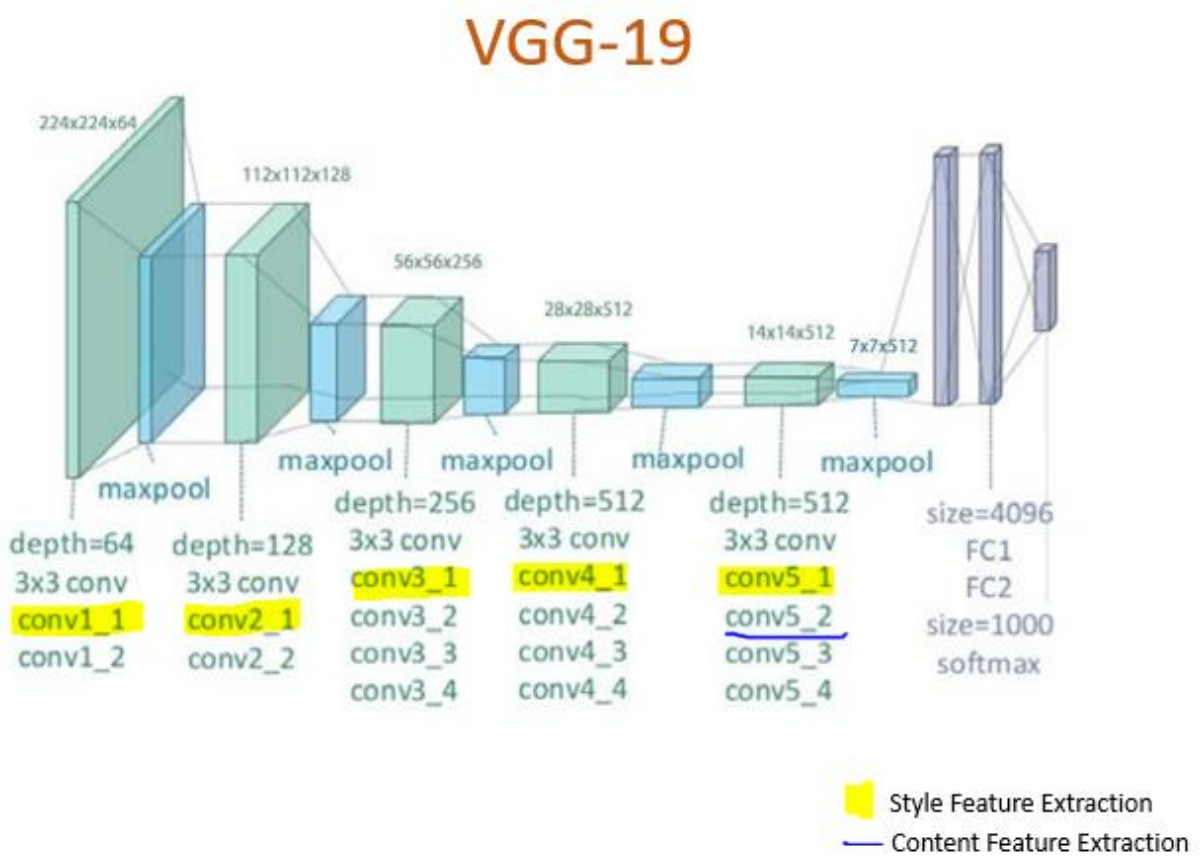
Artistic Neural Style Transfer with TensorFlow 2.0, Part 1: Theory

About VGG

In a deep neural network, where hundreds of layers are involved, you face the serious problem of vanishing gradients in which the accuracy of the model decreases with the increase in the number of layers. Hence, efficient networks like ResNet, DenseNet, VGG-Net, etc. were introduced. It is difficult and very time consuming to build dense neural network from scratch for every problem, and ResNet, DenseNet, and VGG-Net are some of the state-of-the-art networks that have been quite successful.

This guide uses a pre-trained VGG-19 model. Pre-trained models are built using transfer learning techniques, where the model uses knowledge to solve a problem (i.e., recognizing a boat) and applies it to a related but similar problem (recognizing a ship).

The VGG-Net model can recognize low-level features using shallow (earlier) layers and high-level features using deeper layers. The images below show the layer structure of the VGG-19 Network:



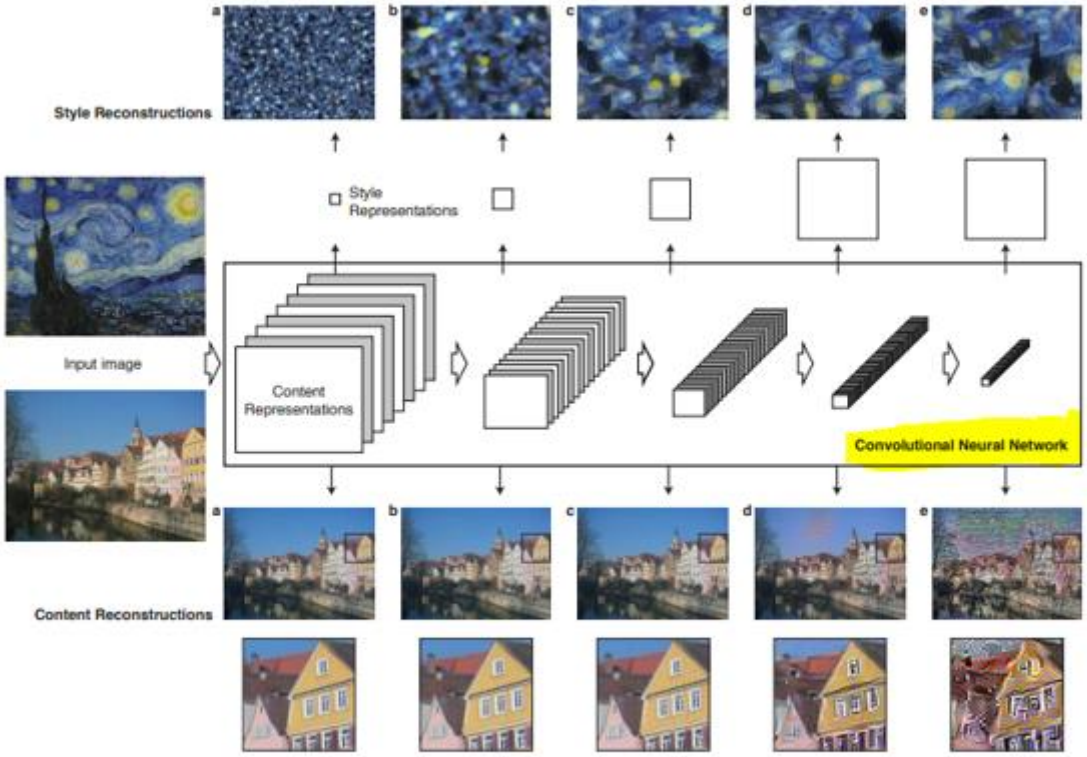


Image representations in a Convolutional Neural Network (CNN)

VGG Architecture

Now you know the building blocks of the VGGNet-19 model. The architecture of its variants is described in the image below.

VGG Net Architectures

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

maxpool layer uses 2x2 filter with stride = 2

Instead of using large filters with larger strides, this model uses smaller filters. For example, the three 3x3 convolution layers will incorporate three non-linear rectification layers instead of one 7x7 convolution layer. This will help the decision function to learn more features. Also, a stack of 3x3 layers will decrease the size in terms of weights, making the model less prone to overfitting.

In artistic neural style transfer, we use three images:

- Content image (image on which style is applied)
- Style image (image used as a style)
- Generated image (a raw image, which will contain the styled content image)

After selecting the images, pre-process them in terms of size, shape, and dimensions.

Intermediate layers are necessary to define the representation of content and style from the images. For an input image, try to match the corresponding style and content target representations at these intermediate layers.

Intermediate Layers for Style and Content

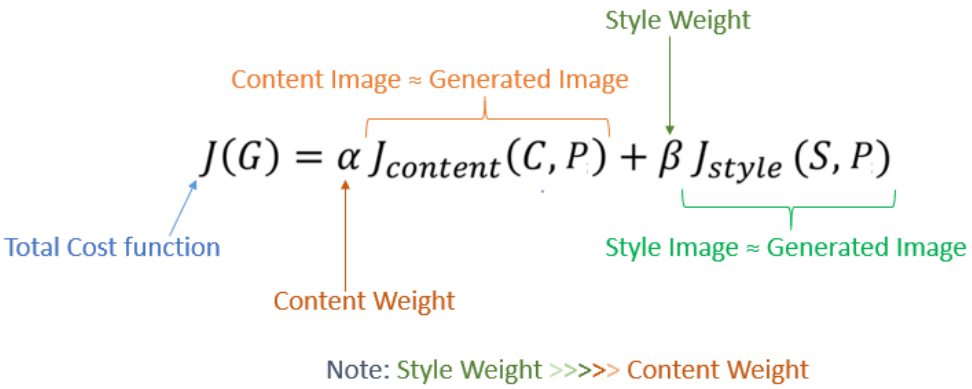
The convolution layers in VGG have the responsibility to separate the style and content of an image. You will use different intermediate layers to extract content and style information.

For the content layer, the second convolutional layer in block 5, block5_conv2 is used. Considering the fact that deeper layers in the network capture the objects and their arrangement in the input image, these are complex features to extract. And near the final layers of the CNN, the best features are found.

For the style layers, use the first convolutional layer in each block of layers, that is, block1_conv1 up to block5_conv5. CNN keeps learning features. At multiple layers, different patterns are detected. Starting layers will detect simple diagonal lines, first layer edges, then certain patterns, and so on.

Cost Function

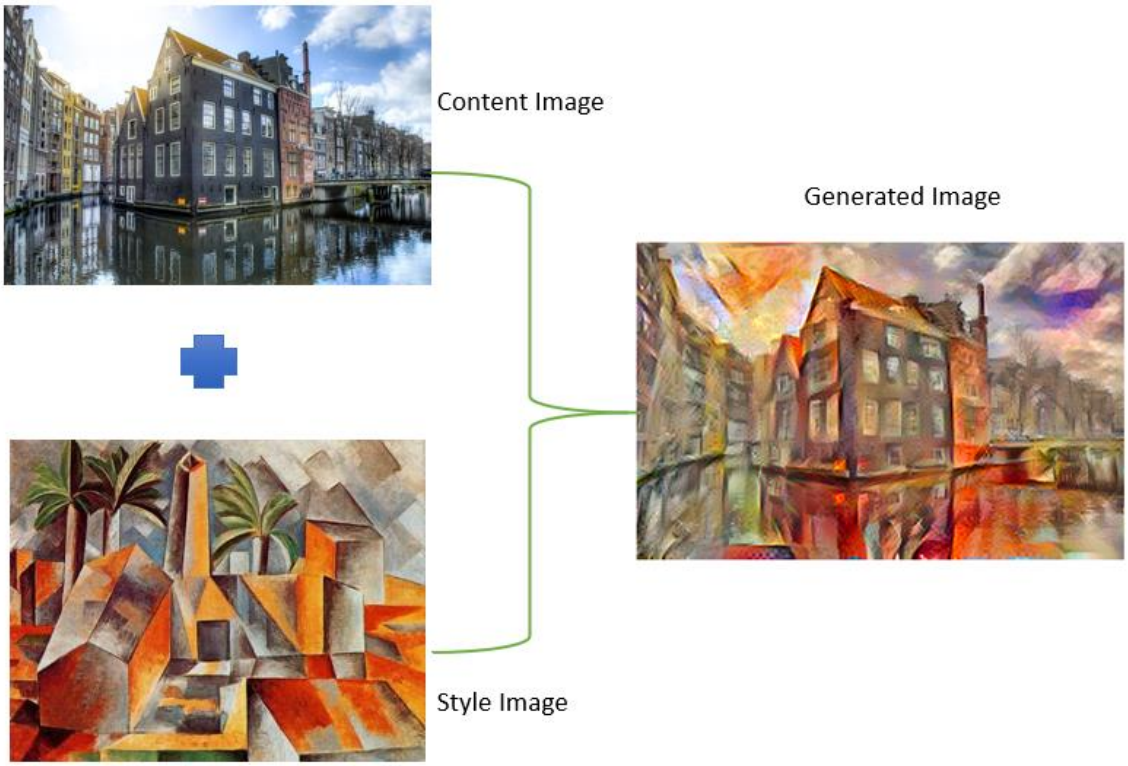
In style transfer, a neural network is not trained. Instead, its weights and biases are kept constant, and an image is updated by changing/modifying the pixel values until the cost function is optimized (reducing the losses). It makes sure that the "content" in the content image and the "style" in the style image are present in the generated image.



Usually, in deep learning, we have only one loss function. However, in neural style transfer, we are generating a new image from two images, so we need more loss functions to generate a new image. We will discuss various loss functions such as content loss, style loss, and variation loss.

There are many approaches to mathematical notation. The equations in this guide are taken from Gatsy et al. (some notations might differ).

Below is a simple representation of how the new image will be generated from the content and style images.



This function helps to check how similar the generated image is to the content image. It gives the measure of how far (different) are the features of the content image and target image. The Euclidean distance is calculated. It is defined as follows:

$$L_{content} = \sum_{l} \sum_{i,j} (\alpha C_{i,j}^{l} - \alpha P_{i,j}^{l})^2$$

Content Loss

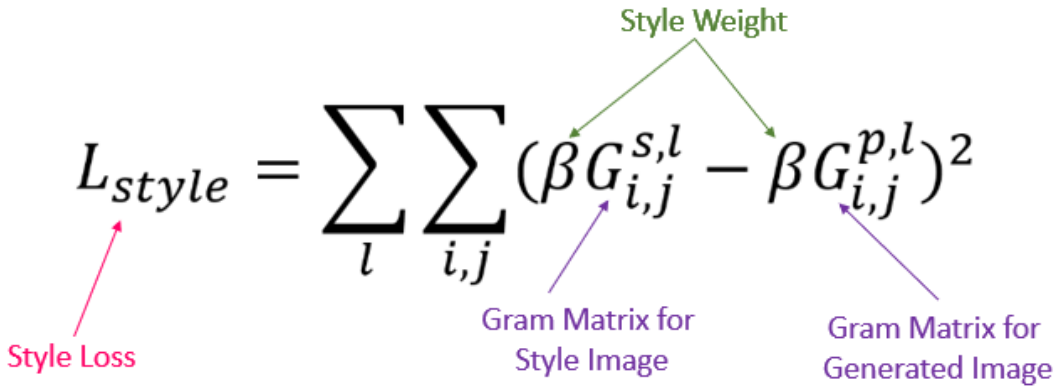
Content Weight

Content Image

Generated Image

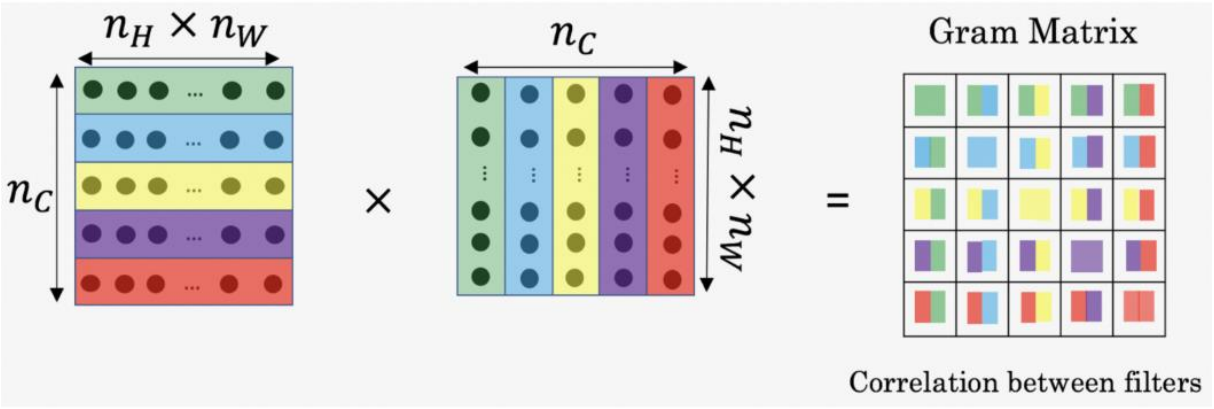
Content Loss, squared-error loss between the two feature representations of C & P

Style Loss measures how different the generated image, in terms of style features, is from your style image. But it's not as straightforward as content loss. The style representation of an image is given by Gram Matrix.



Style Loss, defined as the squared-loss error between the gram matrices between s & p

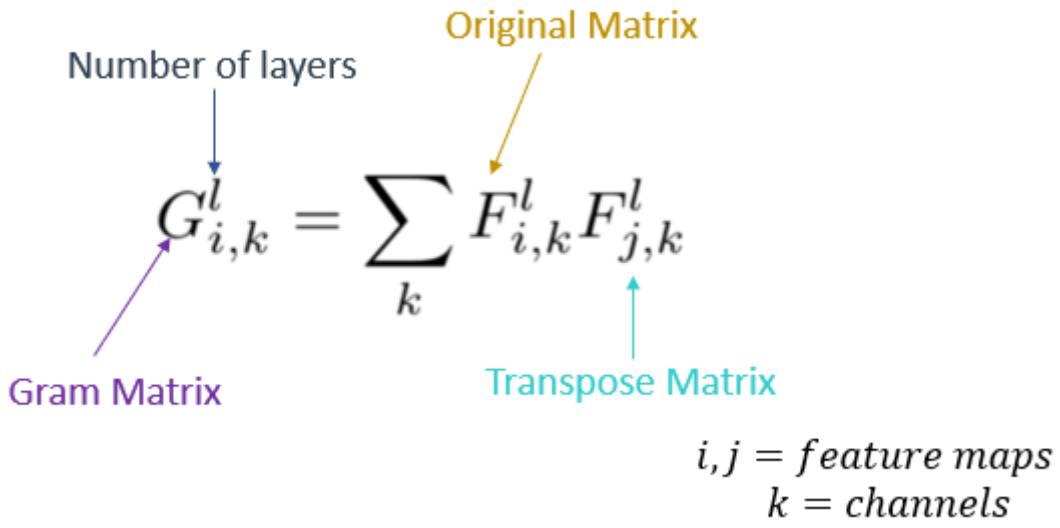
s = Style Image
p = Generated image



Gram Matrix is only concerned with whether the stylish features are present in image weights, textures, and shapes. Hence, it is the best choice. The Gram Matrix G is the set of vectors in a matrix of dot products. For a particular layer, the diagonal elements of the matrix will find how active the filter is. An active filter will help the model find wether it contains more horizontal lines, vertical lines, or textures.

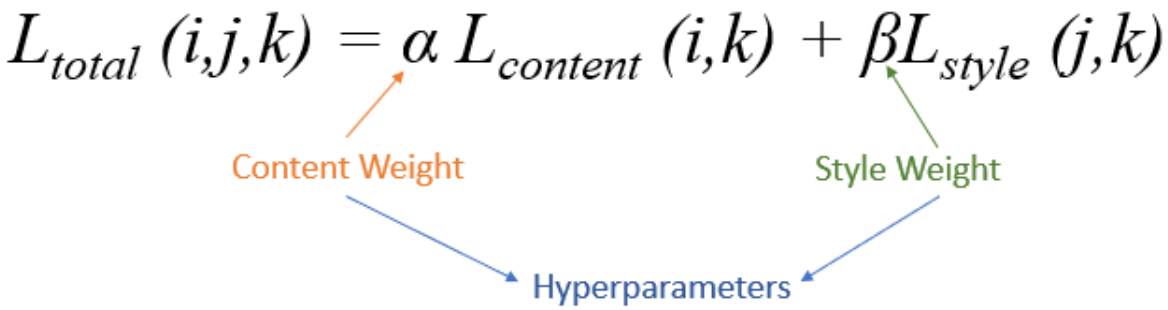
To get the results, the matrix is multiplied by its transposed matrix.

Its equation is as follows:



The GM helps you find how similar Fik is to Fjk. If the dot product is large, they are highly similar.

Finally, total loss will minimize the weighted average.



Variation loss was introduced to avoid highly noisy outputs and overly pixelated results. The main purpose of variation loss is to maintain smoothness and spatial continuity.

Total Variation Loss:

$$E_{tv}(\mathbf{w}) = \sum_{x_k \in \mathcal{I}} \Theta\left(\overline{f}(x_k; \mathbf{w})\right) = \sum_{x_k \in \mathcal{I}} |\nabla f(x_k; \mathbf{w})|$$
$$= \sum_{x_k \in \mathcal{I}} |\nabla_X f(x_k)| + |\nabla_Y f(x_k)|.$$

The change in the combination of images minimizes the loss so that you can have an image combination of both the Picasso painting and the input image. To make the losses bit smaller, an optimization algorithm is used. The first order algorithm uses a gradient to minimize the loss function, famously known as gradient descent. The Adam optimization shows faster results in style transfer.