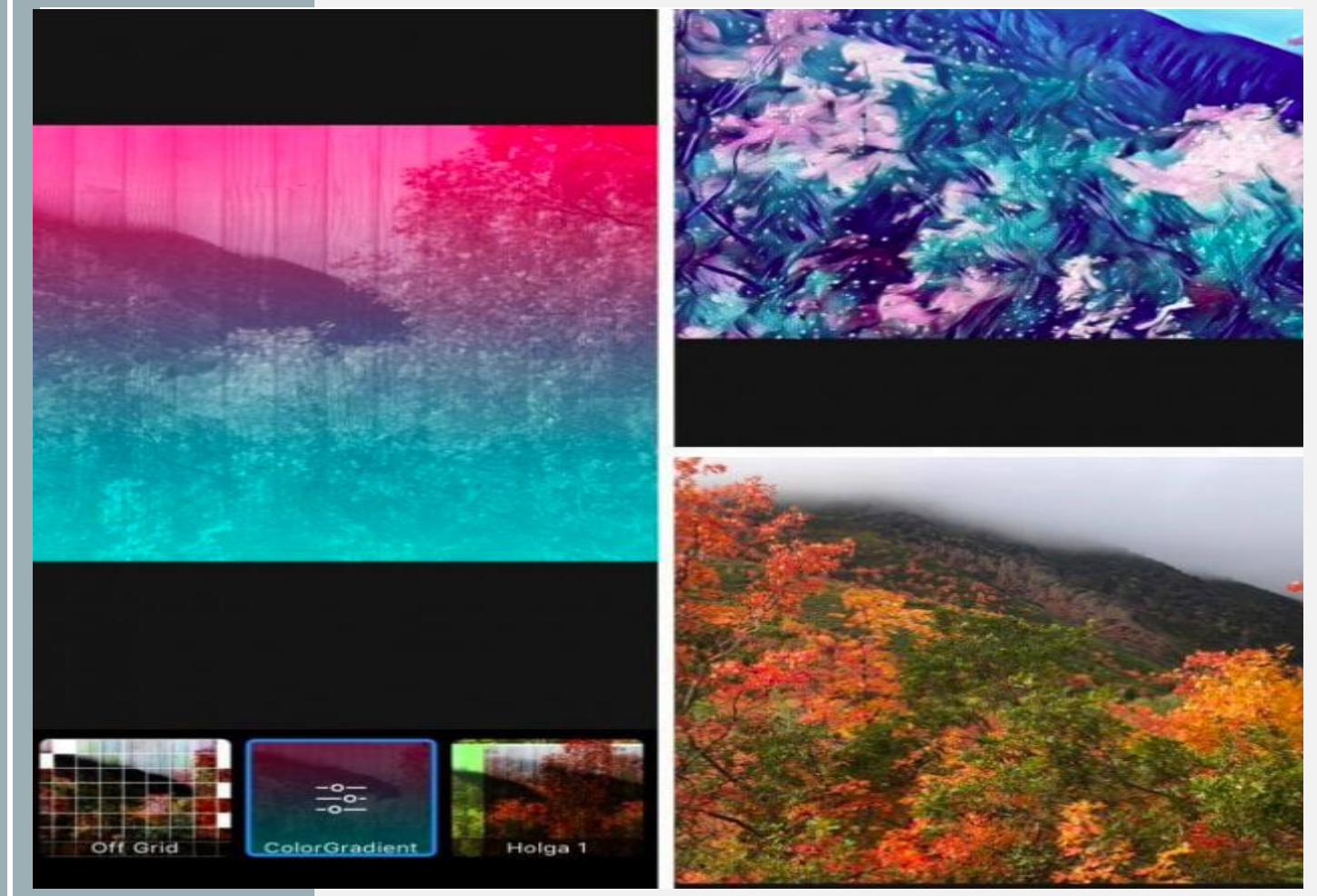
NEURAL STYLE TRANSFER

WHAT IS NEURAL STYLE TRANSFER

- By taking a *content* image and a *style* image, the neural network can recombine the content and the style image to effectively creating an *artistic (*recomposed*)* image.
- Although there are many apps like Prisma which generates artistic styles to pictures taken from mobile, in almost no time. But there are very high chances of image losing its identity
- Hence we tried to implement convolution based neural networks to regenerate the content image through calculating total loss and positioning pixels of the target with respect to the content image.

MOTIVATION

As we see on the right when we tried to yse picsart to transfer style 1 on conted image as shown int only applied the filters of blue and pink color resulting in the image as shown, Although the image that is shown is a more pleasentfull and appeling to see But in itself it has lost its identity, So in order to transfer the style over content preserving its identity we used Neural Style transfer to regenerate a new image based on properties of both content and styled image.



PROBLEM STATEMENT

- Given a content photo X and style photo Y how can we transfer the style of Y to the content X to generate a new photo Z. How can we train CNN to process and optimize the differences(the difference between X and Y)to reach an optimum global(Z)?

COST FUNCTION

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

CONTENT LOSS

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l}\right)^2 .$$

STYLE LOSS

$$\mathcal{L}_{style} = \frac{1}{2} \sum \left(G_{ij}^{l} - A_{ij}^{l}\right)$$

COST FUNCTION

$$\mathcal{L}_{total} = \alpha \mathcal{L}_{content} + \beta \mathcal{L}_{style}$$

CONTENT LOSS

- Say we use hidden layer l to calculate content loss.
- Generally l is chosen somewhere in between of too shallow and too deep.
- Suppose if l was very early layer, then our cost function will try to make even the very minute features like straight line hence resulting to be same as general content image

STYLE LOSS

- Say we are using l's activation channel to measure style.
- Define style as a correlation between activations across channels.
- What to do is look at the activation of in those 2 channels, we look at the n*w pairs & check how these 2 channels are corelated.

IMPLEMENTATION PLAN

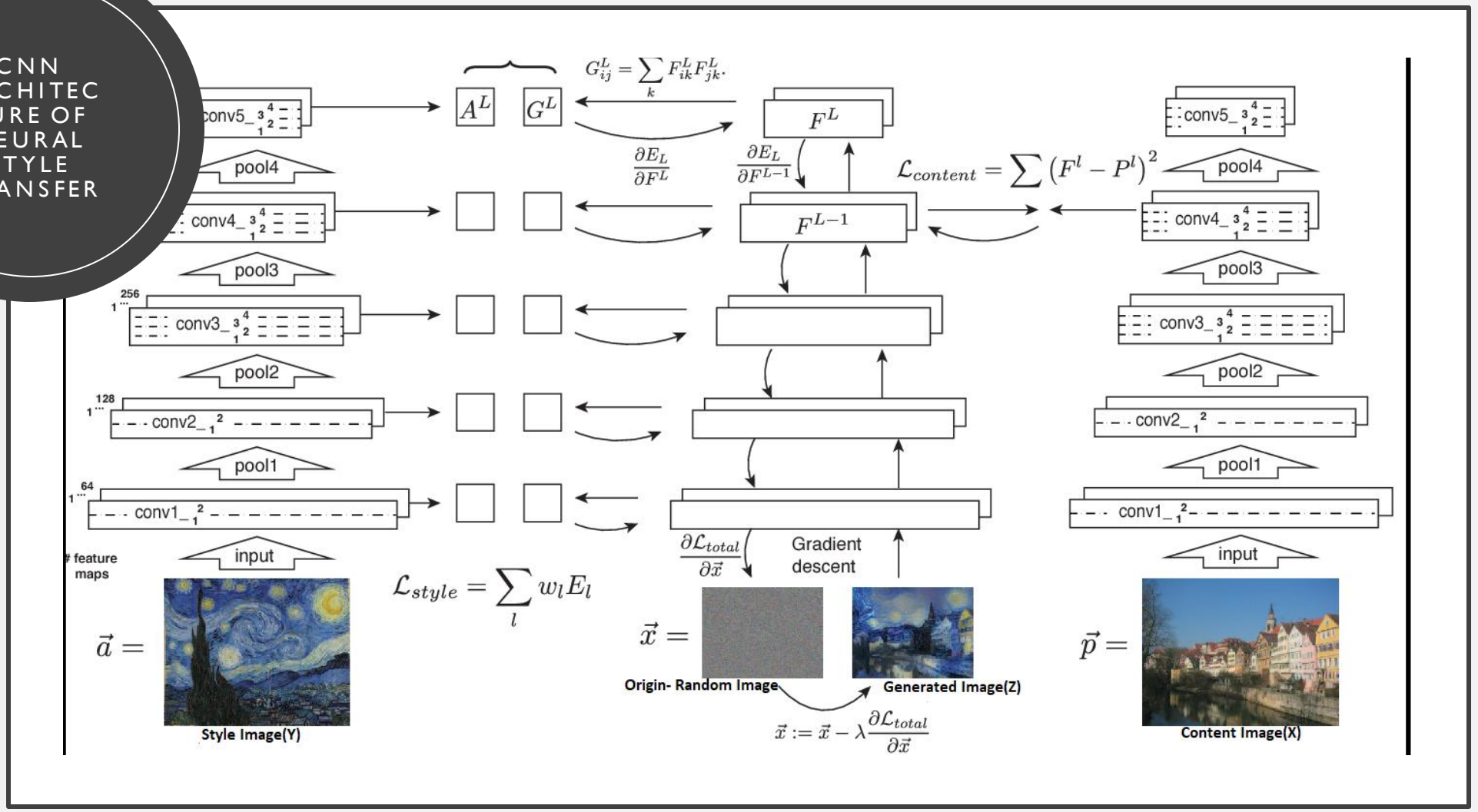
1. Initialize G randomly say (224* 224 * 3).
2. Tried using gradient descent method but the results were not so accurate.
3. So we went on using Broyden–Fletcher–Goldfarb–Shanno algorithm

As mentioned in the paper Neural Style Transfer on Real Time Video

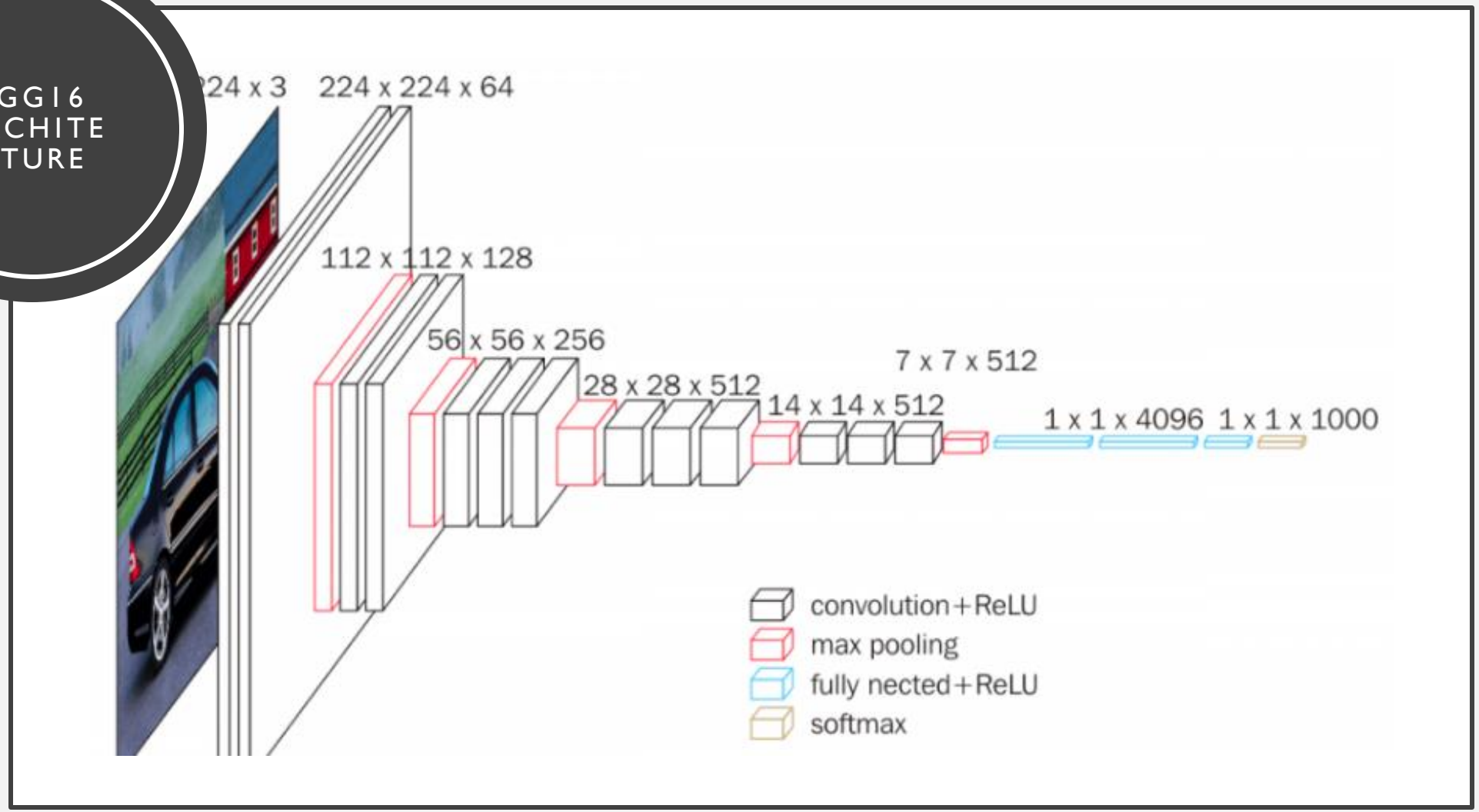
(With Full implementable code) for better results.

Basically we are updating the pixel value of the image G

CNN ARCHITECTURE OF NEURAL STYLE TRANSFER

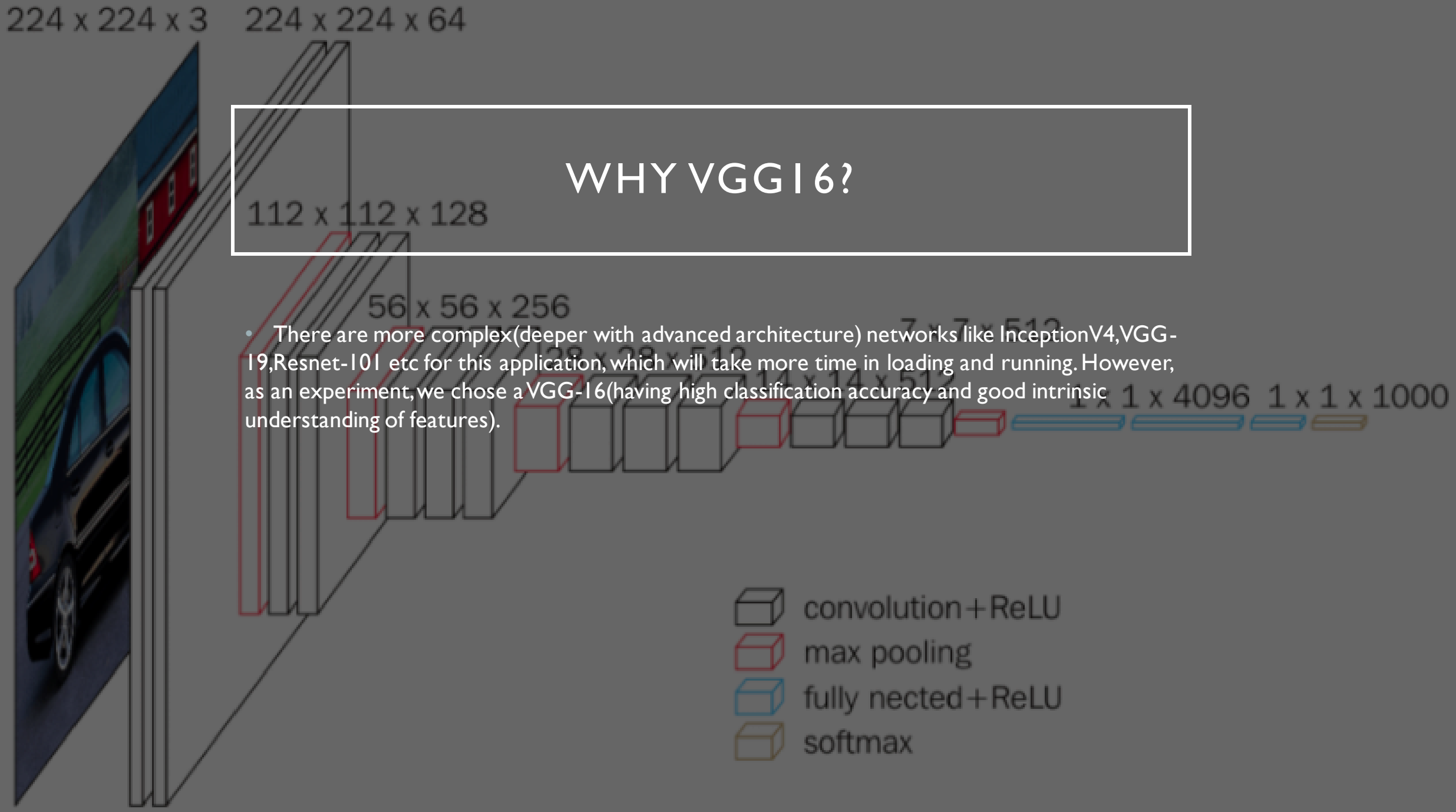


VGG16 ARCHITECTURE



WHY VGG16?

- There are more complex(deeper with advanced architecture) networks like InceptionV4,VGG-19,Resnet-101 etc for this application, which will take more time in loading and running. However, as an experiment, we chose a VGG-16(having high classification accuracy and good intrinsic understanding of features).



BROYDEN–FLETCHER–GOLDFARB–SHANNO ALGORITHM

This algorithm is based on the BFGS recursion for the inverse Hessian as :

$$H_{k+1} = (I - \rho_k s_k y_k^\top) H_k (I - \rho_k y_k s_k^\top) + \rho_k s_k s_k^\top.$$

$$\begin{aligned}
&q = g_k \\
&\textbf{For } i = k-1, k-2, \ldots, k-m \\
&\qquad \alpha_i = \rho_i s_i^\top q \\
&\qquad q = q - \alpha_i y_i \\
&\gamma_k = \frac{s_{k-1}^\top y_{k-1}}{y_{k-1}^\top y_{k-1}} \\
&H_k^0 = \gamma_k I \\
&z = H_k^0 q \\
&\textbf{For } i = k-m, k-m+1, \ldots, k-1 \\
&\qquad \beta_i = \rho_i y_i^\top z \\
&\qquad z = z + s_i(\alpha_i - \beta_i)
\end{aligned}$$

Where,

H_k^0 represents the initial approximate Hessian matrix, which is chosen as a diagonal matrix.

$$g_k \equiv \nabla f(x_k)$$

$$s_k = x_{k+1} - x_k$$

$$y_k = g_{k+1} - g_k.$$

$$\rho_k = \frac{1}{y_k^\top s_k},$$

IMAGE RECONSTRUCTION

CONTENT RECONSTRUCTION

- We can visualise the information at different processing stages in the CNN by reconstructing the input image from only knowing the network's responses in a particular layer.
- We reconstruct the input image from layers & found that reconstruction from lower layers is almost perfect. In higher layers of the network, detailed pixel information is lost while the high-level content of the image is preserved.

STYLE RECONSTRUCTION

- On top of the original CNN representations we built a new feature space that captures the style of an input image. The style representation computes correlations between the different features in different layers of the CNN.
- We recon-struct the style of the input image from style representations built on different subsets of CNN layers This creates images that match the style of a given image on an increasing scale while discarding information of the global arrangement of the scene.

ORIGINAL IMAGES USED

CONTENT IMAGE

STYLE IMAGE

IMAGE RECONSTRUCTION

We can see that after some time the loss for content image regeneration became constant

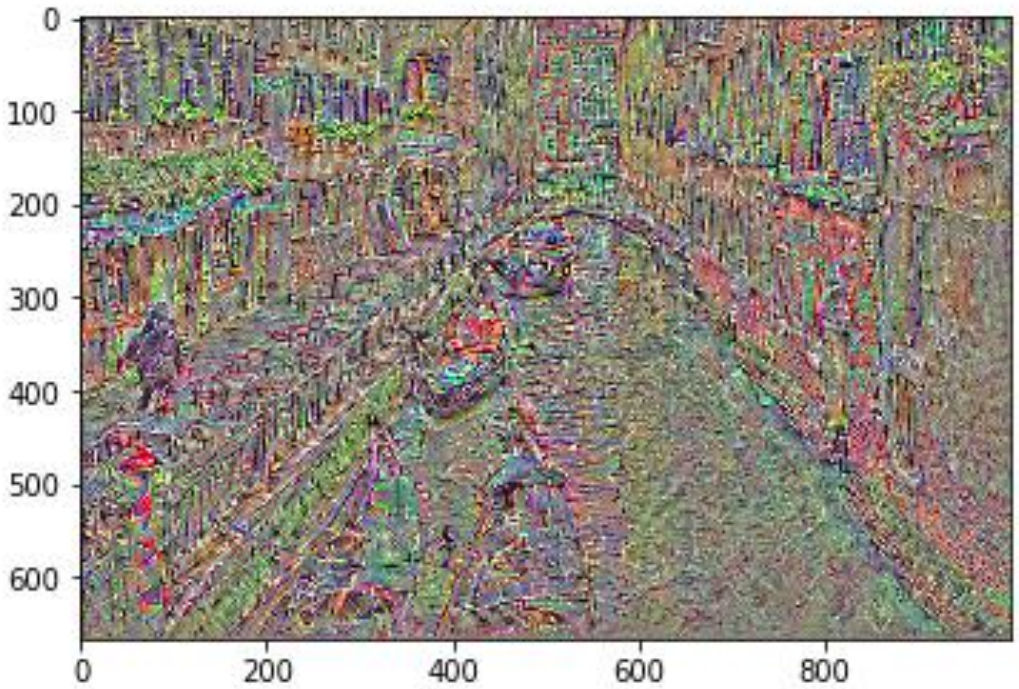
- iter = 0 loss = 45.30365753173828
- iter = 1 loss = 19.052011489868164
- iter = 2 loss = 11.1323194503784 18
- iter = 3 loss = 7.9174900054931 64
- iter = 4 loss = 7.4138126373291 02
- iter = 5 loss = 5.5127797126770 02
- iter = 6 loss = 5.2164139747619 63
- iter = 7 loss = 5.2246618270874 02
- iter = 8 loss = 5.2251839637756 35
- iter = 9 loss = 5.2251839637756 35
- iter = 10 loss = 5.225183963775635
- iter = 11 loss = 5.225183963775635
- iter = 12 loss = 5.225183963775635
- iter = 13 loss = 5.225183963775635
- iter = 14 loss = 5.225183963775635
- iter = 15 loss = 5.225183963775635
- iter = 16 loss = 5.225183963775635 iter = 17 loss = 5.225183963775635 iter = 18 loss = 5.225183963775635 iter = 19 loss = 5.225183963775635 iter = 20 loss = 5.225183963775635 iter = 21 loss = 5.225183963775635 iter = 22 loss = 5.225183963775635 iter = 23 loss = 5.225183963775635 iter = 24 loss = 5.225183963775635 iter = 25 loss = 5.225183963775635 iter = 26 loss = 5.225183963775635 iter = 27 loss = 5.225183963775635 iter = 28 loss = 5.225183963775635 iter = 29 loss = 5.225183963775635 duration: 0:02:09.345389

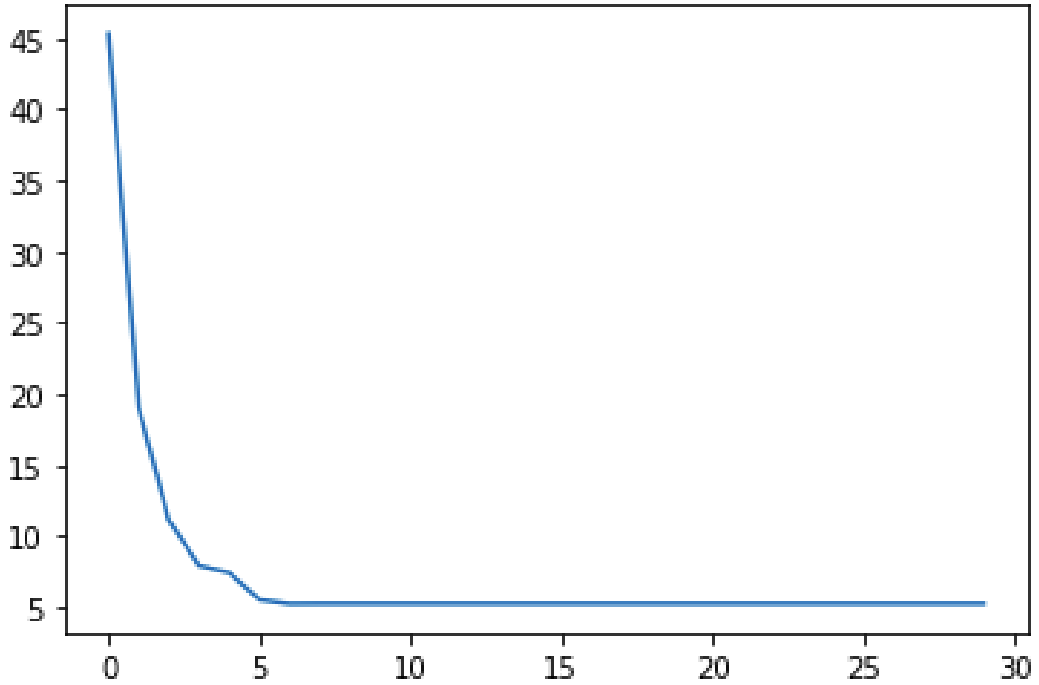
IMAGE RECONSTRUCTION

On right you can see

The image that is reconstructed

The graph plotted for loss calculated

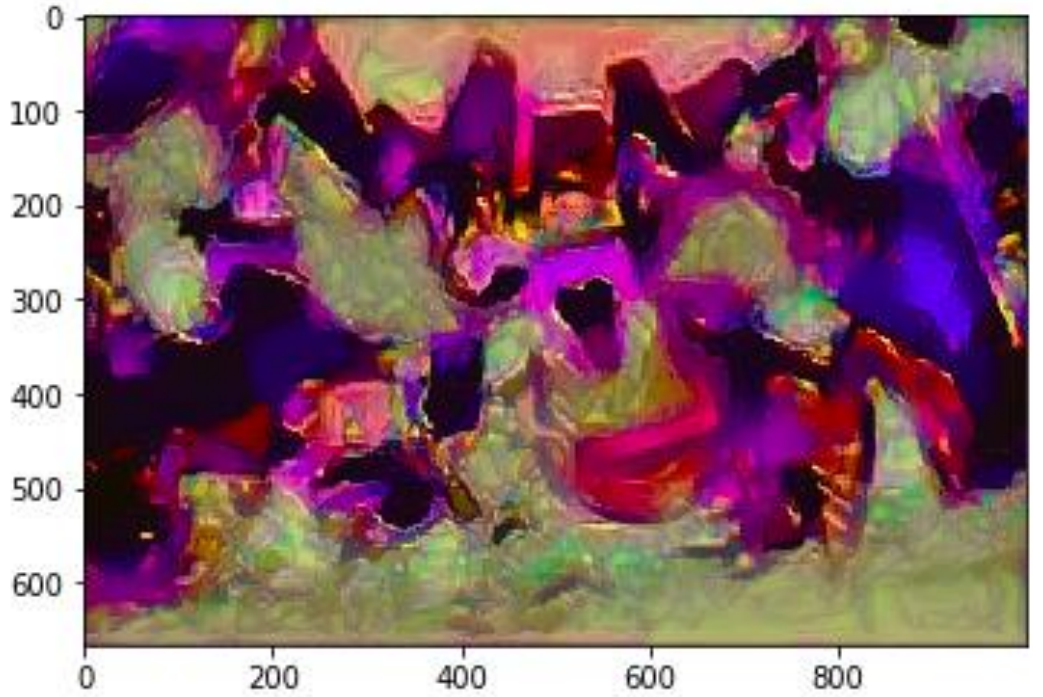


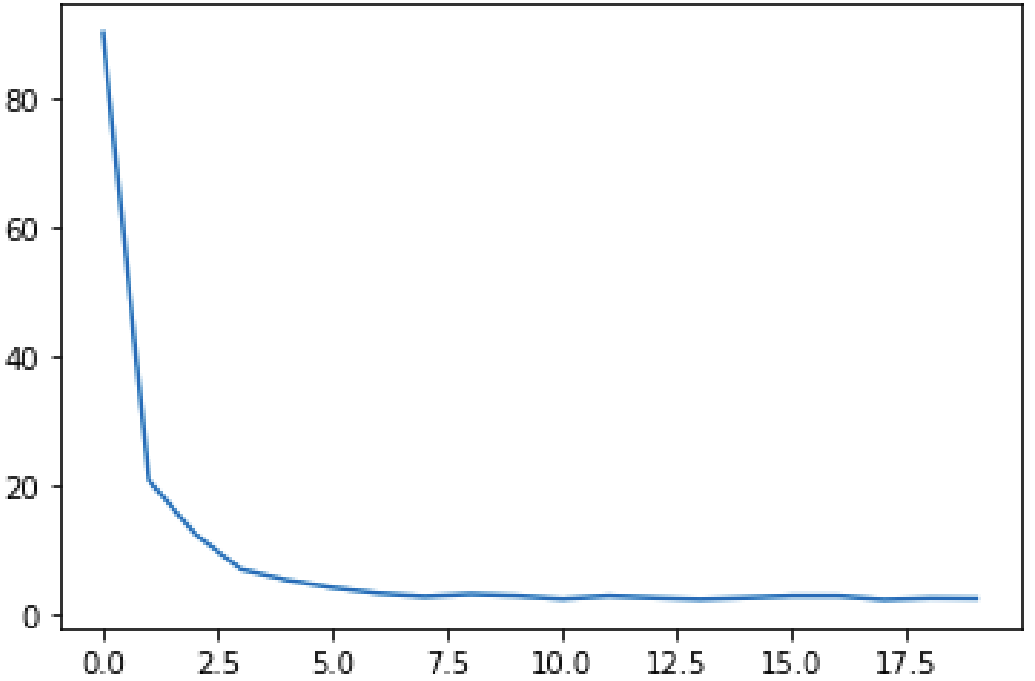


STYLE REGENERATION

Here u can see the style regenerated based on the loss calculated by gram matrices

Second image is of loss plotted



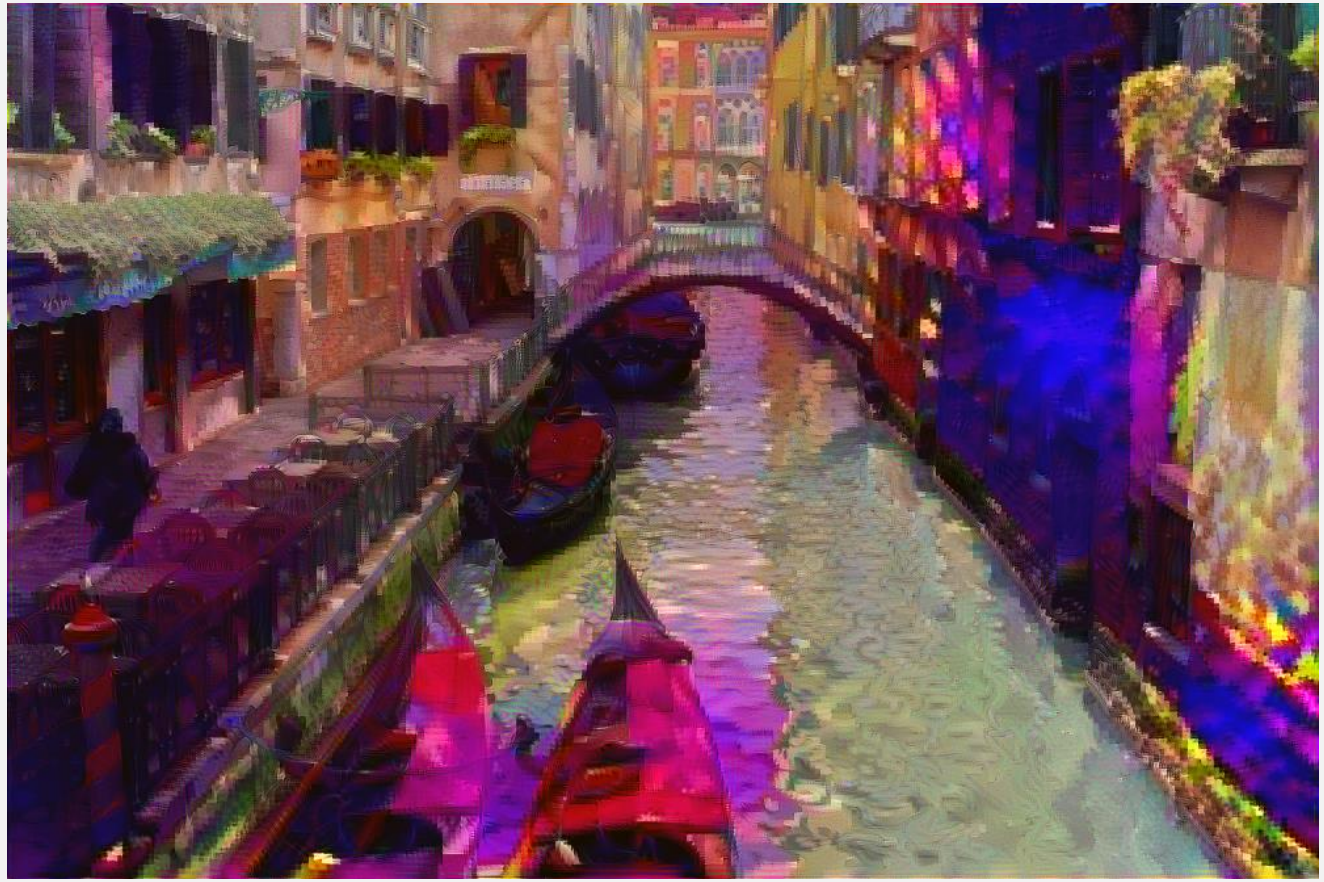


BUSINESS APPLICATION

Apart from personal and artistic usage of this seemingly fancy technique, Neural Style Transfer has its potential to transform any industry, where human creativity has traditionally dominated, such as fine arts, fashion, architecture or new trendy car texture design etc. For example, take the Fashion industry in which requires a deep understanding of mechanisms of fashion: causes and spreading of the trends, principles of cyclic repetition and evolution patterns to develop the fashion of tomorrow.

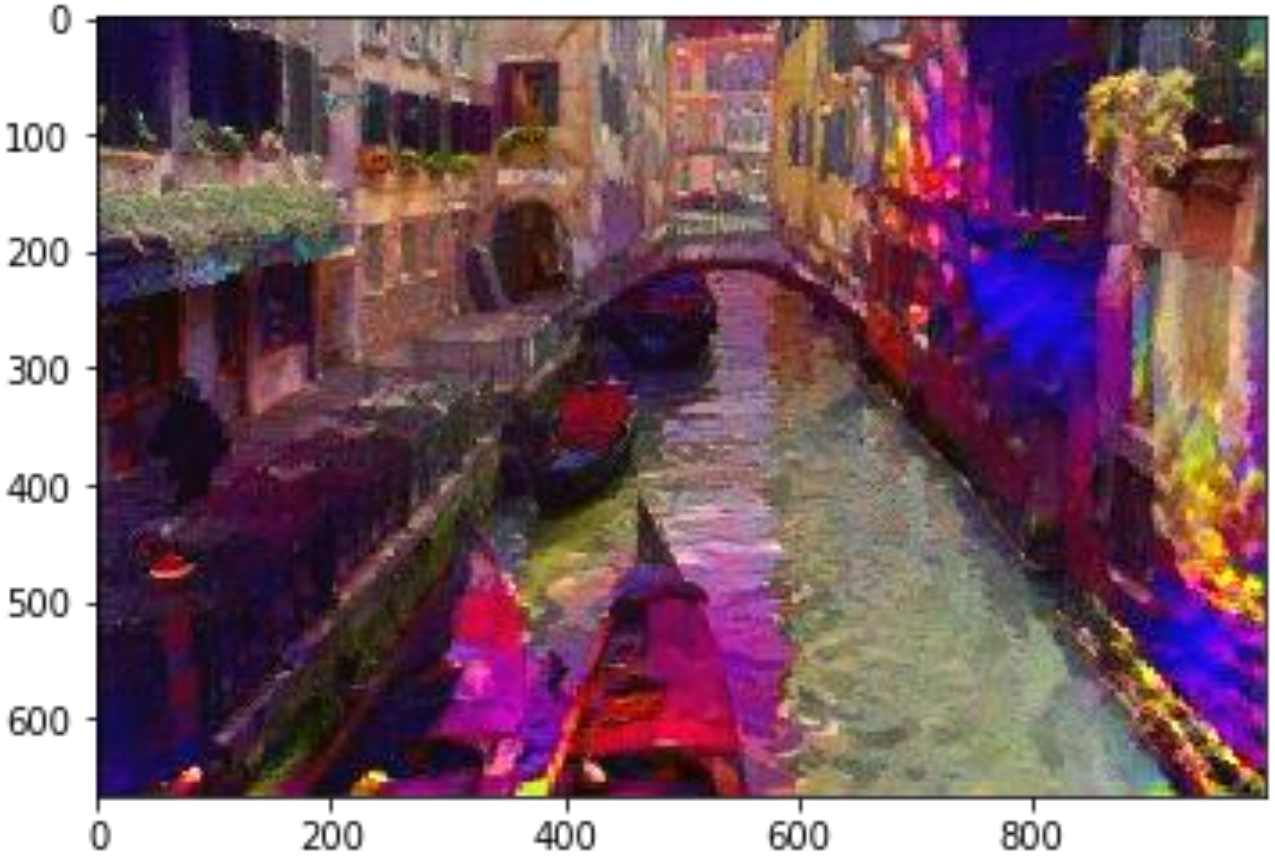
GENERATED IMAGE USING GRADIENT DESCENT

- When we used gradient descent to calculate content loss than the image produced is not of that much high quality as shown.



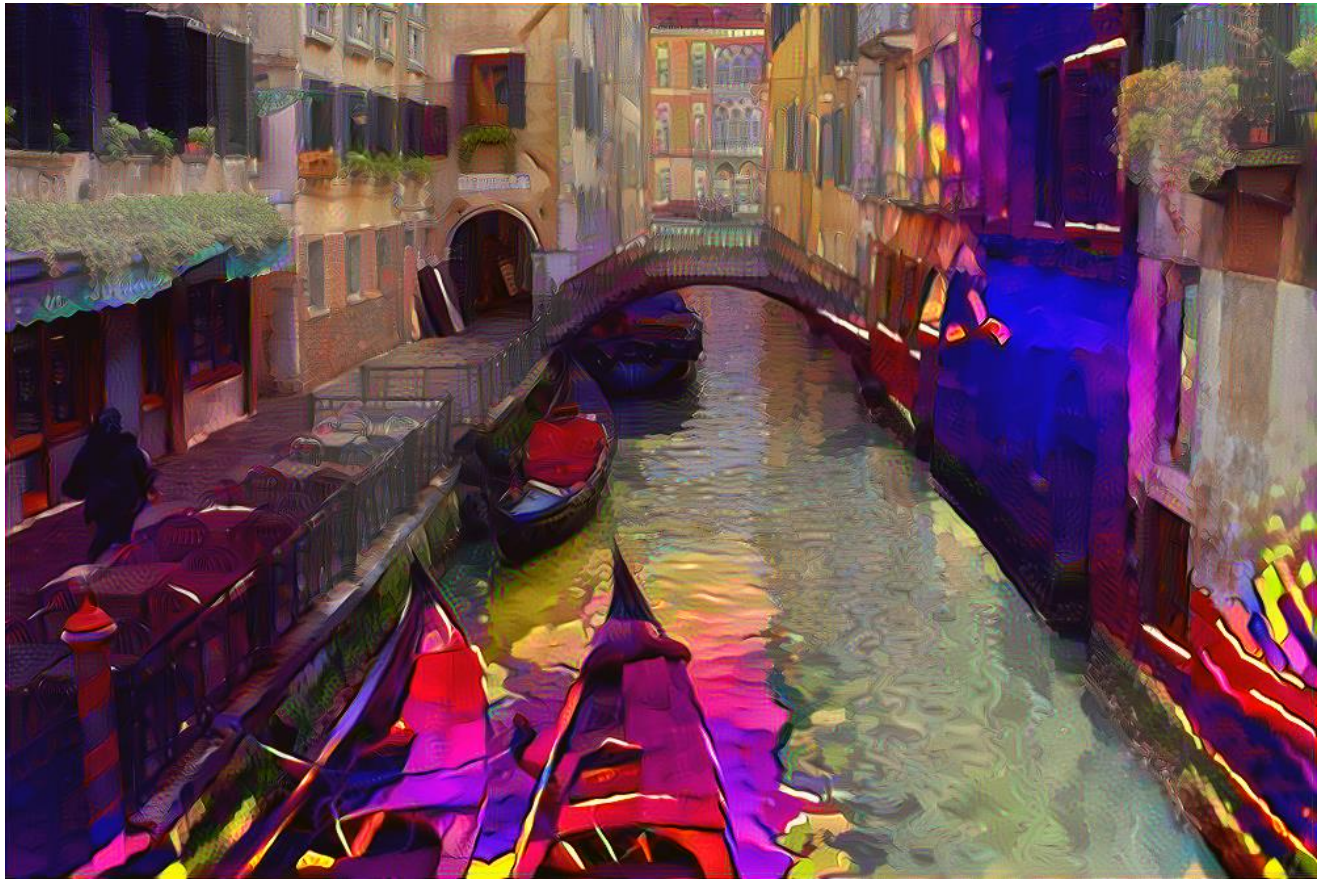
USING BROYDEN–FLETCHER–GOLDFARB–SHANNO ALGORITHM

When we used the above said algorithms for 50 epochs the following result was observed



USING BROYDEN–FLETCHER–GOLDFARB–SHANNO ALGORITHM

When we used the above said algorithms for 80 epochs the following result was observed



USING BROYDEN–FLETCHER–GOLDFARB–SHANNO ALGORITHM

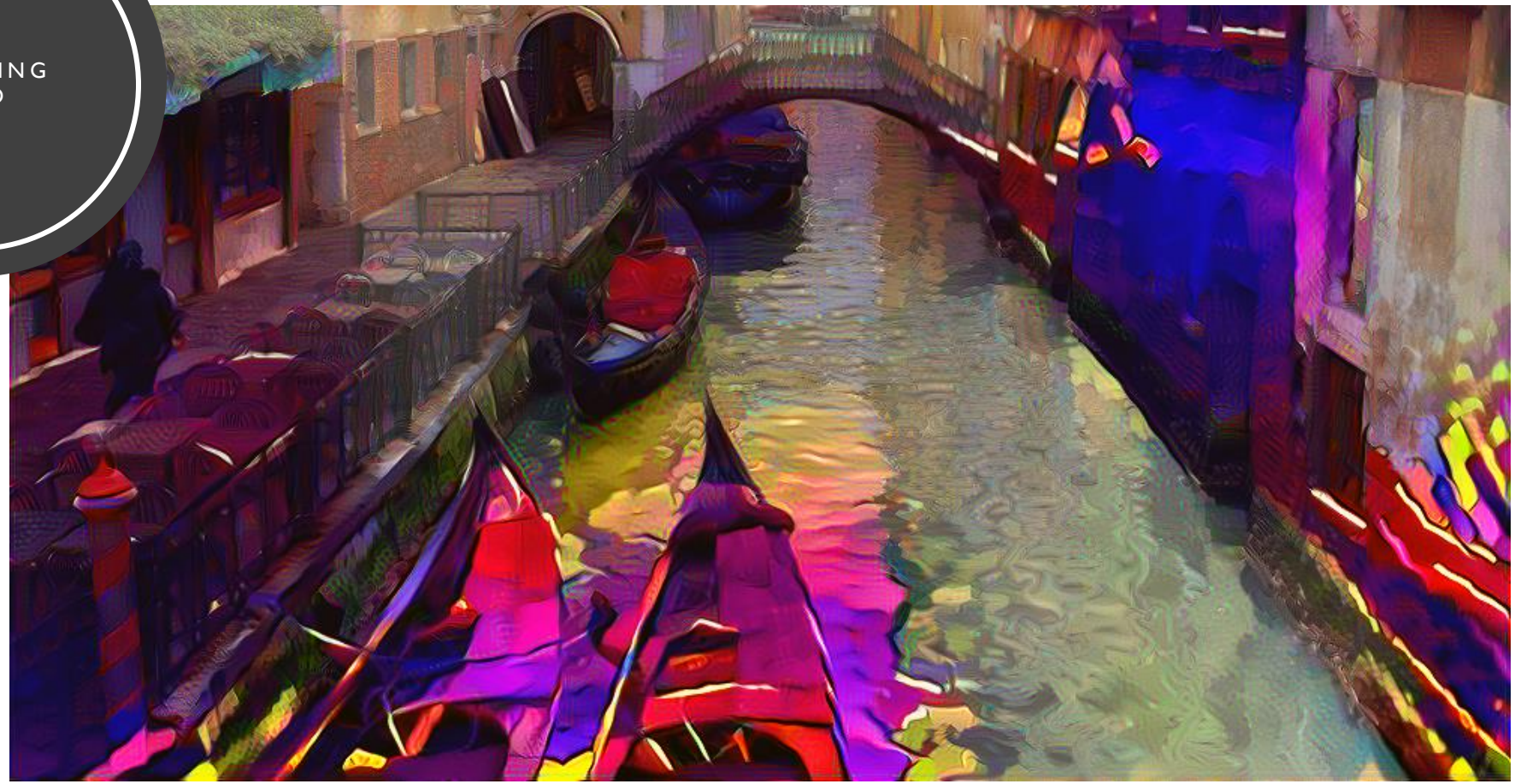
When we used the above said algorithms for 120 epochs the following result was observed

EXPLAINED EXAMPLE

CONTENT IMAGE

STYLE IMAGE

RESULTING INTO



A DIFFERENT EXAMPLE

CONTENT IMAGE

STYLE IMAGE

RESULTING INTO

FURTHER IMPROVEMENT AND EXPERIMENTS:

- 1) More Iterations
- 2) Advanced CNN architecture
- 3) Tweaking content and style loss weights
- 4) Tweaking layer weights for style loss

END