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.

2 Trained networks

2.1 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.

2.2 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.

3 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.

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 the 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.

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

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

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,

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} (G_{i,j}^l - A_{i,j}^l)^2$$

And the gradient for the loss is given by

$$\frac{\partial E_l}{\partial F_{i,j}^l} = \frac{1}{N_l^2 M_l^2} ((F^l)^T (G^l - A^l))_{i,j} if F_{i,j}^l > 00 if F_{i,j}^l < 0$$

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.

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

4.4 Total Loss

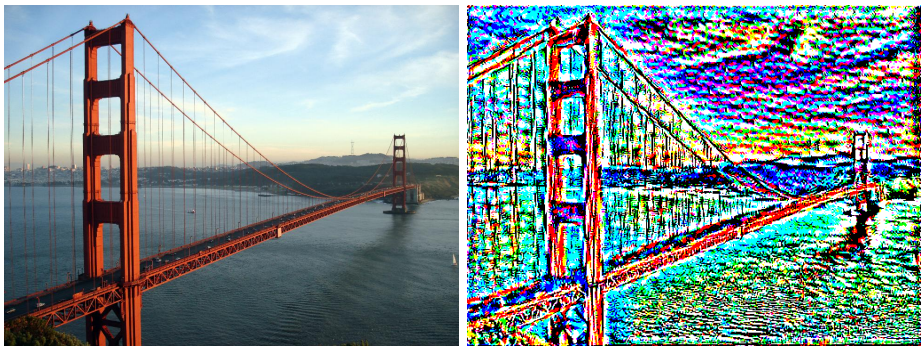
Total loss is given by,

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

We vary these weights $\alpha, \beta and \gamma to get various desired outputs.$

5 Results and Comparisons

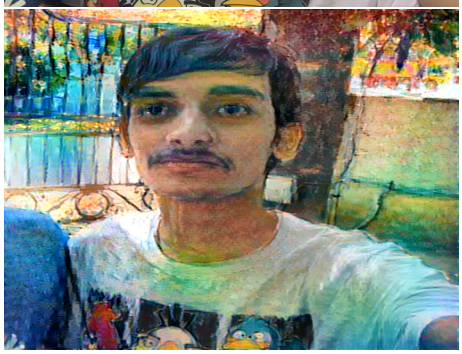
5.1 Considering only content losses



5.2 Final Results





5.3 Comparisons

5.4 Failure Cases

6 Further Extensions

6.1 Automatic Coloring of Sketches