Image Style Transfer Using TensorFlow

Andrews Tito, Jhanvi Shah, Yashasvi Jariwala

Abstract

Rendering the semantic content of an image in different styles is a difficult image processing task. Traditional multimedia techniques require users to go through certain procedures to transfer a content image into a desired art style. Here, we use Gatys et al. and Johnson's work to draw a comparative study which involves separating and recombine the image content and style of natural images. We use pre-trained VGG networks to feature output and optimize to explicitly represent high level image information. With this information, we can generate images combining numerous art style with desired content with high perceptual quality. This project provides a solution for image processing and art composition.

Keywords: Style transfer, super-resolution, deep learning

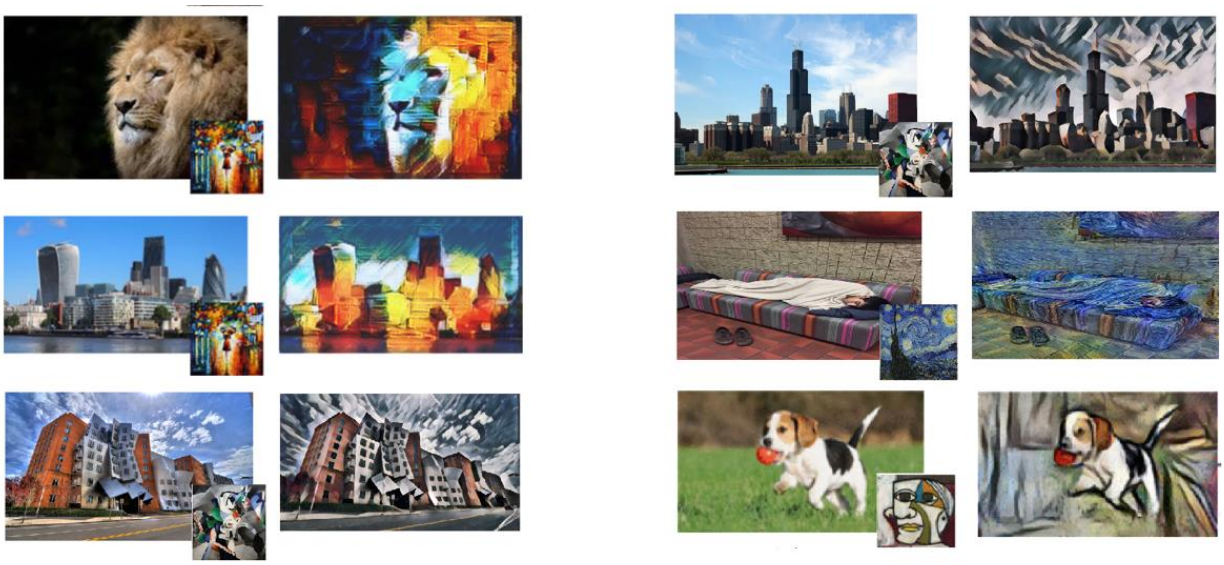


Figure 1: Demonstration of Style Transfer by L. Gatys, A. Ecker, and M. Bethge. A Neural Algorithm of Artistic Style[1]. The content image is displayed on the left, the styling image is on bottom-right and the output image is presented on the right.

Introduction

Motivation

Various applications are used today to stylize images manually to increase the aesthetic value of the photograph. To include the style of the great paintings in our photo would be a great step

towards improvising the aesthetic value of the photo. Generating an output image by giving a content and an styling image with fair accuracy is the main aim. Here, we have evaluated the accuracy of a pre-trained model implementing Convolutional Neural Network.

Background

The idea behind our project is to evaluate the accuracy of the pre-trained model VGG-19 dataset which is trained using the Convolutional Neural Network. Is takes content image and style image as input and gives the output image which gives the contours of the content image and texture and the styling from the style image. The Convolutional Neural Network provides us with powerful tools to extract features and textures from content and style images to give a combined image. Here we have used the pre-trained network VGG-19 which is trained on more than a million images and can classify the objects into 1000 categories.

Our Algorithms

The main idea of our project is to extract features from certain layers of VGG-19 [2] to generate the target image. Here we have to extract the features from some layers of the VGG-19 to obtain the resultant image. To obtain the target image of a greater accuracy we use CONTENT_LAYERS = ('relu4_2', 'relu5_2') to extract the features detecting the objects and STYLE_LAYERS = ('relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', 'relu5_1') to extract the texture and the patterns.

Cost Function defined over the output of content and style layers. And, later ADAM optimizer is used to optimize the network to generate the output image.

Methods

There are three steps to generate a target image. First, extract feature map from content image and a random generated image through VGG-19 and try to minimize the difference. Second, evaluate the correlation between two patterns and to transform it. Here we use dot product of to represent the correlation. Third, define the total loss and use Adam algorithm to optimize the result. The process is described in detail in this chapter. We also took experiments with groups of different hyper-parameters and compared the results, which is further discussed in result chapter.

Environment

Operating System: Windows 10/macOS High Sierra

Python: v3.6.2

TensorFlow: v1.6.0

Pillow: v5.0.10

GPU: NVIDIA GeForce GTX 1060 (optional)

Network

Network Structure: Generally, each layer in the network defines a non-linear filter bank whose complexity increases with the position of the layer in the network. Hence a given input image *x* is encoded in each layer of the Convolutional Neural Network by the filter responses to that image. In this part, we use ImageNet pre-trained VGG-19 network weights. ImageNet is an image dataset organized according to the WordNet hierarchy. It is available publicly. We are only using the feature space of 16 convolutional layers and 5 pooling layers of the VGG-19 network, which is a 19-layer VGG network.

The structure of the network is shown in figure 2.

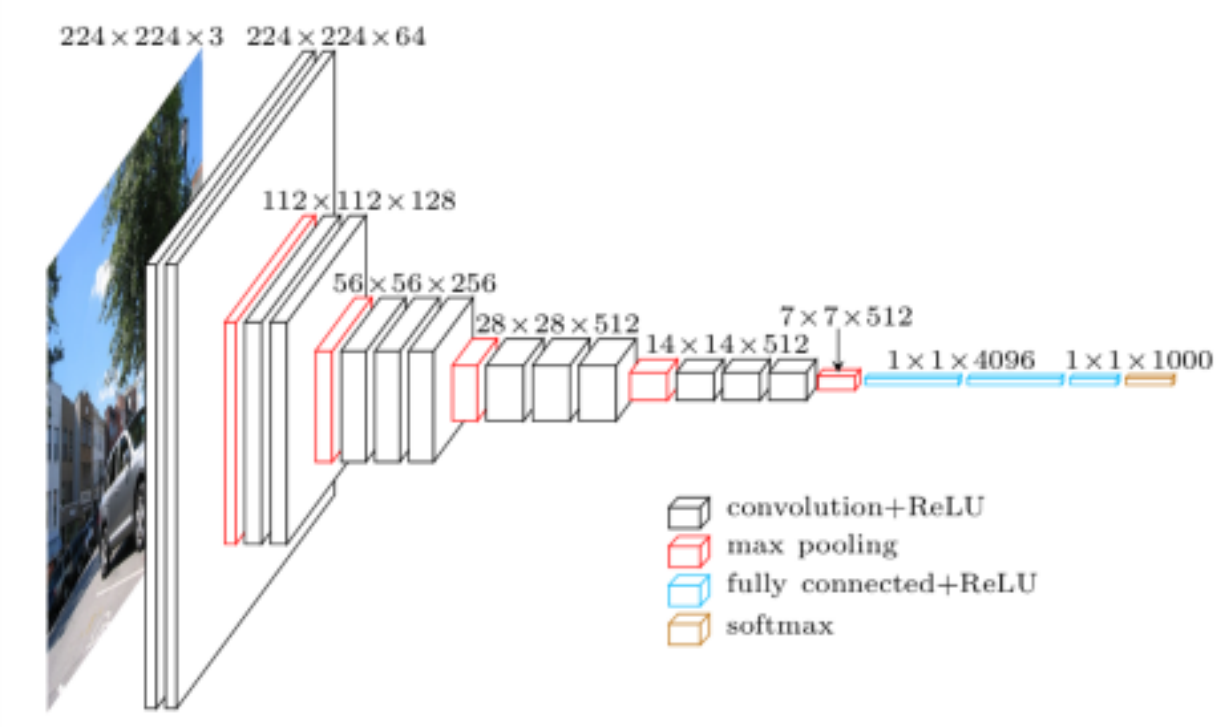


Figure 2: VGG-19 network consists of 16 convolutional layers, 5 pooling layers and 3 fully connected layers [8]

Network Analysis: Take image classification for example. For each layer, every hidden unit is looking for the image patches that maximizes the unit's activation function. This means that they are looking for the image patches that correspond most to the pattern, or feature of the units. The deeper the layer is, the larger region of the image it looks at, and the more complex feature it looks for. Therefore, along the processing hierarchy of the network, the input image is transformed into representations that are increasingly sensitive to the actual content of the image but become relatively insensitive to its precise appearance.

Convolutional Network Neuron Example

To visualize the image information that is encoded at different layers of the hierarchy one can perform gradient descent or other optimizers on a white noise image to find another image that matches the feature responses of the original image.

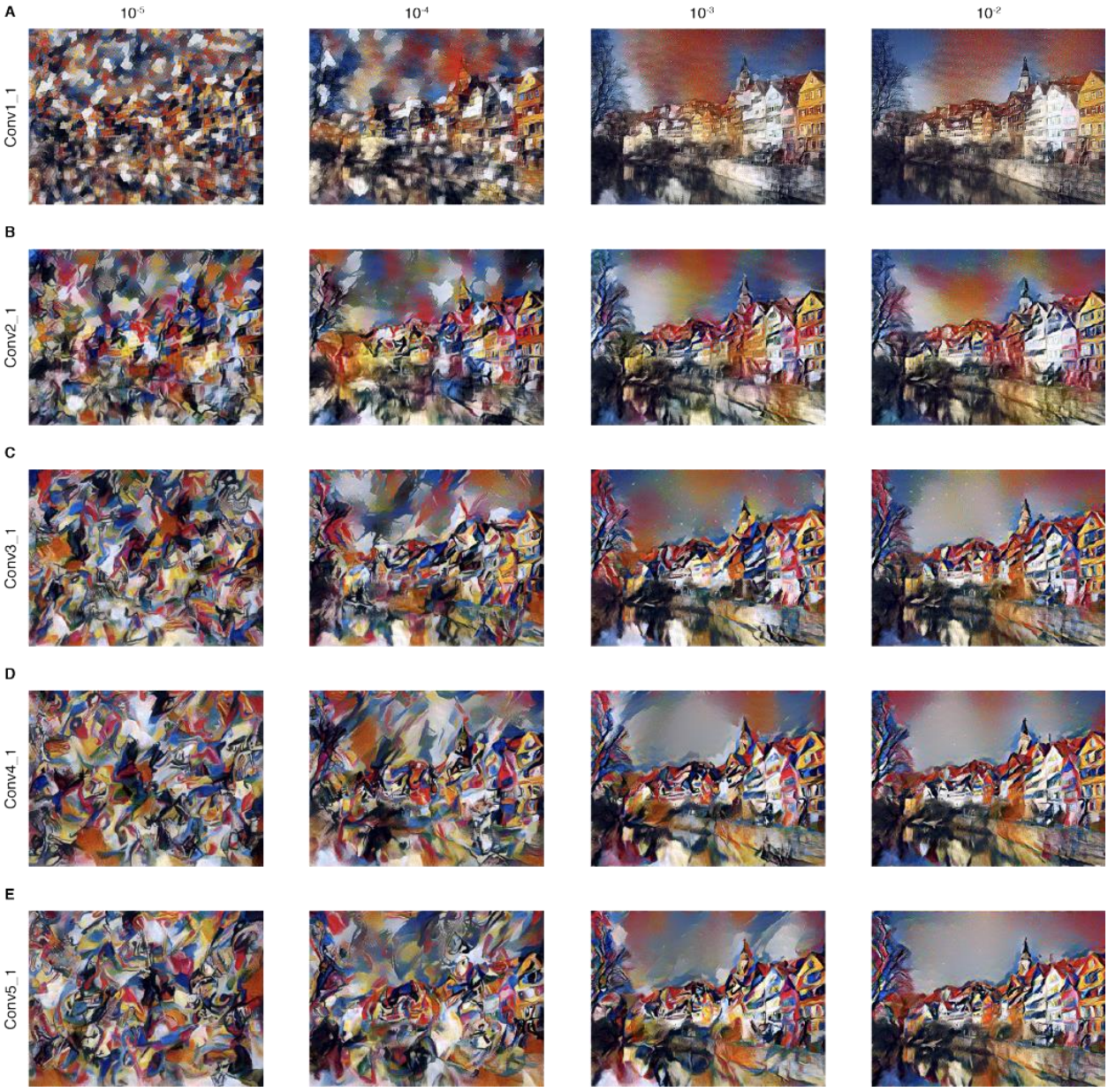


Figure 3: Detailed results for the style of the painting Composition VII by Wassily Kandinsky. The rows show the result of matching the style representation of increasing subsets of the CNN layers. We find that the local image structures captured by the style representation increase in size and complexity when including style features from higher layers of the network. This can be explained by the increasing receptive field sizes and feature complexity along the network's

processing hierarchy. The columns show different relative weightings between the content and style reconstruction. [1]

Content Representation

When we do the content representation, we need a criteria to measure how much information the representation image loses from the content image, and optimize the representation according to it. Content image is basically the for the object representation like person, dog and other objects. In this project, the content weight is the amount of content image to be perceived in the resultant image. Content weight blend specifies the coefficient of content transfer transfer layers. These hyper-parameters are tuned to have significant amount of content image in our target image. From VGG-19 network, relu4_2 and relu5_2 are used for content layers. We have used two methods used to preserve the color of the content image i.e. rgb2gray and gray2rgb.

Style Representation

Style ratio is the amount of style image in the resultant image, we have kept it to default 1e2. When we use style blend weights it represents the amount or the ratio of the content and the styling image in the output image. Usually, we keep the ratio equally weighted but doubling style has better effects than halving content. We have used 5 ReLu layers i.e. 'relu1_1', 'relu2_1', 'relu3_1', 'relu4_1', ‘relu5_1’.

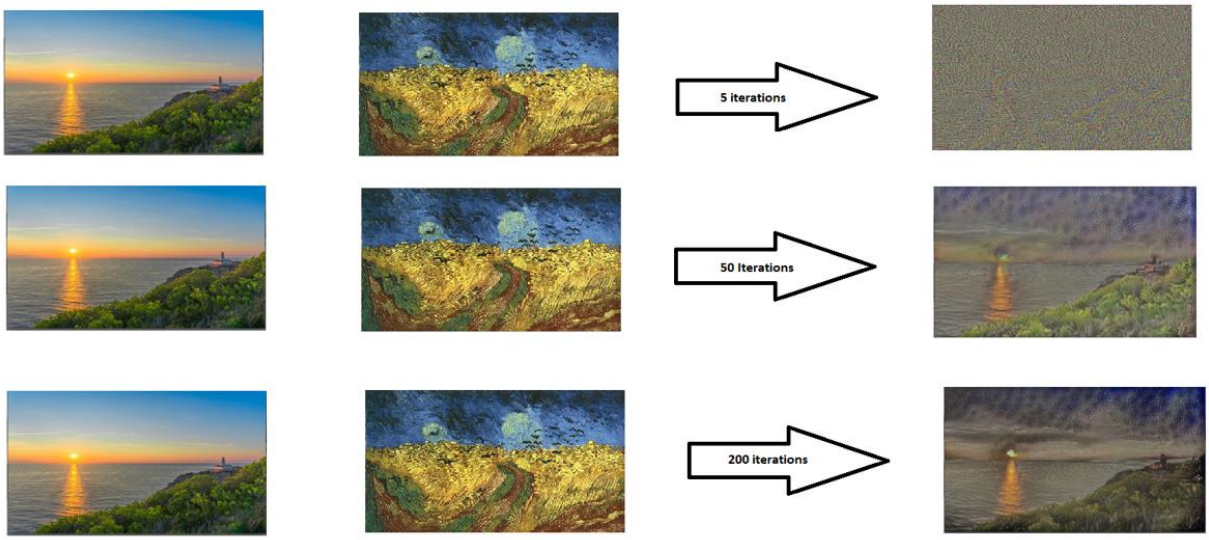
Style Transfer

When we do the transfer, we first initialize the image G as a white noise image with every pixel value generated randomly. Then, use optimizer, such as gradient descent, to minimize the value of the cost function. In this part, we took experiments on three factors which has influence on the generated image. First, we set the optimizing iteration times to several different numbers and compared the result. Second, we changed the weights of style layers and content layers and compared the results. Third, we adjusted the hyper-parameters for weights and biases and got several groups of outputs. All the results are shown in Result chapter.

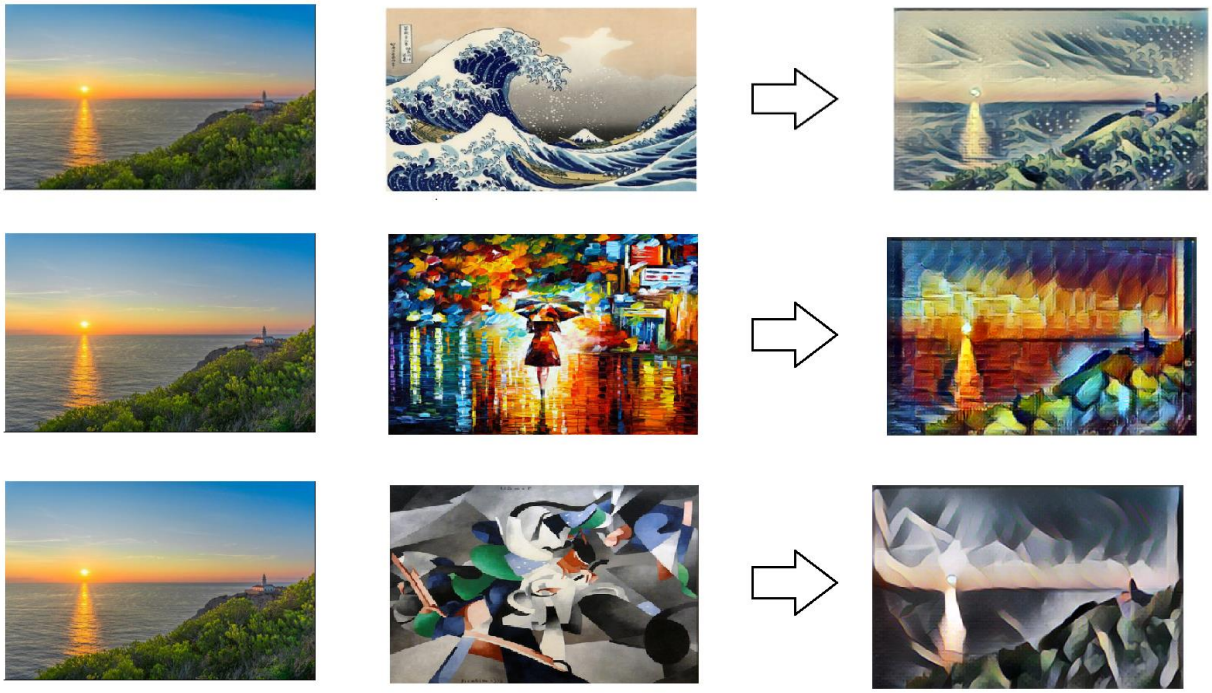
Our Results

General results on different images

We did experiments on many content images and style images, and we found that the content images with clear edges and contrast, and style images with smooth textures and bright colors can generate better results.



Results obtained on pre-trained model:



References

[1] L. Gatys, A. Ecker, and M. Bethge. A neural algorithm of artistic style. arXiv preprint arXiv:1508.06586

[2] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. arXiv:1409.1556 [cs], Sept. 2014. arXiv: 1409.1556. 3

[3] Justin Johnson, Alexandre Alahi, Li Fei-Fei. Perceptual Losses for Real-Time Style Transfer and Super-Resolution arXiv:1603.08155 [cs.CV]

[4] Neural Style by Anish Athalye: https://github.com/anishathalye/neural-style

[5] VGG Network https://www.quora.com/What-is-the-VGG