Shobhit Narayanan (1002315); Vishal Ramanathan (1002319)

Neural Style Transfer

Background

Convolutional Neural Networks were originally created for image classification. Lately they have been used in a variety of other tasks like Image Segmentation, Neural Style transfer and Natural Language Processing tasks as well. CNNs are one of the most interpretable models in Deep Learning because of the ability to visual their representations and understand what they might be learning.

It's safe to assume that CNN does not learn to encode what image is but it actually learns to encode what image represents or what contents are visible in the image. Due to the inherent nonlinear nature of neural networks as we go from shallow layers to deeper layers, the hidden units become capable of detecting more complex feature from a given image. This nature of encoding representations itself is the key to style transfer. It is used to calculate loss between the generated image with respect to content and style image.

There are two input images namely content image and style image that are used to a generate a new image called stylized image. The output has the same content as the content image and has a style similar to that of the style image.

Expected input

Style Image:

Dataset used

For style

Google Images of paintings

For Content

MSCOCO

Group members and contribution

Shobhit: Primary incharge of Model and Training

Vishal: Primary in charge of testing and Visualization

Validation

For the validation process we used the manual validation technique.

There are 2 main variables that can affect validation.

1. Noise factor
2. Style loss weights / Content loss weights ratio

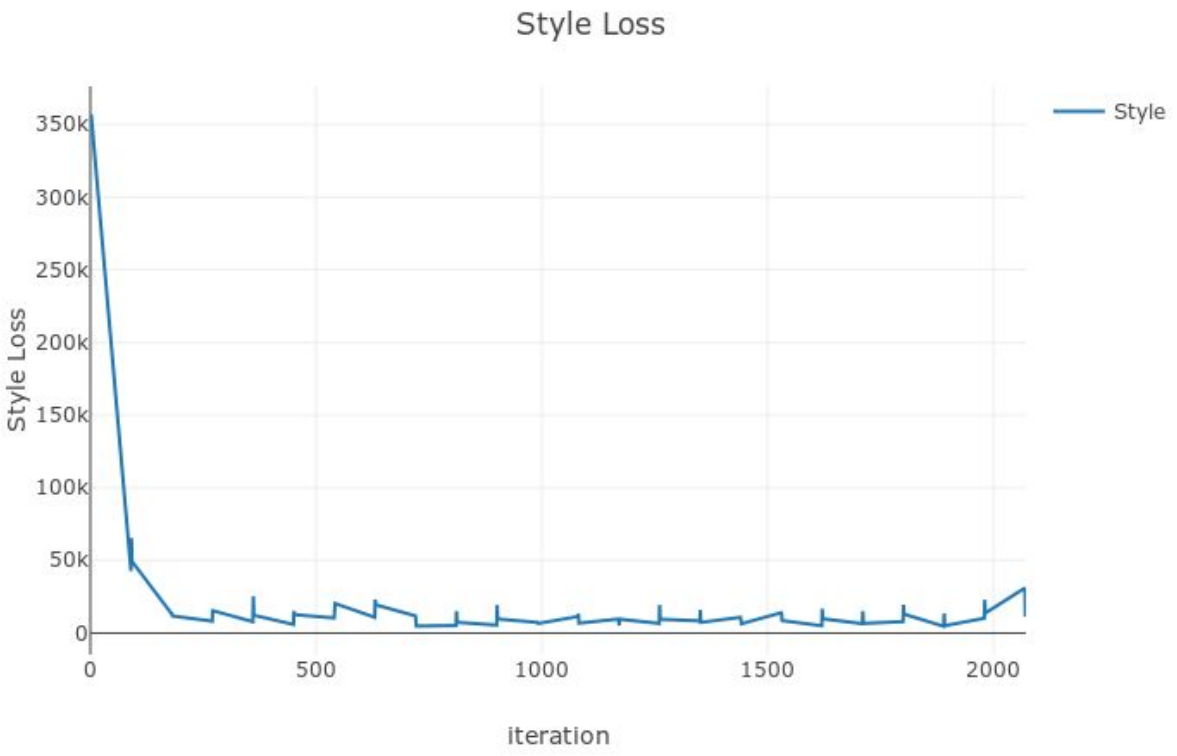
Using

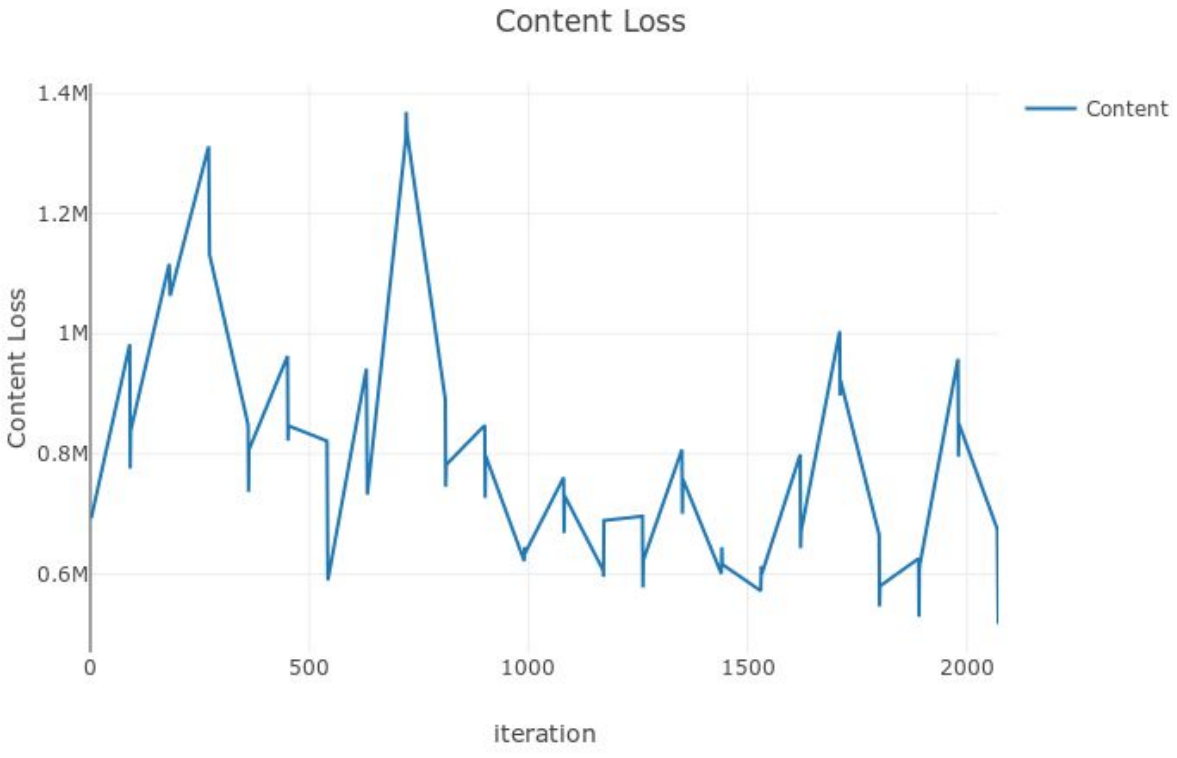
```
style_weights = [5,6,2,2,2]
content_weights = [1e3]
```

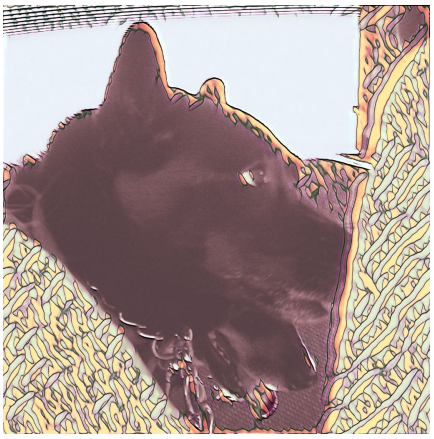
leads us with the best result.

```
Style Loss:  356688.90625
Content Loss:  705239.25
Style Loss:  356505.84375
Content Loss:  699085.0625
Style Loss:  356120.71875
Content Loss:  693082.625
Style Loss:  42685.30078125
Content Loss:  982656.75
Style Loss:  65362.91796875
Content Loss:  775690.3125
Style Loss:  49349.06640625
Content Loss:  838679.125
Style Loss:  13229.3955078125
Content Loss:  1116379.5
Style Loss:  13110.64453125
Content Loss:  1098964.5
.
```

```
.
.
Style Loss:  12664.740234375
Content Loss:  847008.875
Style Loss:  10377.296875
Content Loss:  821544.9375
Style Loss:  19290.583984375
Content Loss:  617036.0
Style Loss:  20327.95703125
Content Loss:  589505.25
Style Loss:  10870.7841796875
Content Loss:  941990.0
Style Loss:  23111.919921875
Content Loss:  747384.875
Style Loss:  19450.939453125
Content Loss:  732163.0
Style Loss:  11874.4765625
Content Loss:  1324799.375
Style Loss:  4895.2958984375
Content Loss:  1369556.0
Style Loss:  4816.0927734375
Content Loss:  1338476.25
Style Loss:  5138.3115234375
Content Loss:  890824.0625
Style Loss:  15075.9755859375
Content Loss:  745314.125
Style Loss:  7370.34375
Content Loss:  782077.875
Style Loss:  5523.40673828125
```


Style Loss
Style
350k
300k
250k
200k
150k
100k
50k
0
Style Loss
0
500
1000
1500
2000
iteration

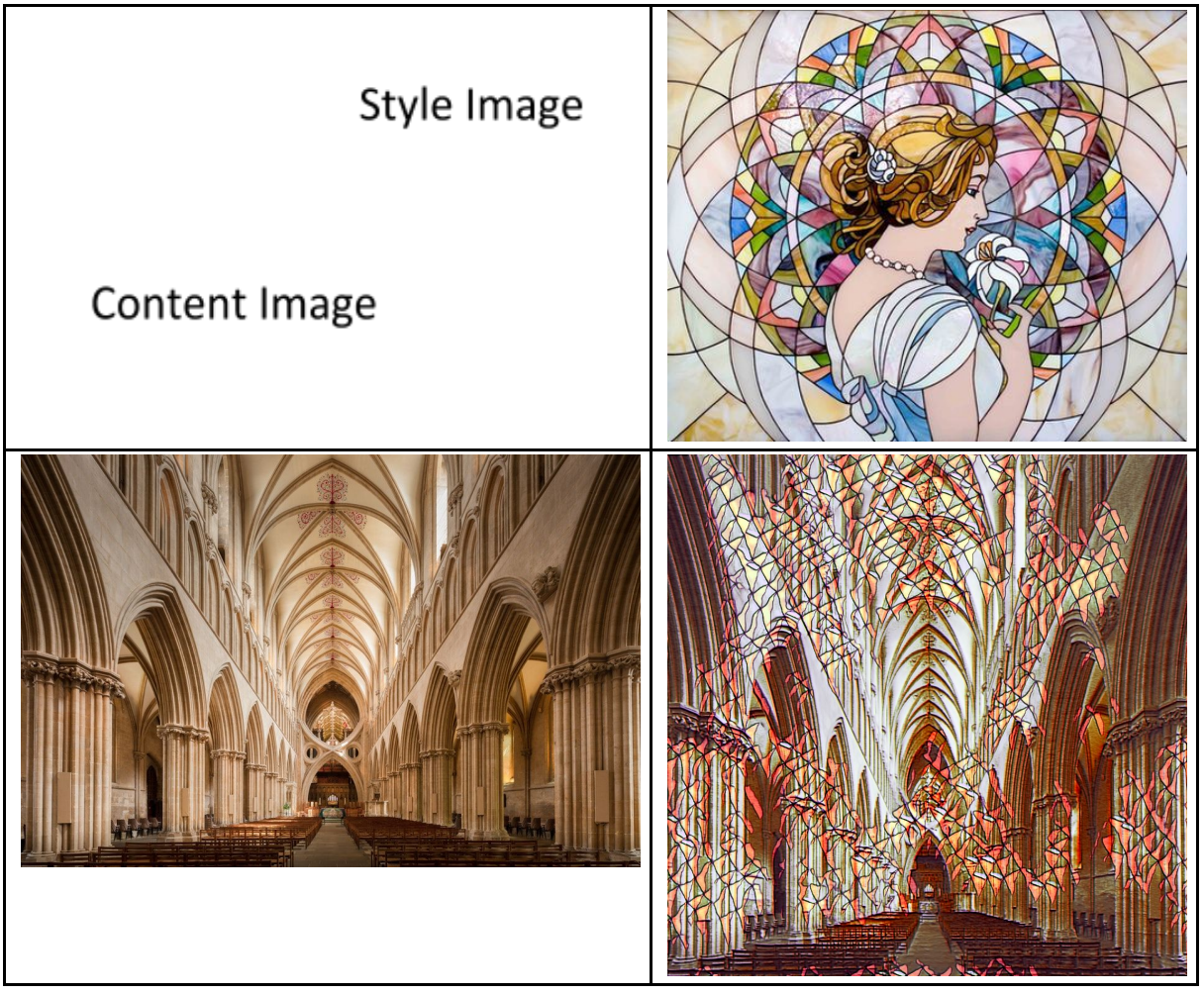

Content Loss
Content
1.4M
1.2M
1M
0.8M
0.6M
Content Loss
0
500
1000
1500
2000
iteration

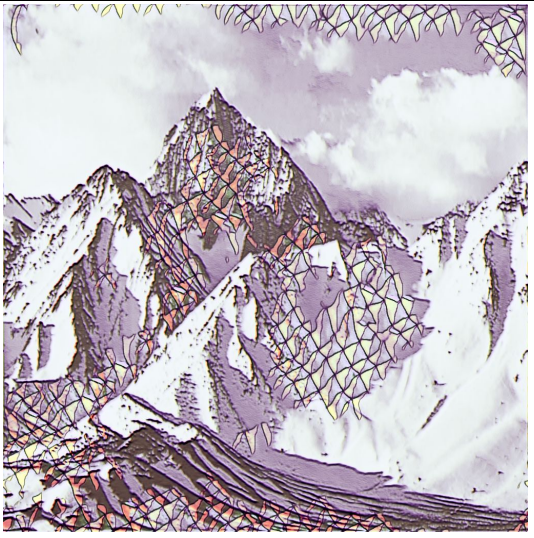


The noise factor decides the strength of the style being applied to the content image.

This is an example image of a dog with a low noise filter level. Since there is an artistic appeal to the images produced we needed to manually find an endpoint for our model and save the state dictionary.

Outputs





Setup details

We are using visdom for checking results. You can install it using

```
pip install visdom
```

```
python -m visdom.server
```

However, you can ignore its use by not adding the --viz argument while training.

For training:

```
python train-3.py --data_dir
/media/omegashenr01n/System1/Users/Shobhit/Documents/DL/COCO/train
2014/ --cuda --texture Textures/mosaic.jpg --verbose --viz
```

The code requires a data directory, the one we used is for MSCOCO. And it needs an input texture image.

For GUI:

a. Install node and npm. It needs to be installed in your PATH variables.
b. Ensure that you have python available on your path variables and that whatever environment you have running is ready.
c. In terminal go to TN-on-node/app directory and enter `npm i`.
d. Enter `node server`.
e. On your browser go to: localhost:3000/
f. Enter the address of your data set. Eg: /media/omegashenr01n/Documents/DL/MSCOCO/
g. The UI should guide you from there.

Bibliography

- Ulyanov, D., Lebedev, V., Vedaldi, A., & Lempitsky, V. (2016, March 10). Texture Networks: Feed-forward Synthesis of Textures and Stylized Images. Retrieved from https://arxiv.org/abs/1603.03417
- https://github.com/JorgeGtz/TextureNets_implementation