Perceptual Losses for Real-Time Style Transfer

Aakef Waris, Ali Issaoui

Table of Contents

Problem:

- Implement style transfer using perceptual loss functions
- Perceptual Loss Functions?
  - Instead of traditional pixel-by-pixel loss, perceptual loss functions are computed from intermediary layers from pre-trained networks
  - This captures higher level feature similarities across images instead of pixel values
  - Effective when trying to capture and recognize features like edges, texture, sharpness, color, etc.
- Compute style AND feature loss functions to converge using Adam optimizer

The Dataset

MSCOCO :

COCO
Common Objects in Context

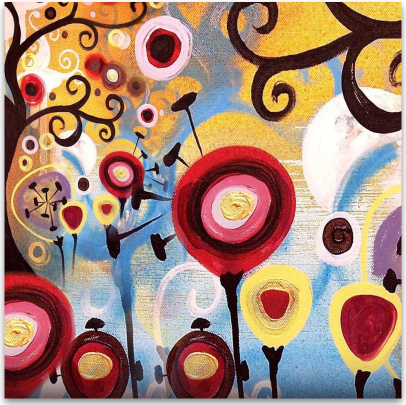
Styles:

COCO is a large-scale object detection, segmentation, and captioning dataset.

Original dataset:

330K images
1.5 million object instances
80 object categories
91 stuff categories

Our subset: **30K images**



Network Architecture

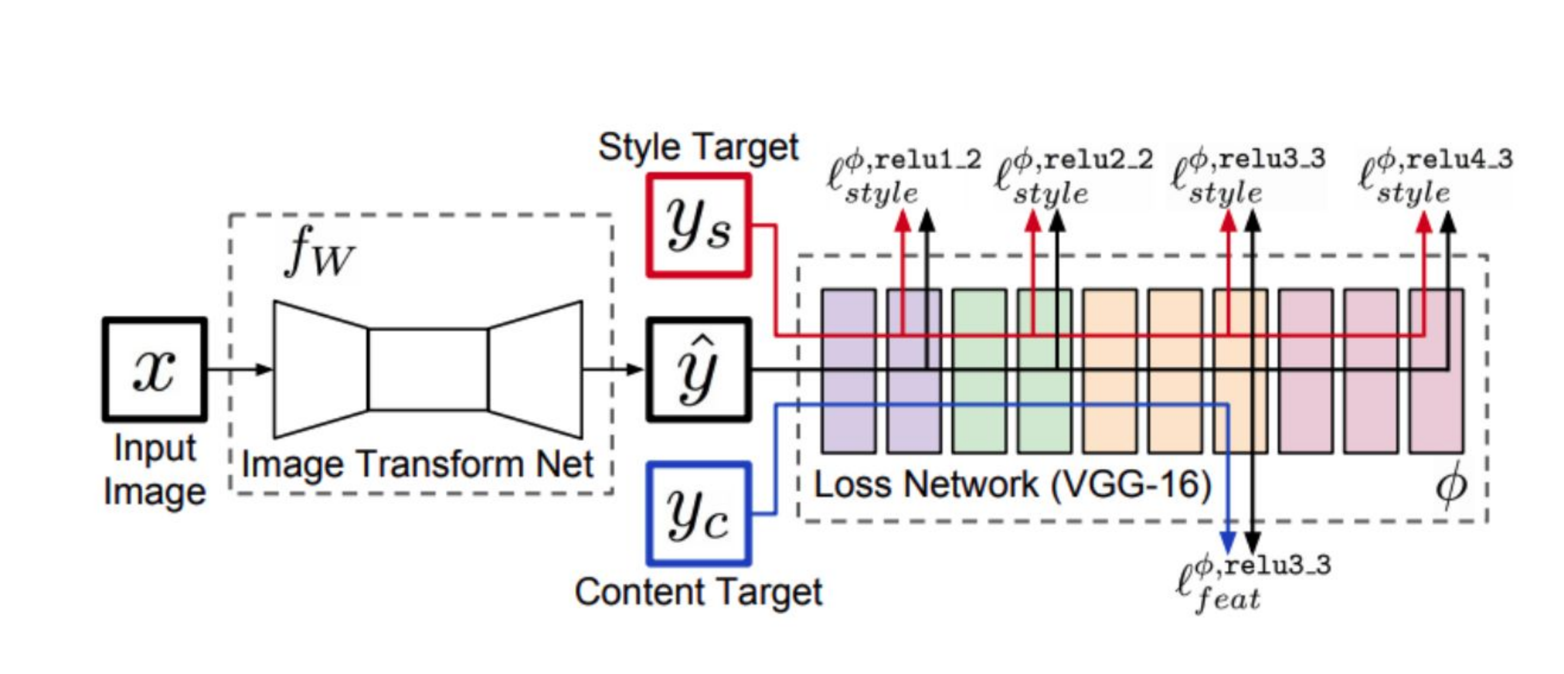


Image Transformation Network

- Initial Down Sampling Layers from 3 to 32 to 64 to 128 channels
- Body of 5 Residual Blocks at 128 channels
- Final Up Sampling Layers from 128 to 64 to 32 to 3 channels for the output image

```
class ImageTransformationNN(torch.nn.Module):

    def __init__(self):
        super(ImageTransformationNN, self).__init__()
        self.down_sample = DownSampleConv()
        self.res = ResidualNet()
        self.up_sample = UpSampleConv()

    def forward(self, X):

        X = self.down_sample(X)
        X = self.res(X)
        y = self.up_sample(X)
        return y
```

Down/Up -Sampling

- 3 Convolutional Layers
- Used Reflection Padding
- Followed by Instance Normalization

Instance Normalization

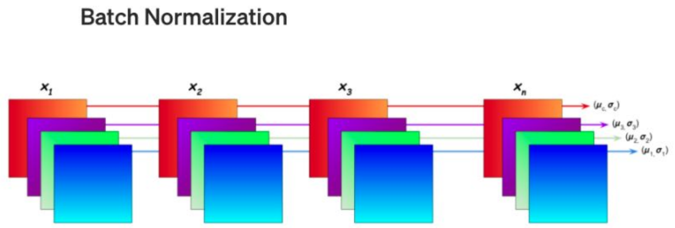


Figure 3:

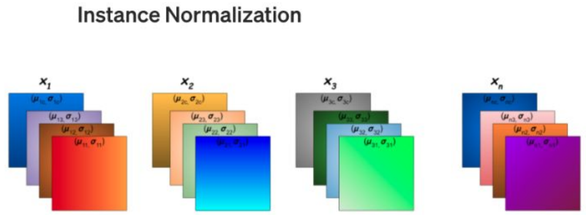


Figure 4:

Reflection Padding

3	5	1
3	6	1
4	7	9

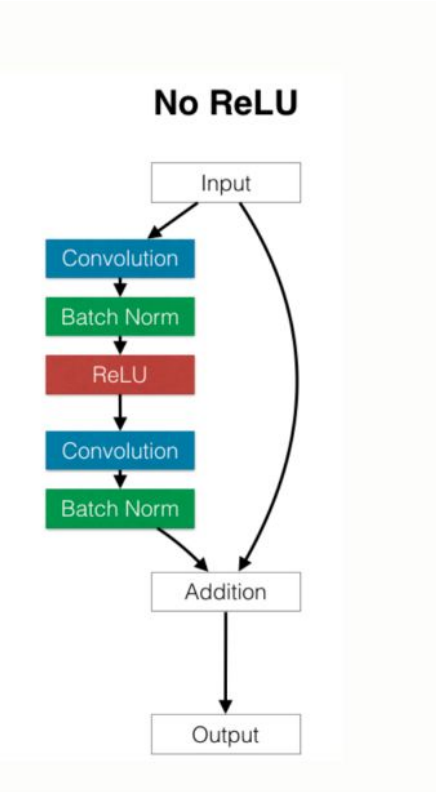
No padding

1	6	3	6	1	6	3
1	5	3	5	1	5	3
1	6	3	6	1	6	3
9	7	4	7	9	7	4
1	6	3	6	1	6	3

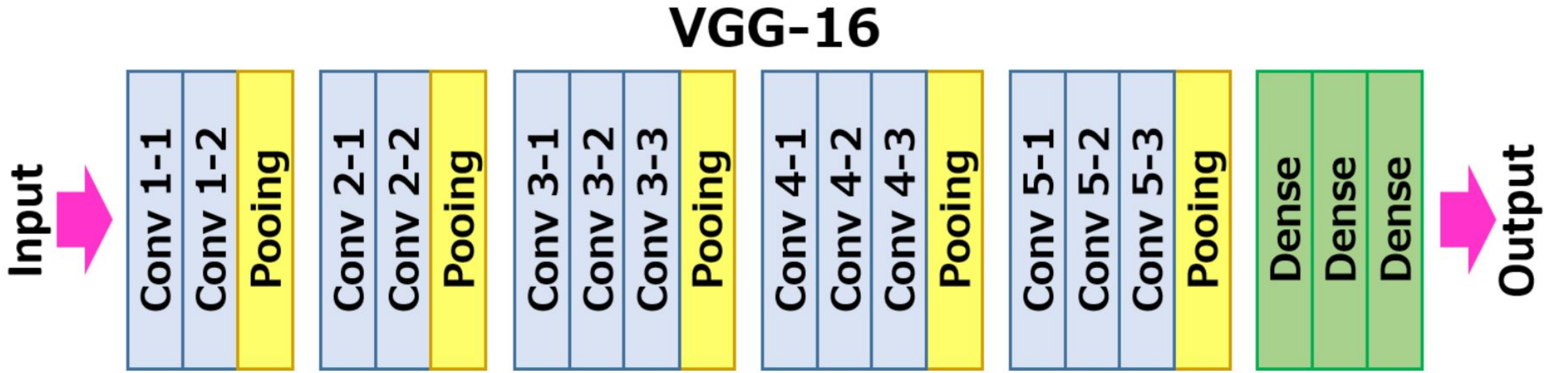
(1, 2) reflection padding

Figure 5:

Residual Blocks



VGG-16 Network (Loss Network)



Training and Loss

Feature reconstruction loss (relu3_3):

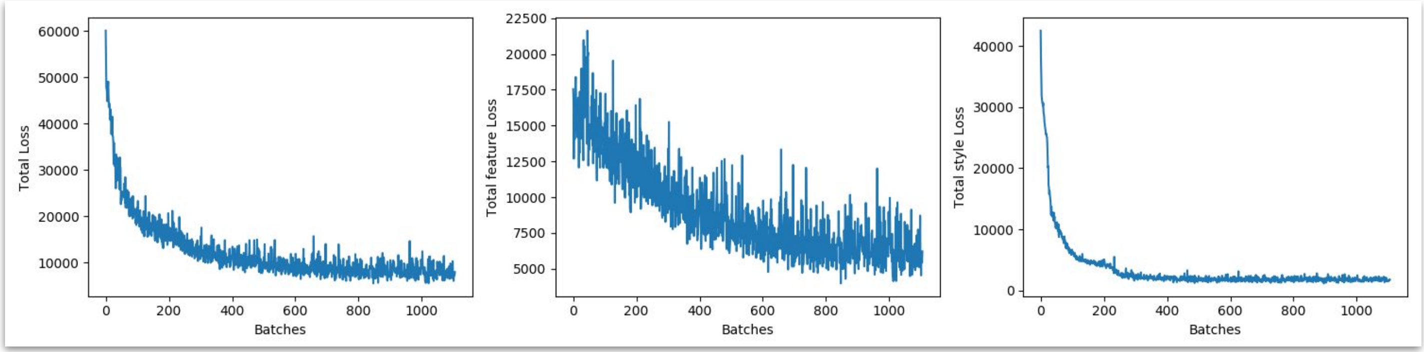
$$l_{feat}^{\theta,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \|\theta_j(\hat{y}) - \theta_j(y)\|_2^2$$

Style Reconstruction Loss(relu1_2, relu2_2, relu3_3, relu4_3):

$$l_{style}^{\theta,j}(\hat{y}, y) = \|G_j^{\theta}(\hat{y}) - G_j^{\theta}(y)\|_F^2$$

Training parameters:

- Epochs: 2
- Batch size: 4
- Optimizer: Adam
- Learning rate: 1e-3



Results

Style Image:

Content Image:

Output Image:

Problems and Further improvements

Problems:

1. Limited computational power
2. Limited time
3. Limited space for the entire COCO dataset (18 Gb)

Further improvements:

1. Continue the training for the Style Transfer task
2. Generalize the perceptual loss method on other transformation tasks:
    a. Image super-resolution
    b. Image colorization, and others

References

- The original paper: https://arxiv.org/pdf/1603.08155.pdf

- Instance Norm: https://becominghuman.ai/all-about-normalization-6ea79e70894b#:~:text=In%20%E2%80%9CInstance%20Normalization%E2%80%9D,%20mean,sample%20across%20both%20spatial%20dimensions.

- Reflection Padding: https://www.machinecurve.com/index.php/2020/02/10/using-constant-padding-reflection-padding-and-replication-padding-with-keras/

- Residual Block Architecture: http://torch.ch/blog/2016/02/04/resnets.html

Thank you