NYU TANDON

FAST NEURAL STYLE TRANSFER

The Great Wave off Kanagawa

Image Source: Wikipedia - https://commons.wikimedia.org/wiki/File:Tsunami_by_hokusai_19th_century.jpg

Submitted by:

Avinash Veershetty (apv280)

Rajat Ravindrakumar Bapuri (rrb398)

Shiva Sanketh Ramagiri Mathad (srm714)

EL-GY 6123: Introduction to Machine Learning (Graduate) course project

Table of Contents

Abstract 2

Problem Statement 2

Architecture: 3

Methodology (Description with formulae) 3

Dataset: 4

Training: 5

Improvements 5

Results 6

Conclusion 9

References 10

Abstract

Rendering the semantic content of an image in different styles is a difficult image processing task. Here, we consider an image transformation problem, where we are transforming a input image into an output image with style derived from another image. We use the approach of feed-forward networks for the task. Different Convolution layers extract different features of an image, we exploit this feature of CNN for the task. We performed above tasks using pretrained models VGG-16 and VGG-19 using image and video as inputs to compare their quality of style transfer and performance. We also experimented by adding texture upon style on the content image.

Problem Statement

We aim at using Neural Style Transfer to produce artistic transformation of images. A task which a human would take days or months to complete is done within seconds using neural networks. With many potential models for Neural Style transfer, we intend to compare the results of two most suited for this application – VGG 16 and VGG 19. Along with comparison of these models, we try to experiment by adding texture along with style on the content image. Extending the task, we try applying VGG 16 model on video input.

Content Image + Style Image = Stylized Image

Architecture:

Following image shows the architecture from *Perceptual Losses for Real-Time Style Transfer and Super-Resolution by Johnson et al* [1]. A Deep Convolutional Neural network such as VGG-16, VGG-19, Inception network or Resnet pre-trained on Imagenet dataset is used as loss network, along with the Image Transform net which has residual connections. The VGG-16 and VGG-19 models have good hierarchy of features like the inception networks, but it is observed that VGG models perform better at this task compared to the inception architectures. The comparison of results of these various models is explained in the results section. The image transformation network is a deep residual network. It may be noted that, the Loss Network weights are frozen, so while training only the Image Transform Network weights are trainable.

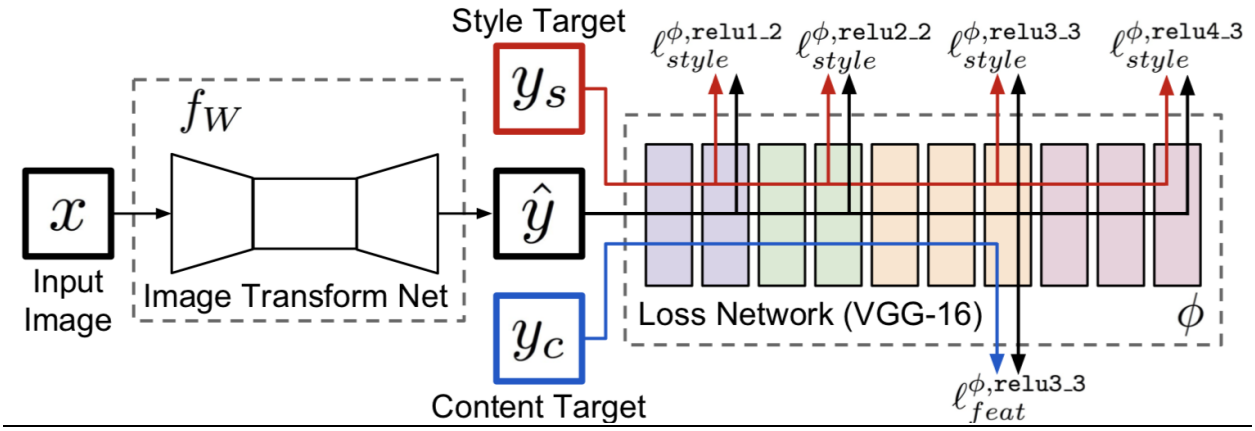


Figure 1: Fast Neural Style transfer System overview.

Methodology

The Fast Neural Style Transfer system consists of two components:

1) Image transformation network: The *image transformation network* f_w is a deep residual convolutional neural network parameterized by weights, i.e. it transforms input images x into output images $\hat{y}$ via mapping $\hat{y} = f_w(x)$. Each loss function computes difference between the output image and a target image $l_i(\hat{y}, y_i)$. The image transformation network is trained using stochastic gradient descent to minimize a weighted combination of loss functions [7–11]:

$$W^* = argmin_w E_{x,\{y_i\}}[\sum_{i=1}^{\cdot} \lambda_i l_i(f_w(x), y_i)]$$

2) Loss network: Loss network is used to define a *feature reconstruction loss* and *style reconstruction loss* that measure differences in content and style between images. For each input image, we have a content target and a style target. We train one network per style target. We define two perceptual loss functions that measure high-level perceptual and semantic differences between images.
   a) Feature Reconstruction Loss: Rather than encouraging the pixels of the output image $\hat{y}$to exactly match the pixels of the target image y, we instead encourage them to have similar feature representations as computed by the loss network. Let $\Phi_j(x)$be the activations of the jth layer of the network when processing the image x. We use a 3×3convolution with C filters on an input of size $C \times H \times W$. if $j$is a convolutional layer then $\Phi_j(x)$will be a feature map of shape $C_j \times H_j \times W_j$. Using a feature reconstruction loss for training our image transformation networks encourages the output image to be perceptually similar to the target image but does not force them to match exactly. The feature reconstruction loss is the (squared, normalized) Euclidean distance between feature representations:

$$l_{feature}^{\phi,j}(\hat{y}, y) = \frac{1}{C_j H_j W_j} \left|\left|\phi_j(\hat{y}) - \phi_j(y)\right|\right|_2^2$$

   b) Style reconstruction loss: The feature reconstruction loss penalizes the output image when it deviates in content from the target. This loss is used to penalize differences in style: colors, textures, common patterns, etc. Let $\Phi_j(x)$be the activations at the jth layer of the network for the input x, which is a feature map of shape $C_j \times H_j \times W_j$. The elements of Gram matrix $G_j^{\Phi}(x)$of size $C_j \times C_j$matrix can be defined as:

$$G_j^{\Phi}(x)_{c,c'} = \frac{1}{C_j H_j W_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{h,w,c} \phi_j(x)_{h,w,c'}$$

   This helps us capture information about which features tend to activate together. The style reconstruction loss is the squared Frobenius norm of the difference between the Gram matrices of the output and target images:

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

   To perform style reconstruction from a set of layers Jrather than a single layer j, we define $l_{style}^{\Phi,j}(\hat{y}, y)$to be the sum of losses for each layer $j \epsilon J$.

Dataset:

Style transfer networks are trained on the MS-COCO dataset [4]. We make use of the dataset consisting 80000 images as content images to train the network. In contrast to the popular ImageNet dataset, COCO has fewer categories but more instances per category.

Training:

We train with MS-COCO training set and prepare low-resolution inputs by blurring with a Gaussian kernel of width σ = 1.0 and down-sampling with bicubic interpolation. We resize each of the 80k training images to 256 × 256 and train with a batch size of 4 for 40k iterations which is effectively 2 epochs on the whole dataset, giving roughly two epochs over the training data. We use Adam [56] with learning rate 1 × 10−3. The output images are regularized with total variation regularization with a strength of between 1 × 10−6 and 1×10−4, chosen via cross-validation per style target. We do not use weight decay or dropout, as the model does not overfit within two epochs. For all style transfer experiments, we compute feature reconstruction loss at layer 'block3_conv3' and style reconstruction loss at layers 'block1_conv2', 'block2_conv2', 'block3_conv3' and 'block4_conv3' of the VGG-16 loss network. For VGG-19, we compute feature reconstruction loss at layer 'block4_conv2' and style reconstruction loss at layers 'block1_conv1', 'block2_conv1', 'block3_conv1', 'block4_conv1' and 'block5_conv1' of the loss network. The code has been implemented in Keras using Tensorflow as the backend. Training takes roughly 4 hours on a single P40 GPU.

Improvements

1) **VGG-19 architecture:** Original paper *Perceptual Losses for Real-Time Style Transfer and Super-Resolution*[1] and its base code implementation[5] incorporates *VGG-16 architecture*[2]. We have improvised on this by implementing the problem statement using *VGG-19 architecture*[3].

2) **Video Transformation:** Base code implementation[5] derives its inference on images. Our implementation extends it to provide fast style transfer on videos. Training part of the implementation remains the same, we use either *VGG-16*[2] or *VGG-19*[3] architecture with *MS-COCO dataset*[4]. During training stage, a model file (with weights and other parameters) is created and saved. Our improvement in the inference includes loading a video file and the saved model file. The output is a video file which is style transformed based on style image from the model file . The loaded video file is processed frame-by-frame using *OpenCV*[6] library and the parameters and weights are extracted from model file. Each frame is then pre-processed and fast neural style transfer is performed. The frames are then stitched together to form a video and saved.

3) **Texture Transformation:** Images with less pixel variation does not yield a distinctive aesthetic look after style transformation. To provide solution for this problem, we introduce a new layer in-between content image and the style layer called as texture layer. A texture layer is also a style transformation but in this, the transformation adds minor variations on the texture of the image. A style transformation after this produces a better artistic look.

a) Training: Texture transformation training part is similar to style transformation training part, in this a suitable image[15] which results in only texture transformation is used for training.
b) Inference: The improvement of Texture transformation is mainly in the inference part. The content image is first preprocessed and then fast style transfer is performed using model from texture image, later a second fast style transfer is performed using model from style image.

Results

Image Style Transfer using VGG-16 and VGG-19

After performing Image Style Transfer using VGG-16 and VGG-19 models, we found following results:

Content Image

Style: The Muse, Pablo Picasso, 1935

VGG-16 Output Image

VGG-19 Output Image

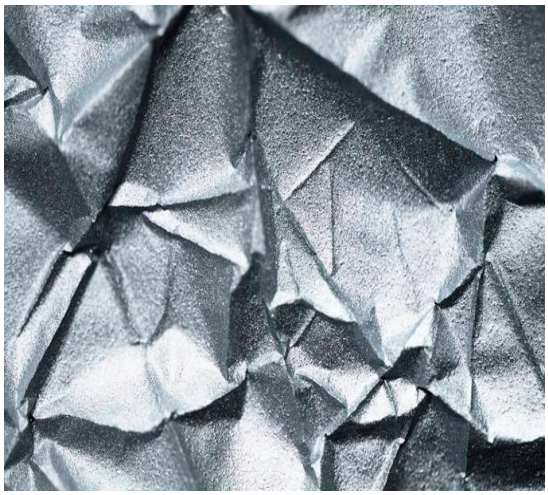
Our models trained for feature reconstruction do a very good job at reconstructing sharp edges such as the building glass panel edges in the content image. The models do not sharpen edges indiscriminately. It sharpens the boundary edges of the building, but the background remains diffuse, suggesting that the model may be more aware of image semantics. Models do a better job at reconstructing fine details, leading to pleasing visual results. As we can see VGG-19 produces slightly better outcome in terms of artistic display. In VGG-19 output image, the building edges are blended well depicting original brush stroked image keeping a proper balance of color and style. Although beautifully incorporating style, the VGG-19 output seems to depend more on the color aspect of the style image.

The outcome of texture and style addition on content image using VGG-16

Content Image

Style Image: wave crop

Texture Image



Output Image without Texture added

Output Image = Content + Style + Texture

Video Style Transfer using VGG-19

The result of Video Style Transfer is stored in Project GitHub portal(https://github.com/Rajat-R-Bapuri/introml-project/tree/master/results).

Although these networks are trained to optimize losses for 256 × 256 images, they also succeed at minimizing the objective when applied to larger images. We repeated the same quantitative evaluation for 512 × 512 and 1024 × 1024. Even at higher resolutions our model achieves a loss comparable to 50 to 100 iterations of the baseline method. Across all image sizes, this method takes about half the time of a single iteration of the baseline. The method processes 512 × 512 images at 20 FPS, making it feasible to run in real-time or on video.

Conclusion

Thus, we would like to make an inference that VGG-19 [3] outperforms VGG-16 [2] in the Style Transfer Task. The style transfer task does not need many epochs to train the network as opposed to classification tasks which require lot of epochs. This task has enabled us to explore other interesting problems such as adding texture to images along with style and stylizing the videos.

Future Work

We would like to extend this approach to other forms of input such as audio and text. It would nice if we could stylize one audio in the form of other for example one person's voice can be transformed to another person's voice or in the context of text one form of text is transformed to another style of writing. Therefore, this task opens up a plethora of many other interesting tasks which can be worked on.

Acknowledgement

We would like to thank Prof. Sundeep Rangan for advising us for the project and providing Google Cloud Platform GPU credits. Also, we would like to thank Sam Lee [5] and Logan Engstrom [14] whose Style Transfer code open-sourced on GitHub has been used and modified in this project.

References

[1] - Johnson et al (2016), Perceptual Losses for Real-Time Style Transfer and Super-Resolution.

[2] & [3]- Simonyan et al (2014), Very Deep Convolutional Networks for Large-Scale Image Recognition

[4] - Lin TY. et al. (2014) Microsoft COCO: Common Objects in Context

[5] - Misgod, fast-neural-style-transfer, (2017), Github Repository, https://github.com/misgod/fast-neural-style-keras

[6] - https://opencv.org/

[7] Mahendran, A., Vedaldi, A.: Understanding deep image representations by inverting them. In: CVPR. (2015)

[8] Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR Workshop. (2014)

[9] Yosinski, J., Clune, J., Nguyen, A., Fuchs, T., Lipson, H.: Understanding neural networks through deep visualization. In: ICML Deep Learning Workshop. (2015)

[10] Gatys, L.A., Ecker, A.S., Bethge, M.: Texture synthesis using convolutional neural networks. In: NIPS. (2015)

[11] Gatys, L.A., Ecker, A.S., Bethge, M.: A neural algorithm of artistic style. arXiv preprint arXiv:1508.06576 (2015)

[12] Kingma, D., Ba, J.: Adam: A method for stochastic optimization. In: ICLR. (2015)

[13] LA Gatys et al (2015), A Neural Algorithm of Artistic Style

[14] Logan Engstrom, fast-style-transfer, GitHub Repository, https://github.com/lengstrom/fast-style-transfer

[15] Courtesy of https://freestocktextures.com/support/

Some more results using VGG 19:

Udnie by Francis Picabia

The Great Wave off Kanagawa