Image Processing with Convolutional Neural Networks (CNN's) and using VGG19 to implement Van Gogh style on any picture.

Stefanos Baklavas 1115201700093

We consider image transformation problems, where an input image is transformed into an output image.Recent work has shown that the best way to do that is by using Convolutional Neural Networks (CNN's) pre-trained by millions of labeled images.We state the benefits of CNN's for image processing , analyze their architecture and their layer responses and cost functions.We also show results of image style transfer using VGG-19 with the parameters proposed by Gatys et al.Different layers are also used to extract results and tune the style image and content image weights to see their effect.

1 Introduction

Image processing is a fast developing science field with a wide variety of new methods taking the place of some older traditional as time passes by. As a result transferring the style of an image into another is now possible and it can be implemented using CNN's. In this paper we use VGG-19 newral network to implemented the Van Gogh style into an image of our choice.

An older approach would require filtering the image 'by hand' using methods such as segmentation , blurring filters , noise and implementing algorithms to change the texture and the shapes of the content image to make it look like it was painted by Van Gogh. This would be almost impossible without reshaping

the image with our hand because it is very difficult to make algorithms that for example recognize the sky, the sun, houses or other objects and then reshaping them in a way that matches Van Gogh's style. Even in this way we would have missed a huge amount of parameters that are to difficult for a person to handle.

In this paper we will present why CNN's are the best for the job of style transfer and their differences from the 'traditional' Neural Networks which are due to the convolutional layers that the first's have.We will use VGG-19 CNN developed by Oxford University in order to transfer the style of an image (in this case Van Gogh's Starry Night) on another content image (in this case a picture of Thames River). We will represent the error functions used in this implementation and the L-BFGS optimization algorithm. We will also explain the differences in VGG-19 layers and present the effect that each layer has on style transferring. Finally different combinations of layers will be tested in order to see which output result has the best aesthetics . Although the Gatys et al suggest using only the first filter from layers conv1-1 to conv5-2 we found than the combination of all layers from 1-1 to 5-2 produces the best output image compared to that of taking all first or second convolutional filters from each layer. Conv5-2 refers to the second convolutional filter of the fifth layer

2 CNN's and the VGG-19 architecture

2.1 Why CNN's

CNNs are used for image classification and recognition because of its high accuracy. It was proposed by computer scientist Yann LeCun in the late 90s, when he was inspired from the human visual perception of recognizing things. The CNN follows a hierarchical model which works on building a network, like a funnel, and finally gives out a fully-connected layer where all the neurons are connected to each other and the output is processed.In our implementation we don't use any fully connected layer as we reach until layer conv5-2.

In the beginning CNN's wasn't the first thought to implement Van Gogh's style on a given picture.Our first thoughts was to pass the picture through filters such as blurring filters to make it look like an oil painting at first, then put an amount of noise in order to transform it into Van Gogh's paintinng texture and then make some kind of segmentation in order to distinguish the sky or some objects like houses and mountains in so as to implement Van Gogh's famous turbulent effect. This kind of approach may cannot be accomplished satisfactorily by traditional algorithms. The biggest difficulty would be the for segmentation and turbulent effect to be implemented and as those problems weren't enough we would also

have to color the segmented image like Van Gogh. So we can understand that CNN's may are a one way road as they take into account a huge amount of parameters and experience has shown that are ideal for image processing.

A simple description of CNN's is the following.

- CNN's are very good in detecting patterns and making a sense of them. That's what make makes them useful for image analysis.
- The thing that separates them from 'traditional' Neural Networks is that their hidden layers are convolutional layers.
- They also include other types of layers (max pooling, Relu etc.) which are used as filters for activation and input passing.
- A convolutional layer:
  - Receives input
  - Transforms the input
  - Outputs the transform to the next layer

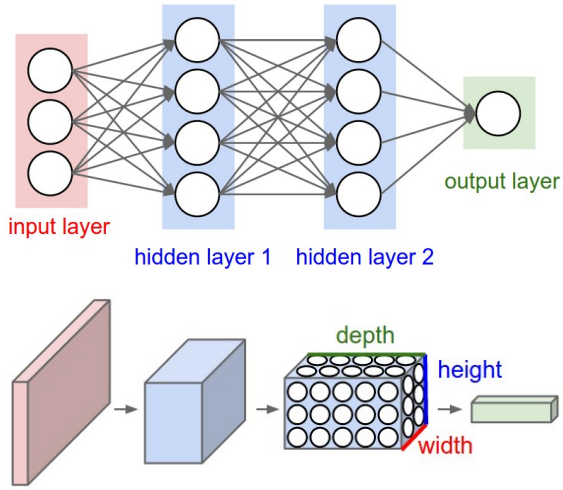
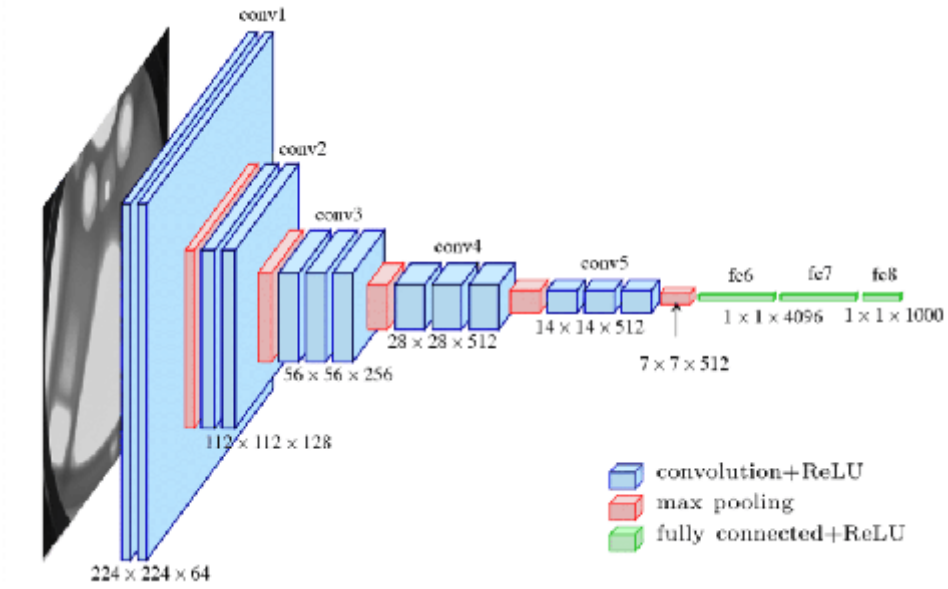


Figure 1: In these two figures, we can see a comparison of a Neural Network and a CNN. The difference is that CNNs have Convolutional hidden layers combined with an activation function, usually ReLU. We can see that each Convolutional layer, instead of being just a node implementing a function on the input, has three dimensions (height, width, and depth) where each depth represents a different Convolutional filter.

2.2 The VGG-19 Architecture

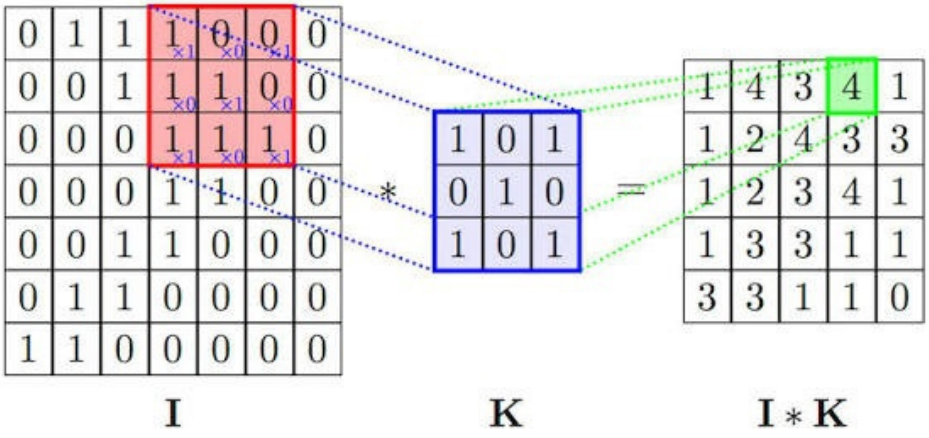
VGG19 is a deep convolutional neural network built at the University of Oxford (see the paper: Very Deep Convolutional Networks for Large-Scale Image Recognition). It has been trained on the ImageNet dataset: 14-million images from 1,000 categories. VGG19's primary purpose is to identify objects in images,and each architecture looks like like this:



VGG-19 is a variant of VGG model which in short consists of 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 SoftMax layer). The final layer outputs the probabilities of each of the 1,000 output categories. But that doesn't concern us and, in fact, to speed things up we won't even be including those layers in our implementation. Those layers weren't also included at the paper "A Neural Algorithm of Artistic Style" from Gatys et al. So the layers that will be used are from 1-1 to 5-2 without the max pooling ones.

2.3 How a Cunvolutional layer works

A convolutional layer works like passing the image through a filter in order to reconstruct it with the feature map that each layer provides. This feature space is built on top of the filter responses in each layer of the network. It consists of the correlations between the different filter responses over the spatial extent of the feature maps.Below we can see an example of how a convolutional layer works.



3 Implementation and methods

3.1 Procedure

The input images are two. One that will be used in order to extract it's style and one that will be used to extract it's content. These two images of our choice will be combined in order to create a third that matches the style of the first and the content of the second as we can see in figure 4.5 where we match Van Gogh's Starry Night style with the content of a picture of Thames River.

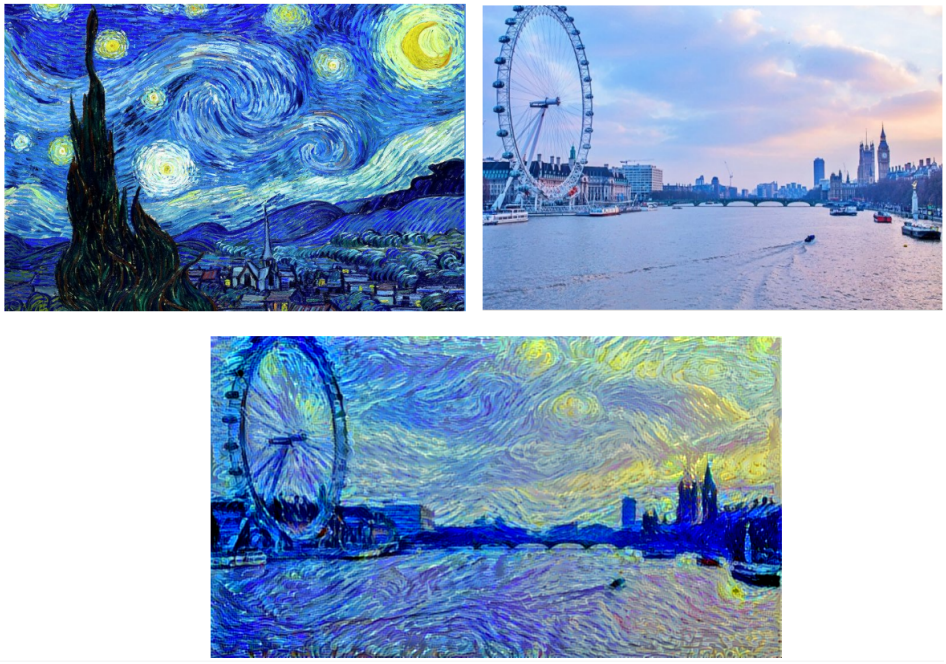


Figure 2: We combine the style of Starry Night with the content of Thames River to create the third image

Before we pass each input image through the network a pre-processing is required because VGG-19 takes only 224 x 224 RGB images, and so we need to transform the two images into this form. After that we will extract the content of the given picture and the style of the artwork by using methods that will be described later. Finally the L-BFGS optimizer will be used in order to reduce the total cost of style and content and produce the final image.

3.2 Extracting Content

To extract the content of a given image(in this case a photo of Thames River) we feed the image into the model end we generate output from layer 5-2. This choice was made because when another layer between 1-1 and 5-2 was used we had a result that was very close to the initial image and that's not what we want as we want to extract the most 'important' characteristics of it. The extracted content will be painted in an initially blank canvas where we will shape the final output of the two images.

- Run the content image p through the model to get the activation output of convolution layer 5-. Let's term that output P5-2.
- Run the (initially random) target image x through the model to get the output of the same layer 5-2. Let's term that output F5-2
- Calculate the error ("content cost") between the two outputs. We can use simple squared error cost function:

$$\sum (P_{5-2} - F_{5-2})^2$$

- Tweak the target image a little to reduce the error. We back-propagate the error through the model back to the target image, and use that as the basis of our tweaking.
- Repeat steps 2-4 until satisfied. A process of gradient descent.

3.3 Extracting Style

The process of extracting the style from an image is very similar. There are two differences:

- Instead of using just one convolution layer we are going to use five. Gatys et al. use a blend of layers 1-1,1-2, 2-1,2-2,3-1,3-2„3-3, 4-1,4-2,4-3 and 5-1. All five layers contribute equally to the total style cost.
- Before we pass the outputs of each layer into the squared error cost function, we first apply a function called the Gram matrix. The Gram matrix looks very simple but it's clever and very subtle.Gam matrix helps us to find correlations between the different filter responses as it applies a dot product between the output of two different layers.

$$Gl_{ij}$$

is the inner product between the vectorised feature map i and j in layer l.

$$G_{ij} = \sum_{k} (F_{ik} - F_{jk})$$

Bellow we can see a visualization of Gam matrix dot product:

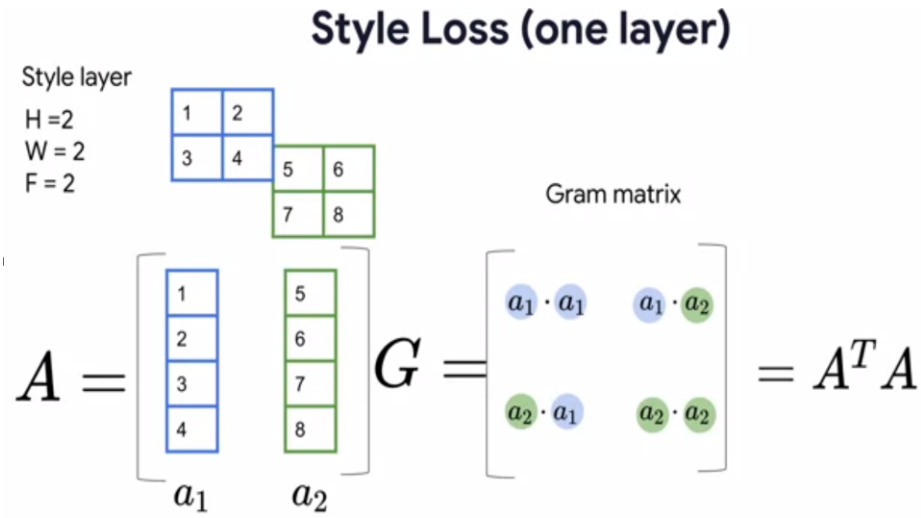


Figure 3: Gam Matrix

3.4 Content + Style

To paint the final image we need to get style-cost and content-cost and combine them to a final image. This is achieved by iteratively tweaking the canvas like before using an optimization function. In this iplementation we follow Gatys et al recommendation and also use the L-BFGS optimisation algorithm. We jointly minimise the distance of a white noise image from the content representation of the photograph in one layer of the network and the style representation of the painting in a number of layers of the CNN. So let $\vec{p}$ be the photograph and $\vec{a}$ be the artwork. The loss function we minimise is:

$$L_{total}(\vec{p}, \vec{a}, \vec{x}) = aL_{content}(\vec{p}, \vec{x}) + bL_{content}(\vec{a}, \vec{x})$$

where α and β are the weighting factors for content and style reconstruction respectively.In the next section we will compare results taken by using different

layers for style and always layer 5-2 for content as when another layer was used we were receiving an image that matched exactly the initial content image.

4 Results

In this section we will do a comparison between output images that we took from different layers. In all cases we used layer 5-2 to extract content so the comparisons are between the different style layers and between the weights for style and content. In figure 4 and figure 5 set of examples style-weight = 2 and content weight = 5 ,while in figure 6 we do weight tuning.

In figure 4 we observe that the more layers of the CNN we use the closer the style is to Van Gogh's Starry Night.

In figure 7 we can see the output of each of the layers 1-1,2-1,3-1,4-1,5-1

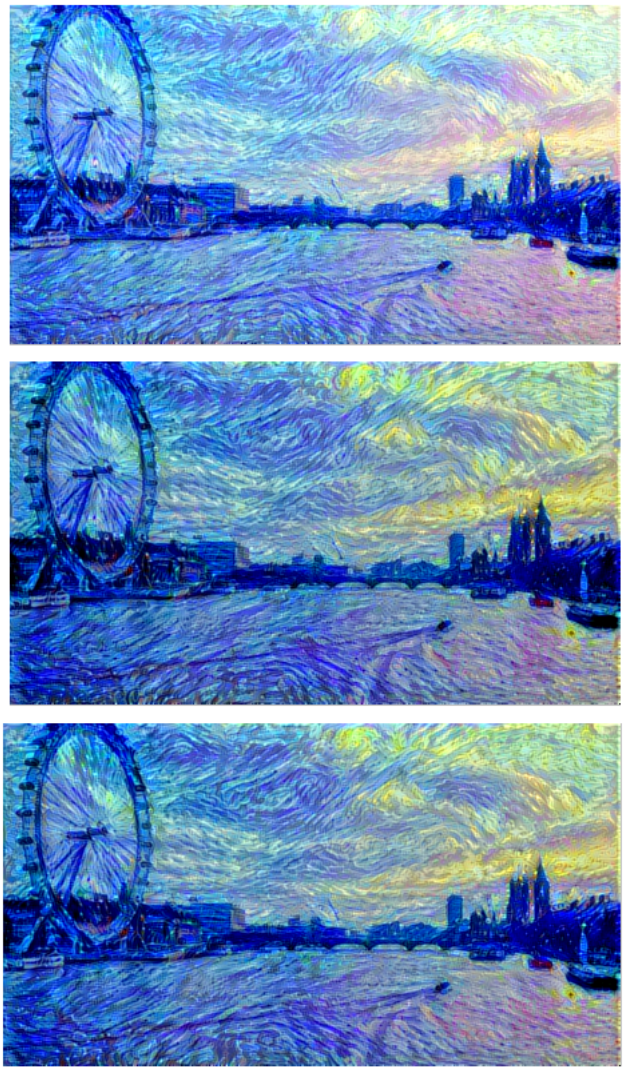


Figure 4: This image was created using layers: style(1-1,2-1,3-1,4-1,5-1),the second using style(1-1,1-2,2-1,2-2,3-1,3-2,4-1,4-2,5-1,5-2) and the third using style(1-1,1-2,2-1,2-2,3-1,3-2,3-3,4-1,4-2,4-3,5-1,5-2).

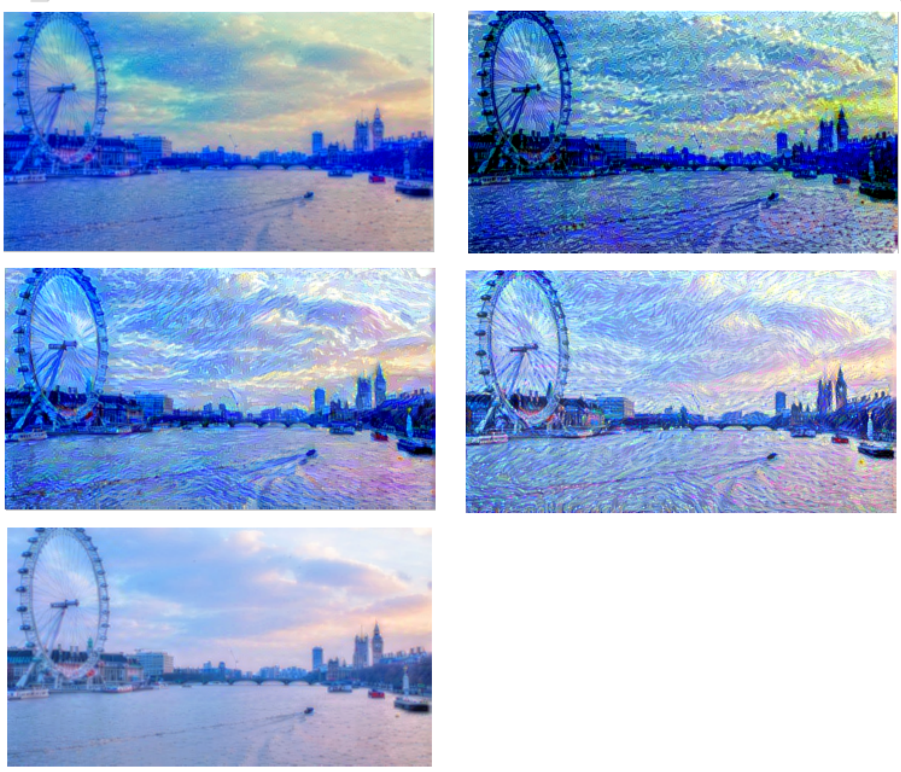


Figure 5: Bellow we can see the output of each of the layers 1-1,2-1,3-1,4-1,5-1 .We can see that as we go deeper into the network's layers we extract more sophisticated patterns (except for layer 5-1).

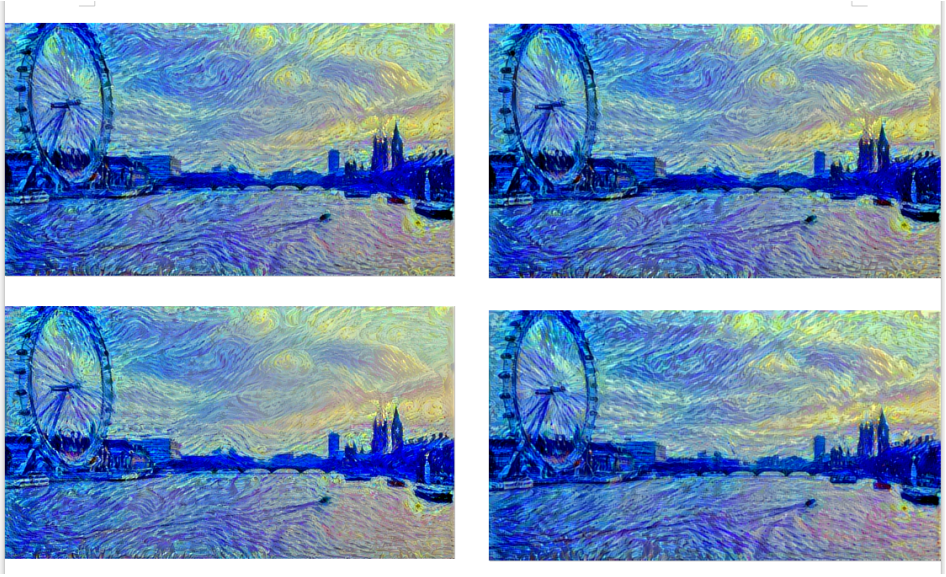


Figure 6: in this set we used all layers from 1-1 to 5-2 to extract style and we tuned the style and content weights differently.In the upper left photo style = 5 and content = 5.In the upper right photo style = 5 and content= =2.In the down left photo style = 8 and content = 2.In the down right photo style = 2 and content = 8.