use a number of layers, 'conv1_1', 'conv2_1', 'conv3_1', 'conv4_1' and 'conv5_1'. This is because the style and texture found at different scales can lead to strikingly different results. Once we extract the activation volumes at a particular style layer for S, we compute the Gram matrix, which is a representation of the style features at that scale for S. Let's assume that we have n filters at a larticular layer l. Now we can unroll an activation volume of dimensions $n \times w \times h$ into a matrix of $n \times wh$, and lets's call this matrix $V \in \mathbb{R}^{n \times wh}$. The Gram matrix $\mathbb{G}^l \in \mathbb{R}^{n \times n}$ is found by performing VV^T. Gram matrices

[1] https://github.com/torrvision/crfasrnn

found across all the style layers give us all the style features. Let's represent these features as s_S.

Now, to produce the desired image X, we begin with an image that is essentially white noise. We run it through the VGG 19 model, and compute the content features for X as x_C in a similar fashion as we did for the content image I. We can also find the style features x_S. Now, we define the content loss function, an L2 loss between x_C and i_C, as $\mathcal{L}_{content}(x_C, i_C)$. Similarly, the style loss is an L2 loss between x_S and s_S, and is given by $\mathcal{L}_{style}(x_S, s_S)$. Thus, the final loss is given by

$$\mathcal{L}_{\text{total}}(I, X, S) = \alpha\mathcal{L}_{content}(x_C, i_C) + \beta\mathcal{L}_{style}(x_S, s_S) \quad (1)$$

We also have a total variational (TV) loss, which takes into account the local gradient differences in the image X we are building, and helps smoothen the final result. This loss isn't mentioned in the result above.

We have consistently used two style images in this work, Starry Night by Van Gogh (as seen in Figure 6) and Self Portrait By Picasso (as seen in Figure 7) for our style transfer. This is because visually, the colors of the two pieces of art are very contrasting, with the Starry Night being mostly blue with some yellow, and Picasso having primarily brown and orange hues. In addition, they have differing patterns and textures where Starry Night has smooth fine paint brush strokes while the Picasso painting has a sharper and more jagged style.

We have also used an existing repository[2] written in Tensorflow for Vanilla Neural Style, whih we extended according to the following sections.

Figure 6. Starry Night Painting with Vanilla Style Transfer

Figure 7. Picasso Self Portrait Painting with Vanilla Style Transfer

[2]https://github.com/anishathalye/neural-style

3.3. Neural Style with Capped Gradients

This is a very minor modification of the Vanilla Neural Style. Instead of just one pass through the CNN in every iteration, we do two. We first just compute content and TV loss, and apply the gradients found on the entire image. In the next pass, we compute the style loss, and the style gradients obtained are only propagated to the desired segments of the image. This is done using our mask as a filter. For example, if we wanted neural style done only on Emma Watson in our reference image, all the style gradients for the background would be made 0.

3.4. Neural Style with Gram Matrix manipulation

In this method, we made changes to the Gram matrix computation for the image X that we are creating. Computing i_C, s_S and x_C remains unchanged in this case. Let's consider the computation of the Gram matrix at a layer l.

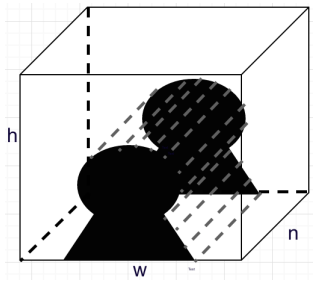

Figure 8. Masking of Activation Volume

At a layer l for the image X we obtain the activation volume with n filters, w width and h height $A \in \mathbb{R}^{n \times w \times h}$. Before we unroll this matrix to perform the matrix multiplication, we use our mask $M \in \mathbb{R}^{w \times h}$ to do an element by element multiplication with the activation of every filter. If M had 0s for the region that was the background, we would now have a 0 throughout the volume that corresponds to that region. This is illustrated in Figure 8. Having done this multiplication with the mask, we can now go on to unroll the volume, and compute Gram matrices like we did earlier.

One thing to notice is that the mask image is of the same size, whereas w and h chnage from style layer to style layer as we move up the network. This is because of pooling layers in the VGG net. To counter this, we can apply the same Max-pooling layer of size 2 and stride 2 to our mask, and we get the mask for the next layer.

The entire architecture of our system is depicted in Figure 9.

4. Dataset and Features

4.1. Dataset

The Pascal VOC and Pascal Context datasets were used to train CRF-RNN framework that was used for the seman-

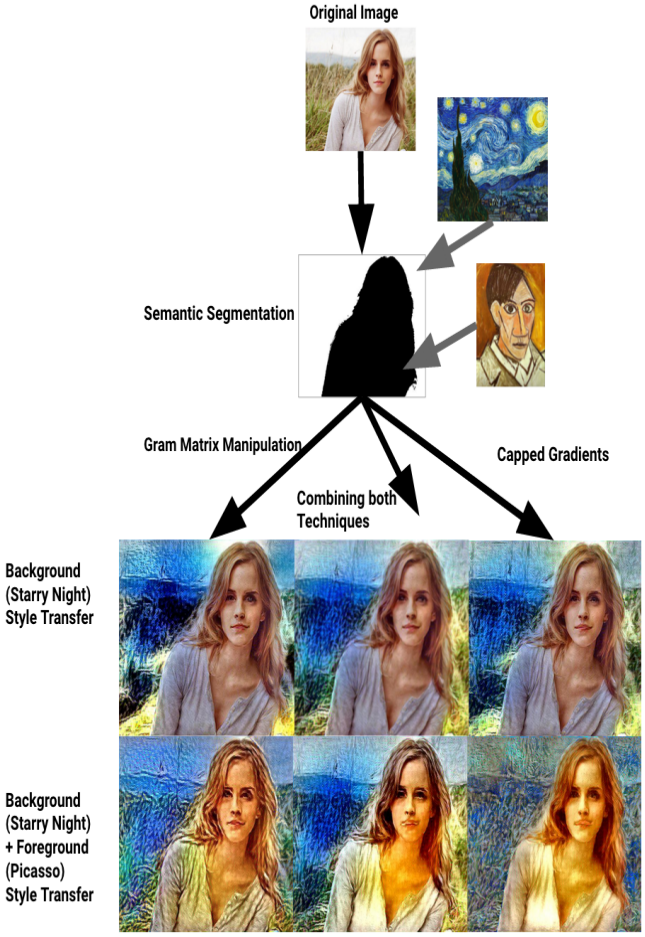

Figure 9. Architecture Layout

tic segmentation. We used a VGGNet 19 layers in the implementation of Neural Style algorithm. Since we do not do any training, we do not have an explicit training dataset. We chose this image Figure 10 as our input image to run as it had a clear foreground and somewhat noisy background with the grass that made it a good candidate to highlight the selective neural style transfer.

5. Results and Discussion

The only way to evaluate our style transfer quantitatively is to look at the style loss after the image has been produced. However, we are actually defining the loss function to reduce this style loss, as a result, it is circular to quantify our results by using the same loss function that we are optimizing towards. As a result, our way of evaluation will be to visually inspect if the stylized images produced achieve the desired goal where the background and object have different style transfers onto them. Another point to mention is that we haven't explicitly mentioned different hypeparameters

Figure 10. Original Emma Watson Image

involved in the creation of any our images. This is because we were focusing on making our methods work, and we found different hyperparameters working well for different methods. This just means that better hyperparameters can be found with a more extensive search.

5.1. Neural Style with Capped Gradients

Looking at Figure 11, where we only had minimized the style loss function of only the background pixels, while applying content loss to the all of the pixels, we can see that the background has a Starry Night filter applied on the background and not on the person in the picture. One very interesting observation noticed is that the background does not match exactly the vanilla neural style Starry Night styled image that was produced earlier in Figure 10 where the "paintbrush" strokes were more apparent. We hypothesise that this might be because in order to build those large scale stylistic features, we need the loss throughout the image X being reduced. If that doesn't happen, then our optimiser keeps trying to push those gradients to the masked parts of the image, but they don't cause an update. This means that the image gets to convergence quickly (stylistically speaking). [h] Looking at Figure 12, where we chose to apply Starry Night on the background and Picasso's Self Portrait Style on the person, we can see that the background actually becomes somewhat mixture of Starry night and the original content image, while the person is now has some sort of light Picasso filter on her that accentuates orange hues as in the original Picasso painting. As with the single style transfer, the "paintbrush" strokes are now less apparent then a vanilla style transfer image.

5.2. Neural Style with Gram Matrix manipulation

Let's now consider the image produced using Gram matrix manipulation with Starry Night as the background (Figure 13). In this case, we have only used 'Conv1_1' as the style layer, rather than all 5 layers in the previous section. This is because this gave us the cleanest results. We can

Figure 11. Capped Gradients: Starry Night Style Transfer on Background

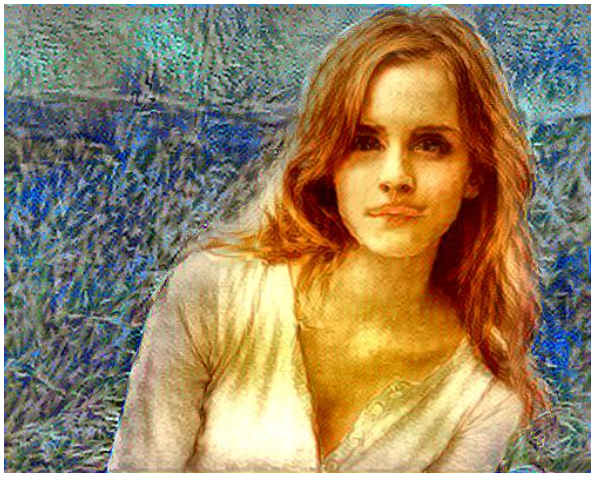

Figure 12. Capped Gradients: Starry Night Style Transfer on Background, with Picasso Style Transfer on Foreground

see that Emma Watson has been nearly perfectly replicated (content loss has been reduced a lot), while her background has transformed completely into a Starry Night landscape. However, the features of the background are very small scale, as is to be expected from just the 'conv1_1' features.

Figure 14 shows Neural Style applied to our image such that we have Starry Night in the background, and Picasso's Self Portrait style done to Emma Watson. In this case, we used 'conv1_1' and 'conv2_1' as our style layers. This can be seen as the size of the features is slightly larger than the previous case. This leads to a unique look, as we see much sharper Picasso strokes than we did in the case of Capped Gradients.

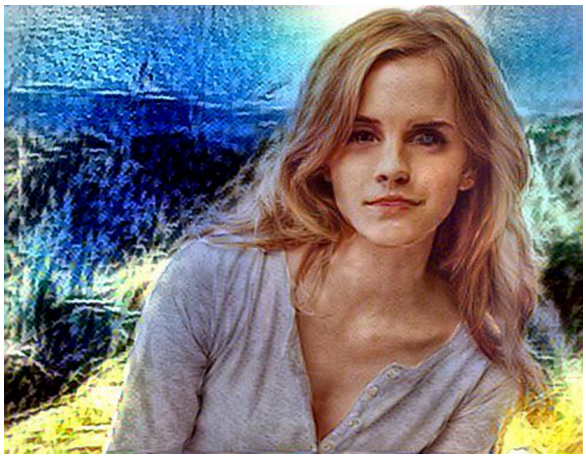

Figure 13. Gram Matrix manipulation: Starry Night Style Transfer on Background

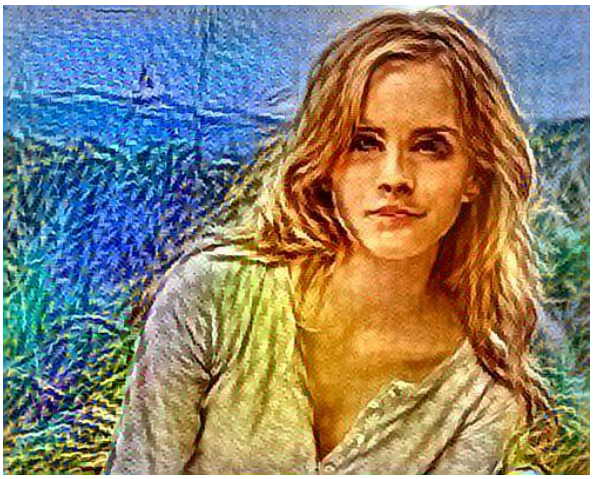

Figure 14. Gram Matrix manipulation: Starry Night Style Transfer on Background, with Picasso Style Transfer on Foreground

5.3. Combined Gram Matrix and Capped Gradients

In this method, we combine the capping of gradients at the time of application of style gradients with the manipulation of the Gram matrices. In doing so, we achieved an image that is visibly more refined in both the background and foreground as can be seen in Figure 15.

5.4. Loss

We plotted the graph Figure 16 of the various losses (styles, content, TV) against iterations and we can clearly see that the content, style and total variation losses go down after each iteration. We also display how the as the various losses decrease, we can visually see from the image snapshots that the generated image is slowly resembling the content as well as style image. For example, at the start, style loss is very high, and at thus, the algorithm rushes to

add the correct style to the image. As the style loss goes down, it startes to compete with the content loss, and the content emerges in our picture.

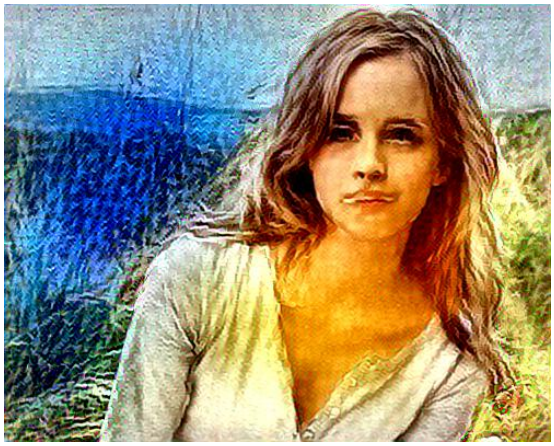

Figure 15. Combined Gram Matrix Manipulation and Capped Gradients

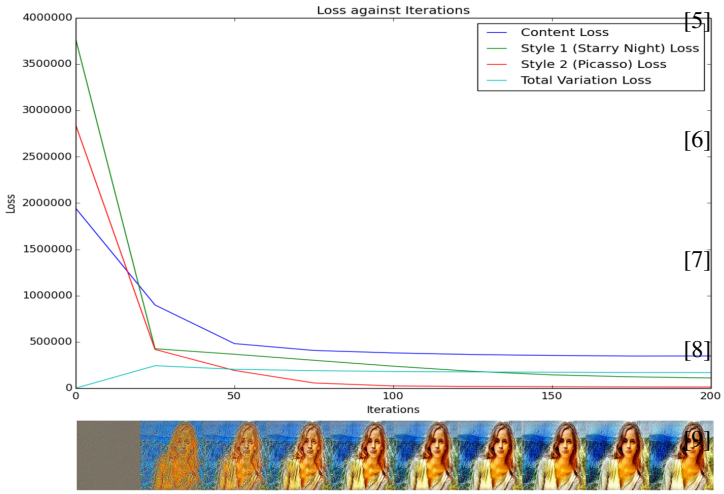

Figure 16. Loss against Iterations with Image Snapshots

6. Conclusions and Future Work

In this paper, we extended the Vanilla Neural style algorithm to be able to identify the main object in the image from the background, and apply Neural style only on the background, only on the object or different neural styles on either. This was achieved using two techniques: either by Capping Gradients or through Gram MAtrix Manipulations. Both resulted in good images, although they were noticebaly different with regards to certain stylistic elements. There are a number of avenues for future work. The semantic segmentation code can only perform segmentation for 20 labels. There exist many more types of objects in the real-world, and we want to be able to apply our neural filter to them too. We would also like to find better hyperparameters such that the images produced look nicer. There are boundary effects in the 2-style images that we would like to fix. One of the applications that we would like to see is the deployment of this algorithm as a phone app, but for that the time required to form the image would need to be reduced drastically.

References

[1] P. Arbeláez, B. Hariharan, C. Gu, S. Gupta, L. Bourdev, and J. Malik. Semantic segmentation using regions and parts. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3378–3385. IEEE, 2012.

[2] A. J. Champandard. Semantic style transfer and turning two-bit doodles into fine artwork. In *nucl.ai Conference 2016*, 2016.

[3] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, 2004.

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

[5] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3431–3440, 2015.

[6] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1520–1528, 2015.

[7] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

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

[9] S. Zheng, S. Jayasumana, B. Romera-Paredes, V. Vineet, Z. Su, D. Du, C. Huang, and P. H. Torr. Conditional random fields as recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1529–1537, 2015.