Neural Style Transfer

Vishal Batchu
IIIT-Hyderabad
vishal.batchu@students.iiit.ac.in

Ameya Prabhu
IIIT-Hyderabad
ameya.prabhu@research.iiit.ac.in

Sri Aurobindo M
IIIT-Hyderabad
s.munagala@research.iiit.ac.in

April 29, 2017

1 Abstract

The aim of the project is to transfer the style of a painting to any given image. Traditionally, the major problem is the lack of image representations that explicitly represent semantic information which allows us to separate content from image. The algorithm we implemented makes use of CNNs which are optimized for object detection.
Essentially given the input content and style images we would generate an image which captures the details and semantic meaning of the content image and it would have the texture and color details of the style image.
We then adapt this model to transfer styles from real images to sketches, using manual segmentation maps. Images have multiple objects at a time. The extra information provided by segmentation masks allow for more meaningful style transfer, as compared to global style transfer which would fail in this case.

2 Data-sets and models used

We used pre-trained VGG_L 19 layer network for our final results. We also tried using the AlexNet but on an average the results with VGG were better, so we used that for all our experiments further on.

3 Style Transfer Algorithm

Initially we add content loss modules and style loss modules at various points in the pre-trained network. The content loss module stores the expected output of the content image when passed through the network up till the selected layer. The style loss module stores the expected style, which is represented by gram matrix of the style image when passed through the network up till the selected layer.

We start by performing a feed forward operation on a white noise image. We then compute the losses at each of the mentioned layers and calculate the gradients based on that and add them to the gradients from the layers ahead in back propagation. This would allow us to back-propagate the gradients all the way to the input image where we modify the input image based on these gradients to reduce the overall error. This is slightly different from training networks since we don't actually modify any of the network weights here but modify the input image (white noise initially) instead. This is basically a deconvolution operation.

Since we minimize this style loss and content loss simultaneously, the (initially)white noise starts to capture semantics of the content input image to minimize content loss and also the texture of the style input image to minimize the style loss. This results in a representation of the style and content in the

Figure 1: Input Content

Figure 2: Input Style

Figure 3: Output

same image that we generate.

4 Modules

4.1 Content Loss

Let $\vec{p}$ and $\vec{x}$ be the original image and the image that is generated and P_l *and* F_l their feature representations. Then, Content loss is given by

$$L_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2$$

, where $F^l_{i,j}$ = activation of i^{th} filter at position j in layer l, where each filter is of size height*width of the image.

The gradient for the content loss is given by

$$\frac{\partial L_{content}}{\partial F^l_{i,j}} = \begin{cases} (F^l - P^l)_{i,j} & if\ F^l_{i,j} > 0, \\ 0 & if\ F^l_{i,j} < 0 \end{cases}$$

4.2 Style Loss

Feature correlations are given by gram matrix Gl. Gram matrix is essentially a kernel that captures the style of an input image very well. Gram matrix can be obtained as follows,

$$G^l_{i,j} = \sum_k F^l_{i,k} F^l_{j,k}$$

This gram matrix was implemented as a small network on which we could perform forward and backward propagation to get the results required. Given an input volume we would get a 2D gram matrix as outputs and given the gradients at the matrix we can get back the gradients at the input volume as well. So, the style loss is given by,

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2$$

And the gradient for the loss is given by

$$\frac{\partial E_l}{\partial F^l_{i,j}} = \begin{cases} \frac{1}{N_l^2 M_l^2} ((F^l)^T (G^l - A^l))_{i,j} & if\ F^l_{i,j} > 0 \\ 0 & if\ F^l_{i,j} < 0 \end{cases}$$

4.3 Regularization Loss

Regularization loss is essentially a loss which stops the image from becoming completely white or completely black and tries to move the pixels towards the

mean values. This is essentially an L2 loss between the image pixels and the zero image pixels.

$$L_{regularize}(\vec{x}, l) = \frac{1}{2}\sum_{i,j}(F^l_{i,j})^2$$

4.4 Total Loss

Total loss is given by,

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha L_{content}(\vec{p}, \vec{x}) + \beta L_{style}(\vec{a}, \vec{x}) + \gamma L_{regula}$$

We vary these weights α, β *and* γ to get various desired outputs. We have also used a TV loss which would try to keep the image smooth so the outputs don't end up having too much variation between nearby pixels.

5 Style Transfer Results

Figure 4: Content

Figure 5: Style

Figure 6: Output

Figure 7: Content

Figure 8: Style

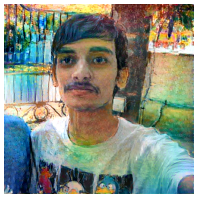


Figure 9: Output

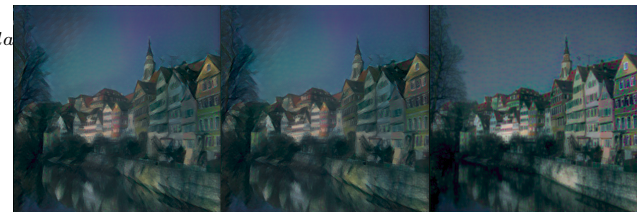
Figure 10: Content

Figure 11: Style

Figure 12: Output

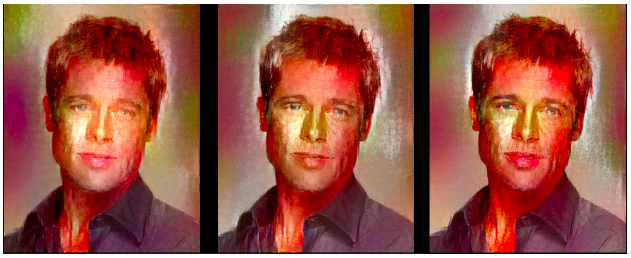
5.1 Comparisons

Selection of different layers to insert style loss modules



From left to right, we use relu1_1, relu2_1, relu3_1 and relu4_1 layers for adding style loss modules in the first image, in the second one we use relu2_1, relu3_1 and relu4_1 layers for adding style loss modules whereas in the third image we only use relu1_1 layer for adding the style loss module and we can clearly see that only the color is captured in this image and not much of the texture and strokes are captured here.

Variation of style weight β



From left to right, we vary the style weight as 0.3, 0.5 and 0.6

5.2 Failure Cases

As we can see here, the current method of style transfer has a global effect and it cannot be applied to

Figure 13: Content

Figure 14: Style

Figure 15: Output

cases where there are multiple objects present. To deal with this we introduce segmented style transfer.

6 Segmented Style Transfer

A local approach to style transfer unlike the standard where we are also presented with segment maps of the input style and content images which we use along with the style and content images to transfer style for each segment locally to the content image.

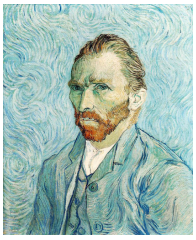


Figure 16: Content

Figure 17: Style

Figure 18: Output

Figure 19: Content

Figure 20: Style

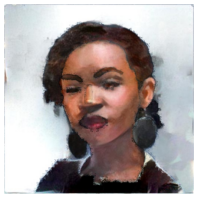


Figure 21: Output

The segment map consists of various color patches where each patch corresponds to a part of the image. Patches that have the same color in the content and style segment maps correspond to the parts of the image that have some similarity which we would like to use to perform the transfer. The style of each segment in the style image gets transferred to the corresponding segment in the content image.

Figure 22: Content

Figure 23: Segment map

Figure 24: Style

Figure 25: Image

This method allows us to transfer style locally between objects that are present at different parts of the image which can be very useful in a variety of cases. Images containing multiple objects etc don't do well with standard style transfer whereas they perform extremely well with segmented style transfer. We have applied segmented style transfer on faces to transfer the features of one person (style image) to another (content image). Performing this normally would miss most of the details and if the images are not perfectly aligned then they give terrible results.

This can also be applied to situations where there is no content image present and we have to generate a content image given the segment map using the style image. This is very useful for artists to quickly make paintings and edit them with ease. This method has

been explored by others in the past but the results are still far from perfection.

Figure 26: Style

Figure 27: Style mask

Figure 28: Content mask

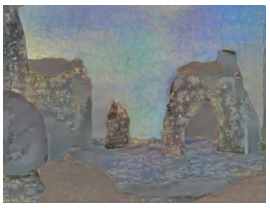


Figure 29: Result

Figure 30: Level 0

Figure 31: Level 1

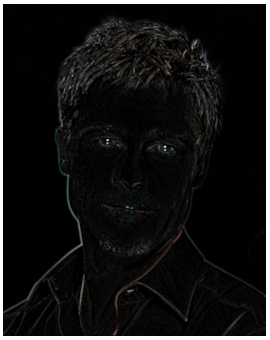


Figure 32: Level 2

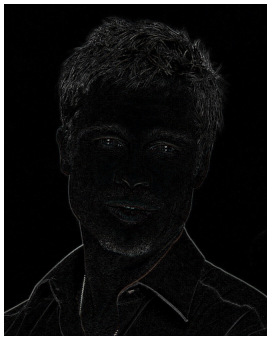


Figure 33: Level 3

7 Laplacian Pyramid based Style Transfer

Laplacian pyramids have been used extensively in image processing and are a very useful tool to analyze a given image at various scales in detail. In style transfer, our goal is to capture the texture of the style image and the semantic content of the content image which mainly consists of edges and finer details. If we observe the activations of the initial layers of a convolutional network when we pass a content image through, it often consists mainly of fine grain details initially which slowly turn into more coarse grained details as we progress through the network. The network requires a huge capacity with a lot of parameters just to get this done. To counter this issue and to use smaller networks which can perform faster and give us more control over the results, we propose the laplacian pyramid based approach.

We generate a laplacian pyramid of a certain size and then assign the highest level laplacian to a content loss layer after one of the initial convolution layers and move on progressively assigning layers to various content loss modules till the end of the network. This tries to ensure that the generated image captures the finer details in the initial layers and the broader, more generalized features in the latter layers. The style is captured from the style image at various stages in the network using a gram matrix like earlier. Once we have the generated image, we then perform ordinary style transfer over the original content image using the generated image as the style image which gives us better results in general when compared to standard style transfer.

Figure 34: Content

Figure 35: Style

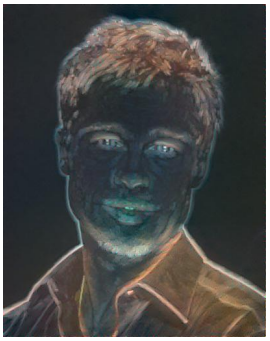


Figure 36: Temporary-Generated

Figure 37: Generated

8 Automatic Coloring of Sketches

First we split the foreground from the background using a binary segmentation on both the sketch and the SBIR retrieved image. We then match the regions based on region statistics such as area and perimeter. We then perform local style transfer from the foreground of the image to the foreground of the sketch and the background of the image to the background of the sketch. This would yield an output that is similar to the style image used. We would then overlay the sketch on top of the image and that would be our final automatic colored sketch. This may seem like a simple approach but there are a lot of issues that arise such as incorrect retrieval of style images to use etc.

Figure 38: Sketch

Figure 39: Image

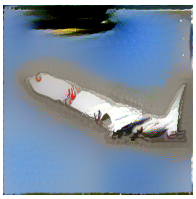


Figure 40: Output

References

[1] Gatys, L. A., Ecker, A. S., and Bethge, M. Image Style Transfer Using Convolutional Neural Networks. *In Proc. CVPR (2016).*

[2] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, Sept. 2014. arXiv: 1409.1556.

[3] Manuel Ruder, Alexey Dosovitskiy, Thomas Brox. Artistic style transfer for videos *German Conference on Pattern Recognition (GCPR).* LNCS 9796, pp. 26-36 (2016).

[4] Vincent Dumoulin, Jonathon Shlens, Manjunath Kudlur. A Learned Representation For Artistic Style *arXiv:1610.07629 [cs.CV]*.24 Oct 2016

[5] Leon A. Gatys, Alexander S. Ecker and, Matthias Bethge. Texture synthesis and the controlled generation of natural stimuli using convolutional neural networks *arXiv:1505.07376 [cs.CV]*.24 Oct 2016