Neural Style Transfer

Vishal Batchu, Raghuram Vadapalli, Ameya Prabhu

November 2016

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. We also trained an auto-encoder which can be used instead of CNN as they are also good at capturing essential details of images.
Essentially given the input content and style images we would generate an image which captures the details and syntactic meaning of the content image whereas it would have the texture and color details of the style image.

Figure 1: Input Content

Figure 2: Input Style

Figure 3: Output

2 Data-sets and models used

The networks we trained are on the data-sets CIFAR10 and CALTECH101. We used pre-trained $\mathrm{VGG}_L 19 layer network for our final results. We also tried using the AlexNet but the results with V$

3 Trained networks

3.1 CNN

We trained a CNN for classification on CIFAR10 data-set. We used Torch(Lua JIT) framework. The underlying assumption here is that CNNs trained for object detection capture semantic content at various levels depending on the depth of layer. This was of very small dimensions (32x32) so style transfer results were not that prevalent, to account for this issue, we moved on to CALTECH101 data-set and considered a subset of the classes(10 classes) which we used to train a classification network.

3.2 Auto-encoder

This is also trained on CIFAR10 data-set. The aim of auto-encoder is to learn a representation for a set of data, typically for the purpose of dimensionality reduction. Since essential details get captured in order to preserve the maximum content information, these are also a good choice.

4 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 style of the style input image to minimize the style loss. This results in representing the style and content in the same image.

5 Modules

5.1 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

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

5.2 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

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

5.3 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.

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

5.4 Total Loss

Total loss is given by,

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

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.

6 Results and Conclusions

6.1 Considering only content losses

Figure 4: Input Content

Figure 5: Output

6.2 Final Results

Figure 6: Content

Figure 7: Style

Figure 8: Output

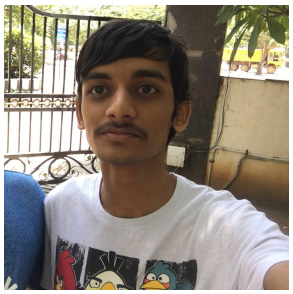


Figure 9: Content

Figure 10: Style

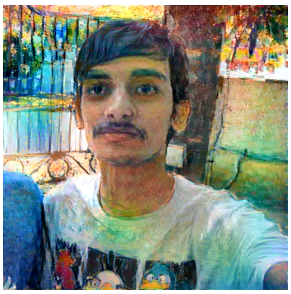


Figure 11: Output

Figure 12: Content

Figure 13: Style

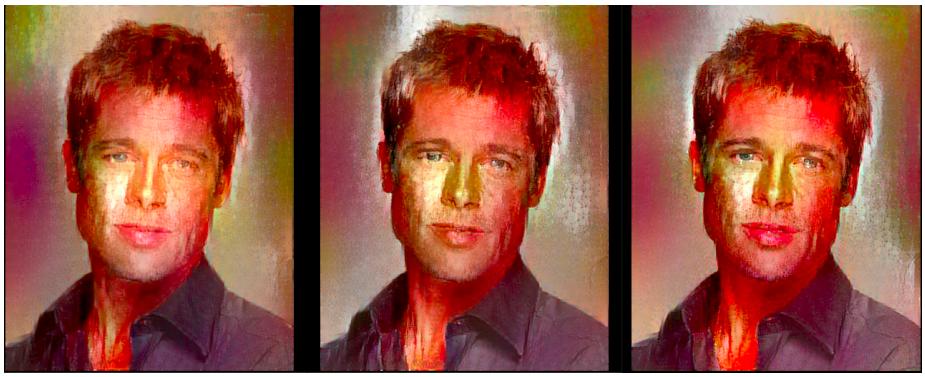
Figure 14: Output

6.3 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

6.4 Failure Cases

Figure 15: Content

Figure 16: Style

Figure 17: Output

The main reason for this failure is that a majority of the style image was mainly black and this was transferred completely to the content image where we just ended up getting black patches. A simple fix for this would be to reduce the style weight so that more of the content details are captured.

7 Research in this field

7.1 Extension to videos

Style Transfer has been extended to videos as well. Some papers directly apply this on videos frame by frame whereas others calculate differences between frames and apply it explicitly so it's a lot faster and doesn't feel too noisy.

7.2 Learning multiple styles

There is also research going on learning multiple styles. One could combine many different styles, exploring unique mixtures of well known artists to create an entirely unique pastiche.

7.3 Local style transfer

People have also tried to consider different regions in the content and style images and tried to transfer the style locally. This is quite useful when we want particular styles to be applied on particular regions of the content image.

8 Further Extensions

8.1 Automatic Coloring of Sketches

Figure 18: Sketch

Figure 19: Image

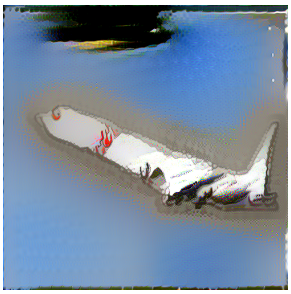


Figure 20: Output

First we split the foreground from the background using 2 level 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 similar to the 3rd image, we would then overlay the sketch on top of the image and that would be our final automatic colored sketch.

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