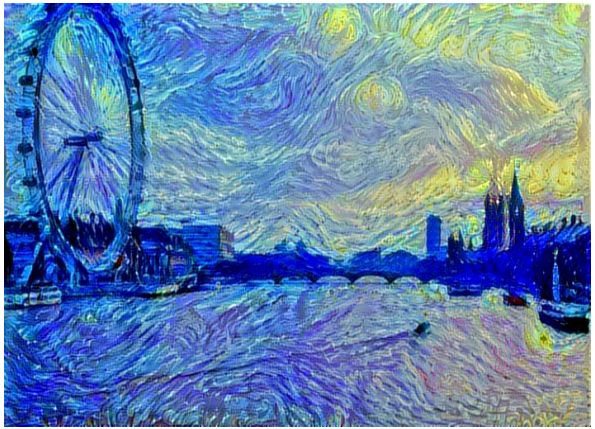
# Image Processing 2021

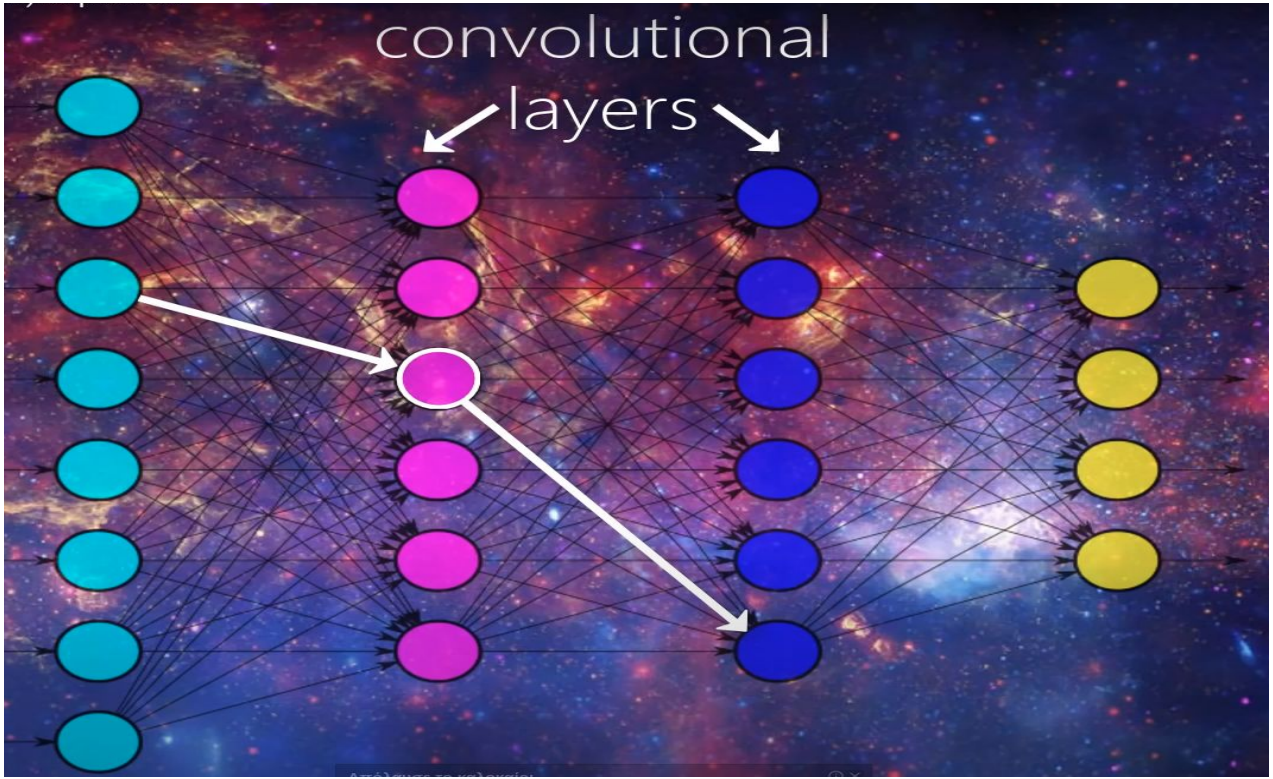
Stefanos Baklavas

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

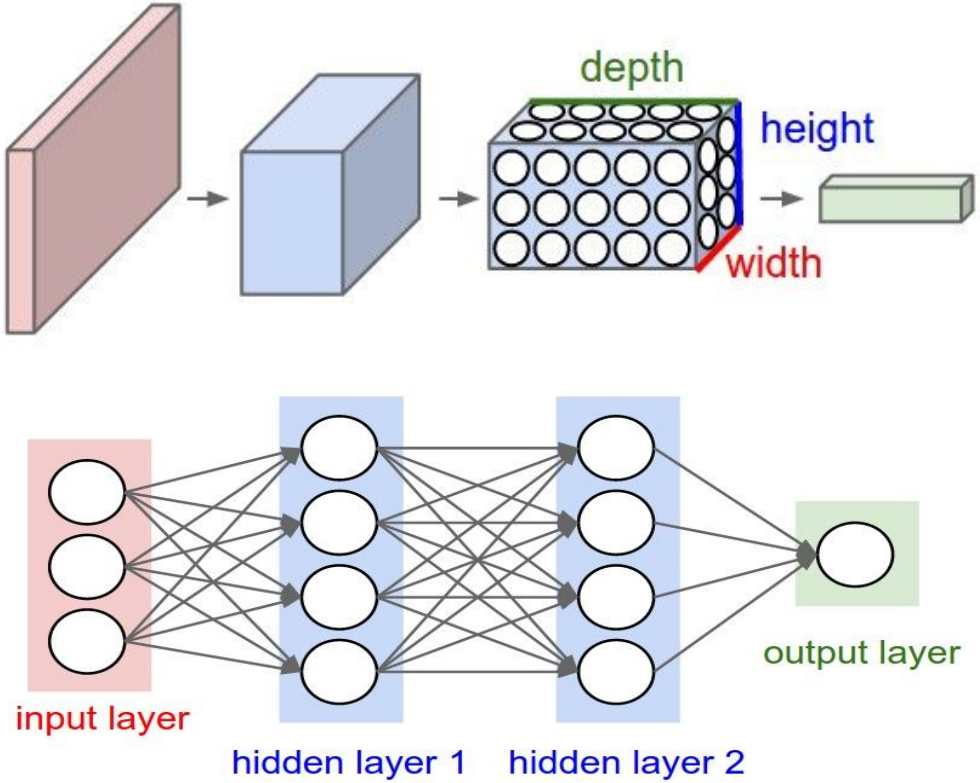


# Why CNN’s for Image Processing?

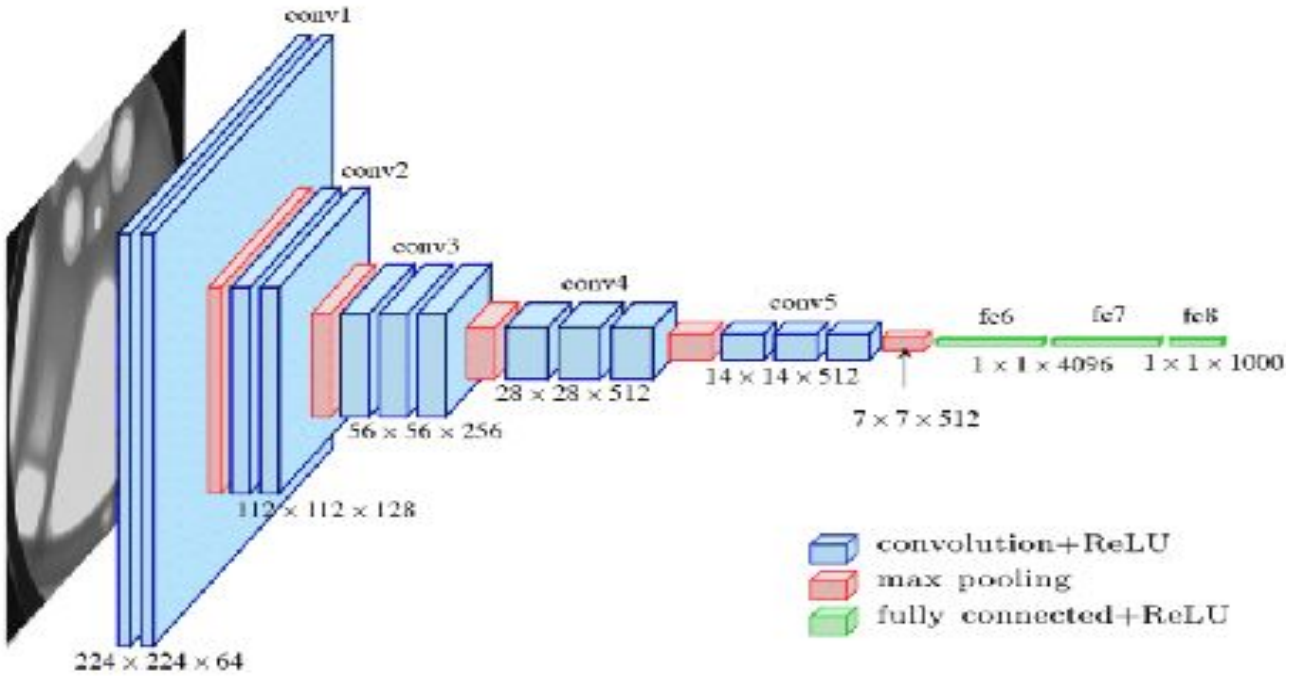
- CNN’s are very good in detecting patterns and making a sense of them. That's what make makes them useful for image analysis.
- The thing that makes them different from other Neural Networks is that their hidden layers are convolutional layers.
- They also include other types of layers (max pooling,relU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


convolutional
layers

# CNN vs Traditional Neural Network

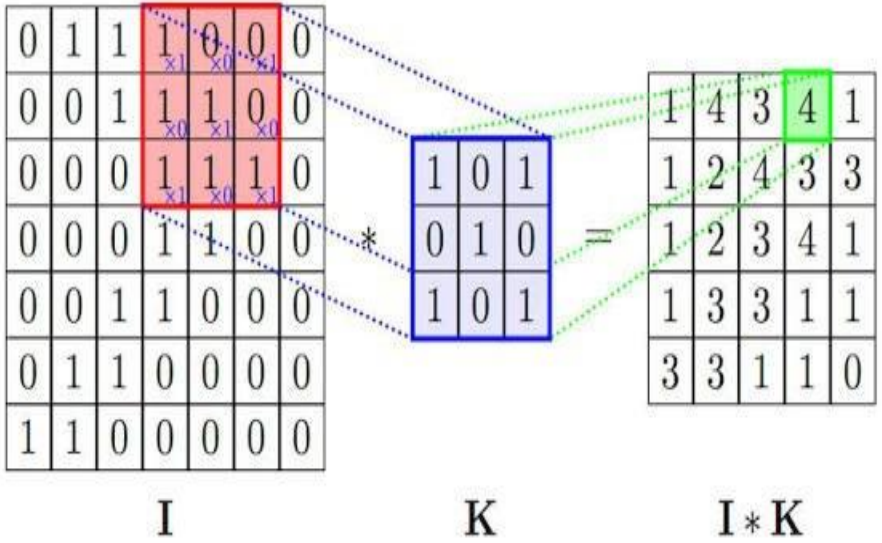


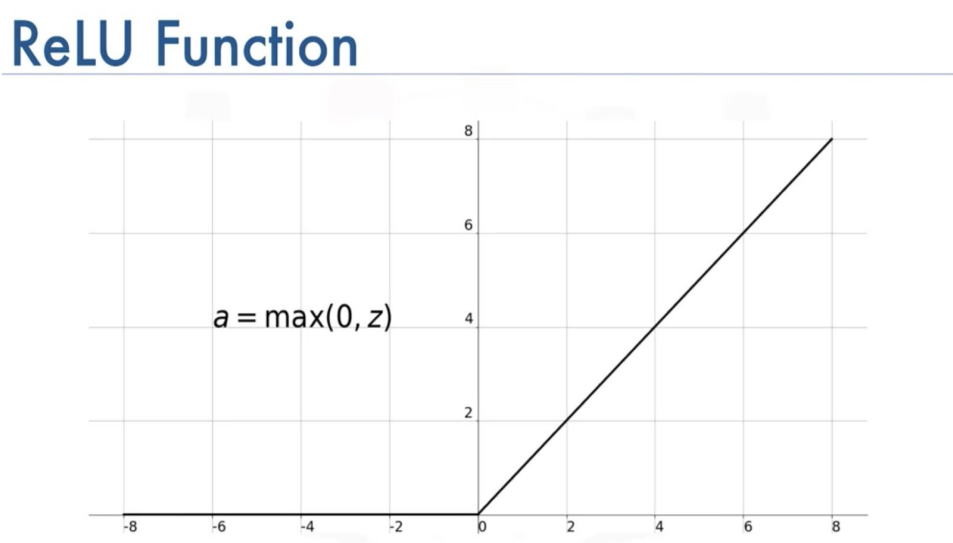
# The VGG19 Architecture



# Types of Layers

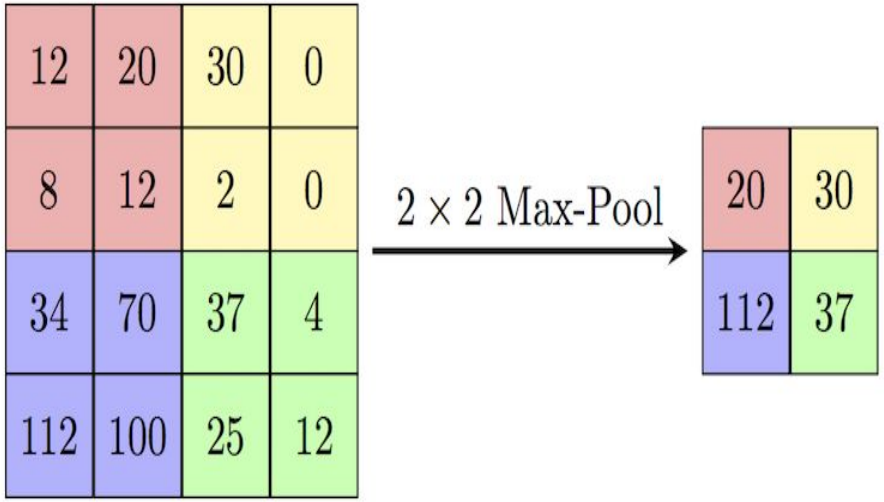
1.Convolutional + ReLu

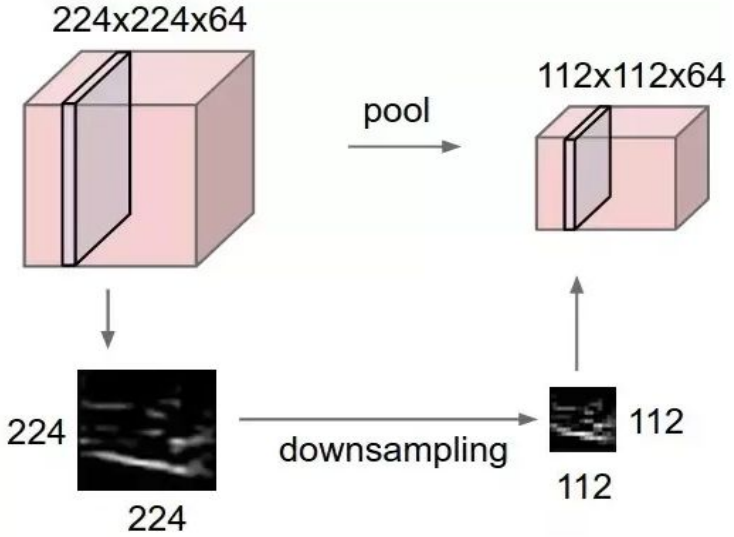


# Types of Layers

## 2.Max Pooling

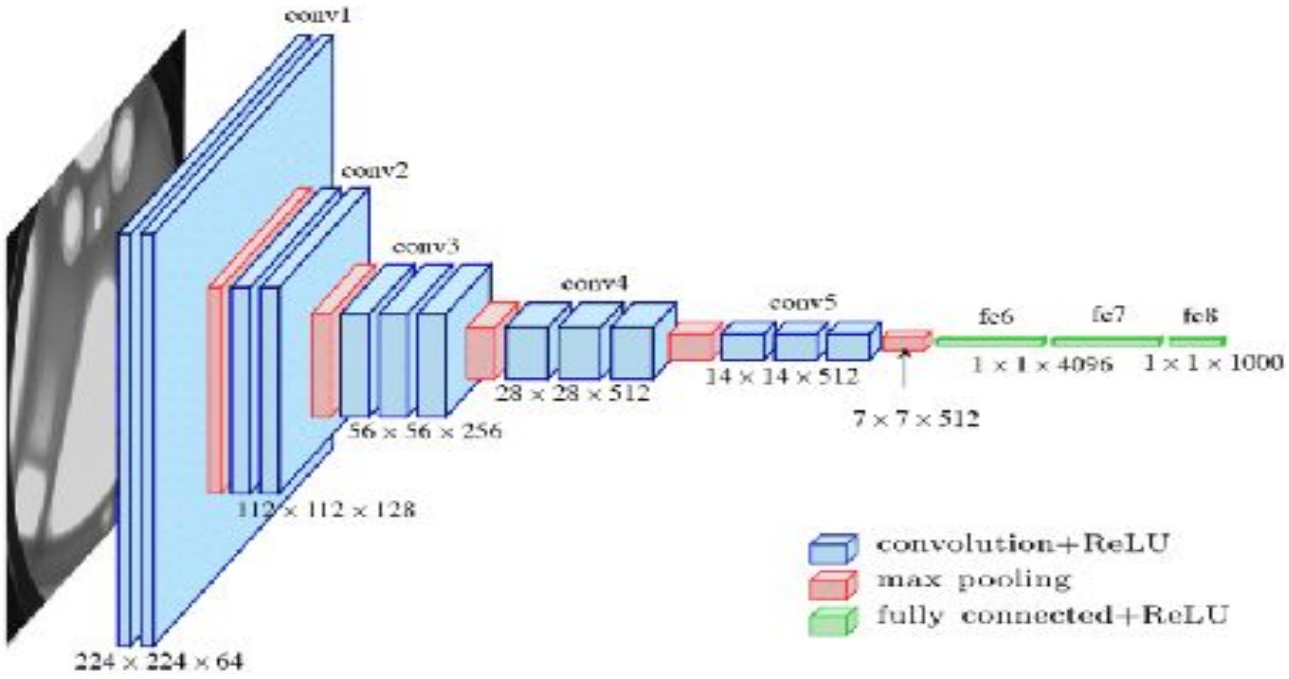


# Types of Layers

In this implementation we do not use the final fully connected layers and we receive our output from layer conv5_2.

# The VGG19 Architecture part2:



# VGG19 characteristics:

- 19 layers (16 convolution layers, 3 Fully connected layer, 5 MaxPool layers and 1 Softmax layer). In this project we use layers from conv1_1 to conv5_2
- Pre trained on millions of samples and has been developed by Oxford University
- The input image to the network should be a 224 × 224 x 3 preprocessed RGB image .During preprocessing we subtract the mean RGB value from each pixel.
- The convolutional filter are 3 x 3 with stride = 1 and padding = 1.
- Max pooling is performed over a 2 x 2 pixel window with stride = 2.
- Fully connected layers are not used in this implementation (the final layer outputs the probabilities of each of the 1,000 output categories. That doesn't concern us )

# Painting like Van Gogh (content)

To extract content from a picture we use the following algorithm:

1. Run the content image through the model and get the activation output of layer conv5_2 (P5_2)
2. Run the (initially random) target image x through the model and get the conv5_2 layer output (F5_2)
3. Calculate the content cost between the two outputs.

$$\sum (P_{5\_2} - F_{5\_2})^2.$$

4.Tweak the target image by back propagating the error back to target image and use that as a basis for out tweaking

5. Repeat 2-4 until satisfied.

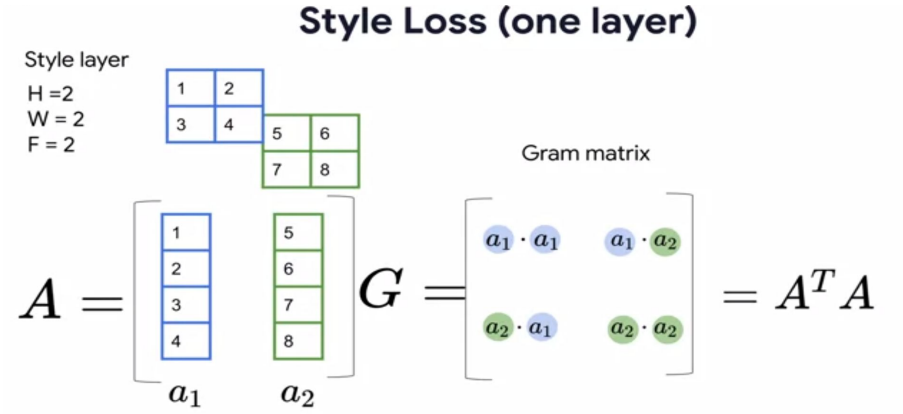
# Painting like Van Gogh (style)

The process is the same as before but instead of passing the output of each layer directly to the squared error function we first apply a function called Gram matrix.

The Gram matrix computes the correlations between the different filter responses.

$G^l_{ij}$ is the inner product between the vectorised feature map i and j in layer l.

$$G^l_{ij} = \sum_k F^l_{ik} F^l_{jk}.$$



# Painting like Van Gogh (content + style)

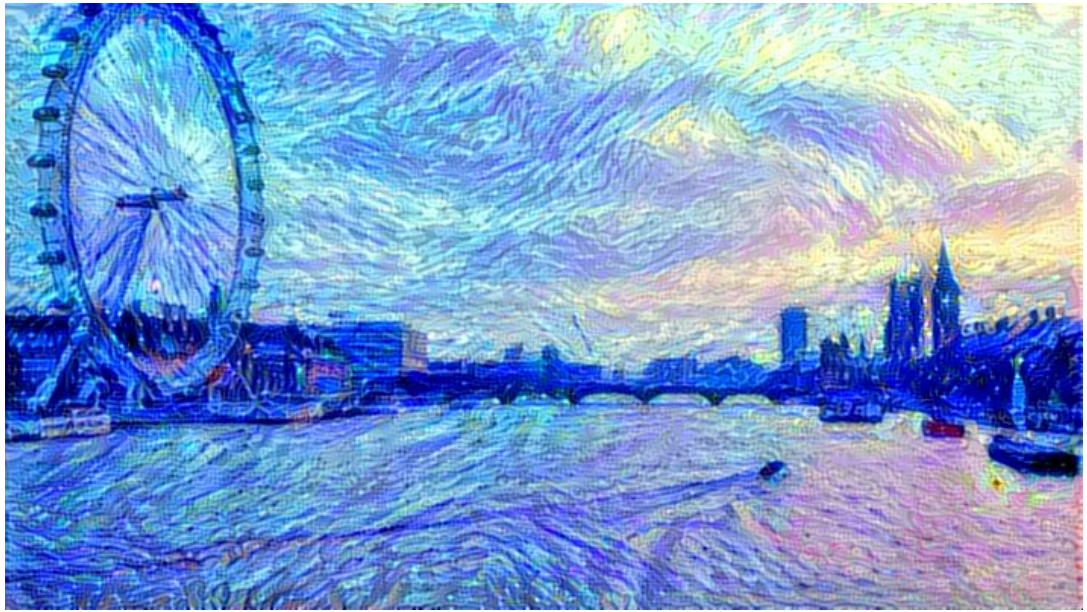
To paint both style *and* content we follow a similar process of iterative tweaking but employing a joint cost function that contains both content and style loss:

```
total cost = content cost + style cost
```

The optimization algorithm used in each iteration is the L-BFGS which is a popular algorithm for parameter estimation in machine learning.Its target is to minimize f(x) over unconstrained values of the real-vector x where f is a differentiable scalar function.

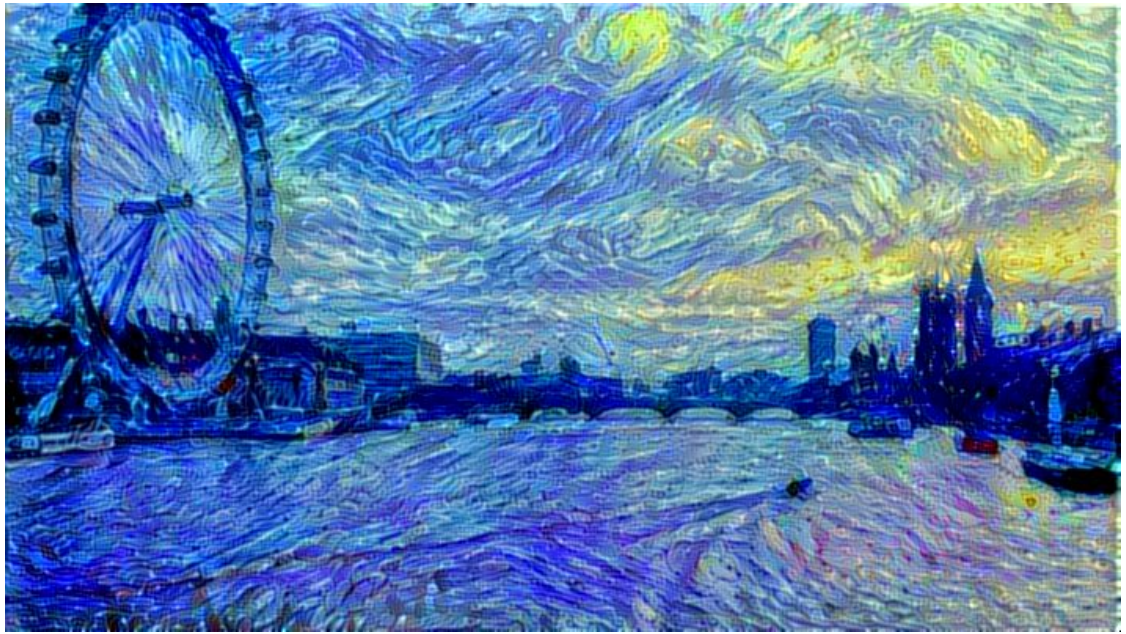
# Comparing Results part 1:

content(5_2) - style(1_1,2_1,3_1,4_1,5_1)



# Comparing Results part 2:

content(5_2) - style(1_2,2_2,3_2,4_2,5_2)



# Comparing Results part 3:

content(5_2) - style(1_2,2_2,3_3,4_3,5_4)

# Comparing Results part 4:

content(5_2) - style(1_1)

# Comparing Results part 5:

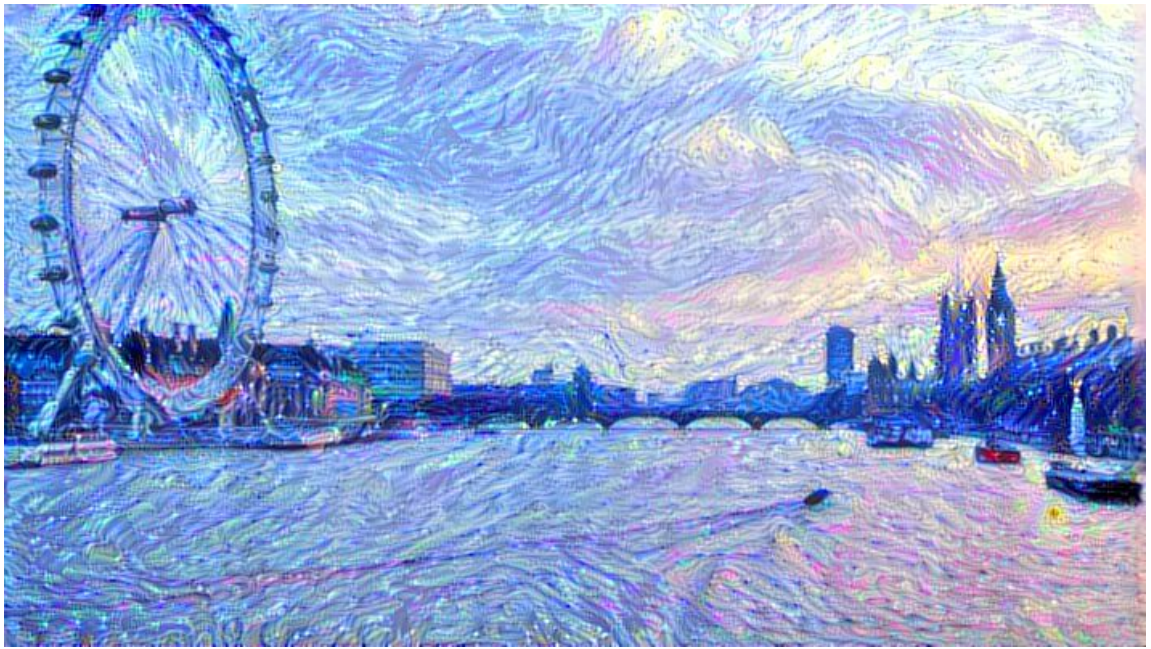
content(5_2) - style(2_1)

# Comparing Results part 6:

content(5_2) - style(3_1)

# Comparing Results part 7:

content(5_2) - style(4_1)



# Comparing Results part 8:

content(5_2) - style(5_1)

# Comparing Results part 9:

content(5_2) - style(3_1,4_1)

# Comparing Results part 10:

content(5_2) - style(2_1)

# Comparing Results part 11:

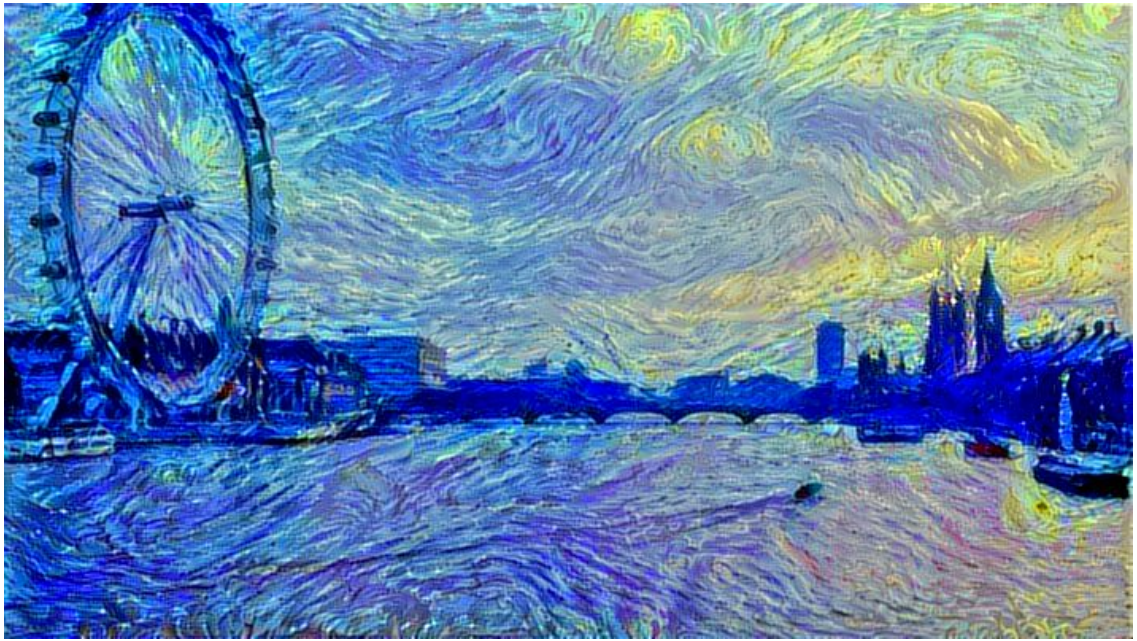
content(5_2) - style(1_1,1_2,,2_1,2_2,3_1,3_2,4_1,4_2,5_1,5_2,5_3)

# Comparing Results part 12:

content(5_2) - style(ολα τα layers)

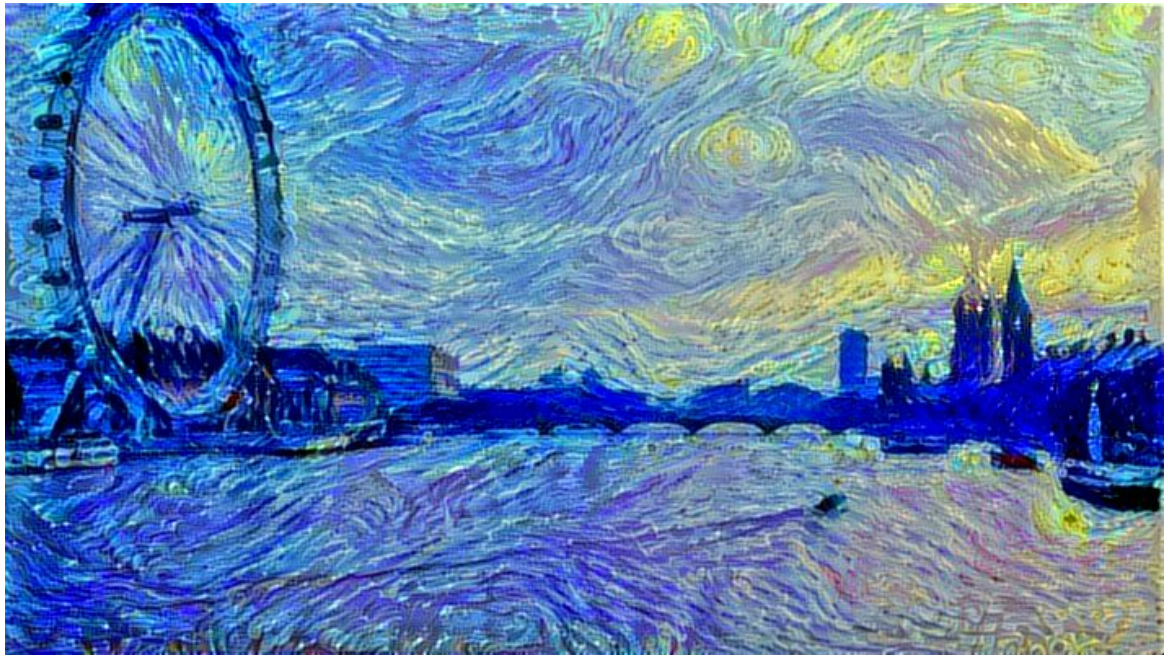
# Comparing Results part 13:

content(5_2) - style(all layers) and weight_style = 5, weight_content = 5



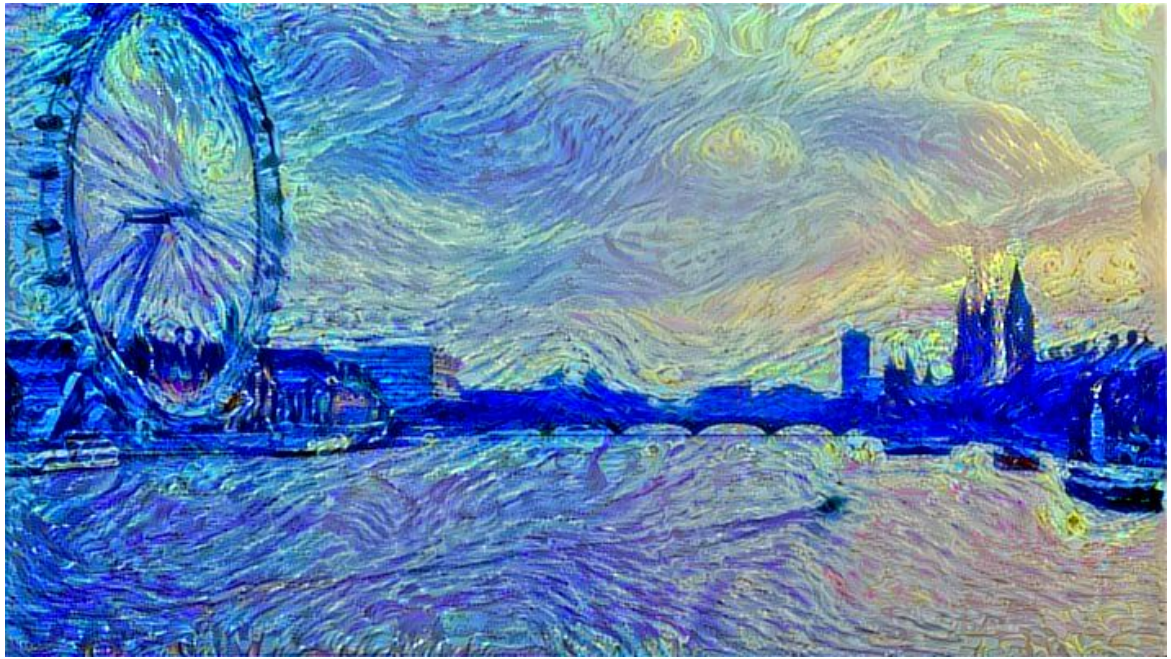
# Comparing Results part 14:

content(5_2) - style(all layers) and weight_style = 5, weight_content = 2



# Comparing Results part 15:

content(5_2) - style(all layers) and weight_style = 8, weight_content = 2



# Comparing Results part 16:

content(5_2) - style(all layers) and weight_style = 2, weight_content = 8