

Réseaux de neurones convolutifs pour le transfert de style

Bouziane Othmane

CentraleSupelec

Abstract. L'objectif de cette étude est de transférer le style d'un tableau réalisé par un artiste vers une photographie numérique pour qu'elle ressemble à une peinture faite par cet artiste. On définit le transfert de style comme étant un processus de modification du style d'une image tout en préservant son contenu. Pour réaliser cette tâche, je vais implémenter un réseau de neurones permettant d'extraire des features clés du tableau pour les appliquer ensuite à une image cible. Je propose donc dans cet article une méthode basée sur un réseau pré-entraîné de type VGG16 implémenté sous PyTorch. L'algorithme utilise donc ce réseau de neurones convolutifs pour séparer et recombiner le contenu et le style des images, fournissant ainsi un algorithme fonctionnel pour la création d'images artistiques.

Keywords: Style Transfer · VGG16 · CNN.

1 Introduction

Les réseaux de neurones convolutifs sont aujourd'hui les outils les plus efficaces, puissants et adaptés pour le traitement d'image.

Le problème étudié est le suivant: on veut transformer une image prise par un appareil photo en une peinture dans le style d'un artiste choisi pour lequel on ne possède qu'une seule oeuvre.

Pour pouvoir réaliser ce transfert de style nous allons extraire de l'oeuvre les features de style pour pouvoir ensuite les appliquer à notre image cible tout en nous assurant que les features de contenu de la cible sont conservées.

2 État de l'art

Le problème du transfert de style est un sujet attrayant car il permet de démontrer la puissance des CNN grâce à un cas d'utilisation spectaculaire. De ce fait, plusieurs travaux ont déjà été réalisés dans le domaine. L'article de Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu et M. Song [1] présente un état de l'art complet des différentes méthodes existantes dans le cadre du transfert de style utilisant des réseaux de neurones. Vous trouverez ci-dessous une sélection des méthodes les plus concluantes présentées dans l'article.

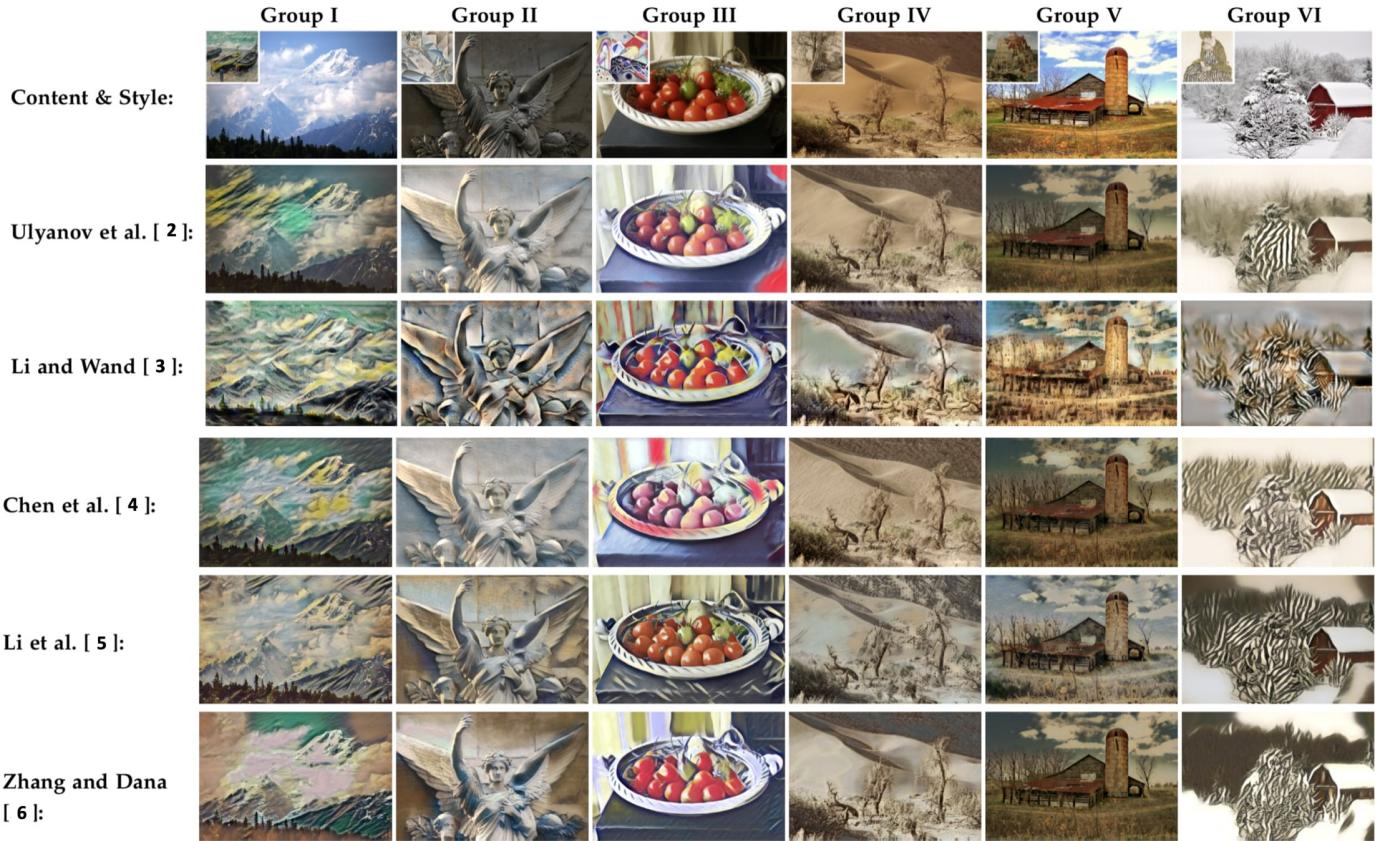


Fig. 1. Présentation d'une sélection des méthodes

Dans le cadre de ce projet, je propose de partir d'un modèle qu'on retrouve dans plusieurs de ces articles [7] et qui utilise un réseau VGG-16 pour effectuer le transfert de style. Je vais d'abord l'implémenter sous PyTorch pour ensuite essayer de l'améliorer en testant de nouveaux paramètres d'entraînement. L'article sur lequel se base principalement la méthode de départ choisie est *Image Style Transfer Using Convolutional Neural Networks* par A. Gatys, S. Ecker et M. Bethge

3 Principe de la méthode

Le principe du transfert de style neuronal consiste à définir deux fonctions de distance, l'une **content_loss** qui décrit à quel point le contenu de deux images est différent, et l'autre **style_loss** qui décrit la différence entre deux images

en fonction de leur style. Ensuite, nous utilisons l'image cible en entrée et nous optimisons les poids du réseau de neurones en minimisant d'une part la distance de style `content_loss` avec l'oeuvre d'art tout en conservant une distance de contenu `content_loss` avec l'image en entrée minimale. L'oeuvre d'art ainsi créée aura donc le même style que l'oeuvre d'art mise en entrée tout en ayant le contenu de l'image.

3.1 Structure du réseau de neurone

Nous construisons un réseau feed-forward basé sur un simple auto-encodeur d'image (illustré à la figure 2). Ce réseau transforme l'oeuvre d'art (image de style) ainsi que l'image en entrée (image de contenu) en un ensemble de features grâce à de multiples convolutions. À chaque couche, la distance de style est calculée, la distance de contenu, elle, est calculée uniquement à la quatrième couche. Une distance globale est ensuite déduite, c'est cette distance qui devra être minimisée pour avoir l'oeuvre d'art en sortie.

Le système est basé sur réseau de neurones convolutif de type VGG-16 pré-entraîné sur PyTorch, les couches de ce réseau sont représentées ci-dessous:

```
(0): Normalization()
(conv_1): Conv2d(3, 64, kernel_size=(3, 3), stride=(1, 1),
    padding=(1, 1))
(style_loss_1): StyleLoss()
(rect_1): ReLU()
(conv_2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1),
    padding=(1, 1))
(style_loss_2): StyleLoss()
(rect_2): ReLU()
(pool_2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ceil_mode=False)
(conv_3): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1),
    padding=(1, 1))
(style_loss_3): StyleLoss()
(rect_3): ReLU()
(conv_4): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1),
    padding=(1, 1))
(content_loss_4): ContentLoss()
(style_loss_4): StyleLoss()
(rect_4): ReLU()
(pool_4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
    ceil_mode=False)
(conv_5): Conv2d(128, 256, kernel_size=(3, 3), stride=(1, 1),
    padding=(1, 1))
(style_loss_5): StyleLoss()
```

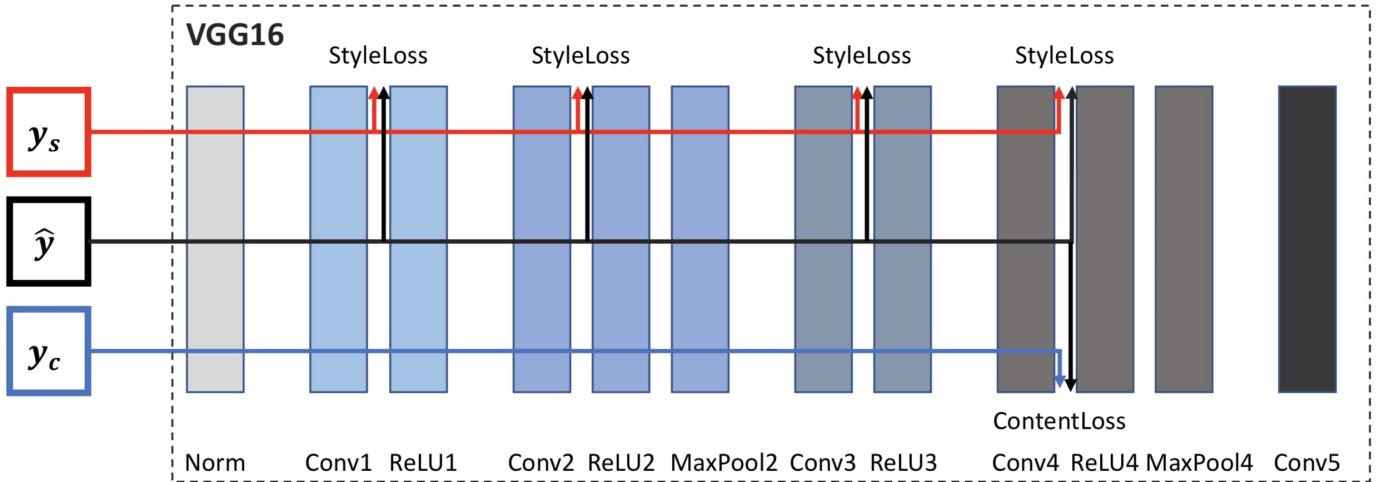


Fig. 2. Vue d'ensemble du système

Le réseau de transformation d'image permet de produire des images stylisées grâce à un ensemble de calcul de fonction de distance (loss functions). On définit plusieurs variables :

1. x est l'image en entrée à laquelle on veut appliquer un style
2. y_s est l'oeuvre d'art à partir de laquelle nous allons extraire le style que nous voulons appliquer
3. $y_c = x$ Image en entrée que nous allons utiliser pour calculer la distance de contenu avec l'image initiale
4. \hat{y} est l'oeuvre d'art en sortie ayant le contenu de l'image initiale avec le style de l'oeuvre d'art en entrée

3.2 Métriques de distances utilisées

Distance de contenu : content_loss

La distance de contenu à la sortie de la couche 1 est définie comme étant un MSE pixel par pixel, ainsi :

$$\text{content_loss}(l) = \frac{1}{2} \sum_{i,j} (\hat{y}_{i,j}^l - y_{c,i,j}^l)^2$$

Avec :

- $\hat{y}_{i,j}^l$ pixel (i,j) de l'image cible à la sortie de la l^{eme} couche du réseau
- $y_{c,i,j}^l$ pixel (i,j) de l'image de contenu à la sortie de la l^{eme} couche du réseau

Étant donné qu'on ne calcule qu'une seule distance de contenu par parcours du réseau de neurones, on définit $\text{content_loss} = \text{content_loss}(4)$

Distance de style : `style_loss`

Les corrélations entre les différentes features d'une feature map constituent un moyen efficace de caractérisation du style d'une image. Nous proposons donc d'utiliser ces corrélations afin de comparer les styles de deux images. Nous utiliserons la matrice de Gram pour calculer ces corrélations. Cette matrice est définie par :

$$G(y^l) = y^{lT} y^l$$

Avec y^l image à la sortie de la l^{me} couche du réseau Pour $G(i, j)$, la matrice aura donc une valeur importante si les features i et j sont fortement corrélées.

On définit ensuite la distance de style `style_loss` à chaque couche de notre réseau comme étant la distance MSE entre les matrices de Gram de l'image de style et de l'image cible à la sortie de la l^{me} couche du réseau. Ainsi, on obtient :

$$style_loss(l) = \frac{1}{4 * (N^l)^2 * (M^l)^2} \sum_{i,j} (G(\hat{y}^l)_{i,j} - G(y_s)_{i,j})^2$$

Avec :

- $G(\hat{y}^l)_{i,j}$ pixel (i,j) de la matrice de Gram de l'image cible à la sortie de la l^{me} couche du réseau
- $G(y_s)_{i,j}$ pixel (i,j) de la matrice de Gram de l'image de style à la sortie de la l^{me} couche du réseau
- N^l et M^l respectivement le nombre de ligne et de colonnes de l'image \hat{y}^l

Finalement, on définit `style_loss` comme la somme des `style_loss(l)` de chaque couche

$$style_loss = \sum_{l=0}^4 style_loss(l)$$

3.3 Méthode d'optimisation

Comme mentionné dans la partie 3.1, l'erreur à minimiser est en réalité une somme de distance calculées à la sortie de chaque couche du réseau.

```
total_loss = content_weight*content_loss + style_weight*style_loss
```

Avec `content_weight` et `style_weight` deux constantes que nous devrons choisir par la suite pour avoir un rendu optimal. Ces deux constantes traduisent l'importance donnée à la transmission du style par rapport à la conservation du contenu.

Pour optimiser cette fonction, nous choisissons d'utiliser un L-BFGS (Limited-Memory Broyden-Fletcher-Goldfarb-Shanno).

4 Résultats

Pour pouvoir utiliser l'algorithme, la dernière étape est de fixer la valeur des hyperparamètres :

- `content_weight`
- `style_weight`
- `num_steps` : Nombre de parcours du réseau

Je propose donc dans un premier temps de tester plusieurs paramètres de `style_weight` et de `content_weight` en prenant une valeur fixe de `num_steps` = 300. Nous allons ensuite tester plusieurs valeurs de `num_steps` pour avoir un résultat de qualité optimale.

4.1 Détermination de `content_weight` et de `style_weight`

Étant donnée que les valeurs de `content_weight` et `style_weight` donnent l'importance **relative** de l'image de style par rapport à l'image de contenu, ces deux variables sont intrinsèquement liées et doivent donc être optimisées en même temps. On remarque que la valeur de `style_weight` doit être supérieure à `content_weight`.

Nous allons donc fixer `content_weight` à 1 et faire varier `style_weight` entre 1 et 100 000 000.

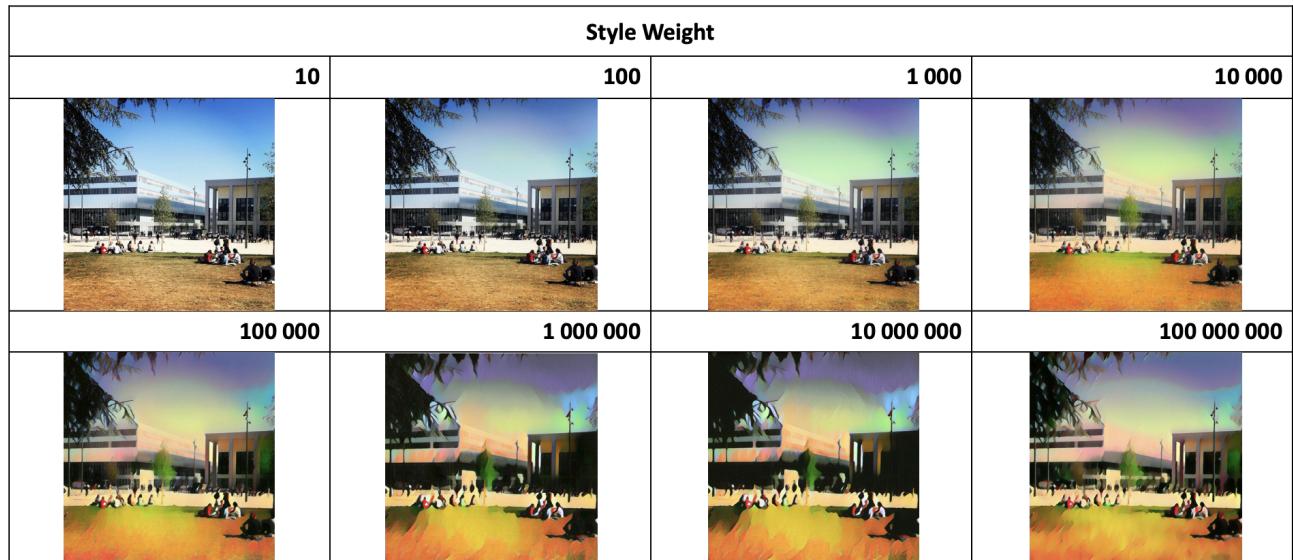


Fig. 3. Grid search sur `style_weight`

On remarque qu'à partir de `style_weight = 1 000 000` les formes commencent à devenir très floue même si l'image reste quand même reconnaissable.

4.2 Détermination de `num_step`

Nous allons maintenant fixer `content_weight = 1 000 000` et `style_weight = 1`. Nous allons ensuite faire varier `num_step` entre 100 à 500 pour choisir la plus petite valeur permettant d'avoir un résultat satisfaisant.

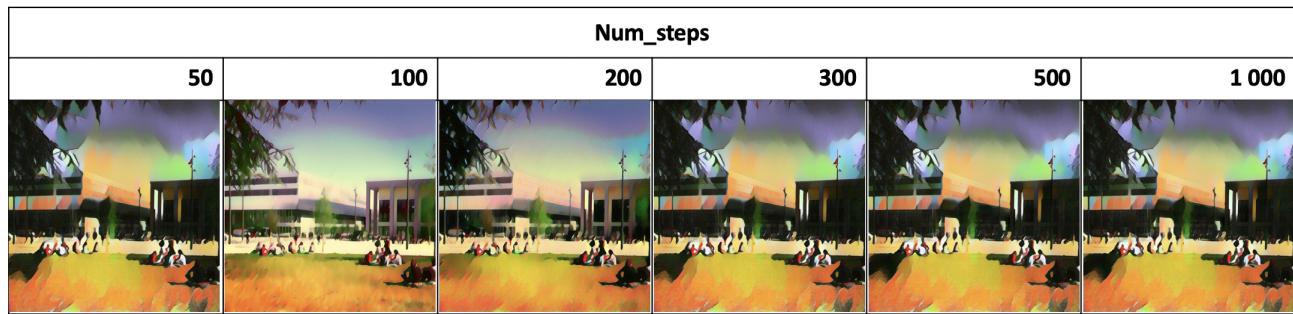


Fig. 4. Détermination de `num_step`

On remarque qu'à partir de 100 itérations, on obtient un résultat satisfaisant

4.3 Résultats avec les paramètres optimaux sur plusieurs images

Maintenant que nous avons choisi les paramètres optimaux sur cette image de style, nous allons vérifier si ces paramètres restent bon pour d'autres images. Ces images sont réalisés avec les paramètres suivant:

- `content_weight = 1`
- `style_weight = 1 000 000`
- `num_step = 100`

On remarque qu'avec les paramètres choisis, le réseau de neurones transmet bien les tons de couleurs ainsi que la "technique de peinture" de l'image de style vers l'image de contenu. Néanmoins, on remarque que les objets ne sont pas déformées et que la géométrie des images de style n'est que très peu transférée.

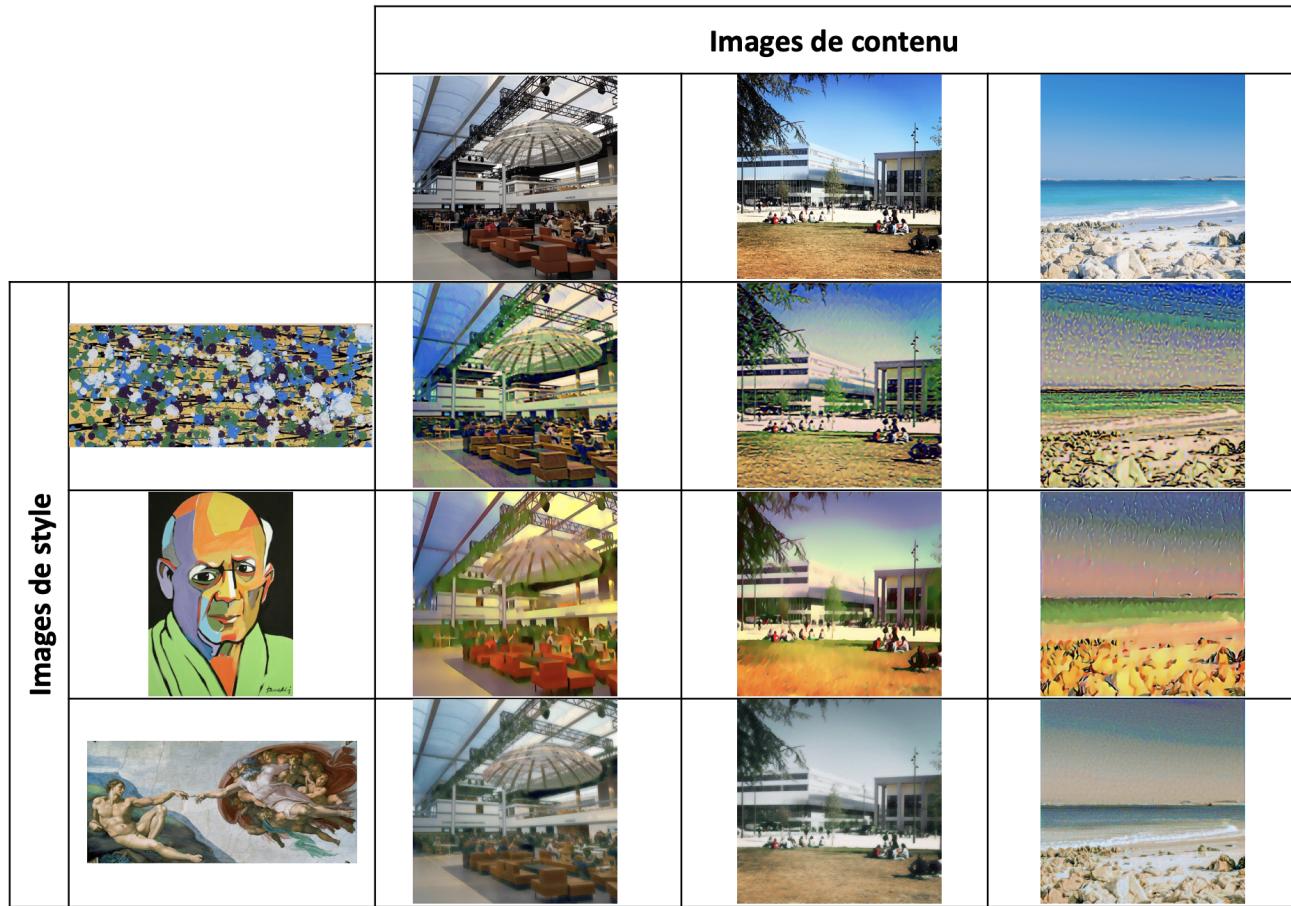


Fig. 5. Résultats sur différentes images de style et de contenu

5 Conclusion

Ce projet a été pour moi l'occasion de découvrir la structure du VGG-16, un réseau de neurones d'extraction de features que j'ai pu adapter pour l'utiliser pour faire du transfert du style. La prise en main de ce réseau m'a permis de réaliser que des réseaux de neurones assez simples permettent déjà de produire des résultats satisfaisants. Étant donné la qualité des images en sortie du réseau, la principale composante à améliorer dans le cadre de cette expérimentation est le temps de convergence de l'optimisation.

En effet, on pourrait penser à faire du transfert de style sur des flux vidéos, or pour l'instant, le temps de traitement de chaque image bien supérieur aux besoins de ce genre de système. Étant donné que dans ce type d'application les

images successives sont très similaires, nous pourrions donc penser à un réseau à mémoire qui sauvegardera les poids de l'image en t pour les réinjectés en $t+1$ et ainsi réaliser un nombre limité d'itérations à chaque image.

References

1. Zunlei Feng Jingwen Ye Yizhou Yu Yongcheng Jing, Yezhou Yang and Mingli Song. Neural style transfer: A review. *IEEE*, 2018.
2. A. Vedaldi D. Ulyanov and V. Lempitsky. Improved texture networks: Maximizing quality and diversity in feed-forward stylization and texture synthesis. *IEEE*, 2017.
3. C. Li and M. Wand. Combining markov random fields and convolutional neural networks for image synthesis. *IEEE*, 2016.
4. J. Liao N. Yu D. Chen, L. Yuan and G. Hua. Stylebank: An explicit representation for neural image style transfer. *IEEE*, 2017.
5. L. Nie S. Li, X. Xu and T.-S. Chua. Laplacian-steered neural style transfer. *ACM*, 2017.
6. H. Zhang and K. Dana. Multi-style generative network for real- time transfer. *arXiv*, 2017.
7. M. Bethge A. Gatys, S. Ecker. Image style transfer using convolutional neural networks. *CVF Computer Vision Fondation*, 2015.