Journal Homepage: -www.journalijar.com

INTERNATIONAL JOURNAL OF ADVANCED RESEARCH (IJAR)

Article DOI:10.21474/IJAR01/24479
DOI URL: http://dx.doi.org/10.21474/IJAR01/24479

ISSN (O): 2320-5407
ISSN (P): 3107-4928

*RESEARCH ARTICLE*

# IMAGE STYLE TRANSFER BASED ON A PRE-TRAINED VGG16 NETWORK: DETAILED FORMULATION, JUSTIFICATION FOR THE CHOICE OF NETWORK AND QUANTITATIVE EVALUATION

**Soro Mamadou, Mouale N'dah M. Bienvenue and Edi Yapi Fiacre Aristide**

1. UFR SFA, Department of Mathematics and Computer Science, Nangui Abrogoua University, Abidjan, Côte d'Ivoire.

## *Manuscript Info*

***Manuscript History***
Received: 15 Aug 2026
Final Acceptance:5 September 2026
Published: October 2026

***Key words:-***
neural style transfer; VGG16; pre-trained convolutional neural network; Gram matrix; perceptual loss; image quality assessment.

## *Abstract*

Neural style transfer involves representing the content of one image in the visual style of another. This paper presents a comprehensive and reproducible formulation of style transfer via optimisation, based on a pre-trained VGG16 network, in line with the work of Gatys et al. The content is represented by the activations of the relu4_2 layer and the style by the normalised Gram matrices of five layers (relu1_1 to relu5_1). The generated image is initialised using the content image and optimised using the Adam optimiser. We justify the choice of VGG16 based on its architecture and a controlled comparison with VGG19, ResNet-50 and the fast AdaIN method across 16 content–style pairs. Evaluation uses SSIM, PSNR, style distances (Gram matrices and colours) and computation time. Across the 16 pairs, the method preserves the content structure (average SSIM of 0.319) whilst transferring the style, and converges within a few hundred iterations. VGG16 matches VGG19 in terms of quality at a lower cost and significantly outperforms ResNet-50; it preserves the structure better than AdaIN without any training, albeit at the cost of increased computation time. All parameters are specified and the code is provided.

*"© 2026 by the Author(s). Published by IJAR under CC BY 4.0. Unrestricted use allowed with credit to the author."*

## Introduction:-

Image style transfer involves generating an image that preserves the content of a source image (shapes, objects, spatial arrangement) whilst adopting the style of a reference image (colour palette, textures, brushstrokes). Gatys et al. [2, 3] demonstrated that convolutional neural networks trained for image classification separate these two types of information to a certain extent. Deep activations encode the content, whilst the correlations between channels—summarised by Gram matrices—encode the style. The stylised image is then obtained by directly optimising the pixels of an image so that it simultaneously reproduces these two representations.This optimisation-based approach was followed by forward-propagation methods, which train a generative network for a given style (Johnson et al. [4], Ulyanov et al. [5]) or for arbitrary styles (AdaIN [6], WCT [7]). These methods are much faster, but require prior training on large image corpora. A detailed review is provided by Jing et al. [8].

**Corresponding Author:-**Soro Mamadou
**Address:-**UFR SFA, Department of Mathematics and Computer Science, Nangui Abrogoua University, Abidjan, Côte d'Ivoire.

The VGG16 network [1], trained on ImageNet [14, 15], achieves a top-5 accuracy of 92.7 per cent and is one of the most widely used feature extractors. In this work, we use it as a fixed feature extractor for style transfer. Our aim is not to propose a new loss function, but to provide a comprehensive, justified and evaluated formulation of this approach.

**Our contributions are as follows:-**

(C1) a comprehensive and reproducible formulation of the VGG16-based style transfer chain: choice of layers, normalisation of Gram matrices, weights, analytical gradients, initialisation using the content image, optimisation via Adam and a stopping criterion (§4);
(C2) an architectural and experimental justification for the choice of VGG16, through a controlled comparison with VGG19, ResNet-50 and the fast AdaIN method (§3.2 and §6.3);
(C3) a quantitative evaluation across 16 content/style pairs, combining measures of content preservation (SSIM, PSNR), style fidelity (Gram and colour distances) and computational cost (§6.2);
(C4) an analysis of convergence that links the evolution of content and style losses to the visual quality of the generated image, as well as a sensitivity analysis regarding weights and initialisation (§6.4 and §6.5).

Section 2 presents related work. Section 3 describes VGG16 and justifies its selection. Section 4 details the method, Section 5 the experimental protocol, Section 6 the results, and Section 7 concludes.

**Related work:-**

**Style transfer via optimisation.** Gatys et al. [2, 3] define a content loss on the activations of a deep layer of VGG19 and a style loss on the Gram matrices of several layers. They optimise the image, initialised with white noise, using the L-BFGS algorithm. Li et al. [20] show that aligning Gram matrices is equivalent to minimising a maximum mean deviation (MMD) between feature distributions, which provides a statistical justification for this representation of style.

**Forward-propagation methods.** Johnson et al. [4] and Ulyanov et al. [5] train a style-generative network using perceptual losses computed with VGG. Huang and Belongie [6] propose AdaIN, which aligns the mean and standard deviation of content features with those of the style; a trained decoder then reconstructs the image, enabling arbitrary style changes in real time. Li et al. [7] generalise this idea using whitening and colouring transformations (WCT).

**Choice of extractor network.** Virtually all studies use VGG16 or VGG19. Wang et al. [10] analyse why residual networks such as ResNet [9] yield poorer results. Their activations are highly concentrated around a few extreme values, which renders Gram statistics uninformative; a smoothing step is therefore necessary.

**Evaluation.** There is no single metric for assessing the quality of a style transfer [8]. Measures of structural preservation (SSIM [11], PSNR), perceptual distances such as LPIPS [12], style distances based on feature statistics, and computation time are generally combined. We adopt this multi-criteria approach.

**The VGG16 network and rationale for its selection:-**

**Architecture:-**

VGG16 [1] comprises thirteen convolutional layers and three fully connected layers. The convolutional layers all use 3 × 3 filters, with a stride of 1 and padding of 1, which preserves the resolution. They are organised into five blocks of 2, 2, 3, 3 and 3 layers, with 64, 128, 256, 512 and 512 channels respectively. Each block ends with a 2 × 2 max-pooling operation with a stride of 2, which halves the resolution. The three fully connected layers (4,096, 4,096 and 1,000 neurons) perform ImageNet classification. We denote the output, following ReLU activation, of the Cth convolutional layer of block B as reluB_C. For style transfer, only the convolutional part is used and the fully connected layers are ignored.

**Why VGG16?:-**

The choice of extractor network determines the quality of the content and style representations. Table 1 compares VGG16 with the architectures under consideration. For each network, it gives the number of convolutional parameters actually utilised up to the deepest layer used, and the time taken for one iteration (forward and backward propagation, 256 × 256 image, 2-core processor) as measured in our experiments.

**Table 1. Characteristics of the compared extractor networks**

| Network | Conv. layers | Parameters (total) | Parameters used | Time per iteration | Structure |
|---|---|---|---|---|---|
| VGG16 [1] | 13 | 138 M | 10.0 M | 0.39 s | sequential, without normalisation |
| VGG19 [1] | 16 | 144 M | 12.9 M | 0.49 s | sequential, without normalisation |
| ResNet-50 [9] | 49 (+ 4 for projection) | 25.6 M | 8.6 M | 0.16 s | residual, batch normalisation |

Parameters used: up to relu5_1 for VGG16 and VGG19, up to the output of the res4 layer for ResNet-50. Times measured under the conditions described in §5.2.

**Three arguments underpin the choice of VGG16:-**

- **Feature statistics suited to style transfer.** VGG16 is purely sequential, with no residual connections or batch normalisation. Its activations retain a regular distribution, which the Gram matrix accurately summarises. Conversely, the activations of residual networks are dominated by a few extreme values, which degrades style transfer [10].
- **A hierarchy equivalent to that of VGG19 at a lower computational cost.** VGG16 and VGG19 share the same five blocks, with the same resolutions and the same number of channels; VGG19 adds only one layer to each of blocks 3 to 5. The layers used for style (the first layer of each block) and for content (relu4_2) are present in both networks. VGG16 therefore offers the same multi-scale description with less computation.
- **Availability and reproducibility.** VGG16 is available pre-trained in all common libraries and serves as a benchmark in the style transfer literature [4, 8, 12], which facilitates comparisons.

These claims are verified experimentally in Section 6.3, where the same algorithm is applied using VGG19 and ResNet-50.

**Reuse of pre-trained features:-**

The convolutional layers of VGG16, trained on ImageNet, can be reused for other tasks. In a preliminary study, we therefore adapted VGG16 through fine-tuning for CIFAR-10 classification. The convolutional layers were frozen and the 1,000-class classifier replaced by a layer of 10 neurons; the accuracy achieved, limited to 61 per cent, is due to the very low resolution of the images (32 × 32). In this work, we employ another form of re-use: the network remains **entirely fixed** and is used solely to measure content and style. No weights are modified; only the generated image is optimised.

**Proposed method:-**

**Overview:-**

Let c be the content image, s the style image and x the generated image, all of size 3 × H × W and with values in [0, 1]. The method seeks the image x that minimises a combination of a content loss—which measures the deviation from c—and a style loss—which measures the deviation from s.

**The processing chain (Fig. 1) comprises six steps:-**

**Étape 1.** pre-processing of images c and s, and initialisation of $x_0 = c$;

**Étape 2.** extraction, using a fixed VGG16 model, of the feature maps of c on the content layer and of s on the style layers, calculated once only;

**Étape 3.** calculation of the reference Gram matrices $A^l$ for the style image;

**Étape 4.** at each iteration, extraction of features from x and calculation of content and style losses;

**Étape 5.** calculation of the gradient of the total loss with respect to the pixels of x, via backpropagation through VGG16;

**Étape 6.** updating x using the Adam optimiser and projecting onto [0, 1], until the stopping criterion is met.

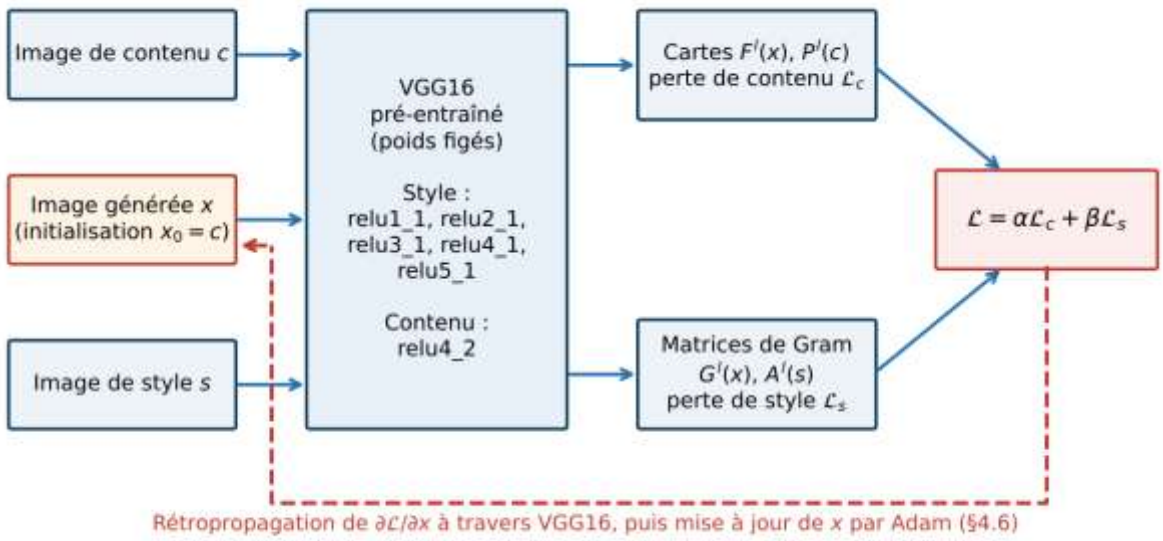



**Fig. 1. VGG16-based style transfer processing pipeline. The network weights are fixed; only the generated image x is optimised.**

**Feature extraction:-**

The VGG16 weights are the original ones [1], trained on ImageNet. The images are converted to the format used by these weights: channels in BGR order, values between 0 and 255, with the ImageNet mean subtracted (103.939; 116.779; 123.68). For a layer l comprising $C_l$channels of size $H_l \times W_l$, we denote $M_l = H_l W_{(l)\ as}$the number of spatial positions. We also denote $F^l(x) \in \mathbb{R}^{C_l \times M_l}$as the activation matrix: each row i is the flattened map of filter i, and the element $F^l_{ik}$is the activation of filter i at position k.

**Content representation and content loss:-**

Content is represented by the activations of a deep layer, which encode objects and their layout without pixel-level detail. We use the relu4_2 layer (C = 512 channels, M = 32 × 32 positions for a 256 × 256 image). Let $P^l = F^l(c)$; the content loss is the mean squared error:

$$L_c(x, c) = \frac{1}{C_l M_l} \sum_{i,k} (F^l_{ik}(x) - P^l_{ik})^2 \quad (1)$$

**Style representation: Gram matrix:-**

Style is described by the correlations between the responses of the filters within the same layer. For layer l, the Gram matrix $G^l \in \mathbb{R}^{C_l \times C_l}$ is defined by:

$$G^l_{ij}(x) = \frac{1}{C_l M_l} \sum_k F^l_{ik}(x)\, F^l_{jk}(x) \quad (2)$$

**In matrix form, $G^l = F^l (F^l)^T / (C_l\ M_l)$. This matrix has four useful properties:-**

- **Statistical interpretation.** $G^l_{ij}$ is the mean, over all positions, of the product of the activations of filters i and j. This is an uncentreed second-order moment: it is high when both filters are active at the same locations, for example a given colour associated with a given stroke orientation.
- **Spatial invariance.** The sum is taken over all positions k. Any permutation of the positions therefore leaves $G^l$unchanged: the matrix describes which patterns co-exist, not where they are located. This is what allows the style to be applied without copying the composition of the style image.
- **Layer-by-layer computation.** Correlations are calculated **within each layer**, between its $C_l$channels. No correlations are calculated between two different layers, which, incidentally, have different resolutions. The multi-scale aspect arises from the combined use of five layers (table below): the shallow layers capture colours and fine textures, whilst the deep layers capture larger patterns.
- **Link to distributions.** Aligning $G^l(x)$ with $G^l(s)$ amounts to minimising the maximum mean deviation (MMD), using a quadratic polynomial kernel, between the distributions of the feature vectors of the two images [20].

**Dimensions of Gram matrices for a 256 × 256 image**

| Style layer | Block | Channels $C_l$ | Positions $M_l$ (256 × 256) | Size of $G^l$ |
|---|---|---|---|---|
| relu1_1 | 1 | 64 | 256 × 256 = 65,536 | 64 × 64 |
| relu2_1 | 2 | 128 | 128 × 128 = 16,384 | 128 × 128 |
| relu3_1 | 3 | 256 | 64 × 64 = 4,096 | 256 × 256 |
| relu4_1 | 4 | 512 | 32 × 32 = 1,024 | 512 × 512 |
| relu5_1 | 5 | 512 | 16 × 16 = 256 | 512 × 512 |

**Numerical example.** Consider a layer with C = 2 channels and M = 3 positions, with F = [[1, 0, 2], [0, 1, 1]]. Then F $F^T$ = [[1·1 + 0·0 + 2·2, 1·0 + 0·1 + 2·1], [2, 0·0 + 1·1 + 1·1]] = [[5, 2], [2, 2]], from which G = F $F^T$ / (C M) = [[5/6, 1/3], [1/3, 1/3]]. The off-diagonal term 1/3 reflects the fact that both channels are active simultaneously at the third position. If we permute the columns of F, i.e. the positions, G remains the same.
The style loss of layer l compares the Gram matrices of the generated image and the style image, $A^l = G^l(s)$:

$$E_l = \frac{1}{C_l^2}\sum_{i,j}(G_{ij}^l(x) - A_{ij}^l)^2 \quad (3)$$

**Objective function and gradients:-**
**The total style loss is a weighted sum over the set Λ = {relu1_1, relu2_1, relu3_1, relu4_1, relu5_1} of style layers, with equal weights $w_l$ = 1/5:-**

$$L_s(x,s) = \sum_{l\in\Lambda} w_l E_l \quad (4)$$

**The objective to be minimised combines the two losses. The ratio β/α determines the trade-off between content preservation and style intensity; the baseline configuration uses α = 1 and β = 1:-**

$$L(x) = \alpha L_c(x,c) + \beta L_s(x,s) \quad (5)$$

**The gradients with respect to the activations are obtained analytically. For the content loss:-**

$$\frac{\partial L_c}{\partial F^l} = \frac{2}{C_l M_l}(F^l - P^l) \quad (6)$$

**For the style loss, we start with $\partial G^l_{ij}/\partial F^l_{ik} = (F^l_{jk} + \delta_{ij}F^l_{ik})/(C_l M_l)$. Using the symmetry of $G^l - A^l$, we obtain:-**

$$\frac{\partial E_l}{\partial F^l} = \frac{4}{C_l^3 M_l}(G^l - A^l)F^l \quad (7)$$

These gradients are then backpropagated through the layers of VGG16 down to the pixels of x, which is achieved by PyTorch's automatic differentiation [16]. The style loss gradient pulls $F^l(x)$ in the directions where the correlations of the generated image differ from those of the style.

**Image optimisation:-**
The generated image is initialised using the content image ($x_0$ = c), which preserves the shapes from the outset and accelerates convergence.

**It is then updated using the Adam algorithm [13], with learning rate η, followed by a projection Π onto [0, 1] to ensure valid pixel values:-**

$$x_{t+1} = \Pi\left(x_t - \eta \frac{\hat{m_t}}{\sqrt{\hat{v_t}}+\varepsilon}\right) \quad (8)$$

In this equation, $\hat{m}_t$ and $\hat{v}_t$ are the corrected estimates of the first- and second-order moments of the gradient $\partial L/\partial x$. One iteration consists of a forward pass, a backpropagation step and an update. This is not an 'epoch' in the sense of learning, as no dataset is processed and no network weights are modified.

**Differences from the baseline method:-**

**Table 2. Differences between the proposed method and that of Gatys et al.**

| Element | Gatys et al. [2] | Proposed method |
|---|---|---|
| Extractor network | VGG19 (normalised weights) | VGG16, original ImageNet weights |
| Content layer | relu4_2 (VGG19) | relu4_2 (VGG16) |
| Style layers | relu1_1 to relu5_1 | relu1_1 to relu5_1, $w_l = 1/5$ |
| Style loss normalisation | $1/(4\ N_l^2\ M_l^2)$ on the unnormalised Gram | Gram normalised by $C_l\ M_l$, then $1/C_l^2$ |
| Initialisation | white noise | content image ($x_0 = c$) |
| Optimiser | L-BFGS | Adam ($\eta = 0.02$) |
| Stopping criterion | not specified | 300 iterations, justified by the convergence analysis (§6.4) |
| Evaluation | Qualitative | SSIM, PSNR, style distances, time |

**Experimental protocol:-**
**Images:-**
**To assess the generality of the method, we use 4 content images and 4 style images, making a total of 16 pairs (Fig. 2).:-**

- **Content images:** the parrot from our initial study, along with three royalty-free reference images from the scikit-image library [17]. These cover different scenarios: an animal with fine textures (cat), a human portrait with large areas of solid colour (astronaut) and an object in an indoor scene (coffee).
- **Style images:** the style image from the initial study (a landscape with saturated colours) and three reference styles commonly used in the literature: Candy, Mosaic and Udnie. These three are taken from PyTorch's public example repository; they differ in their colour palette, the scale of their patterns and their texture.

All images are resized (short side to 256 pixels) and then centred and cropped to 256 × 256.

**Implementation parameters:-**

**Table 3. Parameters of the baseline configuration**

| Parameter | Value |
|---|---|
| Software framework | Python 3, PyTorch 2 [16], scikit-image [17] |
| Network | VGG16, original ImageNet weights [1], fixed; BGR pre-processing, mean subtracted |
| Image size | 256 × 256 pixels (3 channels) |
| Content layer | relu4_2 |
| Style layers (weights) | relu1_1, relu2_1, relu3_1, relu4_1, relu5_1 ($w_l = 1/5$) |
| Weights | $\alpha = 1$; $\beta = 1$ (sensitivity analysis: $\beta \in \{0.1; 1; 10\}$) |
| Initialisation | $x_0 = c$ (check: uniform noise) |
| Optimiser | Adam [13]: $\eta = 0.02$; $\beta_1 = 0.9$; $\beta_2 = 0.999$; $\varepsilon = 10^{\wedge -8}$ |
| Constraint | projection of pixels onto [0, 1] after each update |
| Number of iterations | 300 (evaluation); 600 (convergence study) |
| Stopping criterion | fixed number of iterations (300), justified by the convergence analysis in §6.4 |
| Reproducibility | Random seed 0; single-precision calculation |
| Hardware | 2-core processor, without a GPU (the times in Section 6.3 are given for this hardware) |

**Evaluation metrics:-**

**The quality of a style transfer cannot be measured by a single criterion [8]. We use four families of metrics.:-**

- **Content preservation.** SSIM [11] (structural similarity index, in [0, 1]) and PSNR, calculated between the generated image and the content image:

$$PSNR = 10\ log_{10} \frac{1}{EQM(x,c)}\ (dB) \quad (9)$$

- **Perceptual similarity.** LPIPS [12], a learned perceptual distance (AlexNet version), calculated between the generated image and the content image. The lower the value, the closer the images are.
- **Style fidelity.** The average distance between the Gram matrices of the generated image and the style image. This is calculated using a different evaluation network from the one used for optimisation (VGG19), so as not to evaluate the method against its own criteria, and with VGG16 for illustrative purposes. We add a distance metric independent of any network: the sliced Wasserstein distance (SWD, 64 projections) between the RGB colour distributions of the generated image and the style image [18].
- **Cost.** Computation time per image and per iteration.

A clarification is required: SSIM, PSNR and LPIPS, measured relative to the content image, are not quality scores to be optimised in isolation. An image identical to the content would achieve the highest score without exhibiting any style. These metrics are therefore interpreted in conjunction with the style distances; together, they describe the position of each method within the content–style trade-off.

## Results and Discussion:-

**Qualitative results:-**

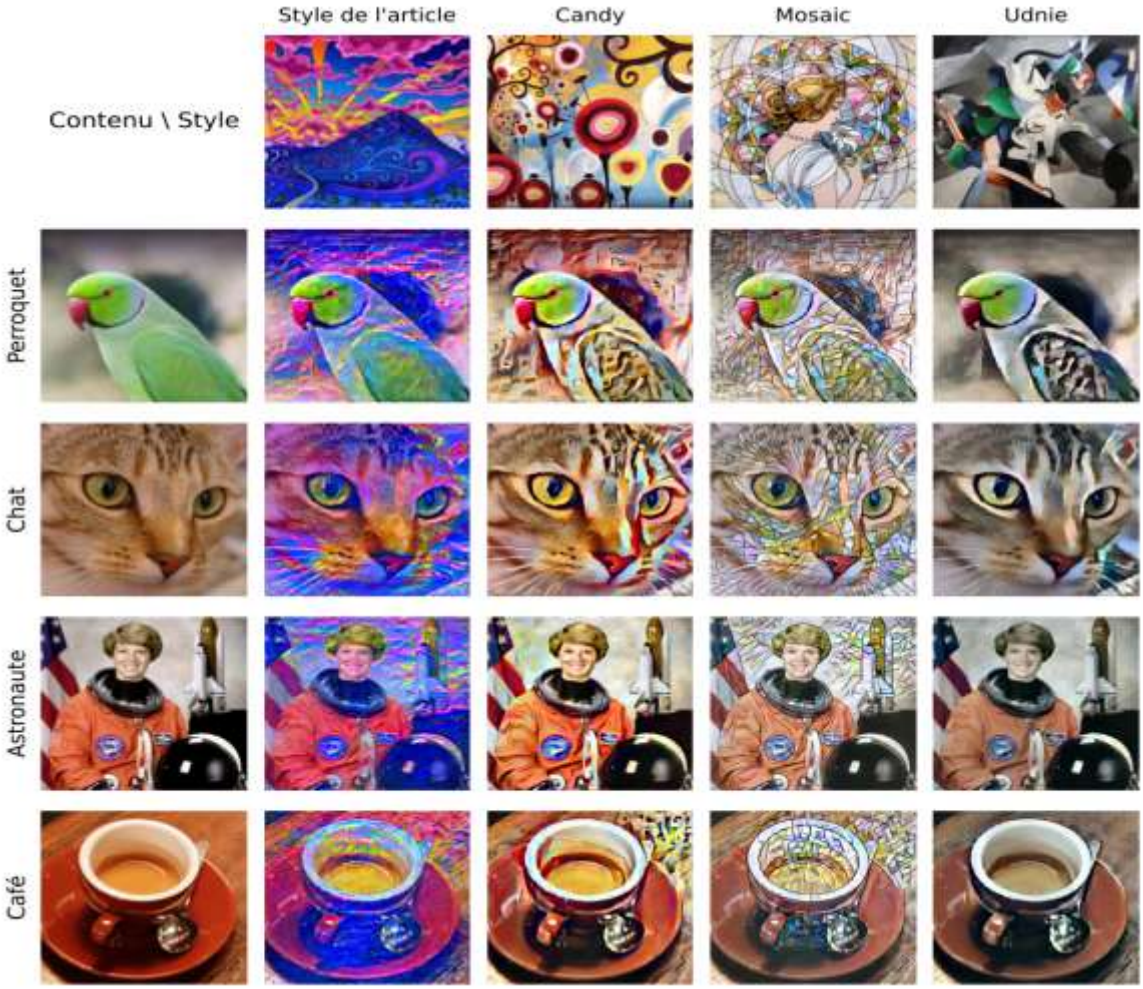



**Fig. 2. Results of the proposed method (VGG16) on the 16 content–style pairs (300 iterations, α = β = 1). Top row: style images; first column: content images.**

Fig. 2 shows that the method can be applied to a variety of content and styles. In all cases, the main contours (the parrot's silhouette, the cat's head, the astronaut's face, the cup) remain recognisable, whilst the colour palette and textures of the style image are transferred. The 'Article' and 'Candy' styles, which are very colourful, significantly recolour the image. 'Mosaic' imposes a pattern of small tesserae that fragments the solid areas. 'Udnie', with its dark hues and geometric shapes, darkens the image and breaks up surfaces into facets, for example on the parrot's body; its effect remains more subtle on large, uniform areas.

**Quantitative evaluation:-**

**Table 4. Proposed method (VGG16): average metrics (± standard deviation) across the 4 content images, for each style**

| Style | SSIM ↑ | PSNR (dB) ↑ | LPIPS ↓ | Gram distance (VGG19) ↓ | Colour SWD ↓ |
|---|---|---|---|---|---|
| Article style | 0.252 ± 0.050 | 11.68 ± 0.71 | [to be completed] | $1.03 \times 10^5 \pm 4.12 \times 10^3$ | 0.074 ± 0.012 |
| Candy | 0.325 ± 0.113 | 14.75 ± 1.59 | [to be completed] | $1.96 \times 10^5 \pm 3.62 \times 10^4$ | 0.057 ± 0.019 |
| Mosaic | 0.258 ± 0.086 | 13.16 ± 1.00 | [to be completed] | $9.69 \times 10^4 \pm 4.65 \times 10^3$ | 0.086 ± 0.028 |
| Udnie | 0.440 ± 0.090 | 16.15 ± 1.74 | [to be completed] | $4.83 \times 10^4 \pm 1.54 \times 10^4$ | 0.052 ± 0.005 |
| **Total (16 couples)** | 0.319 ± 0.116 | 13.94 ± 2.14 | [to be completed] | $1.11 \times 10^5 \pm 5.71 \times 10^4$ | 0.067 ± 0.022 |

SSIM, PSNR and LPIPS: relative to the content image. Gram distance: averaged across the five style layers of VGG19. SWD: sliced Wasserstein distance between the colours of the generated image and the style image. The values per pair are shown in Table A1.Across the 16 pairs, the method produces images that preserve the content structure (average SSIM of 0.319 ± 0.116) whilst reducing style loss by an average of 96.4 per cent compared with the original image. The results vary mainly depending on the style: the average SSIM ranges from 0.252 (Article Style) to 0.440 (Udnie), with styles featuring the most distinctive patterns altering the image to a greater extent. The average PSNR is 13.94 dB; this moderate value is to be expected, as style transfer deliberately alters the colours and textures of the pixels. The Gram and colour distances confirm that the generated image approximates the statistics of the style image. These distances vary from one style to another, as they depend on the richness of each style.

**Comparison with established methods:-**

**We compare four methods across the four pairs shown in Fig. 3, then plot trade-off curves for two pairs (Fig. 7):-**

- **VGG16**, the proposed method;
- **VGG19**, the baseline configuration from Gatys et al. [2], run with the same algorithm, layers, weights and parameters as the proposed method, in order to isolate the effect of the network;
- **ResNet-50** [9], with the same formulation: content at the output of the res3 layer, style on conv1, res2, res3 and res4. Its activations, normalised in batches, are of much lower amplitude than those of VGG: with $\beta = 1$, the image remains almost unchanged. We therefore calibrated $\beta$ across two pairs ($\beta \in \{10^3; 10^4; 10^5; 10^6\}$) and selected $\beta = 10^4$, whose content distance is closest to that of VGG16;
- **AdaIN** [6], a forward-propagation method, using the weights published by its authors.




**Fig. 3. Qualitative comparison of the methods across four content/style pairs.**

**Table 5. Comparison of methods (averages across the 4 pairs in Fig. 3)**

| Method | SSIM ↑ | PSNR (dB) ↑ | LPIPS ↓ | Gram VGG19 ↓ | Gram VGG16 ↓ | SWD ↓ | Time per image | Time per iteration |
|---|---|---|---|---|---|---|---|---|
| **VGG16 (proposed)** | 0.319 | 13.99 | [to be completed] | $1.16\times10^{5}$ | $7.91 \times 10^{3}$ | 0.076 | 136 s | 0.39 s |
| VGG19 (Gatys) | 0.354 | 16.43 | [to be completed] | $1.03\times10^{5}$ | $4.14 \times 10^{4}$ | 0.109 | 160 s | 0.49 s |
| ResNet-50 ($\beta = 10^{4}$) | 0.262 | 12.35 | [to be completed] | $5.02 \times 10^{5}$ | $1.21 \times 10^{5}$ | 0.042 | 59 s | 0.16 s |
| AdaIN | 0.205 | 10.94 | [to be completed] | $7.58 \times 10^{4}$ | $2.22 \times 10^{4}$ | 0.040 | 0.48 s | — |

Optimisation methods: 300 iterations. AdaIN: a single pass (pre-trained encoder and decoder). Times measured on a dual-core processor.

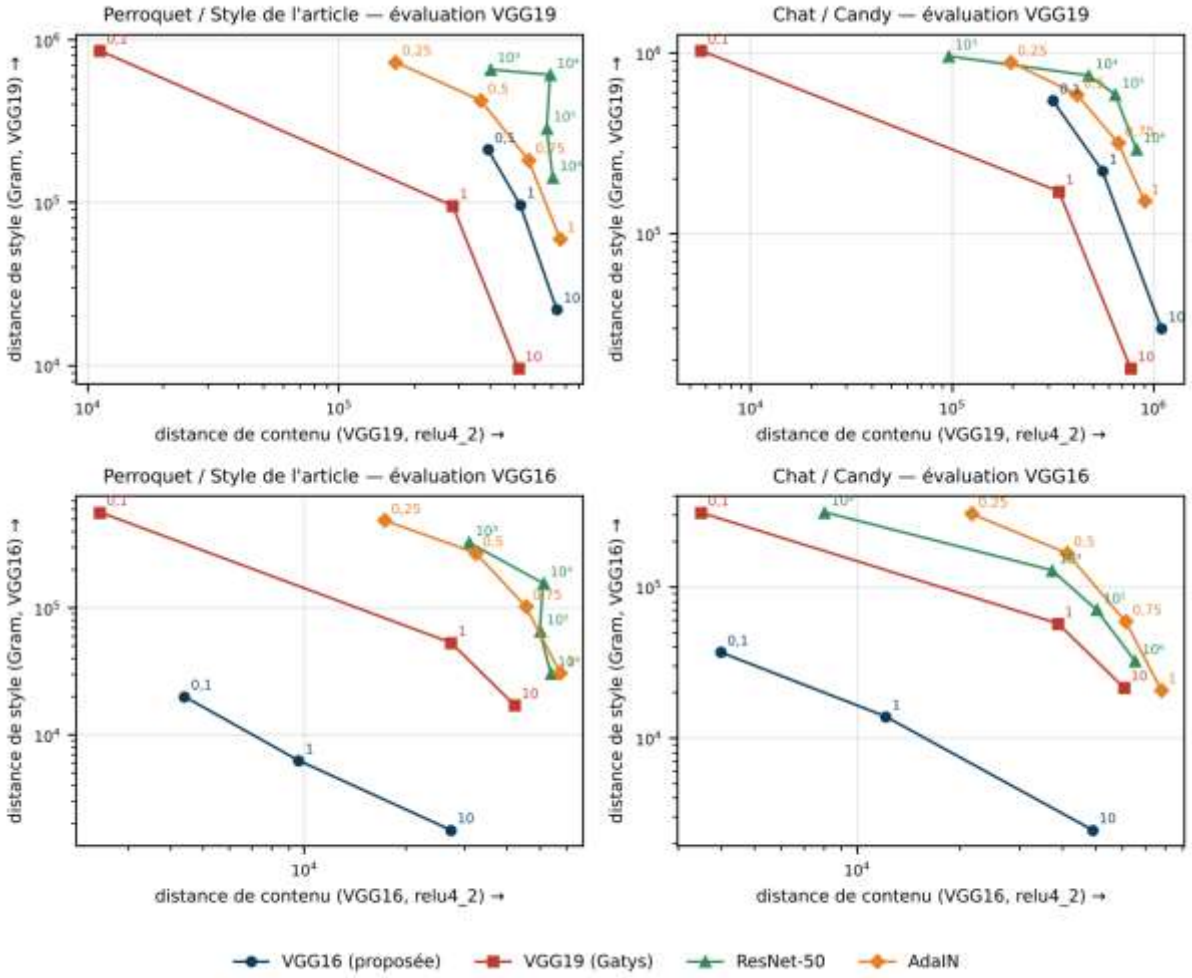



**Fig. 7. Content–style trade-off plots for two pairs, obtained by varying β (VGG16, VGG19, ResNet-50) or α (AdaIN). Top: distances measured with VGG19; bottom: with VGG16. A point located lower and further to the left is better.**

Of the four pairs compared using the same parameters, VGG16 and VGG19 produce visually very similar results, but they fall at different points on the content–style trade-off. VGG19 preserves the structure slightly better (SSIM 0.354 versus 0.319). VGG16 transfers the style's colour palette to a greater extent (SWD colour distance 0.076 versus 0.109). When measured against VGG19, the Gram distance is similar ($1.^{16\times10^5}$versus $1.^{03\times10^5}$). The trade-off curves obtained by varying β (Fig. 7) show that each network appears to perform better when evaluated on its own metrics: none objectively outperforms the other. At comparable quality, VGG16 is less computationally expensive: 0.39 s per iteration versus 0.49 s (−21 per cent), or 136 s versus 160 s per image. ResNet-50 does not transfer style well. With β = 1, the image remains almost unchanged (average SSIM 0.60), as its activations, normalised by batch, are of much lower amplitude.

Even with a calibrated β ($^{10^4}$to$^{10^6}$), it reproduces only the colour palette, without the textures: for a comparable content distance, its Gram distance remains several times greater than that of VGG16 (Fig. 7). This behaviour is consistent with the analysis by Wang et al. [10]. AdaIN [6] is by far the fastest (0.48 s per image, without optimisation). It strongly imposes the style (SWD distance 0.040) but distorts the structure more (SSIM 0.205) and requires a pre-trained decoder. For the parrot/style pair described in the paper, Fig. 7 allows a comparison at a comparable content distance, measured using VGG19. With β = 10, VGG16 achieves a Gram distance of $2.^{21\cdot10^4}$, compared $^{\text{with}}$ $6.^{01\cdot10^4}$for AdaIN (α = 1) and $1.42\cdot10^5$ for ResNet-50 (β = $10^6$ ). These three configurations have similar content distances ($7.35\cdot10^5$ , $7.58\cdot10^5$ and $7.^{09\times10^5}$ ). These results justify the choice of VGG16: transfer quality comparable to that of VGG19 at a lower cost, significantly better than that of ResNet-50, and better

preservation of structure than AdaIN, without any prior training. The trade-off is a computational time that is considerably longer than that of direct propagation methods.

**Convergence analysis:-**

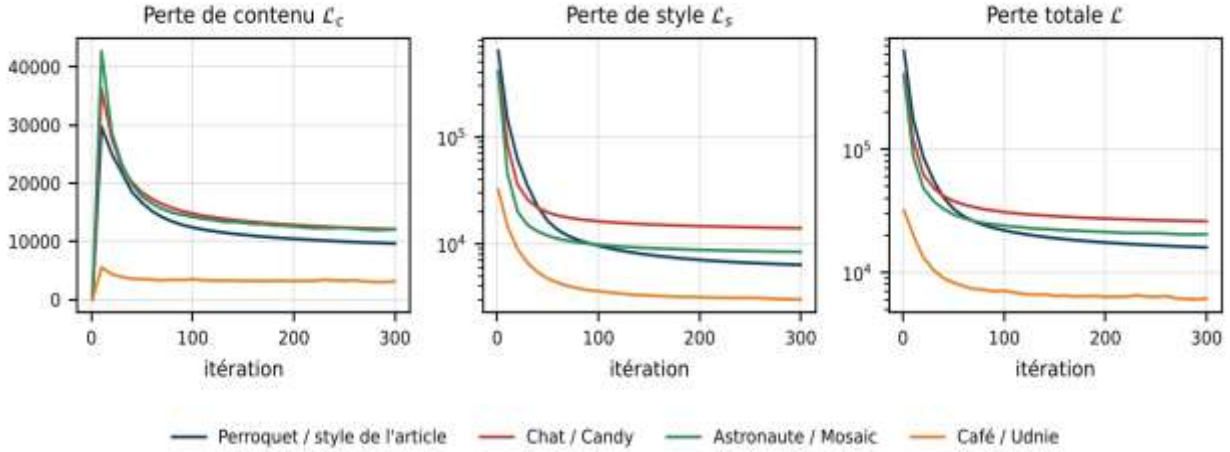



**Fig. 4. Evolution of content, style and total losses during optimisation (VGG16, α = β = 1) for four pairs. The style and total losses are plotted on a logarithmic scale.**

The curves in Fig. 4 exhibit the same behaviour for all four pairs. The content loss starts at zero, since $x_0 = c$. It increases sharply during the first few iterations, peaking around the $10^{th}$ iteration, when the style significantly alters the image. It then decreases as the contours reform, before stabilising. The loss of style decreases by several orders of magnitude, and the total loss decreases monotonically. Across the 16 pairs, the total loss reaches 90 per cent of its final reduction in 29 ± 8 iterations, and 99 per cent in 121 ± 28 iterations. For the 'parrot/style' pair in the article, the loss of style decreases from $6.35 \cdot 10^5$ to $6.33 \cdot 10^3$, a decrease of 99.0 per cent, whilst the content loss stabilises at around $9.61 \cdot 10^3$. This figure is supported by the convergence curves: for the 16 pairs, 99 per cent of the reduction in total loss is achieved in 121 iterations on average, and within 170 iterations at most. Over the last 50 iterations, the total loss varies by no more than 2.7 per cent on average (5.3 per cent at most). These values are not absolute 'errors': their magnitude depends on the normalisation (equations (1) to (3)) and the weights α and β. Only their relative change can be interpreted. The final value of the content loss measures the proportion of the structure sacrificed for the sake of style; that of the style loss measures the residual deviation from the style statistics.

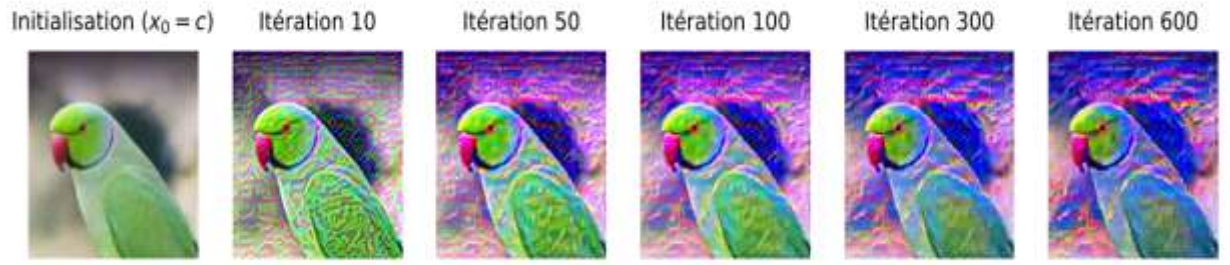



**Fig. 5. Image generated during optimisation (parrot, article style): from initialisation using the content image up to 600 iterations.**

Fig. 5 plots these curves against the image. From 10 iterations onwards, the style palette is largely established and spurious textures appear; this is when the content loss reaches its maximum (Fig. 4). Between 10 and 100 iterations, the parrot's contours reform and the textures organise themselves according to the image's shapes, whilst the content loss falls again and the style loss continues to decrease. Beyond 300 iterations, the image hardly changes at all, which corresponds to the plateau in the curves. Extending the optimisation to 600 iterations yields only a marginal refinement of the textures; 300 iterations therefore represent a good compromise between quality and computation time.

**Sensitivity to weights and initialisation:-**

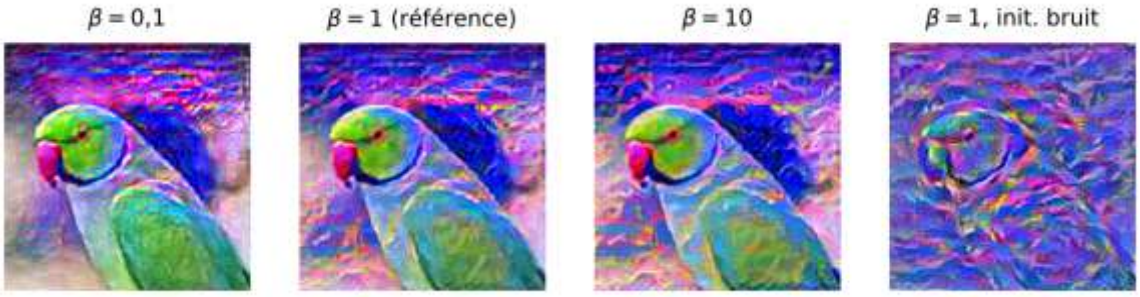



**Fig. 6. Effect of the style weight β and initialisation (parrot, article style, 300 iterations).**

**Table 6. Sensitivity to style weight β and initialisation**

| Configuration | SSIM ↑ | PSNR (dB) ↑ | VGG16 content ↓ | Gram VGG19 ↓ | SWD ↓ |
|---|---|---|---|---|---|
| β = 0.1 | 0.196 | 13.11 | $4.39 \times 10^{3}$ | $2.13 \times 10^{5}$ | 0.115 |
| β = 1 (reference) | 0.193 | 12.71 | $9.60 \times 10^{3}$ | $9.75 \times 10^{4}$ | 0.092 |
| β = 10 | 0.197 | 12.74 | $2.71 \times 10^{4}$ | $2.21 \times 10^{4}$ | 0.093 |
| β = 1, initialised by noise | 0.022 | 8.94 | $1.23 \times 10^{4}$ | $9.27 \times 10^{4}$ | 0.093 |

The weight β controls the content–style trade-off (Fig. 6, Table 6). When β changes from 0.1 to 10, the content distance (measured using VGG16) changes from $4.^{39\cdot10^{3}}$ to $2.^{71\cdot10^{4}}$, and the style distance (Gram, VGG19) from $2^{\cdot13\cdot10^{5}}$ to $2^{\cdot21\cdot10^{4}}$. The baseline configuration β = 1 occupies an intermediate position ($9.^{60\cdot10^{3}}$ and $9.^{75\cdot10^{4}}$). The SSIM varies little for this pair (from 0.196 to 0.197). The content image of the parrot, which is of low resolution to begin with, is in fact very smooth; any added texture, however slight, already reduces the SSIM. This is why the distances calculated on the features are more informative than the SSIM for describing this trade-off. With noise initialisation, as in Gatys et al., and 300 iterations, the style distance is similar but the structure is reconstructed much less effectively (SSIM 0.022 versus 0.193). Initialisation using the content image is therefore significantly more effective for the same number of iterations.

**Appendix — Detailed results by pair:-**

**Table A1. Proposed method (VGG16, 300 iterations): metrics for each of the 16 pairs**

| Content | Style | SSIM | PSNR (dB) | LPIPS | Gram VGG19 | SWD | Time (s) |
|---|---|---|---|---|---|---|---|
| Parrot | Article style | 0.193 | 12.71 | [to be completed] | $9.75 \times 10^{4}$ | 0.092 | 134 |
| Parrot | Candy | 0.169 | 13.34 | [to be completed] | $2.01 \times 10^{5}$ | 0.037 | 133 |
| Parrot | Mosaic | 0.138 | 13.76 | [to be completed] | $9.17 \times 10^{4}$ | 0.057 | 131 |
| Parrot | Udnie | 0.393 | 15.51 | [to be completed] | $5.73 \times 10^{4}$ | 0.052 | 133 |
| Chat | Article style | 0.246 | 11.47 | [to be completed] | $1.09 \times 10^{5}$ | 0.068 | 134 |
| Chat | Candy | 0.323 | 14.01 | [to be completed] | $2.24 \times 10^{5}$ | 0.059 | 134 |
| Chat | Mosaic | 0.272 | 12.67 | [to be completed] | $9.74 \times 10^{4}$ | 0.065 | 136 |

| Content | Style | SSIM | PSNR (dB) | LPIPS | Gram VGG19 | SWD | Time (s) |
|---|---|---|---|---|---|---|---|
| Chat | Udnie | 0.392 | 15.11 | [to be completed] | $6.27 \times 10^4$ | 0.052 | 134 |
| Astronaut | Article style | 0.331 | 11.83 | [to be completed] | $1.01 \times 10^5$ | 0.059 | 132 |
| Astronaut | Candy | 0.489 | 17.45 | [to be completed] | $1.36 \times 10^5$ | 0.047 | 132 |
| Astronaut | Mosaic | 0.379 | 14.39 | [to be completed] | $9.44 \times 10^4$ | 0.094 | 140 |
| Astronaut | Udnie | 0.597 | 19.13 | [to be completed] | $2.28 \times 10^4$ | 0.046 | 131 |
| Coffee | Article style | 0.238 | 10.72 | [to be completed] | $1.03 \times 10^5$ | 0.076 | 132 |
| Coffee | Candy | 0.320 | 14.21 | [to be completed] | $2.24 \times 10^5$ | 0.087 | 133 |
| Coffee | Mosaic | 0.243 | 11.80 | [to be completed] | $1.04 \times 10^5$ | 0.128 | 131 |
| Coffee | Udnie | 0.380 | 14.87 | [to be completed] | $5.04 \times 10^4$ | 0.060 | 135 |

## Discussion and limitations:-

The experiments confirm that the proposed formulation is applicable to a variety of content and styles, with stable convergence behaviour and a trade-off that can be adjusted via β. The choice of VGG16 is justified: it offers transfer quality comparable to that of VGG19 at a lower cost, and significantly superior to that of ResNet-50. The approach has three limitations. (i) Compared with forward-propagation methods such as AdaIN, the computation time is high, of the order of two minutes per 256 × 256 image on a dual-core processor; it would be significantly reduced on a GPU. (ii) Automatic metrics capture only part of the perceived quality; an evaluation by human observers would usefully complement the study. (iii) The weights $w_l$ are uniform and the resolution is limited to 256 × 256; adjusting these weights and moving to higher resolutions, for example via a multi-scale approach, are avenues for improvement.

## Conclusion:-

We have presented a comprehensive formulation of image style transfer based on a pre-trained and frozen VGG16 network. The content is represented by the activations of the relu4_2 layer, and the style by the normalised Gram matrices of five layers. The generated image, initialised by the content image, is optimised using the Adam optimiser. We have formalised the losses and their gradients, clarified the role of the Gram matrix and specified all parameters. Evaluation on 16 content/style pairs shows that the method preserves the content structure (average SSIM of 0.319) whilst transferring the style, with convergence achieved within a few hundred iterations. The controlled comparison shows that VGG16 offers quality comparable to that of VGG19 at a lower cost, and significantly better than that of ResNet-50. It also shows that the method preserves the structure better than AdaIN, without prior training, at the cost of increased computation time. Future work will focus on GPU acceleration, adaptive layer weighting, high resolutions and evaluation by human observers.

## References:-

1. K. Simonyan, A. Zisserman, 'Very deep convolutional networks for large-scale image recognition', International Conference on Learning Representations (ICLR), 2015. arXiv:1409.1556.
2. L. A. Gatys, A. S. Ecker, M. Bethge, 'Image style transfer using convolutional neural networks', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 2414–2423, 2016.
3. L. A. Gatys, A. S. Ecker, M. Bethge, 'A neural algorithm of artistic style', arXiv:1508.06576, 2015.

4. J. Johnson, A. Alahi, L. Fei-Fei, 'Perceptual losses for real-time style transfer and super-resolution', European Conference on Computer Vision (ECCV), pp. 694–711, 2016.
5. D. Ulyanov, V. Lebedev, A. Vedaldi, V. Lempitsky, 'Texture networks: feed-forward synthesis of textures and stylised images', International Conference on Machine Learning (ICML), 2016.
6. X. Huang, S. Belongie, 'Arbitrary style transfer in real-time with adaptive instance normalisation', IEEE International Conference on Computer Vision (ICCV), pp. 1501–1510, 2017.
7. Y. Li, C. Fang, J. Yang, Z. Wang, X. Lu, M.-H. Yang, 'Universal style transfer via feature transforms', Advances in Neural Information Processing Systems (NeurIPS), 2017.
8. Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, M. Song, 'Neural style transfer: a review', IEEE Transactions on Visualisation and Computer Graphics, vol. 26, no. 11, pp. 3365–3385, 2020.
9. K. He, X. Zhang, S. Ren, J. Sun, 'Deep residual learning for image recognition', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778, 2016.
10. P. Wang, Y. Li, N. Vasconcelos, 'Rethinking and improving the robustness of image style transfer', IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 124–133, 2021.
11. Z. Wang, A. C. Bovik, H. R. Sheikh, E. P. Simoncelli, 'Image quality assessment: from error visibility to structural similarity', IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600–612, 2004.
12. R. Zhang, P. Isola, A. A. Efros, E. Shechtman, O. Wang, 'The unreasonable effectiveness of deep features as a perceptual metric', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 586–595, 2018.
13. D. P. Kingma, J. Ba, 'Adam: a method for stochastic optimisation', International Conference on Learning Representations (ICLR), 2015.
14. J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, 'ImageNet: a large-scale hierarchical image database', IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 248–255, 2009.
15. O. Russakovsky, J. Deng, H. Su et al., 'ImageNet Large-Scale Visual Recognition Challenge', International Journal of Computer Vision, vol. 115, no. 3, pp. 211–252, 2015.
16. Paszke, S. Gross, F. Massa et al., 'PyTorch: an imperative-style, high-performance deep learning library', Advances in Neural Information Processing Systems (NeurIPS), 2019.
17. S. van der Walt, J. L. Schönberger, J. Nunez-Iglesias et al., 'scikit-image: image processing in Python', PeerJ, vol. 2, e453, 2014.
18. J. Rabin, G. Peyré, J. Delon, M. Bernot, 'Wasserstein barycentre and its application to texture mixing', Scale Space and Variational Methods in Computer Vision (SSVM), LNCS 6667, pp. 435–446, 2011.
19. Krizhevsky, I. Sutskever, G. E. Hinton, 'ImageNet classification with deep convolutional neural networks', Advances in Neural Information Processing Systems (NeurIPS), 2012.
20. Y. Li, N. Wang, J. Liu, X. Hou, 'Demystifying neural style transfer', International Joint Conference on Artificial Intelligence (IJCAI), pp. 2230–2236, 2017.