# Neural Style Transfer

**Mrinal Jain**
New York University
mrinal.jain@nyu.edu

**Jairam Venkitachalam**
New York University
jv1589@nyu.edu

## Abstract

Neural style transfer involves manipulating an image to have another image's artistic qualities and style using deep learning. Gatys et al.'s formative work in this domain is influential and has attracted a wide variety of research to improve the algorithm, either in terms of speed or quality of the transfer. Moreover, many extensions have been studied that adapt style transfer for specific applications (e.g., preserving color, photo-realism, etc.). Our project aims to explore and re-implement some of these techniques and analyze the plausible directions of research in this domain. Our code is publicly available.[1]

## 1 Introduction

The very idea of how humans perceive and create artistic imagery has fascinated people throughout history. However, until quite recently, artificial systems with similar capabilities did not exist. Advancements in deep learning have ushered a new era in the domain of image stylization, with the work by Gatys et al. in [2] gaining much traction with researchers. They proposed an algorithm for Neural Style Transfer, which at a high-level is the technique of recomposing one image (e.g., a photograph) in the artistic style of another image (e.g., a famous piece of artwork). The algorithm proposed by Gatys utilizes the fact that feature representations generated by pre-trained convolutional neural networks can be segregated to have the individual style and content information of the original image. This opens up the possibility to mix the style and content of two different images to generate the so-called "stylized image."

In the following years, research has been conducted to improve style transfer, both in terms of quality [3] and speed [5]. The authors in [3] explored many techniques providing minor to significant improvements in the quality of the generated stylized images. On the other hand, Justin et al. attempted to overcome a shortcoming of the algorithm by training a network that can generate these stylized images instead of directly synthesizing a specific image. [4] extends the style transfer procedure to preserve the content image's color in the synthesized image - which the original method failed to do. We re-implement the methodologies mentioned above and analyze their advantages and limitations.

The remainder of this paper is organized as follows:

- *Section 2* goes in detail into each of the methods under consideration and attempts to explain the rationale behind different techniques.
- In *Section 3*, we present our results and observations.
- *Section 4* explores some potential directions of research in the domain of neural style transfer, and we conclude our work in *Section 5*.

[1] https://github.com/MrinalJain17/neural-style-transfer

## 2 Method

### 2.1 A Neural Algorithm of Artistic Style

The seminal work by Gatys et al. [2] published in 2015 laid much of the foundation for modern style transfer methods. The basic idea was to use a convolutional neural network to extract an image's style and content representations. The authors proposed that these representations of content and style are separable - and one can intermix the style and content representations of two different images to produce a "stylized image."

The new image is synthesized to match one image's content (e.g., a photograph) and another image's style (e.g., a piece of art). An important aspect is that instead of manipulating the pixel space, the authors work in the images' feature space. They used a pre-trained VGG network [1] to extract these feature representations, which were then used to optimize the third image - the stylized image jointly. *Figure 1* represents the network used to extract the feature representations.

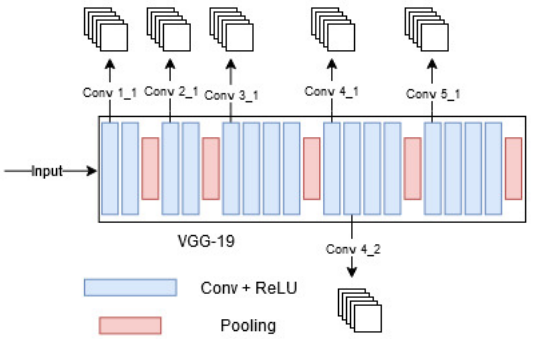



Figure 1: Extracting feature maps from a VGG-19 Network.

**Content Representation** In general, the hierarchical nature of neural networks enables deeper layers of a network to learn global information. Therefore, as we move forward in the hierarchy, the layers are concerned more with the overall content than individual pixel-level information. Based on this reasoning, the authors use the feature maps generated by layer `conv4_2` of the VGG-19 network as the representation for the content of an image.

Let $\overrightarrow{p}$ and $\overrightarrow{x}$ be the original image and the image to be generated, and $P^l$ and $X^l$ be their respective feature maps in layer $l$. Here, $P^l, X^l \in R^{N_l \times M_l}$ where layer $l$ has $N_l$ distinct feature maps of size $M_l$ (when flattened). For example, $P^l_{ij}$ is the activation in the $i$th filter at the (spatial) position $j$ in layer $l$. The content loss at layer $l$ is then defined as:

$$L_{\text{content}} = \frac{1}{2N_l M_l} \sum_{i,j} (X^l_{ij} - P^l_{ij})^2$$

**Style Representation** The authors capture the style of an image by constructing a feature space on top of these feature responses generated by the VGG network. It involves computing a "Gram matrix" - which intuitively, is like a co-occurrence matrix consisting of the correlations between different pairs of feature maps. *Figure 2* demonstrates the construction of the gram matrix. Five layers from the VGG network (`conv1_1`, `conv2_1`, `conv3_1`, `conv4_1`, and `conv5_1`) were used to capture the style of an image.

For an image with feature maps $F^l \in R^{N_l \times M_l}$, the gram matrix in layer $l$ is defined as:

$$G^l = \langle F^l_i, F^l_j \rangle \; ; G^l \in R^{N_l \times N_l}$$

Where $F_i$ and $F_j$ are two feature maps in the same layer. The loss from layer $l$ is defined as:

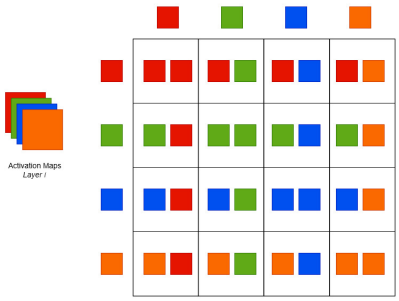



Figure 2: Construction of gram matrix from feature maps of a specific layer.

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum (G^l - A^l)^2$$

Where $G^l, A^l$ correspond to the gram matrix of the original image and the image to be generated respectively. The total style loss is then computed as:

$$\boxed{L_{\text{style}} = \frac{1}{L} \sum_{l=1}^{L} E_l}$$

The aggregate loss is a weighted sum of the content and the style loss, with $\alpha$ being the content-style trade-off:

$$L_{\text{total}} = \alpha L_{\text{content}} + L_{\text{style}}$$

We summarize the entire style transfer algorithm in *Figure 3*. Note that the generated image $y_g$ is iteratively optimized by the gradients computed with respect to $L_{\text{total}}$.

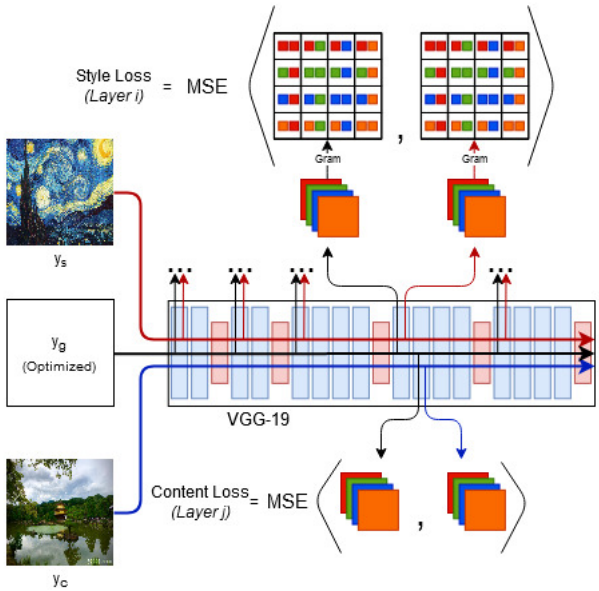



Figure 3: Overview of Neural Style Transfer by Gatys et al.

### 2.2 Improving the Neural Algorithm of Artistic Style

The work in [3] attempts to improve the original style transfer algorithm **qualitatively**. They use the same setup as described in *Figure 3*, but propose the following improvements:

**Activation shift** The authors argue that only a few activations in the feature maps are non-zero, and as a consequence, the gram matrices are sparse (visualized in *Figure 4*) - something which degrades the quality of the style transfer. The idea is that the gram matrix entries represent the correlation between a pair of features, and such a value being zero is inherently ambiguous - are either of the features absent, are both of the features absent, or do the two features never appear together? It is then up to the optimizer to interpret these ambiguous values, essentially being detrimental to convergence. The solution provided by the authors is to shift the activations by a constant (selected to be 1) before computing the gram matrix.

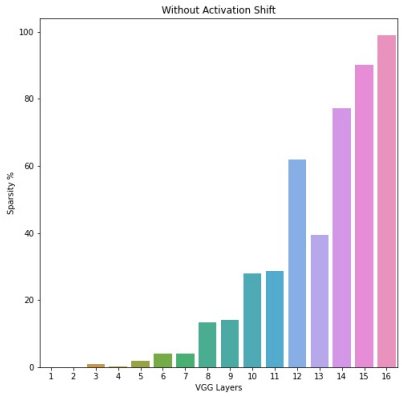



Figure 4: Sparsity in the gram matrix without activation shift.

**More style layers** Intuitively, assuming that having a set of style layers leads to better transfer of the style, it makes sense to extend the five layers that were used in [2] to include all the 16 convolutional layers in the VGG network for the computation of the style loss $L_{\text{style}}$.

**Chain of inter-layer feature correlations** Note that when computing the style loss in [2], the gram matrix's feature correlations are computed only within the same layer $l$. The authors extend the existing set of gram matrices $\{G^l\}$ to capture feature correlations from responses belonging to different layers $l$ and $k$. More specifically, the variant they boiled down to involved computing the feature correlations (and hence the gram matrix) using the feature maps from adjacent layers $l$ and $(l+1)$. This minor difference in construction is visualized in *Figure 5*.

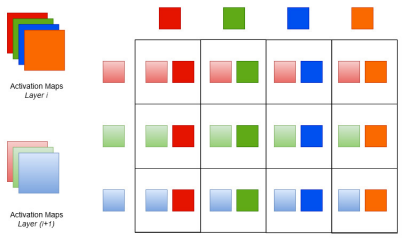



Figure 5: Construction of a chained gram matrix: Capturing feature correlations from adjacent layers of the VGG network.

Let $F^l \in R^{S_l \times M_l}$ and $F^k \in R^{T_l \times O_l}$ correspond to the feature maps from layer $l$ and $k$ respectively. Note that since the two layers could have different spatial dimensions, the smaller of the two feature maps are interpolated to match the larger one's size. The gram matrix is then computed as:

$$G^{lk} = \langle F^l, \text{Up}(F^k) \rangle \; ; G^l \in R^{S_l \times T_l}$$

**Geometric weighing scheme** We know that most of the content information is contained in the deeper layers. However, the earlier layers capture the style information better. The authors propose a geometric weighting scheme that gives more importance to the earlier layers. If there are a total of $N$ layers, then:

$$w_i = 2^{N-i} \; ; i \in [1, N]$$

### 2.3 Preserving Color in Neural Artistic Style Transfer

While the original algorithm in [2] transfers the style onto the content image, it inadvertently transfers the color scheme from the style image onto the content image. This paper [4] was an extension by Gatys et al. to overcome the shortcoming. We use one of the two methods proposed by the author, where the style transfer is applied only on the luminance of the style and content image. This is because the human perception of artistic style is highly dominated by the image's luminance rather than the color content. Finally, the original content image's color is added back to the generated (luminance) image. This color-luminance separation is performed in the *YIQ* color space. The color preserving procedure is surmised in *Figure 6*.

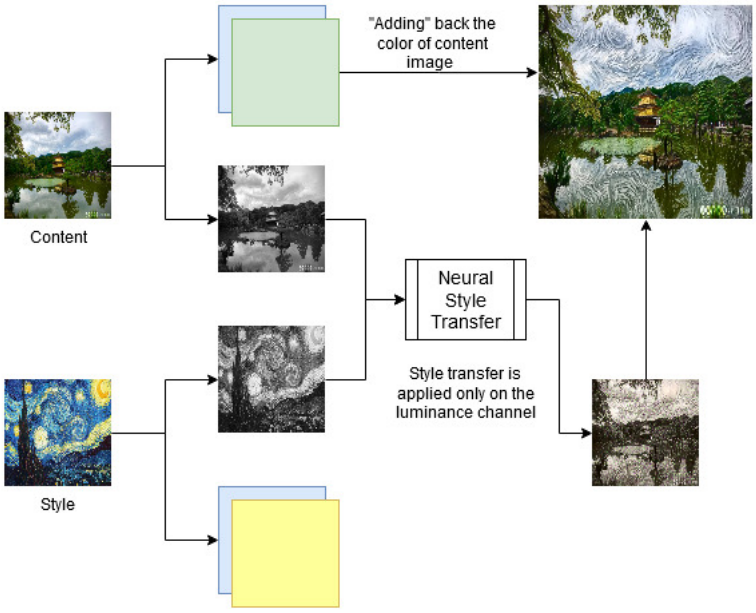



Figure 6: Color preservation as an extension of Neural Style Transfer.

### 2.4 Perceptual Losses for Real-Time Style Transfer

A limitation of the original algorithm proposed by Gatys et al. is that it works just for a single style and content image at a time by directly synthesizing the stylized image - being inherently slow. Justin et al. in [5] proposed a method capable of performing style transfer in real-time. The authors used a very similar setup as in [2], and combined the style and content loss functions under a unified umbrella called the *perceptual loss*.

However, instead of directly optimizing to generate a stylized image, they trained a network (called the *Transformation Network*) to learn to produce the said image. Their network can learn to generate stylized images for a specific style, but **any given content image**.

A high-level overview of the fast style transfer method is presented in *Figure 7*. The authors iterated over the images in the COCO data set [7] and used each image as a content image to train the transformation network.

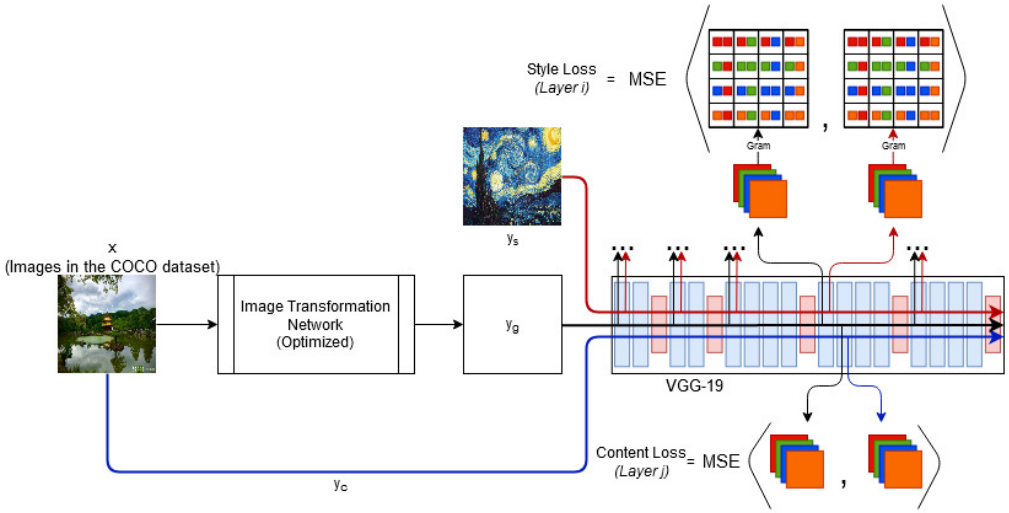



Figure 7: Fast Style Transfer by Justin et al.

**Total Variation Loss** The authors in [5] added a term called the total variation loss to smooth the generated (stylized) images. This loss penalizes the variation between adjacent pixels (obtained by shifting the image in both x and y directions by 1 pixel). However, we did not observe any visual differences when using this and the existing style and content losses.

## 3 Experiments

A primary hyper-parameter to tune in the existing setup is $\alpha$: the content-style trade off. After performing a bunch of experiments, we got the best results for $\alpha = 5.25 \times 10^{-7}$, weighing the style orders of magnitude more than the content loss. ***Figure 10* portrays the effect of the different methodologies presented in *Section 2*.**

Some notable differences between our implementation and the papers:

1. The authors in [2] replaced the max-pooling operation in the VGG with average-pooling. However, we obtained better (qualitative) results with the former.
2. Originally, the L-BFGS optimizer was used to synthesize the new image. We used an out-of-the-box Adam optimizer, which produced visually more pleasing results than the L-BFGS counterpart.
3. Initializing the image to be generated with the content image instead of random white noise often led to improved results.

**Normalized VGG** The authors in [2, 3, 5] used a normalized variation of the pre-trained VGG network. What this means is that the activations produced by the layers of the normalized VGG are centered at $\mu = 1$. The reason is that since the magnitude of the activations can vary significantly among the layers, the value of the style loss would be dominated by those higher-magnitude activations. This effect is verified in *Figure 8*. However, based on our experiments,[2] a normalized VGG led to a sub-par transfer of style in general.

[2]Our implementation of the normalized VGG: https://github.com/MrinalJain17/vgg-normalized

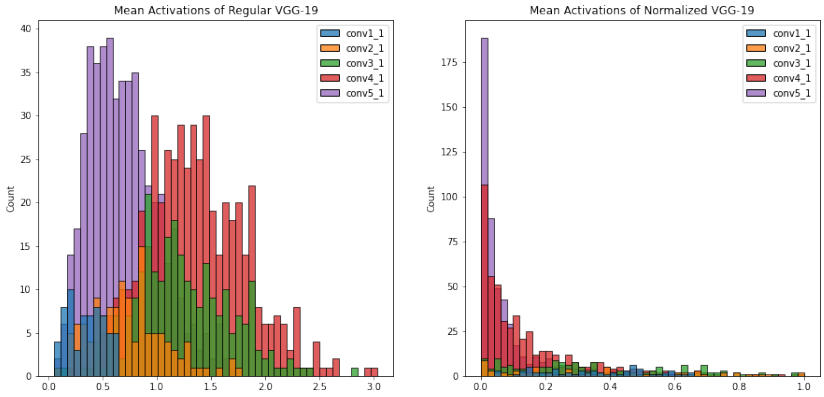



Figure 8: Distribution of activations in regular vs. normalized VGG network.

**Training Curves** Finally, before presenting our results, *Figure 9* displays the behaviour of the content and style loss over the optimization procedure for the original [2] and the improved [3] methods. Note that the magnitude of both style and the content loss is better for improved style transfer. However, as we will discuss later in *Section 4*, the value of the loss function does not indicate the quality of the style transfer.

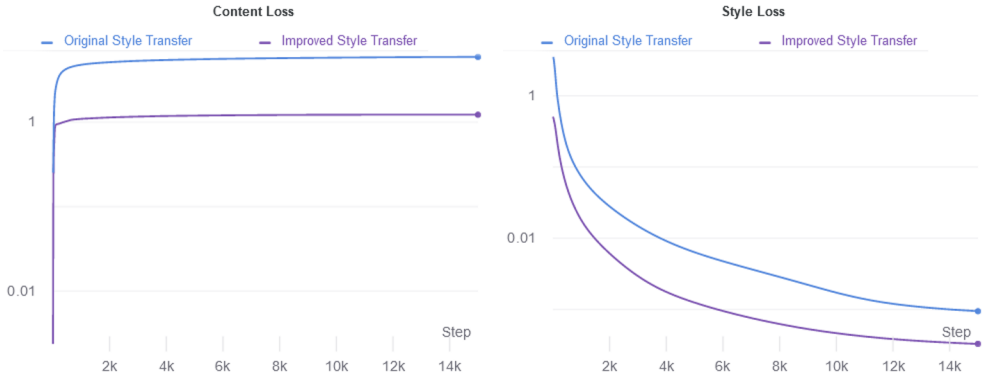



Figure 9: Content and style loss for original vs. improved style transfer.

## 4 Discussion

Following are the areas which we think are ripe for research:

**No Quantitative Measure to Evaluate the Quality of Style Transfer** To date, there is little to no research that can quantitatively evaluate the quality of a transfer. This poses a significant problem because researchers have to rely on crowd-sourcing to quantify their experiments and compare them among different techniques. Though unique, human judgment is neither a reliable nor scalable metric to evaluate different style transfer techniques.

**The Trade-off between Speed and Quality** It can be observed in *Figure 10 (d) vs. (f)* that the image generated using the fast style transfer is not as visually pleasing as the one generated by direct optimization. Fast style transfer offers the ability to work in real-time, but it seems like that comes with a compromise. It should be interesting to try and match the quality of fast methods with their original counterparts.

**Virtually No Generalization** The original algorithm of neural style worked for a specific style and content image. The fast style transfer improved upon this by working for any content image (although one still needs to train a separate network for each style image). A significant leap would be to have general methods that can work for arbitrary style and content images - while still maintaining the generated results' quality.

## 5 Conclusion

We surveyed and re-implemented the original artistic algorithm of neural style [2], followed by improvements in terms of the quality [3] and speed [5] of the style transfer. Moreover, we added the capability to preserve color from the content image in the generated image following the work in [4]. Reproducibility of experiments and extensive visualization was something that we focused on predominantly.

We have created an interactive report with many more sample results obtained using our implementation of Neural Style Transfer. **The visualizations can be viewed here - Link to visualizations.**

## References


[1] Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. CoRR, abs/1409.1556. https://arxiv.org/abs/1409.1556 2

[2] Gatys, L.A., Ecker, A.S., & Bethge, M. (2015). A Neural Algorithm of Artistic Style. ArXiv, abs/1508.06576. https://arxiv.org/abs/1508.06576 1, 2, 4, 5, 6, 7, 8

[3] Novak, R., & Nikulin, Y. (2016). Improving the Neural Algorithm of Artistic Style. ArXiv, abs/1605.04603. https://arxiv.org/abs/1605.04603 1, 4, 6, 7, 8

[4] Gatys, L.A., Bethge, M., Hertzmann, A., & Shechtman, E. (2016). Preserving Color in Neural Artistic Style Transfer. ArXiv, abs/1606.05897. https://arxiv.org/abs/1606.05897 1, 5, 8

[5] Johnson J., Alahi A., Fei-Fei L. (2016) Perceptual Losses for Real-Time Style Transfer and Super-Resolution. In: Leibe B., Matas J., Sebe N., Welling M. (eds) Computer Vision – ECCV 2016. ECCV 2016. Lecture Notes in Computer Science, vol 9906. Springer, Cham. https://doi.org/10.1007/978-3-319-46475-6_43 1, 5, 6, 8

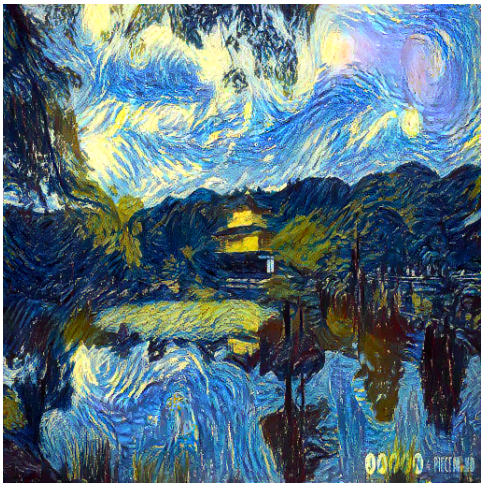
[6] Ulyanov, D., Vedaldi, A., & Lempitsky, V. (2016). Instance Normalization: The Missing Ingredient for Fast Stylization. ArXiv, abs/1607.08022. https://arxiv.org/abs/1607.08022

[7] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollar, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Computer Vision–ECCV 2014. Springer (2014) 740–75 6

(a) Content

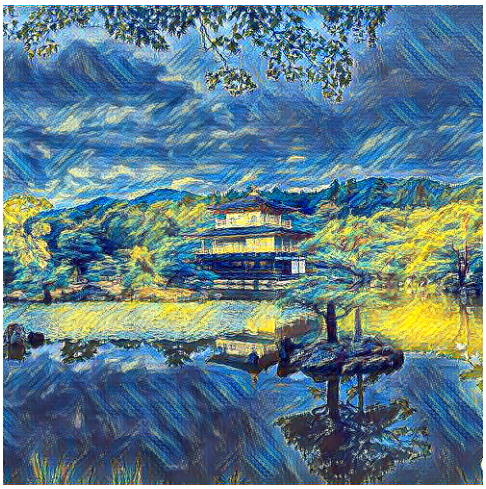
(b) Style

(c) Original Style Transfer (*Section 2.1*)



(d) Improved Style Transfer (*Section 2.2*)

(e) Color Preserving Style Transfer (*Section 2.3*)



(f) Fast Style Transfer (*Section 2.4*)

Figure 10: Neural Style Transfer Methodologies.