Neural Style Transfer in Practice
Final Project Report

Bergaoui, Khalil
khalil.bergaoui@student.ecp.fr

Hachicha, Mohamed Amine
mohamed-amine.hachicha@student.ecp.fr

1. Abstract

In this project, we consider the deep learning-based approaches to performing Neural Style Transfer (NST) on images. In particular, we intend to assess the Real-Time performance of this approach, since it has become a trending topic both in academia and in industrial applications.

For this purpose, after exploring the perceptual loss concept, which is used by the majority of models when performing NST, we conducted a review on a range of existing methods for this practical problem. We found that the feed-forward based methods allow to achieve real time performance as opposed to the framework of iterative optimization proposed in the original Neural Style Transfer algorithm introduced by Gatys et al. Which is why we mainly focused on two feed-forward methods proposed in the literature: one that focuses on Single-Style transfer, TransformNet, and one that tackles the more generic problem of Multiple Style Transfer, MSG-Net.

We tested the existing implementations using pretrained models and proposed a study case benchmark to compare, for different image resolutions, both the required run time as well as the visual rendering of each method. Additionally, we contributed to the quantitative evaluation and comparison of artistic style transfer by applying quantitative perceptual and distortion metrics inspired from the literature. Finally, in addition to the real-time performance comparison, we picked up the original NST algorithm and tried to experiment with Combined Style Transfer, which aims at applying, at the same processing step, multiple styles to a single content image.

2. Motivation

In our project proposal, initially, we proposed to conduct a comparison between two possible approaches to Neural Style Transfer, one based on convolutional networks, the other combining convolution and adversarial training. Moreover, we were planning to perform the comparison both during training and inference modes and assess the run time required to learn the task as well as asses the performance.

However, due to a lack of computational resources on the one hand, and more importantly technical issues with respect to running the available code in the initially proposed references mainly because of deprecated/incompatible versions of pytorch and missing/new dependencies between the different libraries (torch lua, torchvision, etc..) we found ourselves spending considerable amount of time debugging code in order to run training of existing models. That is why we decided to adapt our project to existing well-executable code, and focus our comparison only on inference mode. In this new perspective, we remain respectful to our initially proposed constraints in the sense that we include in our work Multi Style transfer, considering only comparisons during inference mode, analyzing the following methods that we picked : the original method proposed by Gatys et al. [9], its feed-forward extension allowing real time performance [7] and a Multi-Style Generative approach[15] that also achieves real-time performance for multiple styles.

3. Problem Definition

In general, the problem of NST can be defined as follows:

Given a set M of input images, and **Given** a style image S, **Generate**, for each input image I, an output image S(I) that adapts the style S to the image I.

In our particular case, we would like that our style image S can be arbitrarily chosen with at least one the following constraints that naturally emerge in practice :

- What are the degrees of freedom one has when performing the Transfer, in terms of transfering the style while for e.g preserving the original colors, controling the brush size, etc. ?

- How much time is required to process a single frame at a given resolution ?

Note that the problem is variational in the sense that it defines a criteria to optimize during training. Formally, given a content image x_c and a style image x_s, the objective takes the following form:

$$argmin_{\hat{y}}\ l^c_{feat}(\hat{y}, x_c) + \alpha l^s_{style}(\hat{y}, x_s)$$

where l^c_{feat} and l^s_{style} are respectively content loss and style loss terms and α is a trade-off constant. These terms are explicitly detailed in the next section (4.1).

It is worth mentioning however,that, after the optimization process, there is no evident quantitave metric to evaluate the quality of the output. We try to address this issue by studying the quality of the output in terms of the perception-distortion tradeoff, as discussed in [13]. More details will be provided in the Methodology and Evaluation sections.

4. Related Work

Before we dive into NST methods, we must present some key concepts and methods that are used to perform style transfer.

4.1. Perceptual loss

A key notion in almost all existing methods in the literature of Neural Style Transfer is the VGG-based Perceptual Loss. It was for example introduced in [7] and in [3]. This loss aims, given an input content image and a style image, to produce an output image that contains the high-level features of the style image while preserving the low-level content of the content image.

To extract the high and low level features of an image, the majority of related works exploited the VGG network, which is a CNN submitted by Karen Simonyan and Andrew Zisserman in Large Scale Visual Recognition Challenge in 2014 to perform image classification on the ImageNet dataset.

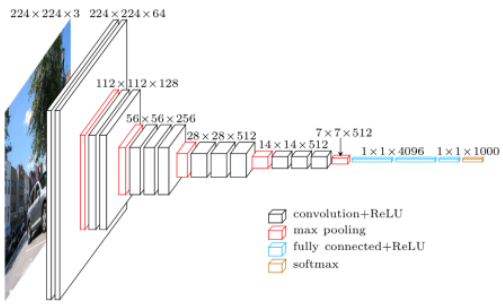

Figure 1: VGG Network Architecture

It is well know, like it is shown for example in [1], that VGG can be used for feature extraction in images: the first convolutional layers encode the low-level features of the image while the deeper layers encode the high-level features. In the NST context, VGG is referred to as the loss network.

The perceptual loss is finally a wrighted sum of a feature reconstruction loss and a style construction loss like it was explained in [7].

If we denote ϕ ou loss network (VGG for instance), ϕ_j the j-th activation of this network, y our content image and $\hat{y}$ the output image, then the feature reconstruction loss is :

$$l^j_{feat}(y, \hat{y}) = \frac{1}{C_j.W_j.H_j} \|\phi_j(y) - \phi_j(\hat{y})\|^2_2$$

where j, the layer index, is typically small in the case of VGG since the low-level features of the content image are encoded by the first layers. (C_j, H_j and W_j are the tensor dimensions).

The style reconstruction loss was introduced by Gatys et al. in [3] and [4]. It's given by the formula (where y is now the style image):

$$l^j_{style}(y, \hat{y}) = \left\| G^\phi_j(y) - G^\phi_j(\hat{y}) \right\|^2_F$$

where G^ϕ_j is the **Gram Matrix** at the j-th layer, defined by :

$$G^\phi_j(x)_{c,c'} = \frac{1}{C_j.W_j.H_j} \sum_{h=1}^{H_j} \sum_{w=1}^{W_j} \phi_j(x)_{c,h,w} \phi_j(x)_{c',h,w}$$

In the case of VGG, the index j of style layers is typically high since deep layers are the one that encode high-level features.

4.2. Original NST algorithm

One of the first algorithms in Neural style transfer was the one introduced by Gatys et al. in [3]. The idea behind the algorithm is quite simple. We don't need a dataset of images for training; we only need our content and style images and the pretrained VGG model.

In the original work, the authors suggested to start with a randomly generated image $\vec{x}$ and in each iteration of the algorithm we forward it through the VGG network and calculate the perceptual loss using the activations at different levels of the network.

The perceptual loss is a weighted sum of the feature and style reconstruction losses defined in 4.1. Then, we perform backpropagation to update our image $\vec{x}$. We notice here that the optimization is done directly on the image and not on some network weights.

Many implementations suggest to replace the randomly generated image at step 0 by the content image, which helps reduce optimization time especially if we put more weight on the feature reconstruction loss.

4.3. Extensions and Improvements

For controlling perceptual factors in performing Neural Style Transfer, Gatys et al. themselves [3] proposed several slight modifications to improve their previous algorithm. We ill focus on two particular factors : colour control and brush/Stroke size control.

For the colour control, the original NST algorithm produces stylisedimages with the colour distribution of the style image. However, sometimes people prefer a colour-preserving style transfer, i.e., preserving the colour of the content image during style transfer. In [8], the authors propose two methods to achieve color preservation: Color histogram matching and Luminance-only transfer.The latter is motivated by the observation that visual perception is far more sensitive to changes in luminance than in color.In our experiments, mainly section 6, we adopt this method which principle is very simple : Before style transfer, perform a luminance-transfer on the style image's luminance channel L_s, using the YIQ color space such that :

$$L'_s = \frac{\sigma_c}{\sigma_s}(L_S - \mu_s) + \mu_c [8]$$

where L_c is the luminance channel of the content image, μ_c and σ_s the respective mean and standard deviation.

For stroke/brush size control, we mention a few interesting points. One possible strategy to contol brush size is to resize the input image to different scales before the forward pass, which would hurt stylisation quality. Another alternative is to train multiple models with different scales of a style image, which is space and time consuming.

For a thourough and extensive discussion we refer the reader ro section 5 of [14] .

When it comes to arbitrary style transfer, there exists methods using iterative optimization-based approaches. This framework, however, suffers from expensive runtime during inference and this particular problem was addressed in [11] by considering a local optimization objective, simpler than the perceptual loss of VGG. On the other hand, fast approximations with feed-forward neural networks have been proposed to speed up neural style transfer. Unfortunately, the speed improvement comes at a cost: the network is usually tied to a fixed set of styles and cannot adapt to arbitrary new styles. In this paper [6], the authors present a simple yet effective approach that enables arbitrary style transfer in real-time, which they refer to as Adaptive Instance Normalization. In [15], only Instance Normalization is being used, which entails that the model is bound to learn a fixed number of styles. Nevertheless, the authors show that this number can be as high as 1,000 with higher performance[15] in terms of inference speed in compared to the work of [6].

5. Methodology

Now that we have reviewed the related literature to Neural Style Transfer, we will first introduce the basic principles of the methods we are trying to compare. We will use pre-trained models to evaluate and compare the processing time during inference as well as the quality of the style transfer.

5.1. TransformNet

This model was introduced by Johnson et al. in [7] in which NST is performed along with Super-Resolution. The model is inspired from the method employed by Gatys et al. and it envolves training a network called TransformNet on Coco dataset (83k images). The idea is to forward each image y in TransformNet giving an output image $\hat{y}$ that we pass into VGG network along with the content image y and a style image S. Then, the perceptual loss can be calculated using the feature extraction in VGG and we backpropagate through TransformNet to update its weights. The global architecture is illustrated below.

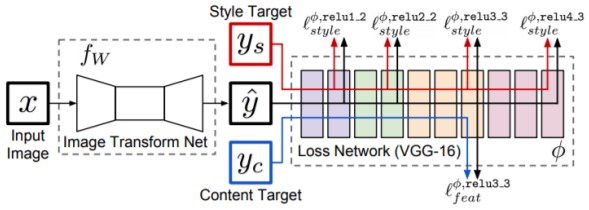

Figure 2: TransformNet Overview [7]

TransformNet's architecture envolves 3 first convolution layers, 5 residual blocks and then 3 upsampling convolution layers. Instead of using batch normalization, where the mean and variance are computed across the batch elements, the network uses Instance Normalization which was introduced in [12] for it's efficiency in NST. Instance Normalization computes the mean and variance for each image and for each channel.

5.2. MSG-Net

The particularity of this method is that it allows Multiple-Style transfer during training.

Similarly to the previous method (TransformNet), given a content image x_c and a style image x_s, the authors get inspiration from the below minimization problem (where we use the same notations as in section 4.1) which is solvable by using an iterative approach but infeasible to achieve it in real-time.

$$argmin_{\hat{y}}\ l^c_{feat}(\hat{y}, x_c) + \alpha l^s_{style}(\hat{y}, x_s)$$

The authors propose to approximate the solution and put the computational burden to the training stage by introduc-

ing, through the CoMatch Layer, an approximation which tunes the feature map based on the target style:

$$\Phi^{-1}\left[\Phi\left(\phi_j(x_c)\right)^T W G_j^{\phi}(\phi_j(x_s)\right]^T$$

where$W$ is a learnable weight matrix and$\Phi()$ is a reshaping operation to match the dimension, G_j^{ϕ} is the Gram Matrix of the j^{th} scale of the loss network ϕ (using the same notations as in section 4.1).

This way multiple-style learning is enabled and the overall network uses in total 2 CoMatch layers, one for the siamese network and one for the transformation network :

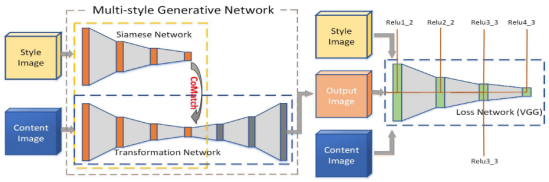

Figure 3: MSG-Net architecture [15]

As part of the Generator Network, we adopt a Siamesenetwork sharing weights with the encoder part of transformation network, which captures the feature statistics of thestyle imagexsat different scales, and outputs the Gram Matrices$\{G_j(\phi(xs)), j = 1, ...K\}$ where K is the total number of scales (a scale corresponds to a specific layer in the loss network). Then a transformation network takes the content image and matches the feature statistics of the style image at multiple scales with CoMatch Layers.

Note the training objective is the above loss averaged accross the entire content and style images of the training set, with a Total Variation regularization term:

$$argmin_{\hat{y}}\ \mathbb{E}_{x_c,x_s}[\lambda_c * l^c_{feat}(\hat{y}, x_c) + \lambda_{TV} * l_{TV}\hat{y} +$$

$$\lambda_s * \sum_{k=1}^{K} l^k_{style}(\hat{y}, x_s)]$$

5.3. Original NST Extension: Combined Style Transfer

During our project, we have observed that the main conceptual difference between the previously described methods in 5.1 and 5.2 is that, after training, TransformNet[7] is given x_c as input, while MSG-Net[15] is given a pair (x_c,x_s) as input, where x_c represents the content image and x_s the target style. Such seemingly small difference, induces changes namely in the design of the feed-forward architecture, but also allows for more flexibility during inference in order to experiment with different runtime manipulations (color preservation, brush size control, etc..). In this perspective, it is somewhat natural to consider the case where the input is extended to (x_c,x_{s1},x_{s2}) where x_{s1} and x_{s2} represent two different style images. This point of view could allow us to investigate questions of this sort: What if Van Gogh and Monet for example decided to make a painting together?

Obviously to consider such point of view in the framework of real-time performance requires a high level of ingenuity regarding the architectural design of the feed-forward network, which is out of the scope of our project. Which is why, we tried to address this question in the final section of our work by integrating some slight modifications to the perceptual loss of the original NST algorithm proposed by Gatys et al. So instead of having only one style loss factor, we would calculate a style loss factor for each style image that we have, while varying the corresponding coefficients.

5.4. Quantitative metrics

In our work, we try to compare methods on a Style Transfer task which is more of a qualitative artistic task. Besides the real-time performance during inference, we do not dispose of an evident metric in order to assess the outputs of both methods and quantify the differences. In fact, since the considered task relies on visual rendering of the style transfer, an important assessment criteria is the perceptual quality of the output image. However, perceptual quality is not directly linked to a measured quantity, for instance, in [13], perceptual quality of an image is defined as the degree to which it looks like a natural image, and has nothing to do with its similarity to any reference image. In many image restoration/generation domains, perceptual quality is commonly evaluated empirically by the mean opinion score of human subjects, and recently, with the usage of GANs, it has become increasingly popular to perform such studies through real vs. fake questionnaires especially when the task is rather qualitative, as in our case, or also in the case of image colorization [16].

Thus, a desirable metric is one that correlates well with the human perceptual evaluation. Moreover, we look for a metric which can be computed using a single reference image, as it is the case for metrics that evaluate distortion such as Root Mean Squared Error (RMSE). Combining both constraints, we found a relevant metric in [10] called Single Image Fréchet-Inception Distance (SIFID), which uses the internal distribution of deep features at the output of the convolutional layer just before the second pooling layer (one vector per location in the map) in the pretrained Inception Network[2].

Given two images, typically the content(1) and the output(2) after style transfer, we can measure the SIFID value as follows:

$$SIFID = |\mu_1 - \mu_2|^2 + tr(\Sigma_1 + \Sigma_2 - 2 * (\Sigma_1.\Sigma_2)^{\frac{1}{2}}$$

where μ_1, Σ_1 (resp μ_2, Σ_2, are the mean and covariance matrix of the Inception features of image 1 (resp image 2).

In addition to the chosen perceptual indicator SIFID, we measure the induced distortion by the style transfer between the content X_1 and the output X_2 using the RMSE:

$$RMSE = \sqrt{\sum_{i=1}^{N} \frac{(X_1^i - X_2^i)^2}{N}}$$

where N is the number of pixels of images X_1 and X_2.

Note that the distortion measure RMSE requires the same shape of the images which is not necessarily the case in the perceptual measure SIFID.

Finally, using these two metrics, the best method in our setting is the one that achieves less distortion (lower RMSE) and better perceptual quality (lower SIFID). Our comparison takes into consideration the perception-distortion tradeoff as it is mathematically proven in [13] that distortion and perceptual quality are at odds with each other.

6. Evaluation

6.1. Real-Time Performance

In our evaluation, we begin by comparing the real-time performance of the two methods. We vary the resolution of the target image (same resolution as the content image used as input) and we report the time, in seconds, that is needed to process a single frame. As our study case targets the common user, we report the results obtained on mainstream Google Colab GPU. We find that TransformeNet is slightly slower than MSG-Net across all ranges of the tested resolutions. This highlights the effectiveness of the implemented feed-forward CoMatch layers in MSG-Net. Furthermore, we observe that for both methods, the processing time increases with the target resolution. However, the rate at which it increases is different depending on the method. For instance, when we multiply by a factor of x4 the target resolution, the single frame processing time is multiplied respectively by ≈ 2 for MSG-Net and ≈ 4 for TransformNet. Therefore, in addition to more flexibility in the style choice, MSG-Net performs better as can be seen in the table below:

Content Image	MSG-Net[15]	TransformNet[7]
512x320	0.034s	0.051s
724x452	0.049s	0.10s
1024x640	0.076s	0.19s

Thus, when dealing with higher resolution images or even videos, one would prefer to deploy MSG-Net (At 1024x640, MSG-Net can process ≈ 12 frames per second vs ≈ 5 frames per second for TransformerNet).

For the interested user, we also report the obtained values when running the experiments on CPU. This could be helpful for those working on local machines and having no access to GPU to speed up computation:

Content Image	MSG-Net[15]	TransformNet[7]
512x320	0.6s	0.1s
724x452	2.7s	1.8s
1024x640	4.6s	4.1s

6.2. Perceptual evaluation

We begin our evaluation by comparing the visual rendering of each of the two methods on the same example. In this setting, we consider "venice-boat" as the content image and "candy" as the style image.

It can be seen that from the figure below, the outputs of the two methods are visually different with the presence of small artifacts in the output of TransformNet. Moreover, we observe that the rendered colors are not the same: MSG-Net seems, on this example, to favor darker shades of colours compared to the more lively colours rendered by TransformNet.

Figure 4: Qualitative comparison between the outputs of MSG-Net and TransformNet on the same style transfer task.

For a quantitative comparison, we measure the SIFID (introduced in section 5) as a distance indicator between (output,style) on the one hand and (output,content) on the other hand. As can be see from this below table, the style transfer is successful in the sense that the output image is perceptually closer to the style image than to the content image, since the corresponding SIFID distance is smaller. This is true for both methods. For both low and high resolutions, MSG-Net gives a smaller SIFID with respect to the style image which means that MSG-Net is more efficient in style transfer.

Content Image	MSG-Net[15]	TransformNet[7]
512x320	0.85 (3.74)	1.17 (3.32)
1024x640	0.71 (3.40)	1.04 (4.29)

For a more extensive comparison, we try to evaluate the distortion caused by the style transfer with respect to the original content image used as input (venice-boat in our example). For this, we compute the RMSE and SIFID metrics between the content image and the output of the style transfer. This allows us to directly compare the two methods by plotting the images shown in Figure 4 as points in the perception-distortion plane, as shown in the figure below:

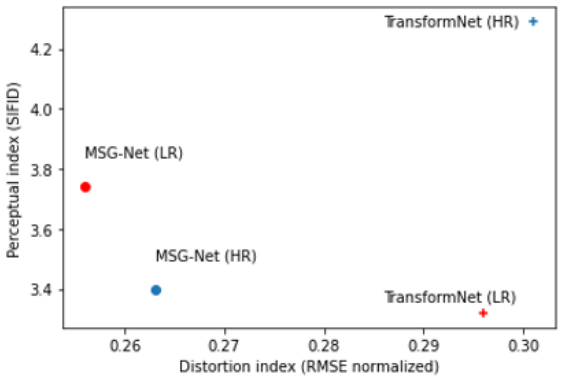

Figure 5: MSG-Net[15] vs TransformNet[7] in the Percepetion-Distortion plane

Using these metrics, we can say, independently of our preference towards lively or dark colours, that MSG-Net renders a better style transfer while at the same time reducing distortion and staying closer to the original content. Only for the considered Low-Resolution case, TransformNet yields a better perceptual index.

6.3. Runtime Manipulations

In this section, we address the common degrees of freedom that the user has, in addition to choosing the target style and resolution. Such flexibility is only allowed when the style image represents an input during inference for the considered method. This exactly shows the limitation of TransformNet when it comes to runtime manipulations and in this section we will evaluate only MSG-Net in our experiments.

For each, degree of freedom that we will consider, we will include both qualitative results to visually observe the differences and quantitative results to quantify the task and highlight the perception-distortion trade-off in a less subjective manner.

6.3.1 Color preservation

As described in the Methodology section, color preserving style transfer aims at producing an output image that reproduces the style of the chosen target but keep the colors of the input content image. In practice, this could be a highly desirable feature, depending on the application case, which why we attempt to include it in our comparison. In all of our experiments, Color preservation is performed using the *Luminance matching* approach proposed in [8]. Note that it is unclear that there is any practical advantage in typical GPU implementations[8].

First, we show our results obtained with MSG-Net. We keep the style image fixed and test two different target resolutions. As we can see in the below figure, with higher resolution, more content details are preserved for both cases as expected. Moreover, with color preservation, the result is found to be more perceptually pleasing in the sense that it is more gradual and less striking:

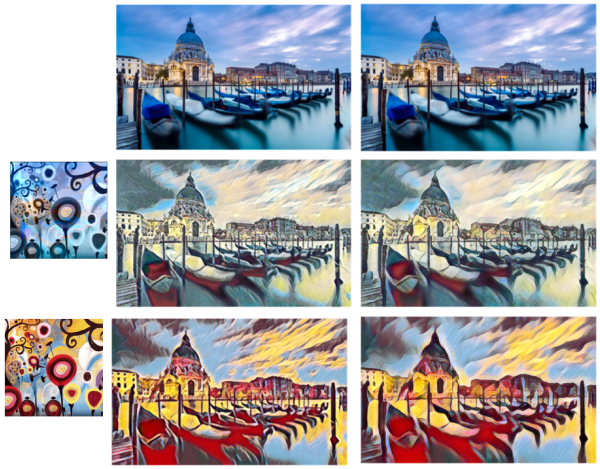

Figure 6: Comparing Color Preserving (second row) and regular(third row) Style Transfer using MSG-Net: We display results obtained for 1024x640 resolution in the left column and 520x320 resolution in the right column.(the left most column displays the resulting style image with and without color preservation)

Quantitatively, we represent the above 4 output images (second and thrid rows) in the perception-distortion plane. In particular, for every output image, we measure a distortion index, given by the Root Mean Squared Error (RMSE) and a perceptual index given by the Single Image Fréchet-Inception Distance (SIFID).

Both indexes are computed w.r.t the input content image. We find that color preservation gives better distortion and perceptual results (the lower the indexes the better) which matches our previous qualitative evaluation.

Moreover, both with and without color preservation, we observe that increasing the resolution yields a lower perceptual index, reflecting the better overall visual rendering. However, we observe that strangely the distortion index increases. This rather counter-intuitive result (since we expect to have a better image using HR, so less distortion) is

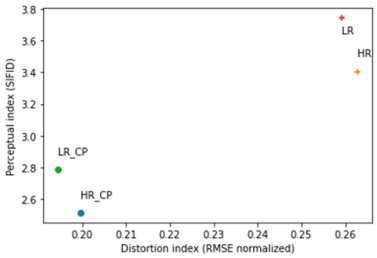

Figure 7: Quantitative comparison between the 4 images displayed in Figure2. Each point in the plane represents one image and the legend is as follows : LR := Low-Resolution, LR_CP:= Low-Resolution with Color Preservation (similarly HR stands for High-resolution)

actually in line with the perception-distortion trade-off discussed in [13].

6.3.2 Brush size control

In this section, we try to evaluate the brush size control feature. In practice, this is achieved by changing the resolution of the style image. Note that we verified that, this change of resolution has, on average (with milliseconds precision) , no impact on the processing time of a single frame (however it is obvious that more memory will be used). In our experiments, we resample the content image to a fixed resolution of 512x320.

In this setting MSG-Net was pretrained on different sizes of the style image[15] for this purpose. We visualize the output of the model for two different styles using two resolutions for the style image. Qualitatively, the differences are not easy to spot visually but we can see from the figure below that overall, using a smaller resolution helps preserve finer content details as it is, to a certain extent, equivalent to a smaller brush stroke used while drawing/painting the new target image (Figure8)

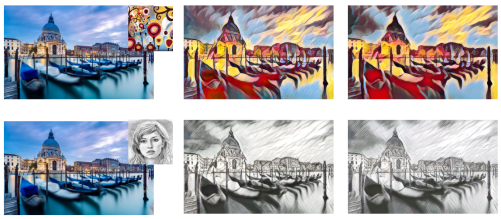

Figure 8: Comparing Brush size control: results are obtained for two different style targets (candy for top row and pencil for bottom row). Left column shows the original content image with corresponding target styles, middle column shows results obtained with a "small brush" and right column shows results obtained with a "large brush"

Quantitatively, we study the effect of the brush size in the perception-distortion plane. We find that higher brush sizes yield a better perceptual index but a worse distortion result. The corresponding "trajectory" seems to converge to a given point which can be interpreted as the following : we approach the limit as the brush size increases and increasing the size of the style image beyond the size of the content image has no impact. Again our quantitative results highlight the aforementioned perception-distortion trade-off.

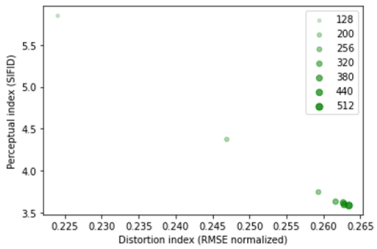

Figure 9: Brush size in the perception-distortion plane: Every point corresponds to an output image obtained using a different brush size. In our plot, a brush size 128 means that the style image is a resampled 128x128 image. The content image size is fixed to 512x320.

Another worth mentioning observation is that, when the style image resolution becomes too small, MSG-Net fails to faithfully transfer to the target style. In fact, we manage to confuse the model, by decreasing the style resolution, and observe that it performs style transfer but to another different style (one of the 21 the styles that were used during training[5]).

Figure 10: Style confusion using too small brush size: When the style image is down-sampled to a point where style details are no longer clear, MSG-Net gets confused and performs style transfer to the wrong target style.Top row (target:= candy, output:=mosaic). Bottom row(target:=pencil, output:=mosaic_duck). MSG-Net was trained on 21 styles including candy, mosaic, pencil, mosaic_duck , etc [5]

6.4. Combined Style Transfer

We performed the combined style transfer as discussed in the methodology with The Scream and Starry Night paintings as style images (Figure 11).

(a) Starry Night (b) The Scream

Figure 11: Used style images

For the NST algorithm, we have defined the following loss function:

$$l(\hat{y}, x_c, x_{s1}, x_{s2}) = l^c_{feat}(\hat{y}, x_c) + \lambda_1 l^s_{style}(\hat{y}, x_{s1}) + \lambda_2 l^s_{style}(\hat{y}, x_{s2})$$

Where λ_1 and λ_2 are the loss weights associated with The Starry Night and The Scream painiting respectively. We used different values for λ_1 and λ_2. We can see the final results, after 100 epochs in Figure 12 below.

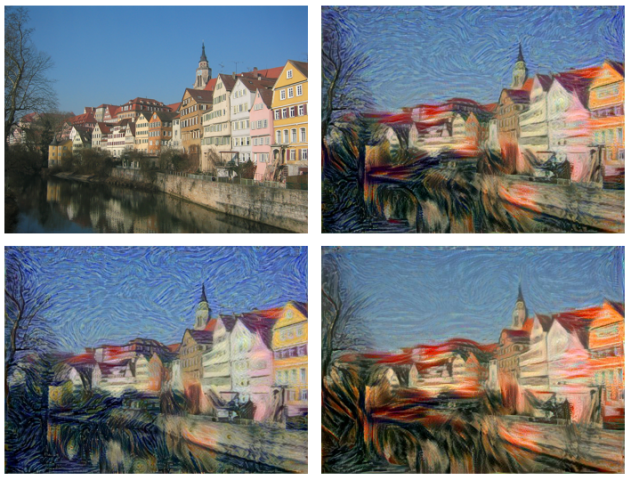

Figure 12: Top Left: The original content image, Top Right: CST with $\lambda_1 = \lambda_2$, Bottom Left: CST with $\lambda_1 = 5\lambda_2$, Bottom Right: CST with $\lambda_2 = 5\lambda_1$

For each combination of λ_1 and λ_2 we can see that the final outcome has low-level features that correspond to the content image and high-level features that correspond to a weighted combination of the style images. For the case in which $\lambda_1 = 5\lambda_2$ for example, it is clear that The Starry Night's style is predominant while The Scream painting's style adds a subtle modification to the output image style. The inverse happens when $\lambda_2 = 5\lambda_1$.

To verify this, and for each configuration, we have calculated the SIFID index between the output image and each of the style images, and also the content image. The results are summarized in the table below.

-	$\lambda_1 = \lambda_2$	$\lambda_1 = 5\lambda_2$	$\lambda_2 = 5\lambda_1$
Starry Night	1.51	0.96	2.49
The Scream	1.25	1.88	0.85
Content Image	1.06	1.15	1.21

We notice that, in fact, with $\lambda_1 = 5\lambda_2$ and $\lambda_2 = 5\lambda_1$ the SIFID is the lowest with the respective style image that has the highest style loss weight, which is what we expect.

For $\lambda_1 = \lambda_2$, we see that the SIFID is not small with any of the two style images. This is due to the fact that the output style in a blended mixture of the two paintings styles.

7. Discussion

In this work, we have tried to experiment with three methods in the existing rich literature of Neural Artistic Style Transfer: NST[9],TransformNet[7] and MSG-Net[15] . We found, that Multiple-Style transfer can be achieved in real-time without loss of runtime performance and with more flexibility in runtime manipulations. We have also tried to quantify the artistic style transfer in order to extend the qualitative assessment that heavily relies on the visual rendering of the used methods. This was possible by comparing the methods in the perception-distortion plane, which is a relevant criteria according to the literature [13]. However, in our project, we only considered a single perceptual index (SIFID) and a single distortion metric(RMSE) and as a further improvement one could consider other metrics as perceptual/distortion indexes and compare the obtained results, since a change in the metric is highly susceptible of inducing a change in interpretation. Moreover, we considered RMSE as a distortion index as it is widely used in image restoration applications, however it is worth mentioning that this metric allows computation between two images (arrays) of the exact same size. Such constraint could be undesirable in practice. In addition to testing different metrics for the perception-distortion study, it is highly desirable to experiment with different content images since some particularities of the considered method can only appear for specific content patterns. Nonetheless, we could not manage to achieve this in our present work, mainly for time constraints.

When it comes to the extension of style transfer to more than one style target at the same processing step, we have

managed to start digging in this direction but clearly more research work must be conducted especially when the goal is real-time performance through a compatible feed-forward architecture.

References

[1] Gayani Chandrarathne, Kokul Thanikasalam, and Amalka Pinidiyaarachchi. A comprehensive study on deep image classification with small datasets. 04 2019. 2

[2] Christian Szegedy et al. Going deeper with convolutions. In *arXiv:1409.4842 [cs.CV]*, 2014. 4

[3] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A neural algorithm of artistic style, 2015. 2, 3

[4] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Texture synthesis using convolutional neural networks, 2015. 2

[5] https://github.com/zhanghang1989/PyTorch Multi-Style-Transfer/tree/master/experiments/images/21styles. 21 styles of pretrained msg-net. 7

[6] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 3

[7] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution, 2016. 1, 2, 3, 4, 5, 6, 8

[8] Aaron Hertzmann Eli Shechtman Leon A. Gatys, Matthias Bethge. Preserving color in neural artistic style transfer. *arXiv:1606.05897 [cs.CV]*, 2016. 3, 6

[9] Matthias Bethge Leon A. Gatys, Alexander S. Ecker. A neural algorithm of artistic style. 2015. 1, 8

[10] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. Singan: Learning a generative model from a single natural image. In *Computer Vision (ICCV), IEEE International Conference on*, 2019. 4

[11] Mark Schmidt Tian Qi Chen. Fast patch-based style transfer of arbitrary style. 2016. 3

[12] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization, 2017. 3

[13] Tomer Michaeli Yochai Blau. The perception-distortion tradeoff. *arXiv:1711.06077 [cs.CV]*, 2020. 2, 4, 5, 7, 8

[14] Zunlei Feng Jingwen Ye Yizhou Yu Mingli Song Yongcheng Jing, Yezhou Yang. Neural style transfer: A review. *CoRR*, abs/1610.07629, 2019. 3

[15] Hang Zhang and Kristin Dana. Multi-style generative network for real-time transfer. *arXiv preprint arXiv:1703.06953*, 2017. 1, 3, 4, 5, 6, 7, 8

[16] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 4