Style Transfer via Multi-Scale Feature CNN

Introduction

Style transfer is a computer vision technique which takes two images – a content image and a style reference image, and blends them together so that the resulting output image retains the core elements of the content image, but yet appears to be as "painted" in the style of the style reference image.

Style transfer, as a task performed in CV world, has obtained much attention in the DL field.
There are several ways in which to create a style transfer outcome. Such include but not limited to, Variational Auto-Encoders (VAE), Generative Adversarial Networks (GANs), Transformers (Encoder-Decoder with Attention), Neural style, and more deep independent architectures.

Our goal in this project is to be able to create a style transferred image depending solely on a some known DL neural architecture without relying on other network model which functions as a Discriminator objective, or just another capacity model for dividing the content and style tasks.

We believe if the style transfer task were thought of as a combination of DC (or base-line hidden distribution) component from the content image (could be thought of as the dc of the image in the fourier domain, plus a certain Heaviside filter for part of the low frequencies of the content – Low Pass FIR Filter). Hence, maintaining Semantic regions.
And an AC distribution part – as for a variant (variance) component which potentially holds Semantic information, object shape, and Semantic Texture of the style image (could also be thought of as for the high frequency part in the fourier domain – as for a High Pass FIR filter).

For instance,
Let's consider an arbitrary neural architecture (drawn below, for demonstration, VGG16 network),

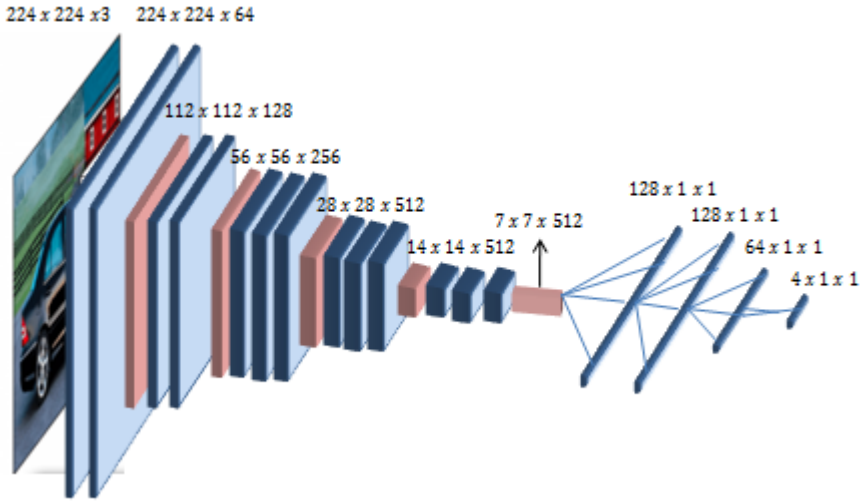

Figure 1: VGG16 network for illustration

Our method then would be to take a single neural network architecture, and while transferring a desired image (content\style image) into, obtaining some the necessary feature maps from different network regions (by deep mean), and building an objective loss function by comparing it with the same feature map obtained when transferring the target image via the same network.

Those feature tensors\maps would be (when dealing with the style image) a multi scale feature tensors taken from different parts of the NN.

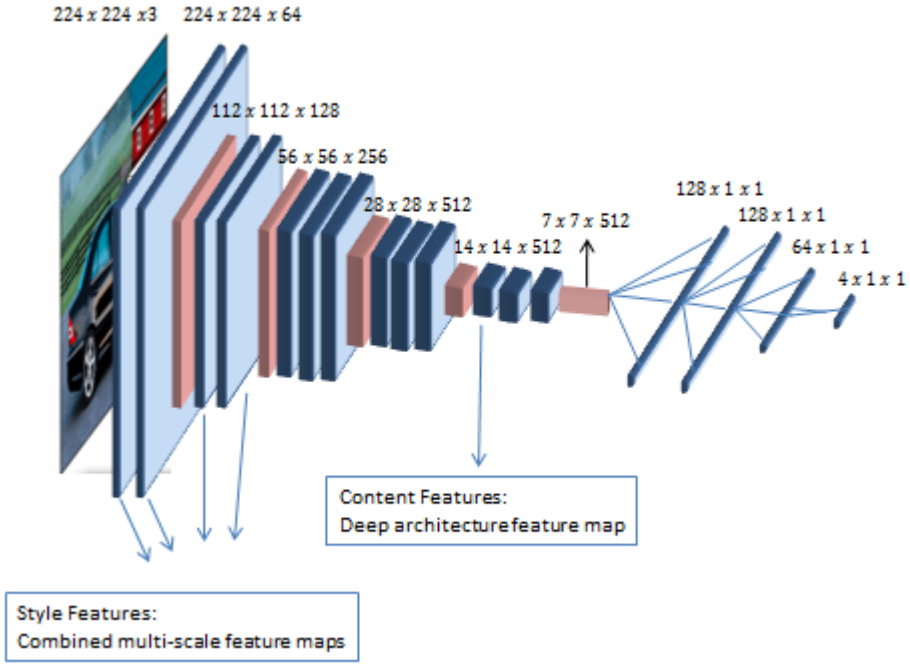

Figure 2: Extracting the desired feature maps (depth neural model wise)

As such, from defining the total loss of the network (by a combination and a certain Norm), we'll be deriving the gradients of this loss function through the network, all the way back to the input target image (which at first will be a random Gaussian Noise or a blank canvas for instance), and performing one way of an SGD method and changing the pixels of the target image in such a way which'll minimize our loss function. Hence, changing its distribution such that when transferred through the model, it'll obtain a close multi scale feature maps (in the used attention layers) within different (deep wise) architecture layers.

By mean, changing target image content (when the "content" layer is to be taken into consideration. That is the deep layer before the FC\dense part) to be as close to the content of the content image, and its style (the layers which are accounted for the texture and variance component in an image. That is when the few first multi scale layers are taken into consideration) to be as close to the style of the style image.

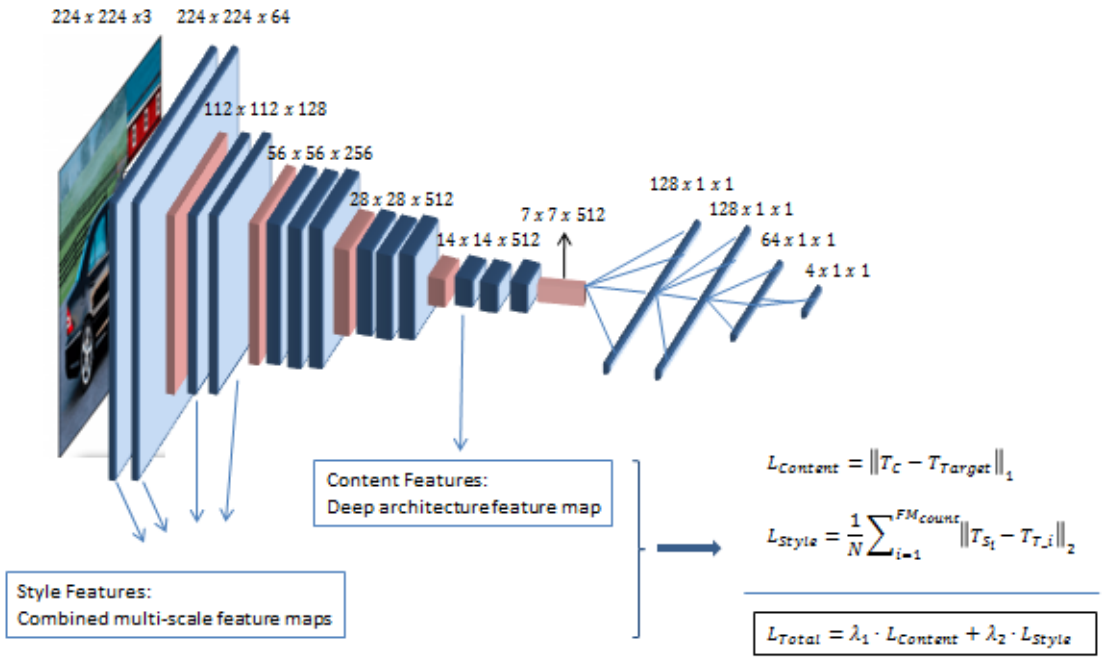

Figure 3: Calculating and Combining the two model losses into one regularized objective Style Transferred Loss

Method

To be able to perform the above task, we offer a novel approach – A Multi Scale Feature Objective Style Transfer method. Which consists of extracting a selected feature maps out of a given neural network (preferably, one which was trained on a classification task. Hence, be able to extract and identify different textures, semantic regions\edges, semantic information, and build an inner object composed structure), in a depth wise manner (in such a way to serve our content\style feature extraction goal), apply some transformation functions, calculate a Loss term, and update\optimize a Target image in such a way to fulfill finding a local\global minima to our regularized pre-defined Loss feature objective. By mean, Style Transfer.

Our pick for a neural network for this specific task, is to be the VGG19 Network.
A pre-trained ImageNet deep NN for classification tasks, and deep enough to separate and be able to extract different features via different deep level wise network layers (to consists various kernel extractors and a high dimensional weight values such that are objective dependent).

P.S. The neural network different weight values, layer wise, can be thought of as Markovian Chain stochastic parameters in an extremely highly dimensional space. This further alleviates the task of training process in the long trajectory, and simplify the joint probability distribution in each layer (if to be consider a stationary process).

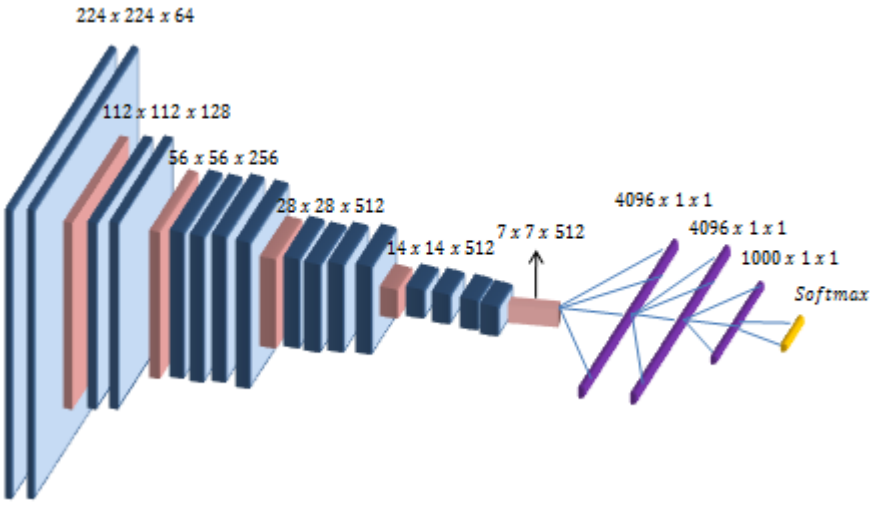

Figure 4: VGG19 Neural Network Illustration

As described previously,
In our case, for the sake of the style transfer task. Our Loss objective will be taken from different layers via network feature maps. Yet, not from the classification layers. Since the latter when trained, is responsible for the final classification. i.e. compressing all the previous learned\extracted information via Fully Connected layers into a fixed special tensor (with an oriented gradient).

Hence, won't be relying on.
As such, we neglect the FC layers and maintain all the rest of the CNN,

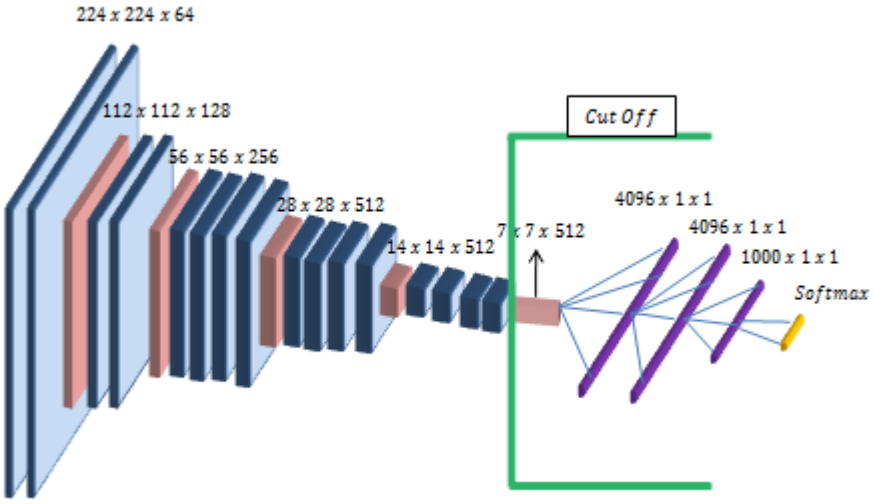

Figure 5: Neglecting the Classification layers and maintaining the Features

The resulted final network,

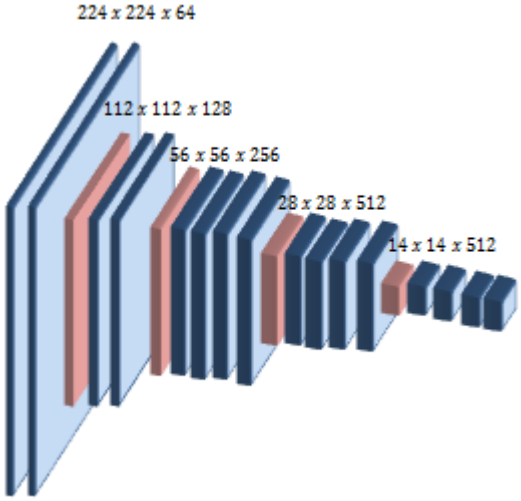

Figure 6: Final Neural Network

After specifying the final neural network, we should elaborate the progress and the Loss objectives for the task.

P.S. Worth mentioning, the above model will be static (fixed weights parameters) through all of the optimization process.

Loss Objectives

In order for the optimization to take place, we need to determine a Loss objective, both for the Content and for the Style images.

Content Loss

Regarding the Content Loss,
As mentioned, our objective is to fine tune the Target image such that it'll contain the content of the Content image. By mean, semantic regions, overall semantic information, Content objects, and its DC shifted (due to texture style changes) component together with a few mid-range frequencies.

For such, we need to extract the feature maps which potentially will be containing the low frequencies together with a larger receptive field information of the image.
Then, extracting the deeper feature tensors would be preferred.

In this research, we've extracted the convolutional layers $Conv_{3\_2}$, and $Conv_{4\_2}$ (for Multi-Scale Content representation) of the vgg19 architecture, both for the Content and Target images.
After, combined via linear superposition (weighted regularized loss) into the Content Loss as an L2 one.

An illustration of the above,

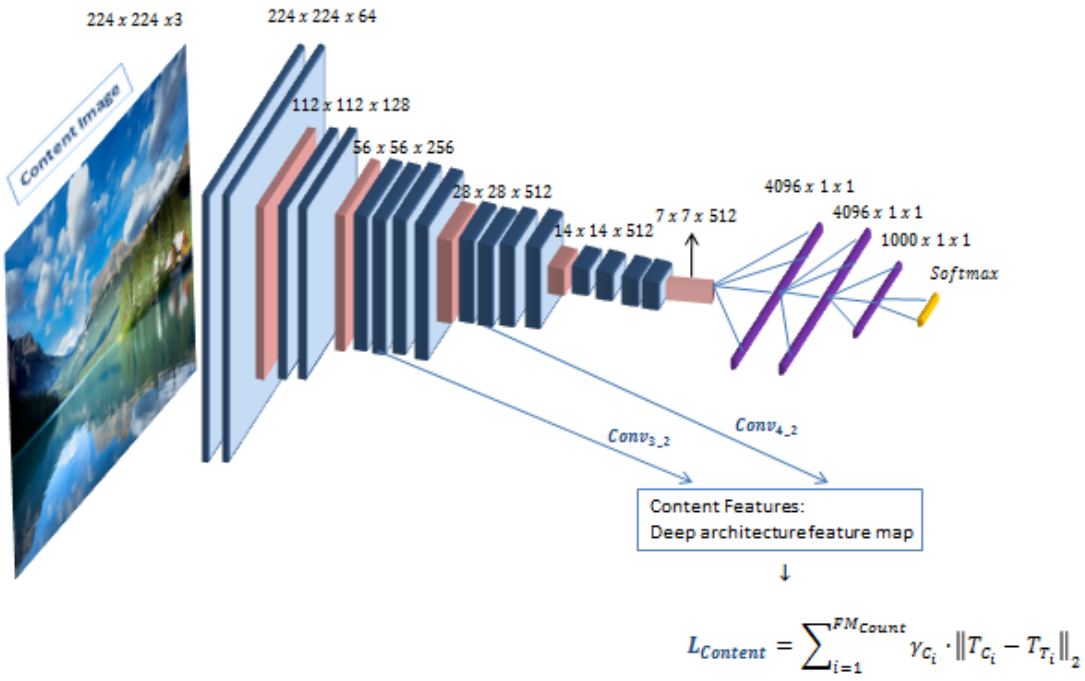

Figure 7: Content Loss illustration

Style Loss

Regarding the Style Loss,
Within the Style manner, our objective is to fine tune the Target image such that it'll contain the style of the Style image. By mean, semantic texture, semantic colors, and its AC component (due to semantic style texture) together with its high frequencies.

For such, we need to extract the feature maps which potentially will be containing the higher frequencies together with a relatively narrow receptive field information of the image.
Then, extracting the shallow feature tensors would serve our cause.

In this research, we've extracted the convolutional layers $Conv_{1\_1}$, $Conv_{2\_1}$, $Conv_{3\_1}$, $Conv_{4\_1}$, and $Conv_{5\_1}$ (for Multi-Scale Style representation) of the vgg19 architecture, both for the Style and Target images.

As for the combination of these multi-scale feature maps,
When dealing with Style objective, if we were to combine the above via a simple Frobenius Norm, we won't get a style transfer effect! Simply because the Content of the image will start to change slightly, and we'll start noticing the Content of the Style image present on the Target image.
In order to obtain a real Style Transfer, we'll need to perform a transformation.
When dealing with Style transfer, we wish to add the Style of the style image into our Target image (together with the Content of the content one via the previous relevant method). And when thinking about Style, it's the high frequency and color information. The information that's not spatial dependent, which will still exists (will still be there) even if we shuffled the image in space!
Therefore, the transformation that we should use needs to consent to the above.
One as such that it'll remove the space dependency, is the Hermitian Transformation.

The Hermitian matrix in general (or self-adjoint matrix) is a complex square matrix that is equal to its own conjugate transpose. That is, the element in the i_{th}row and j_{th} column is equal to the complex conjugate of the element in the j_{th} row and i_{th} column, for all indices i and j.

$$A\ Hermitian\ \leftrightarrow a_{ij} = \overline{a_{ji}}$$

In a matrix form:
$A\ Hermitian\ \leftrightarrow A = \overline{A^T}$, if $\overline{A^T} = A^H$ then, $A\ Hermitian\ \leftrightarrow A = A^H$

The Hermitian Transformation, when used properly, can eliminate the spatial dependency and rather focus on the correlation (Cross Covariance) between the different feature maps of each convolutional block.

In this case, in order for the HT to work, we need to represent our Tensor as a matrix $2D$ one. And in order for it to work on the spatial domain, we need to assure that our representation has all the spatial information as the Hermitian Matrix rows (That will insure, when applying the transformation, that the spatial dimension will contribute to the Hermitian Matrix values (Δ_1) and won't appear in the resulted Transformation).

$Proof\ of\ (\Delta_1)$:

$(H \cdot W)_{Row\_Tensor\_i} \times \overline{(H \cdot W)^T{}_{Column\_Tensot\_J}} \ \leftrightarrow$

$(H \cdot W)_{Row\_Tensor\_i} \times \overline{(H \cdot W)_{Row\_Tensor\_J}} \ \mid (H \cdot W)^T{}_{Column\_Tensot\_j} = (H \cdot W)_{Row\_Tensor\_j} \ \leftrightarrow$

$(H \cdot W)_{Row\_Tensor\_i} \times (H \cdot W)_{Row\_Tensor\_j} \mid \overline{(H \cdot W)_{Row\_Tensor\_J}} = (H \cdot W)_{Row\_Tensor\_j}$, When $(H \cdot W)_{Tensor}$ is a Real Signal Matrix.

■

P.S. The spatial information – (H, W) is a real signal (not complex). Then, $(H, W) = \overline{(H, W)}$. Hence, $(H \cdot W)_{Tensor} = \overline{(H \cdot W)_{Tensor}}$ (The Tensor Conjugate equals the original Tensor when dealing with real signals).

In each desired Convolutional block (For instance, $Conv_{2\_1}$), we reshape the $4D$ (4 dimensional) tensor (B, C, H, W) into a new $2D$ Tensor matrix: $(B \cdot C,\ H \cdot W)$. Then, apply the Hermitian Transformation, to get a $(B \cdot C,\ B \cdot C)$ Matrix Tensor (Δ_2).

$Proof\ (\Delta_2)$:

$(B \cdot C,\ H \cdot W)_{Matrix} \times \overline{(B \cdot C,\ H \cdot W)^T{}_{Matrix}} \ \leftrightarrow$

$(B \cdot C,\ H \cdot W)_{Matrix} \times \overline{(H \cdot W,\ B \cdot C)_{Matrix}} \overset{(\Delta)}{\leftrightarrow}$

$(B \cdot C,\ H \cdot W)_{Matrix} \times (H \cdot W,\ B \cdot C)_{Matrix} \ \leftrightarrow$

$(B \cdot C,\ B \cdot C)_{Matrix}$

■

Further explanation,

Let's consider a Hermitian Transformation Matrix on a general Feature Tensor FM_T of size (B, C, H, W) after a reshape mapping representation ϑ:

$$H\left(\vartheta\left(FM_{T_{[BxCxHxW]}}\right)\right) = H\left(FM_{T_{[BxC,\ HxW]}}\right) =$$

$$= FM_{T_{[BxC,\ HxW]}}\ x\ \overline{FM_{T_{[BxC,\ HxW]}}}^{T} =$$

$$= \begin{pmatrix} \alpha_{1x1} & \alpha_{1x2} & \cdots & \alpha_{1x(H\cdot W)} \\ \alpha_{2x1} & \alpha_{2x2} & & \alpha_{2x(H\cdot W)} \\ \vdots & & \ddots & \vdots \\ \alpha_{(B\cdot C)x1} & \alpha_{(B\cdot C)x2} & \cdots & \alpha_{(B\cdot C)x(H\cdot W)} \end{pmatrix}_{B\cdot C\ x\ H\cdot W}$$

$$\times \overline{\begin{pmatrix} \alpha_{1x1} & \alpha_{1x2} & \cdots & \alpha_{1x(H\cdot W)} \\ \alpha_{2x1} & \alpha_{2x2} & & \alpha_{2x(H\cdot W)} \\ \vdots & & \ddots & \vdots \\ \alpha_{(B\cdot C)x1} & \alpha_{(B\cdot C)x2} & \cdots & \alpha_{(B\cdot C)x(H\cdot W)} \end{pmatrix}_{B\cdot C\ x\ H\cdot W}}^{T} \overset{(\Delta_1)}{=} FM_{T_{[BxC,\ HxW]}}\ x\ FM_{T_{[HxW,\ BxC]}}$$

$$= \begin{pmatrix} \alpha_{1x1} & \alpha_{1x2} & \cdots & \alpha_{1x(H\cdot W)} \\ \alpha_{2x1} & \alpha_{2x2} & & \alpha_{2x(H\cdot W)} \\ \vdots & & \ddots & \vdots \\ \alpha_{(B\cdot C)x1} & \alpha_{(B\cdot C)x2} & \cdots & \alpha_{(B\cdot C)x(H\cdot W)} \end{pmatrix}_{B\cdot C\ x\ H\cdot W}$$

$$\times \begin{pmatrix} \alpha_{1x1} & \alpha_{1x2} & \cdots & \alpha_{1x(B\cdot C)} \\ \alpha_{2x1} & \alpha_{2x2} & & \alpha_{2x(B\cdot C)} \\ \vdots & & \ddots & \vdots \\ \alpha_{(H\cdot W)x1} & \alpha_{(H\cdot W)x2} & \cdots & \alpha_{(H\cdot W)x(B\cdot C)} \end{pmatrix}_{H\cdot W\ x\ B\cdot C} \overset{\Delta_2}{=} FM_{new\_T_{[BxC,\ BxC]}}$$

$$\overset{\Delta_3}{=} \begin{pmatrix} \beta_{1x1} & \beta_{1x2} & \cdots & \beta_{1x(B\cdot C)} \\ \beta_{2x1} & \beta_{2x2} & & \beta_{2x(B\cdot C)} \\ \vdots & & \ddots & \vdots \\ \beta_{(B\cdot C)x1} & \beta_{(B\cdot C)x2} & \cdots & \beta_{(B\cdot C)x(B\cdot C)} \end{pmatrix}_{B\cdot C\ x\ B\cdot C}$$

As for Δ_3,

We have mentioned earlier (Page 7, the second paragraph before Δ_1 proof) that a Hermitian Transformation Matrix measures the correlation in between the different feature maps within the same convolutional block. And as such, it can be illustrated as a Cross Covariance Matrix which its Stochastic Processes are almost standardized (with zero mean).

Proof,

$$H\left(\vartheta\left(FM_{T_{[BxCxHxW]}}\right)\right) = H\left(FM_{T_{[BxC,\ HxW]}}\right) = FM_{T_{[BxC,\ HxW]}}\ x\ \overline{FM_{T_{[BxC,\ HxW]}}}^{T} =$$

$$\overset{(\Delta_1)}{=}\ FM_{T_{[BxC,\ HxW]}}\ x\ FM_{T_{[HxW,\ BxC]}} = (*)$$

By mean,

The correlation is being calculated between each row stochastic vector with all of the other rows of the same ϑ transformed matrix.

This can be described as the Cross Covariance Matrix between stochastic, almost standardized processes,

$$(*)_{[i,\ j]} = \mathbb{E}\left[(\mathrm{X}_i - \mathbb{E}[\mathrm{X}_i]) \cdot \left(\mathrm{Y}_j - \mathbb{E}\left[\mathrm{Y}_j\right]\right)\right]\ |\ \mathbb{E}[\mathrm{X}_i] = \mathbb{E}[\mathrm{Y}_i] = 0\ \Rightarrow\ \mathbb{E}\left[(\mathrm{X}_i) \cdot \left(\mathrm{Y}_j\right)\right]$$

And when X_i , Y_j: $A\ singel\ FM_{[HxW]}$,

$$(*)_{[i,\ j]} \overset{K_{XY}}{=}\ CCM\left(FM_{i_{[HxW]}}\ ,\ FM_{j_{[HxW]}}\right)$$

■

P.S.

- We're not seeking a sparse model (and hence a spanning base kernel model). So a Manhattan Euclidean Distance\Taxicab Norm (L1 Norm). A Frobenius Norm (L2 Norm) is then required for convergence.
- We apply a Multi-Scale Feature extraction method for better objective representation. That way, we can extract different, yet task relevant, features which obtain the required information.
  Using the above, we compose different features across various task-relevant layers of the neural network, each containing different signals in different frequencies, different receptive fields, and a variety of semantic regions and textures. We need to span the dimensionality of the model as a whole.

If to put the above into perspective. An illustration of the Style Loss,

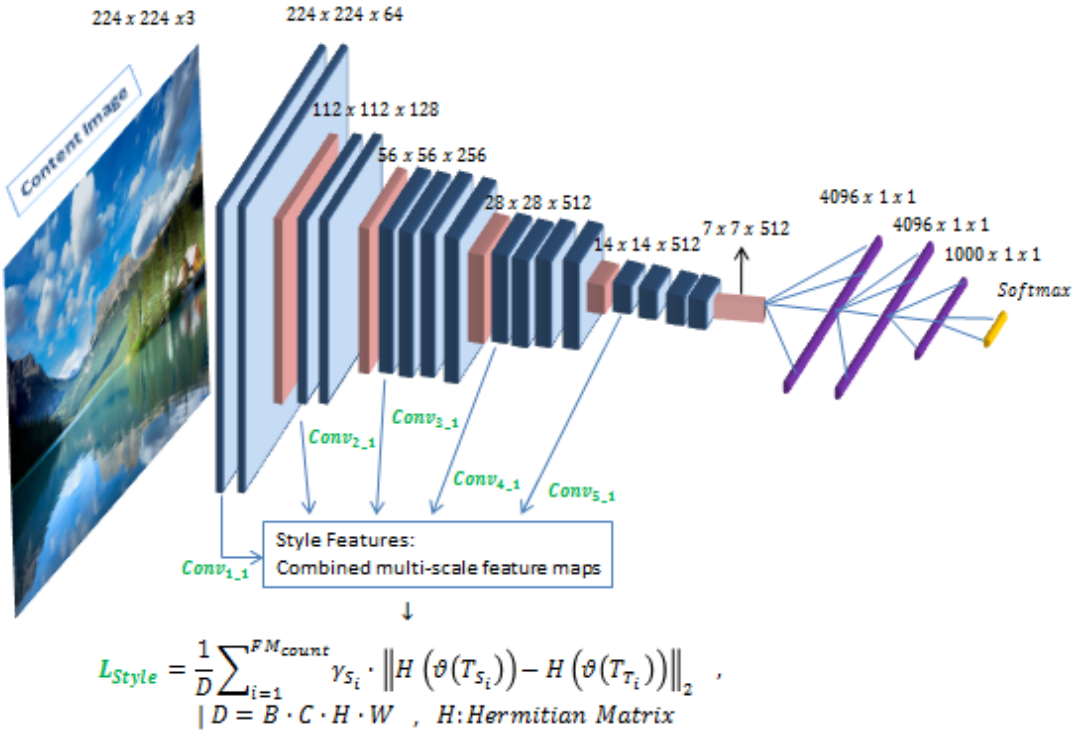

Figure 8: Style Loss illustration

<u>Optimizer</u>,

For our case we used Adam Optimizer for updating our Target image,

$$\theta_t = \theta_{t-1} - \alpha \cdot \frac{\eta}{\sqrt{\widehat{v_t} + \varepsilon}} \cdot \widehat{m_t} \quad , \quad \alpha = 3 \cdot 10^{-4} \left(\sim \frac{1}{3}\ of\ 10^{-3}\right)$$

Where,

$$\widehat{m_t} = \frac{m_t}{1 - \beta_1^t} \quad , \quad \widehat{v_t} = \frac{v_t}{1 - \beta_2^t}$$

And,

$$m_t = \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_{t-1}$$
$$v_t = \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_{t-1}^2$$

$$g_t = \nabla L_{total}(\theta_t) = \frac{\vartheta L_{total}}{\vartheta \theta_t}$$
$$g_t - The\ relative\ gradient\ in\ step\ t$$

Combining all the previous for a Total Target Multi-Scale Loss Objective,

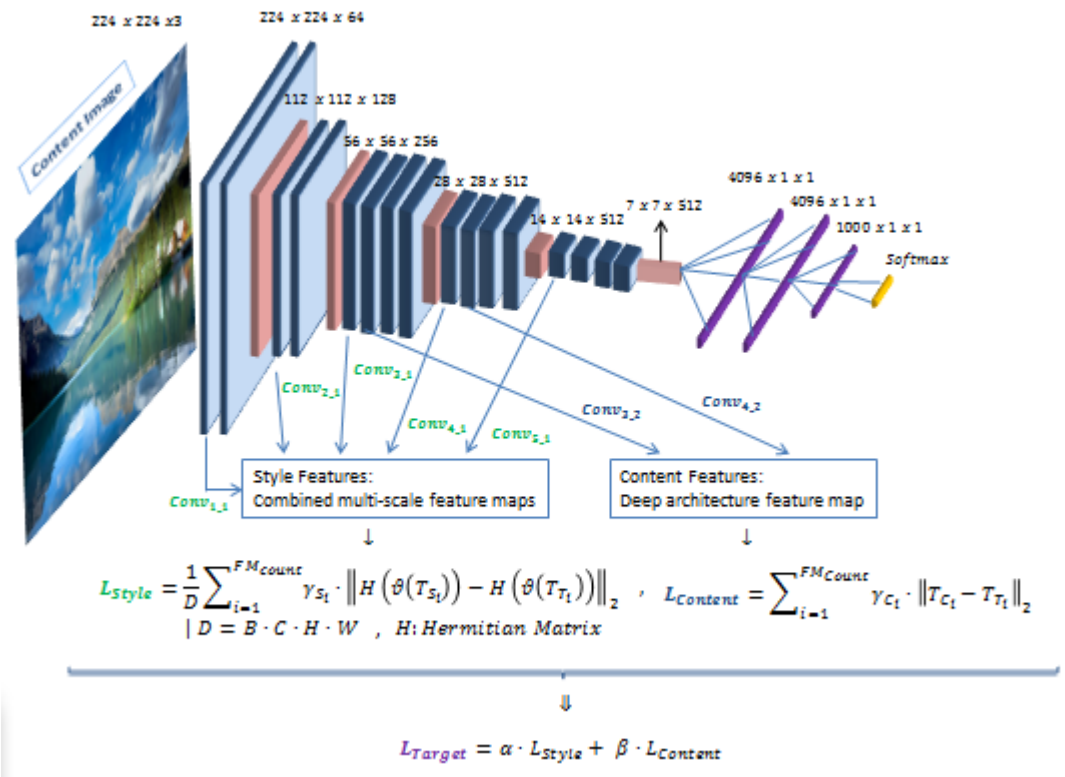

Figure 9: Total Target Loss Objective

Results

For our Content images, we used the following,

For our Style images, we used the following,

After implementing the above via our autonomous Style Transfer code, applying it to multiple Content and Style images, and adjusting and tuning all the network Hyperparameters. We present the following results,

Golden Gate Bridge (San Francisco):

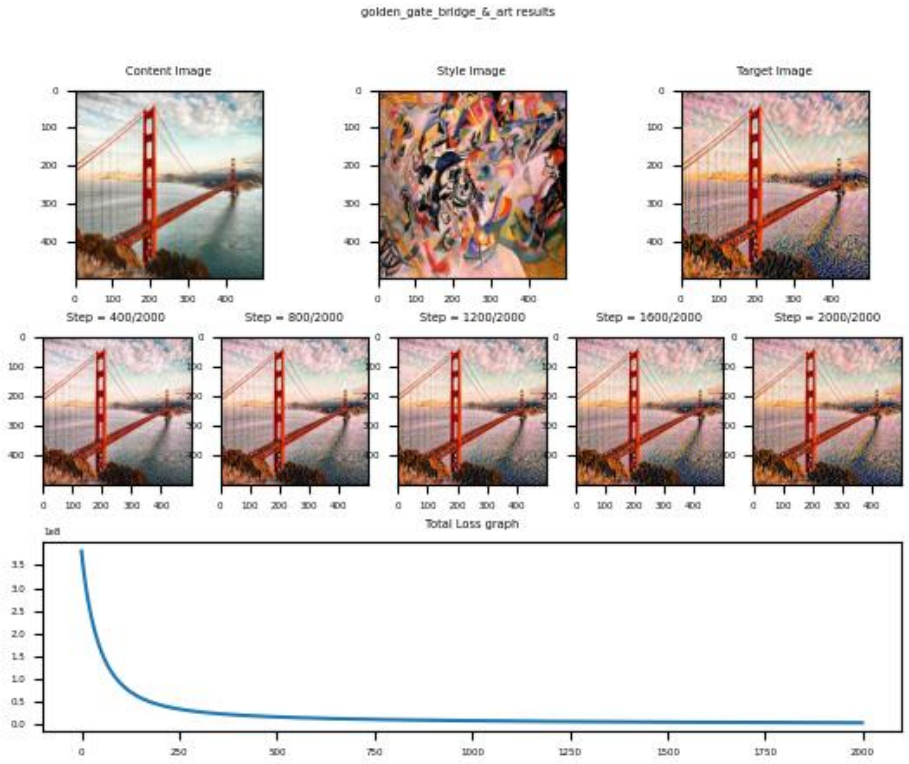

Figure 10: Great Golden Bridge in San Francisco Content image combined with the Art Style image

An actual image of the Golden Bridge at sunset,

Seto Land Bridge (Japan):

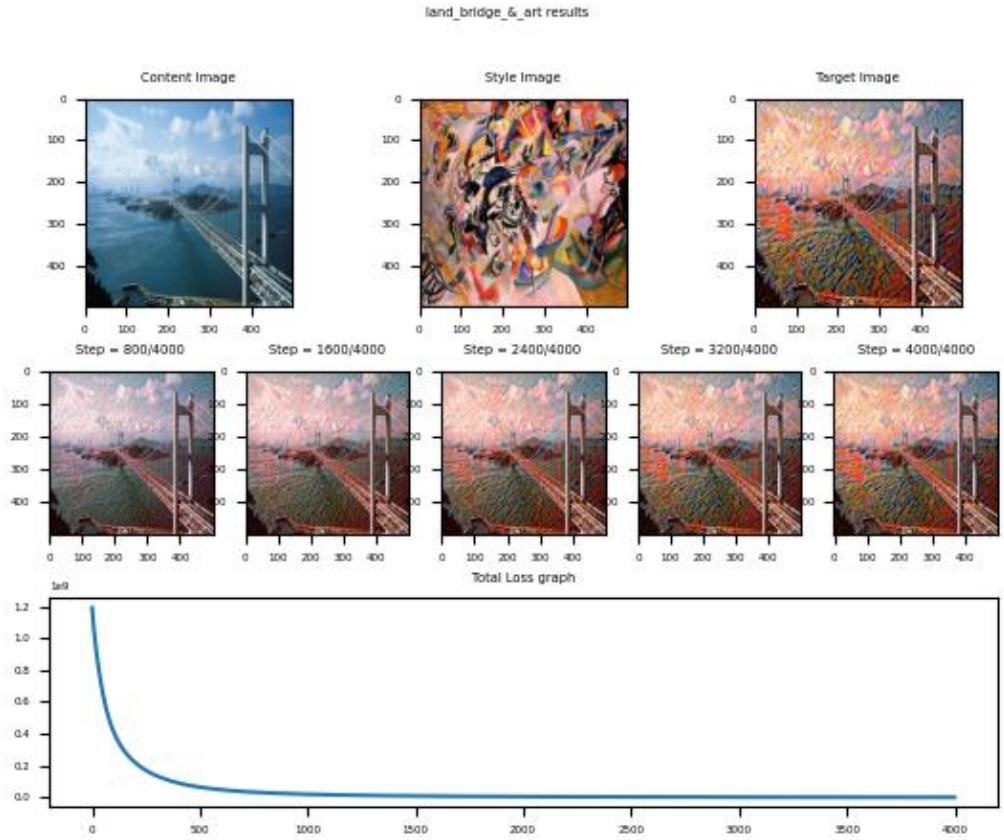

Figure 11: Seto Bridge in Japan Content image combined with the Art Style image

Mona Lisa (Da Vinci):

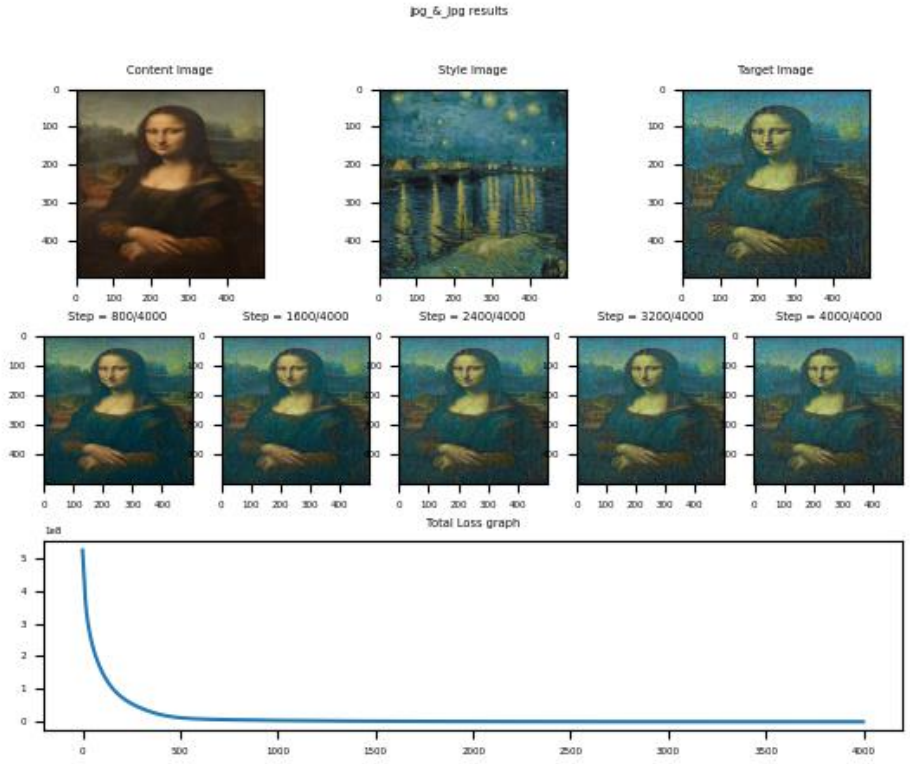

Figure 12: Mona Lisa by Da Vinci Content image combined with the Vincent Van Gogh Style image

Switzerland Houses:

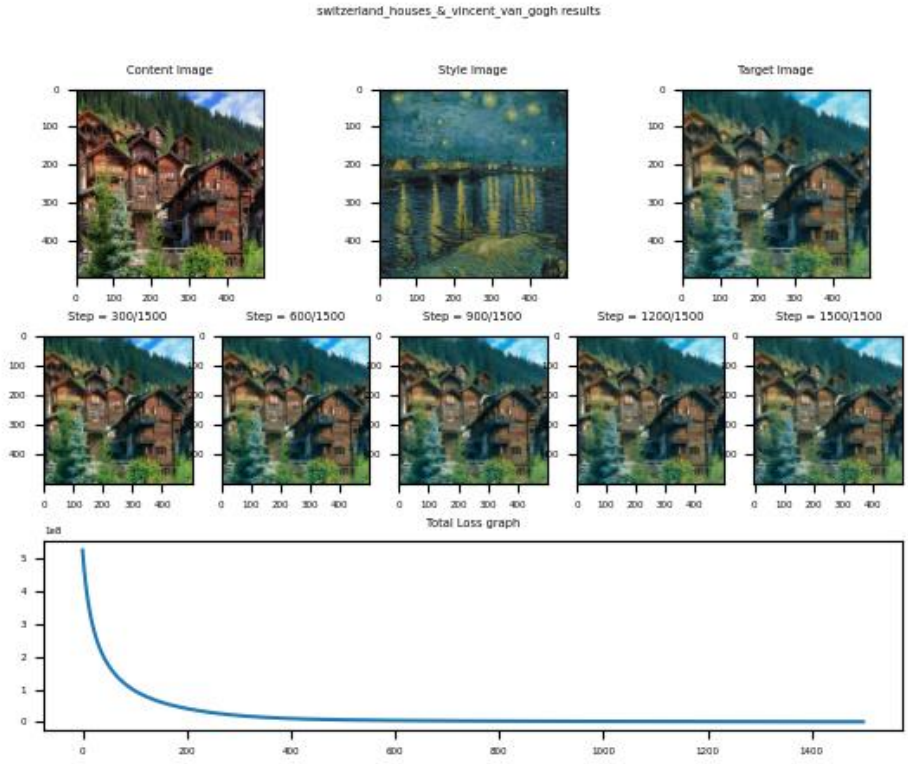

Figure 13: Switzerland Houses Content image combined with the Vincent Van Gogh Style image

First Landscape:

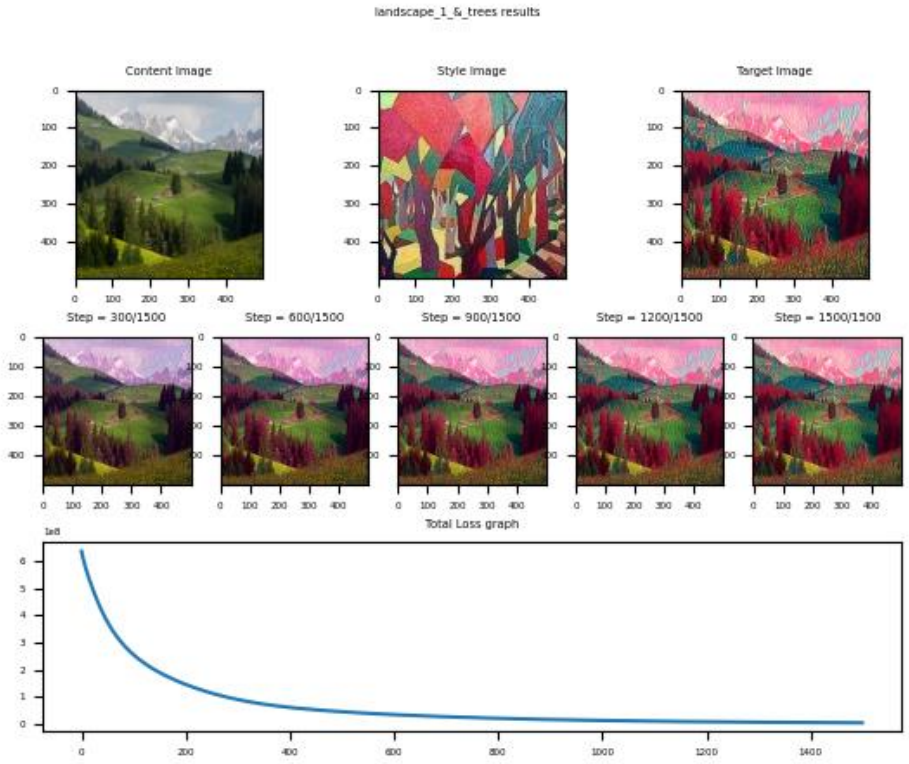

Figure 14: First Landscape Content image combined with the Trees Style image

Second Landscape:

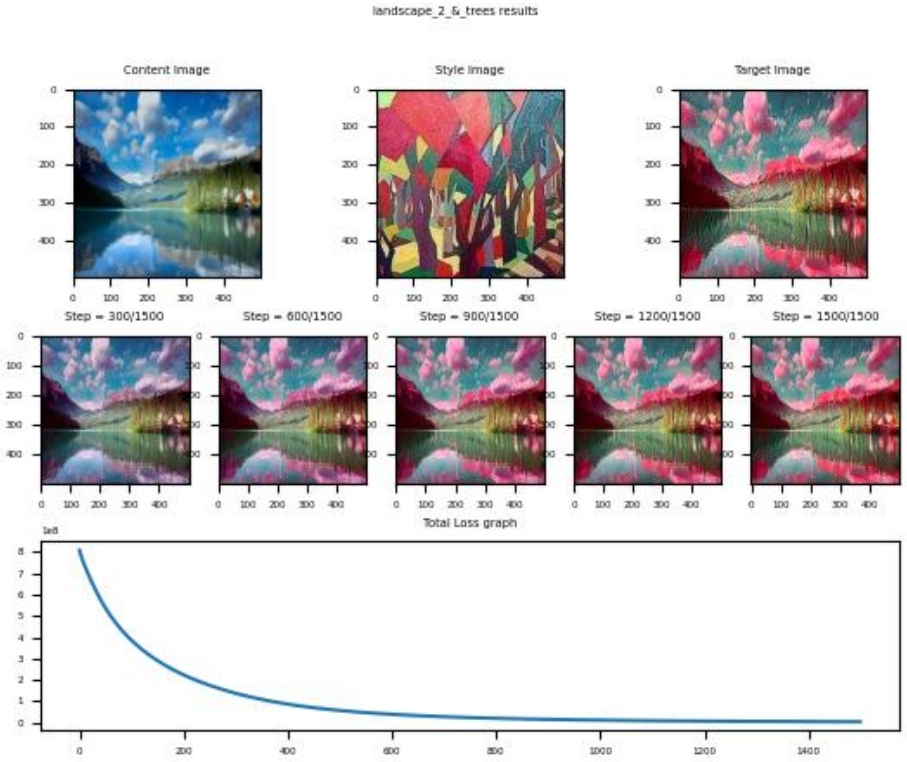

Figure 15: Second Landscape Content image combined with the Trees Style image

And for a bonus image,

It'd be interesting to see how a Style image would be affected when being applied on a Style Transfer with other Content image!

It's rather different situation since for the style transfer downstream task, it's usually chosen a Style image that has a certain style into it. By mean, its variance is steady (only First momentum part in the Gradient objective, rather than second – For example, the Hessian Matrix (Jacobian Matrix of the Gradient of a Signal)).

And when taking into consideration such an image (usually Content one can consent to the above), we'll be having a second and third degree momentum. Thus, enthralling to see what the model will come up with.

Without further ado,

Vincent Van Gogh:

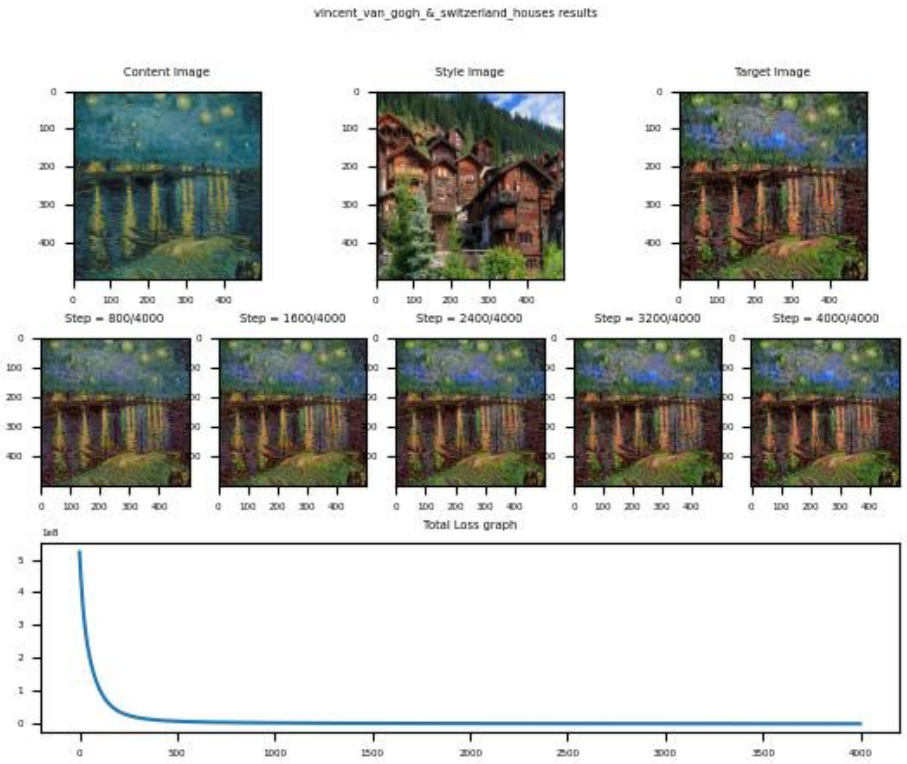

Figure 16: Vincent Van Gogh as Content image combined with Switzerland Houses as Style image

We can notice here that also the color and texture of the major Semantic regions within the Style image were transferred into the Content image. And one might say, in such an interesting manner.

The transferred texture is then applied partially in a vertical direction, due to the nature of houses! Together with some other formats, this is truly exciting.

If going a step further, a very exiting experiment to try, is to run the code (with a few hyperparameter tuning – mostly number of steps and weight values and ratio) when the Target image's initial state is to be,

- A Random Gaussian Noise
- A White Canvas

This is a huge step for both the performance monitoring and the algorithm\model dimensionality reduction (finding the right Probably Approximately Correct Learnable function (model including transformations) in a highly dimensional space).

After implementing some corrections (as mentioned above regarding tuning),

Target initialization as a Gaussian Random Noise

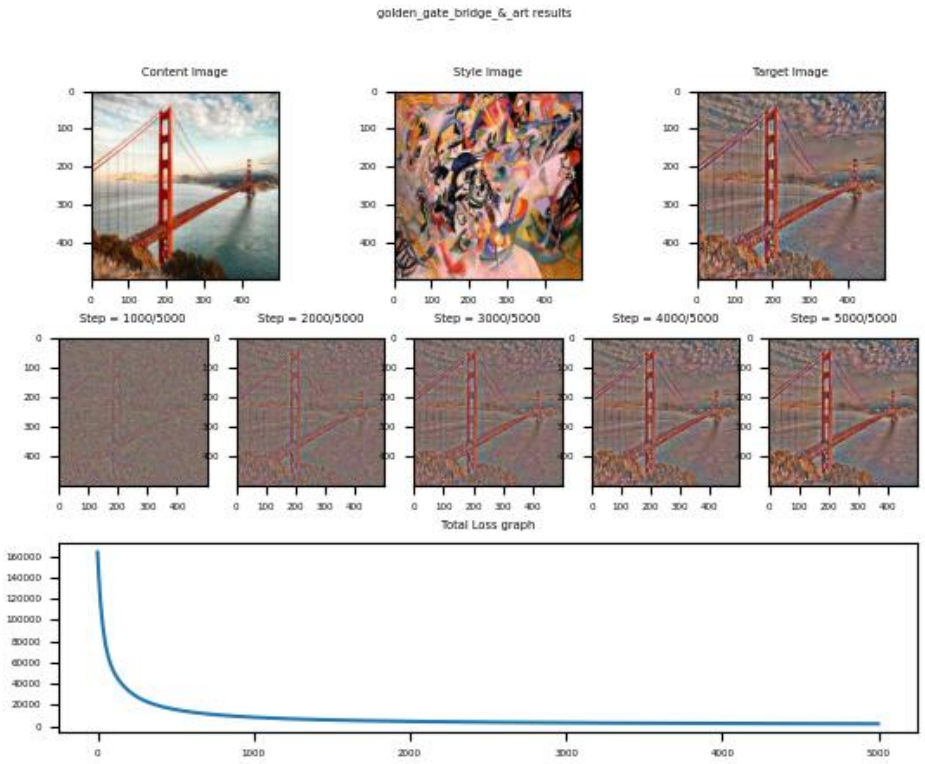

Figure 17: Great Golden Bridge in San Francisco Content image combined with the Art Style image when Target image is initiated to be a RGN

We can notice that the algorithm works! (When the right measures are to be applied)

Here, the parameters differentiation are,

- The optimization is being processed for: $5x10^3$ steps
- The STN ratio of the Content relatively to the Style is: $1x10^2$
- The weight of Content Loss in the Total Loss Objective is: $1x10^3$
- The weight of Style Loss in the Total Loss Objective is: $1x10^1$

Target initialization as a White Canvas

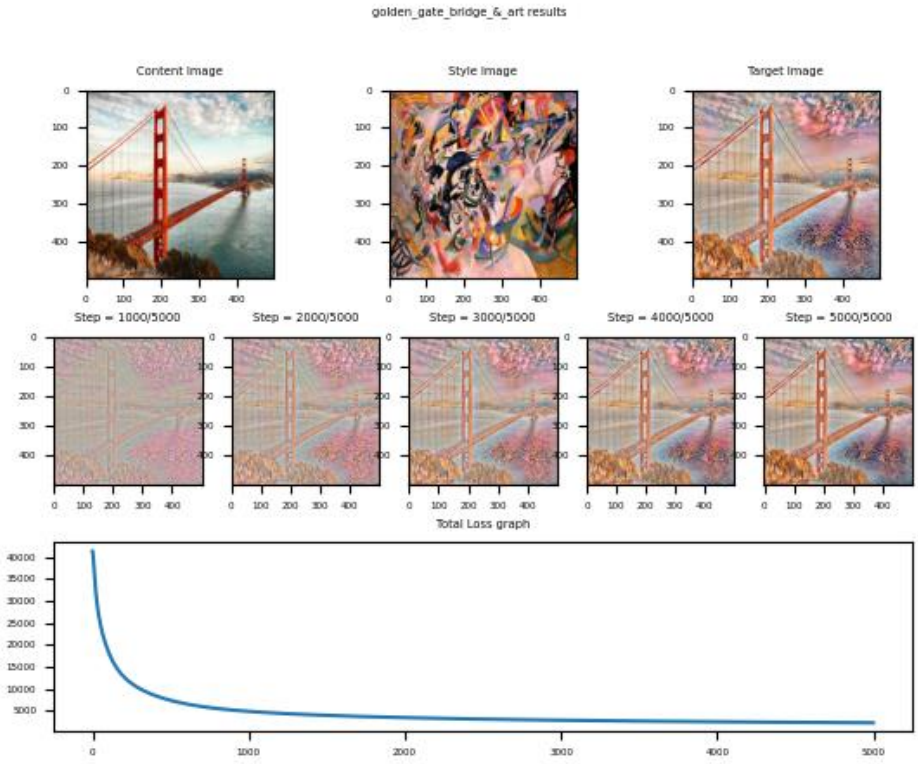

Figure 18: Great Golden Bridge in San Francisco Content image combined with the Art Style image when Target image is initiated to be a WC

We can notice that the algorithm works! (When the right measures are to be applied)

Here, the parameters differentiation are also as above,

- The optimization is being processed for: $5x10^3$ steps
- The STN ratio of the Content relatively to the Style is: $1x10^2$
- The weight of Content Loss in the Total Loss Objective is: $1x10^3$
- The weight of Style Loss in the Total Loss Objective is: $1x10^1$

It is truly exciting to see such results!

Final Thoughts

Building and planning for such a model was pretty intense. The objective is novel, and the method\algorithm is hard as it is.
In order for the idea to work, we had to come up with a Hermitian transformation which represents a Cross Correlation matrix channel wise between feature maps of CNN blocks.

Acing such model, transformations, loss objectives, and extracting FM layers, consists in finding a point of matter via low distortion embeddings through an extremely high dimensional trajectory. We are feasible for finding such models for specific tasks, but not such that can be applied to our case. Especially when we're dealing with highly dimensional parameter space, in-convex functions (they are far from being convex), and the parameters are continuous.

Despite the infeasibility, we've managed to create such a method. And the results are just fascinating!

"Artificial intelligence is built upon the idea of a functional neural brain cell. Deep learning will lead to deep understanding, while shallow observations might work just for now, but won't last for the long run".

M. A