Image Style Transfer

...

Neural Style

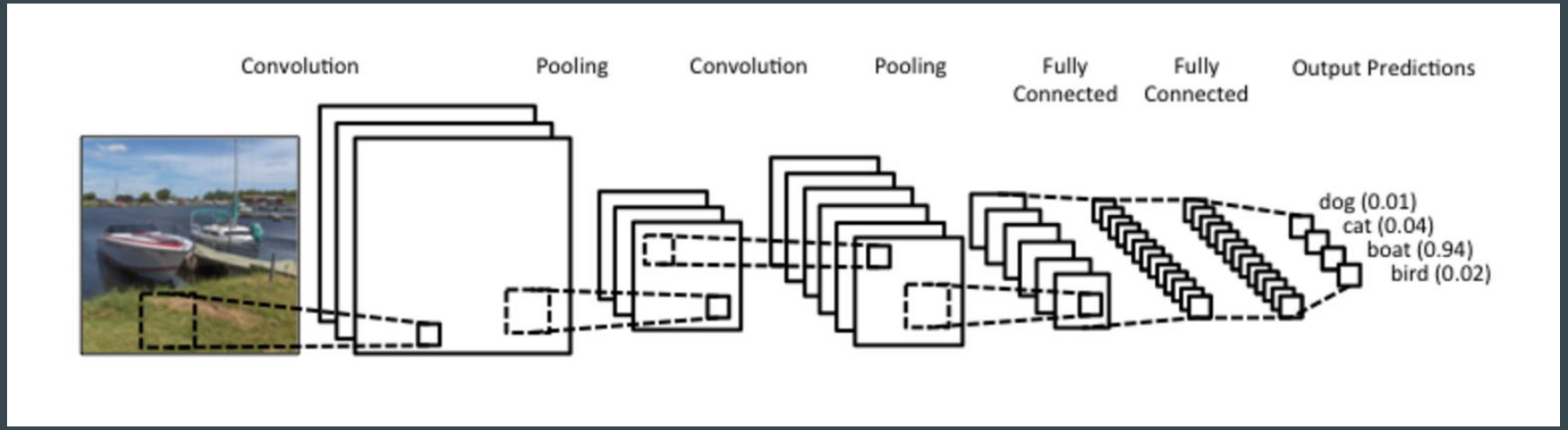
Overview

Combine style and content from different images to generate a new image.

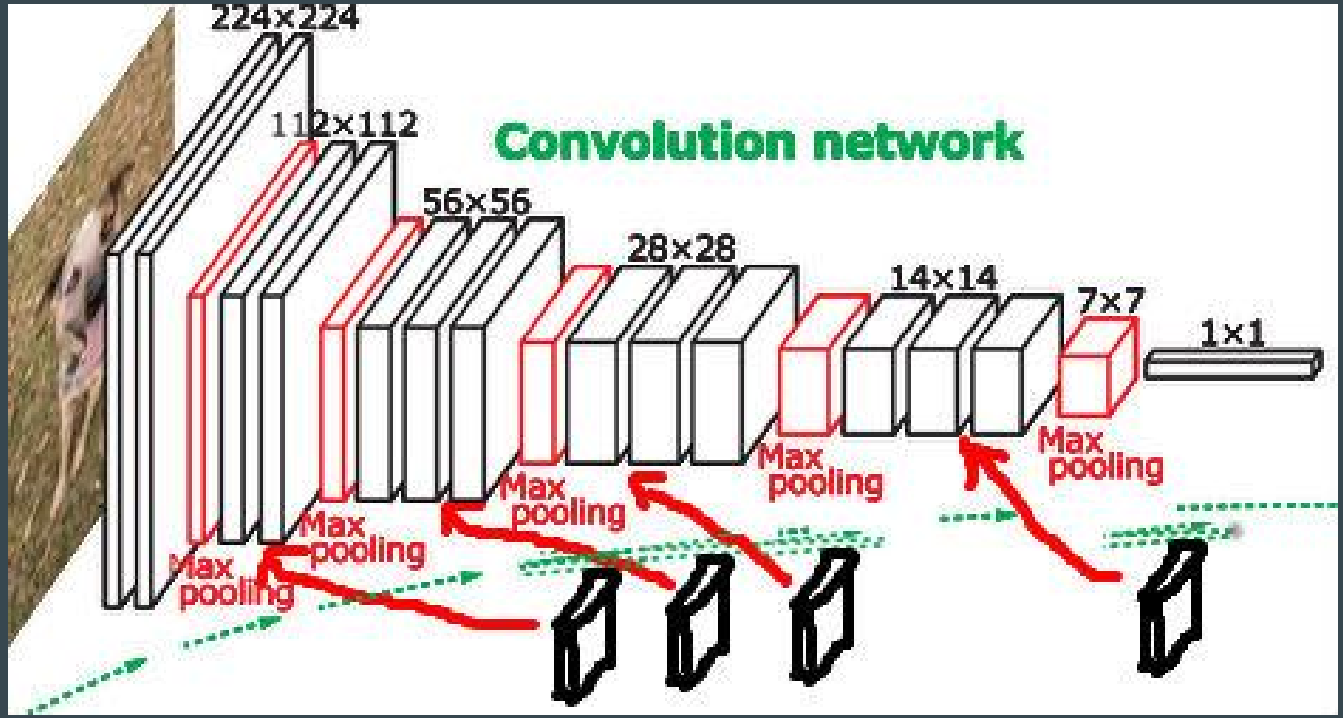
Introduction

- No traditional content representation techniques in the past.
- Use CNNs which are optimized to perform object detection and localization.
- Given a content and style image, generate an image which captures the texture from the style image and the semantic details from the content image.

Convolutional Neural Networks



Module Insertion in Network



VGG

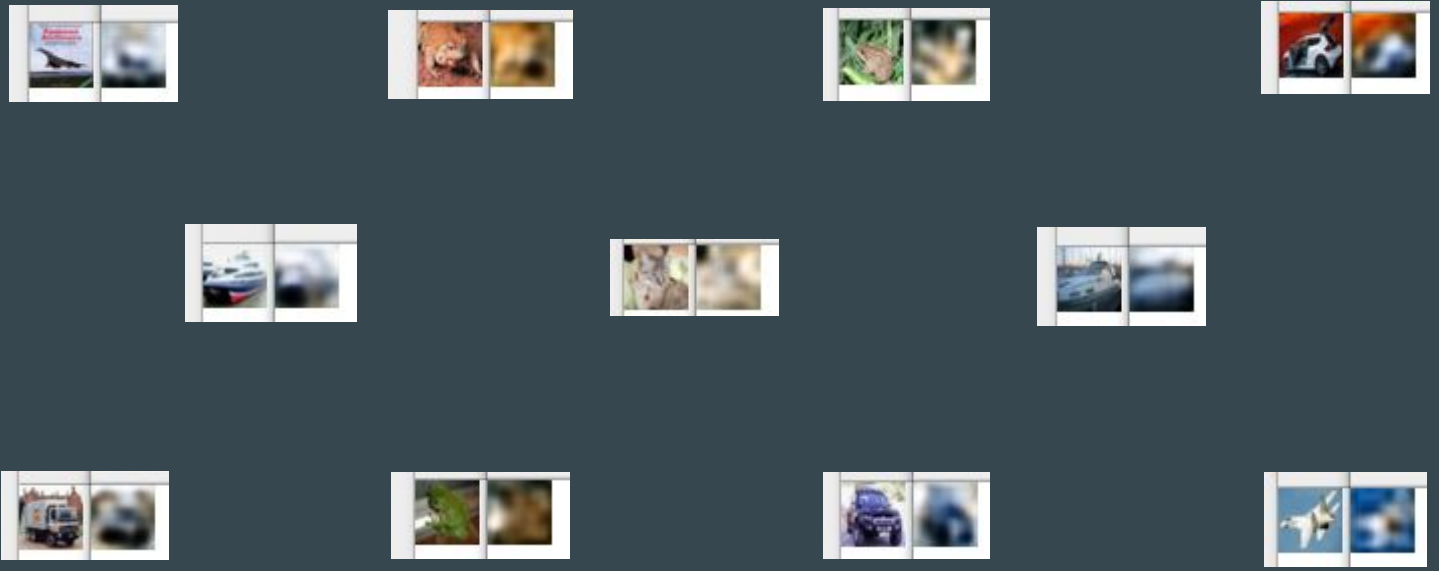
Implementation Overview

CNNs trained and existing models

- We have trained a small autoencoder and a classifier based on the CIFAR10 and CALTECH101 datasets.
- We would be trying style transfer using this network later on to test results based on various networks.
- Mainly we used the VGG_ILSVRC_19_layer model as our main model while running style transfer since this network is a heavily trained network which gave good results.

- Framework used - Torch (Lua JIT)
- Training time: Approx 24hrs on GPU (30 iterations on all images)

Trained Autoencoder (CNN) on 32x32 images



Algorithm Overview

Initialization

- Select layers after which we insert style and content loss modules.
- At each of these layers store the output of the content image at the specific layer as our targets.
- Similarly for the style we store the gram matrix representations of style images as our targets.
- Randomly initialize an image as our input (white noise)

Learning - Deconvolution to Input Layer

- Propagate the current input image through the network and at the end set the gradient to 0.
- On backpropagation as the image goes through the style and content loss modules, gradients are added and they get accumulated.
- Input image modified according to the gradients to reduce overall loss.
- Loss here is CLoss + SLoss + RLoss where CLoss is the content loss, SLoss is the style loss and RLoss is the regularization loss.
- Repeat this process for around 1000 iterations to get good results.

Content Loss

Content Loss module implemented

- A single content loss module has been added in the network
- The 3 most commonly used images for style transfer were used and the results were recorded. The tubingen image, brad pitt's face and the golden gate image.
- This module just stores an expected content image representation during initialization.
- When we pass the white noise image it compares the L2 norm between the noise image representation at the current layer and the expected output of the content image at that same layer and adds this to the gradient while backpropagating.

Style Loss

Gram Matrices

- Represent the style of an input volume with the help of a 2D matrix
- An input volume of dimensions (C,H,W) would generate a matrix of dimensions (C,C)
- The generation of the gram matrix is done as follows,

  (C,H,W) -> (C,HxW) -> ([C,HxW],[C,HxW]) -> ([C,HxW],[HxW,C]) -> (C,C)

- This matrix is symmetric and also satisfies all the other properties of kernels. This is essentially a kernel that captures the style of an image quite well.

Style Loss modules implemented

- Around 4-5 style loss modules have been added in the network at various places
- We ran the network on images of our faces and various other pictures and stored the results.
- This module just stores an expected gram matrix of the style image representation initially
- When we pass the white noise image it compares the L2 norm between the gram matrix of the noise image representation at the current layer and the expected gram matrix at that same layer and adds this to the gradient while backpropagating.

Regularization Loss

Regularization Loss module implemented

What we noticed was that with certain kinds of style and content images often the network ended up making them completely white since the error started to get minimized by doing this. We took two steps to fix this issue.

1. We computed and subtract the means from the original images and then multiplied it with 255.0 to get a representation in the range of -255.0 to 255.0 which helped mainly solve the problem.
2. We also added a regularization loss module at the start of the network which would try to reduce the intensity of the image and move it towards a zero image. In this manner we would not let the image become completely white which was the issue we faced earlier. Finding suitable weights for all the loss modules was a challenge but we were successful after a few iterations of trial and error.

Total Variational Loss

TV Loss module used

- We used an existing TV (Total Variational) loss module and added it to the start of our network.
- This helped reduce the variation between adjacent pixels and helped us achieve a lot of smoothness in the output image based on the weight assigned to the TV Loss module. We ended up using a 1e-3 weight for the TV Loss layer.
- The results seemed fine without this layer as well but it helped reduce the overall random noise in the output image which made it seem unreal and odd at times.

Sample Runs

Examples using only Content Loss

Only Content Loss
Outputs - 1

Tubingen.jpg

Only Content Loss Outputs - 2

brad_pitt.jpg

Only Content Loss
Outputs - 3

golden_gate.jpg

Final Examples

Examples without Input Normalization

(Almost all terrible failures)

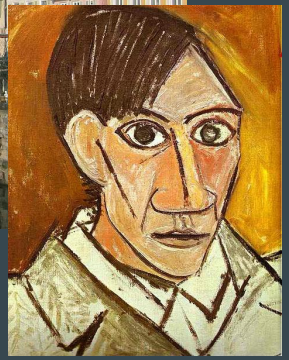
No Input Normalization - 1

golden_gate.jpg

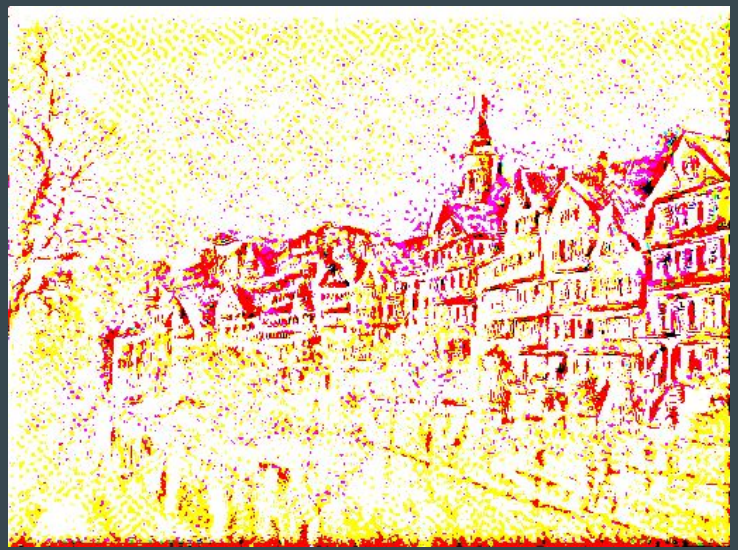
starry_night.jpg

No Input Normalization - 2

tubingen.jpg



picasso_self_portrait.jpg

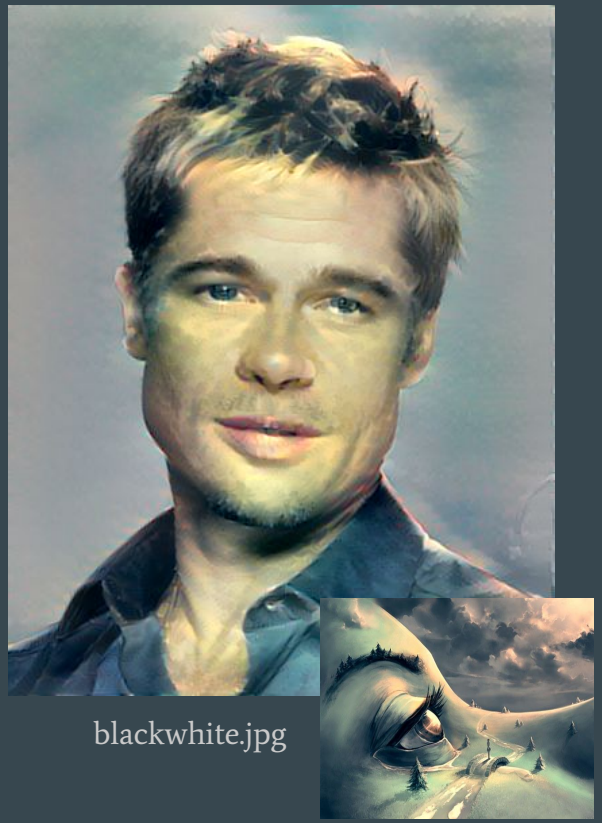


Examples with Input Normalization

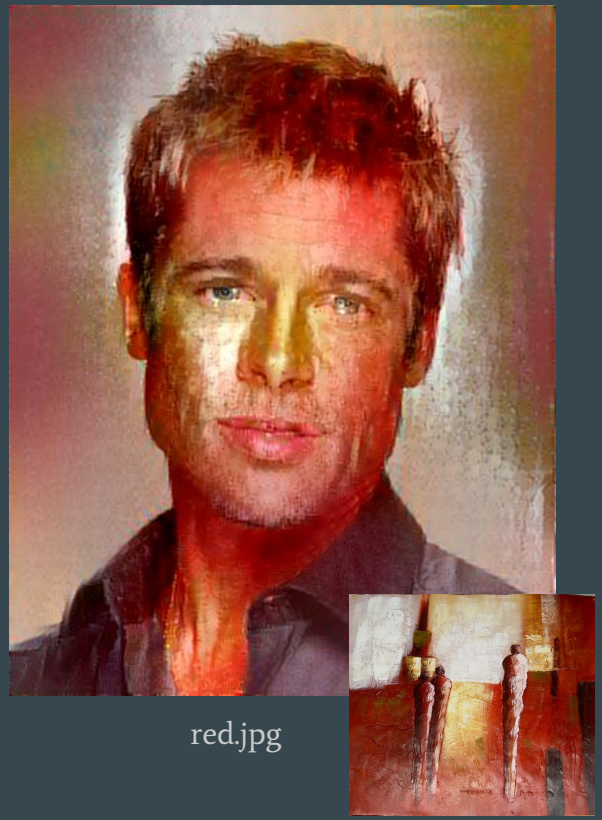
(And TV, Regularization Losses)

Example - 1

brad_pitt.jpg



blackwhite.jpg



red.jpg

Example - 2

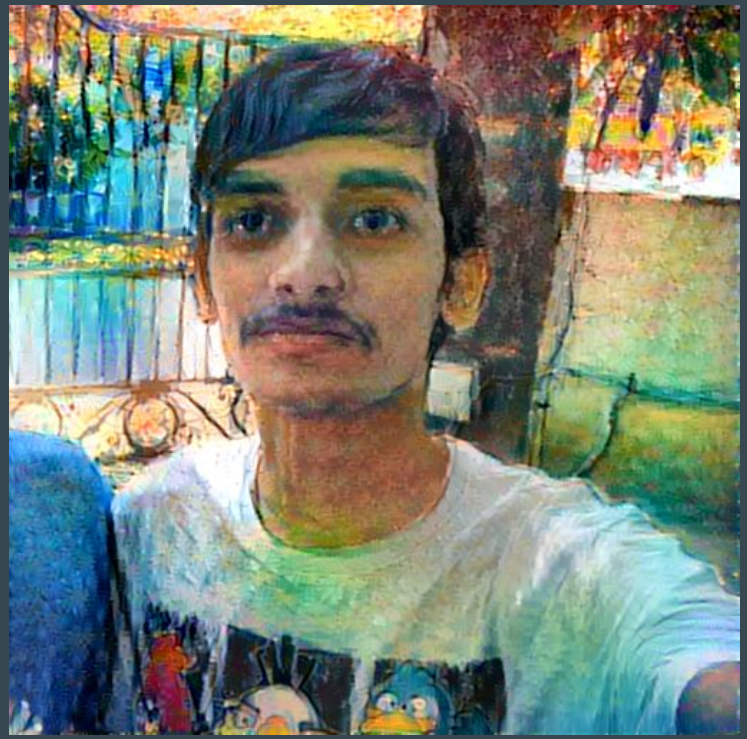
tubingen.jpg

seated-nude.jpg

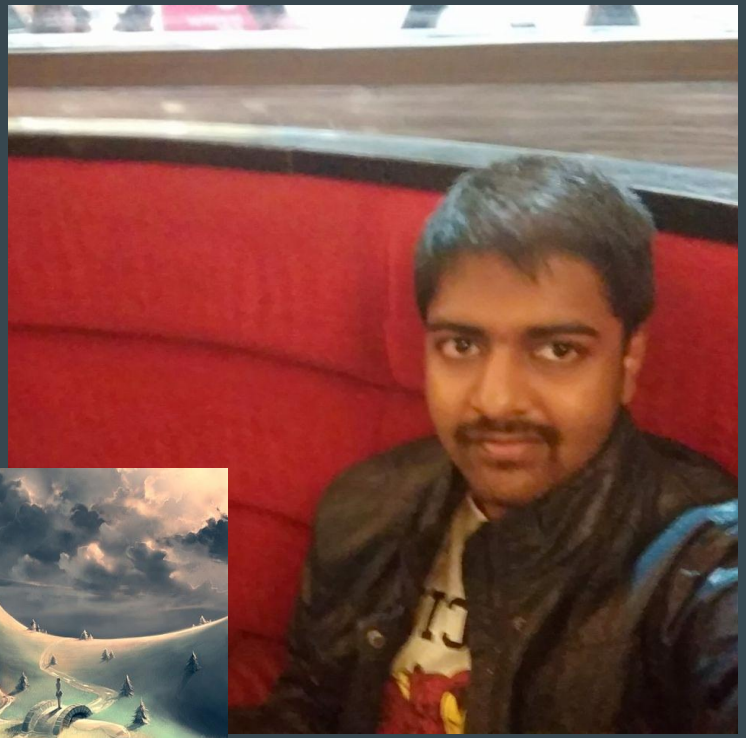
Example - 3

vishal.jpg

scenery4.jpg



Example - 4



raghuram.jpg

blackwhite.jpg

Example - 5

house.jpg

style2.jpg

Example - 6

bear.jpg

walk.jpg

Example - 7

bear.jpg

scenery.jpg

Example - 8

bear.jpg

walk2.jpg

Some Failure Cases

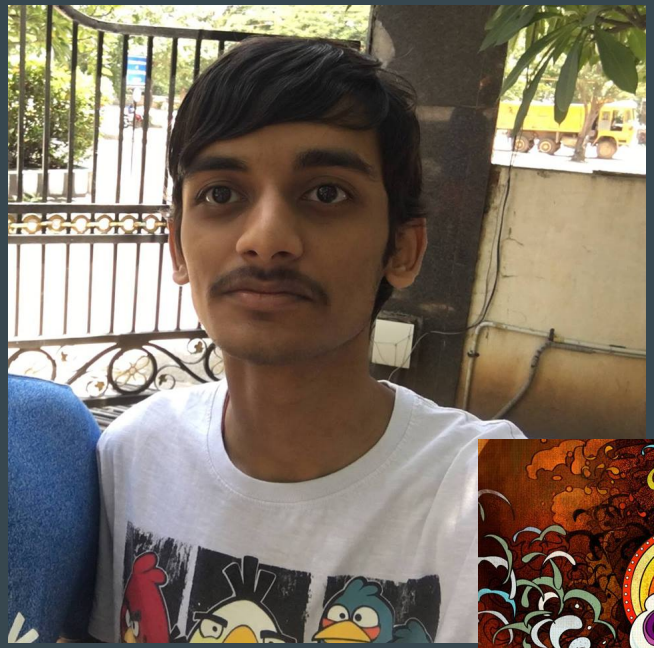
Failure Case - 1

bear.jpg

stars.jpg

Having very dark regions in the style images have a gram matrix which can easily be achieved, which is why the images copy the style with ease and ignore focussing on the content loss that much. Quick fix - Reduce weight of style loss.

Failure Case - 2



vishal.jpg

color.jpg

Technically, not a failure but since the style was so intricate, we ended up prioritizing the style a lot over the content. Quick fix - Reduce the weight of the style loss.

Comparisons varying a few parameters

Comparison 1 - Variation of Style and Content Weights

Small Style weight

Medium Style weight

Large Style weight

Comparison 2 - Using different layers for inserting style loss modules

Using relu1_1, relu2_1, relu3_1 and relu4_1 layers for insertion of style loss modules.

Using relu2_1, relu3_1 and relu4_1 layers for insertion of style loss modules.

Using only relu1_1 layer for insertion of the style loss module.

Further Improvements

- More training on own network for better results
- Local style transfer - Neural Doodle
- Extension to videos along with flow analysis for smoothing transitions
- Using multiple styles along with a single content image.
- Sketch coloring using corresponding images by foreground/background segmentation

Past/Current Research

- 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 - https://arxiv.org/pdf/1604.08610v2.pdf
- 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. - https://arxiv.org/pdf/1610.07629v1.pdf
- A few methods similar to style transfer are being compared. http://cs231n.stanford.edu/reports2016/208_Report.pdf
- Style transfer on face portraits was done in 2014 itself, https://people.csail.mit.edu/yichangshih/portrait_web/ and the results were quite impressive. This was extended to videos as well.
- Generate almost painting like images (Crayon, wax, paint, pastels etc) - http://jaisrael.github.io/ParallelStyleTransfer/
- We have also been planning on extending this to the automatic coloring of sketches. Consider a sketch as an input, get a corresponding image either via an SBIR system or just using standard class images for each of the available sketch classes. We then segment both the sketch and the image and match corresponding regions. We then finally apply the style transfer on each region independently to ensure that we end up coloring the sketch in a manner similar to it's corresponding image.

Sketch automatic coloring

(Work in progress)

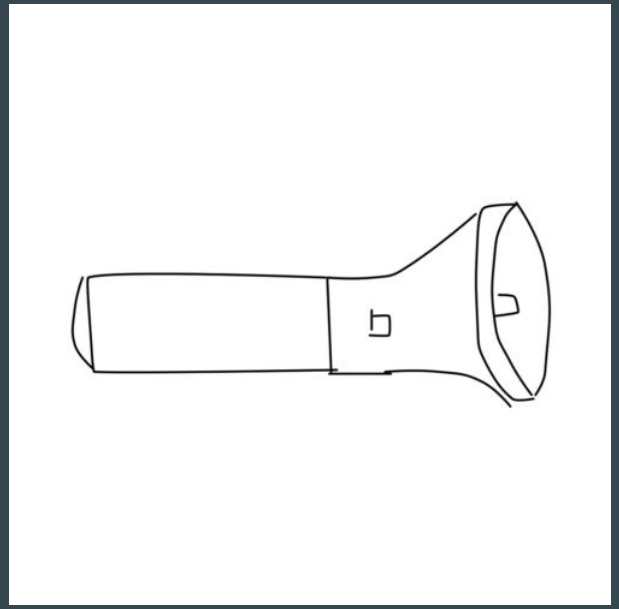
Example - 1

Input Airplane Sketch

SBIR retrieved airplane

Colored Sketch (We will overlay the input sketch on this image to get a good coloring)

Example - 2



Input Torch Sketch

SBIR retrieved torch

The distortion is mainly due to bad segmentation of the images

Thank you!