A

Project report on

Image Style Transfer by using Convolutional Neural Network

by

Vikas Bhadoria (vikasbadhoria69@gmail.com)

Department of Informatics engineering

Wismar University of Applied Sciences: Technology, Business and Design

Wismar, Germany

ABSTRACT

In this project, I have demonstrated how to use deep learning to compose images in the style of another image . This is known as 'neural style transfer'. This is a technique that produces new images of high perceptual quality that combine the content of an arbitrary photograph with the appearance of numerous well known artworks.

Neural style transfer is an optimization technique used to take three images, a 'content image', a 'style reference image' (such as an artwork by a famous painter), and the input image we want to style and blend them together such that the input image is transformed to look like the content image, but "painted" in the style of the style image.

The results achieved by this technique provide new insights into the deep image representations learned by Convolutional Neural Networks and demonstrate their potential for high level image synthesis and manipulation.

The project is coded in python and built on top of Pytorch framework. The project can be divided into 5 steps when it comes to practical implementation:

1. Visualize data
2. Preprocessing/preparing our data
3. Set up loss functions
4. Create model
5. Optimize for loss function

Introduction

Transferring the style from one image onto another can be considered a problem of texture transfer. In texture transfer the goal is to synthesise a texture from a source image while constraining the texture synthesis in order to preserve the semantic content of a target image. For texture synthesis there exist a large range of powerful non-parametric algorithms that can synthesise photorealistic natural textures by resampling the pixels of a given source texture. Most previous texture transfer algorithms rely on these nonparametric methods for texture synthesis while using different ways to preserve the structure of the target image. Although these algorithms achieve remarkable results, they all suffer from the same fundamental limitation: they use only low-level image features of the target image to inform the texture transfer.

To generally separate content from style in natural images is still an extremely difficult problem. However, the recent advance of Deep Convolutional Neural Networks has produced powerful computer vision systems that learn to extract high-level information from natural images. Convolutional Neural Networks trained with sufficient labeled data on specific tasks such as object recognition learn to extract high-level image content and even other visual information processing tasks including texture recognition and artistic style classification. This project shows how the generic feature representations learned by high-performing Convolutional Neural Networks can be used to independently process and manipulate the content and the style of natural images.

Problem Statement and Objectives

To build a target image using content from one input image and style of another.

The objective of this project is:

To take a base input image, a content image that has to be matched, and the style image that has to be matched. Then transforming the base input image by minimizing the content and style distances (losses) with backpropagation, creating an image that matches the content of the content image and the style of the style image.

Deep image representations

The project makes use of 'transfer learning' technique which allows us to use the parameters of a deep and highly trained model which has been extensively trained on huge datasets. This task is otherwise very daunting because of the limited processing power. The model used in this project is the VGG-19 network model, which was trained to perform object recognition and localisation. The model is publicly available and can be explored. VGG19 is trained on 14 million different images and is able to classify 1000 different object categories. This project uses feature space provided by a normalised version of the 16 convolutional and 5 pooling layers of the 19-layer VGG network. We use the 16 layer feature extraction part of VGG-19 and freeze its parameters so as to use the same weights as the model has achieved after extensive training. There is no use of any fully connected layer of VGG19 as for this project only feature extractions of an image are required, not the classification part .

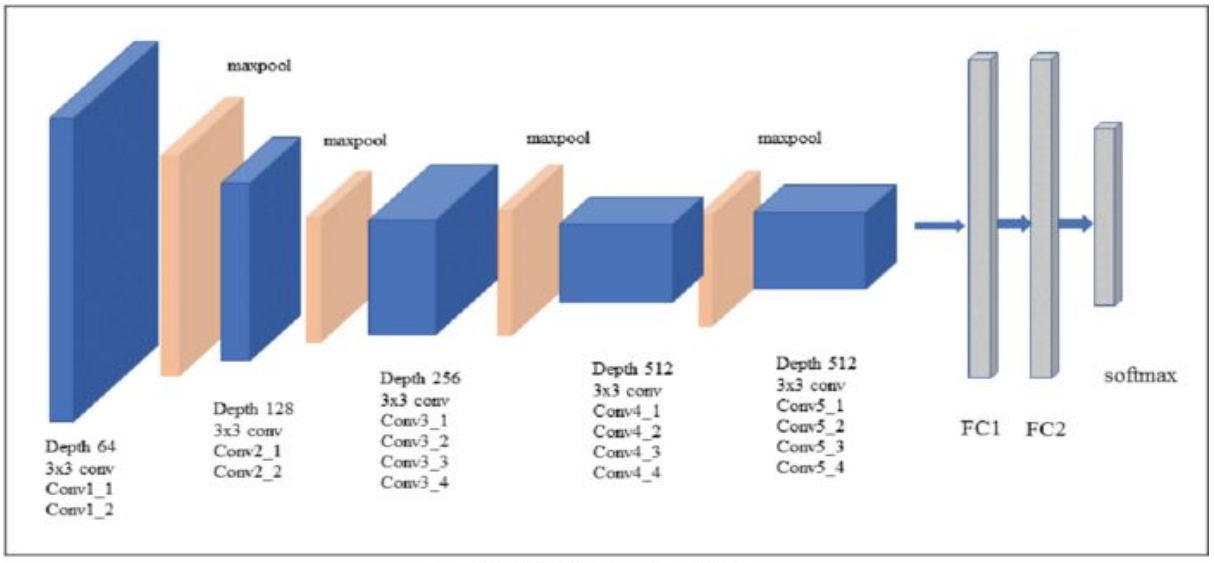


Fig. 3. VGG-19 network architecture

Define content and style representations

In order to get both the content and style representations of our image, some intermediate layers within the model can be used. Intermediate layers represent feature maps that become increasingly higher ordered as one goes deeper. These intermediate layers of VGG-19 are necessary to define the representation of content and style from the images. For an input image, the idea is to try and match the corresponding style and content target representations at these intermediate layers.

When Convolutional Neural Networks are trained on object recognition, they develop a representation of the image that makes object information increasingly explicit along the processing hierarchy. 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 invariant to its precise appearance. Thus, higher layers in the network capture the high-level content in terms of objects and their arrangement in the input image. In contrast, reconstructions from the lower layers simply reproduce the exact pixel values of the original image. Therefore the feature responses in higher layers of the network are referred to as the content representation. For this project the ‘conv4_2’ layer was found optimal for content representation. The representations of content

and style in the Convolutional Neural Network are well separable. That is, one can manipulate both representations independently to produce new meaningful images.

Another important factor in the image synthesis process is the choice of layers to match the content and style representation on. The style representation is a multi-scale representation that includes multiple layers of the neural network. The number and position of these layers determines the local scale on which the style is matched, leading to different visual experiences . Matching the style representations up to higher layers in the network preserves local image's structures on an increasingly large scale, leading to a smoother and more continuous visual experience. Thus, the visually most appealing images are usually created by matching the style representation up to high layers in the network, which is why for all images shown we match the style features in layers 'conv1 1', 'conv2 1', 'conv3 1', 'conv4 1' and 'conv5 1' of the network.

Define and create our loss functions (content and style distances)

Content Loss:

Content loss definition is actually quite simple. The network is passed with both the desired content image and the base input image that needs to be transformed. This will return the intermediate layer outputs from the model. Then simply take the euclidean distance between the two intermediate representations of those images.

More formally, content loss is a function that describes the distance of content from our input image 'x' and the content image 'p' . Let C be a pre-trained deep convolutional neural network. Let X be any image, then C(x) is the network fed by X and P be the content image, then C(p) is the network fed by P.

Let $F^l_i(x) \in C(x)$ and $P^l_i(p) \in C(p)$ describe the respective intermediate feature representation of the network with inputs x and p at layer l . Then we describe the content distance (loss) formally as:

$$L^l_{content}(p, x) = \sum_{i,j}(F^l_{ij}(x) - P^l_{ij}(p))^2$$

Then backpropagation is performed in the usual way such that it minimizes this content loss.Thus keep changing the initial image until it generates a similar response in a certain layer (defined in content_layer) as the original content image.

Style Loss:

Computing style loss follows the same principle, this time feeding the network with base input image and the style image. Instead of comparing the raw intermediate outputs of the base input image and the style image, the Gram matrices of the two outputs are compared.

Mathematically, one can describe the style loss of the base input image 'x' and the style image 'a' as the distance between the style representation (the gram matrices) of these images. The style representation of an image is described as the correlation between different filter responses given by the Gram matrix G^l, where G^l_i is the inner product between the vectorized feature map i and j in layer l.

To generate a style for the base input image, gradient descent is performed to transform it into an image that matches the style representation of the original style image. This is done by minimizing the 'mean squared distance' between the feature correlation map of the style image and the input image. The contribution of each layer to the total style loss is described by

$$E_l = \frac{1}{4N_l^2 M_l^2}\sum_{i,j}(G^l_{ij} - A^l_{ij})^2$$

where G^l_i and A^l_i are the respective style representation in layer l of input image ‘x’ and style image ‘a’. Nl describes the number of feature maps, each of size Ml=height*width. Thus, the total style loss across each layer is

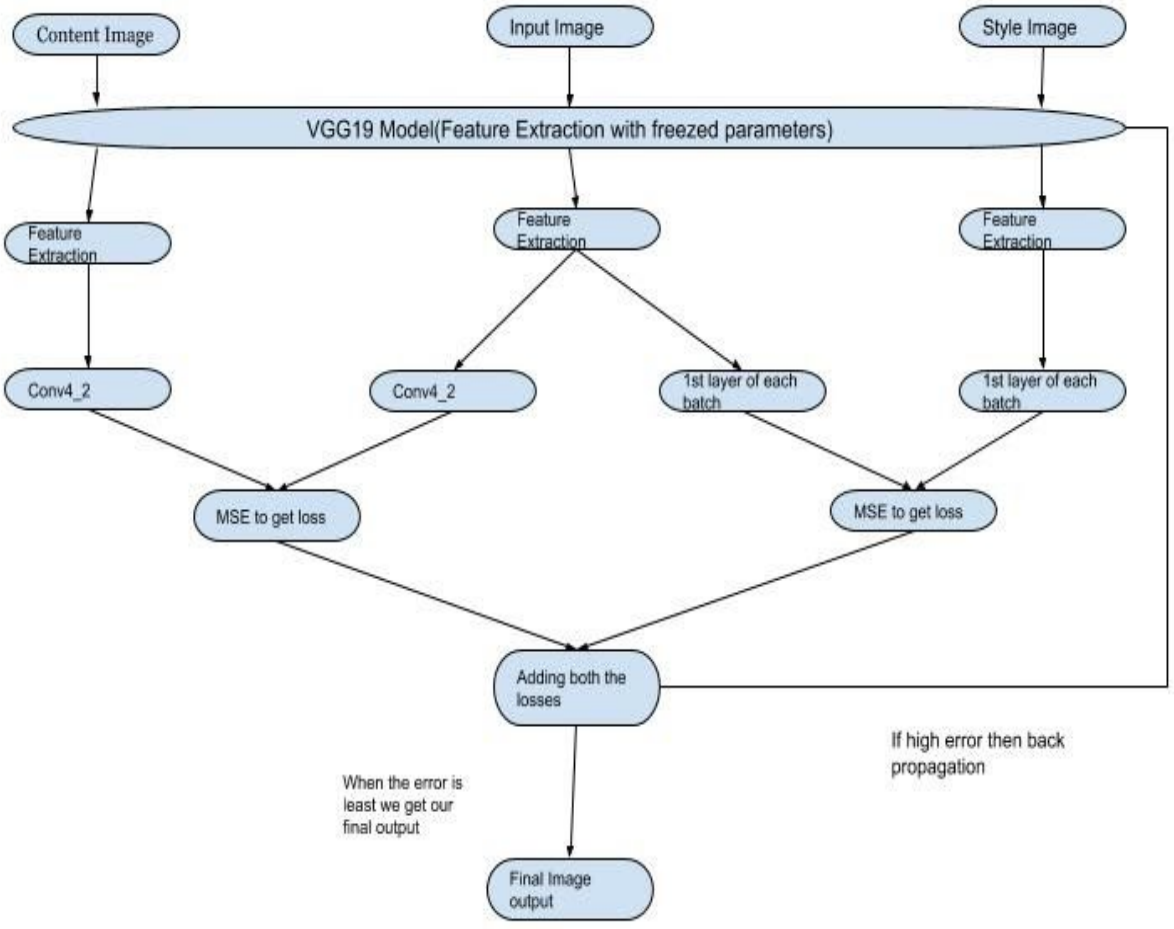
$$L_{style}(a, x) = \sum_{l \in L} w_l E_l$$

where the contribution of each layer’s loss by some factor wl can be weighed.

Run Gradient Descent

For this project the ‘Adam’ optimizer is used in order to minimize the loss. Iteratively the output image is updated such that it minimizes our loss. The weights associated with the model network are not changed, but instead the input image is trained to minimize loss.

The final workflow for this project



Final Output

Below, the first image is the content image followed by style image and the final image is the output image.

It can be seen how the final output image has been created using the content of the first image and style of the second.