Étude des réseaux de neurones pour le transfert de style

Arthur Migliorini (2022096) , **Grégoire Chapeaux (2033122)** , **Rémi Thiébaut (2051818)** et **Maxime Vincent (2052483)**

Étudiants au cours INF8225 à Polytechnique Montréal

{arthur.migliorini, gregoire.chapeaux, remi.thiebaut, maxime-1.vincent}@polymtl.ca

Résumé

La réalisation de transfert de styles dans des images est une problématique complexe, rendue possible par les avancées en intelligence artificielle. L'objectif de cet article est de faire un tour d'horizon des approches existantes. Nous allons donc présenter plusieurs méthodes permettant de réaliser un transfert de style dans des images. Les différentes techniques présentées donnent des résultats de qualité et de vitesse variables.

1 Introduction

Le transfert de style vise à générer de nouvelles images à partir d'une image d'entrée et d'une référence de style. On "reproduit" ce style sur l'image en d'entrée, via un réseau ayant suivi une phase d'apprentissage du style. Bien que l'intérêt immédiat de telles problématiques soit surtout artistique, le transfert de style représente un grand pas en avant pour l'intelligence artificielle, tant sur l'aspect technique (traitement des images) que sur l'aspect plus abstrait (définition d'un "style" et de son apprentissage).

Ce projet se base sur 4 articles : *A Neural Algorithm of Artistic Style* [1], un projet pionnier du transfert de style ; *Perceptual Losses for Real-Time Style Transfer and Super-Resolution* [2], qui introduit l'utilisation des CNN pour le transfert de style ; *Instance Normalization: The Missing Ingredient for Fast Stylization* [3], une amélioration de qualité du modèle précédent ; enfin *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization* [4], qui se propose de fonctionner autour de styles arbitraires.

Chacun de ces articles présente une approche novatrice pour réaliser un transfert de style. Nous avons réimplémenter les méthodes proposées afin de pouvoir comparer leurs résultats et leur vitesse de convergence.

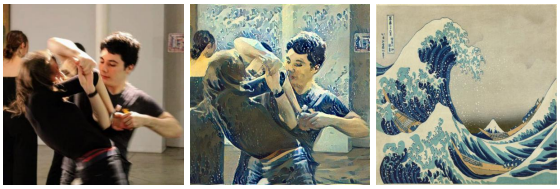
2 Une première approche du transfert de style

La première approche étudiée est celle proposée par l'article *A Neural Algorithm of Artistic Style* qui introduit le domaine du transfert de style. Elle consiste à utiliser un réseau déjà entraîné dans la reconnaissance d'objets, et d'utiliser certaines de ses sorties pour estimer à la fois à quel point le contenu d'une image correspond au contenu de l'image initiale et à quel point le style correspond au style de l'image dont on veut transférer le style. On a ici utilisé VGG19 comme réseau de reconnaissance d'objets. Pour comparer les sorties du réseau sur les différentes images et en déduire la *perte*, on procède différemment pour la *perte* de style et la *perte* de contenu.

Pour la perte de contenu, on fait simplement la somme des carrés des différences des sorties, mais pour la perte de style on calculera la matrice de Gramm de chaque sortie, et la *perte* sera l'erreur quadratique moyenne entre ces deux matrices, comme précédemment.

Dans ce papier, ce qui est *entraîné* est directement l'image de sortie, de ce fait l'algorithme n'a pas besoin d'entraînement préalable, mais peut mettre quelques minutes avant de renvoyer une image satisfaisante selon la taille des images d'entrées.

On peut voir sur la figure 1 un exemple de transfert de style en appliquant le style de *La Grande Vague de Kanagawa*. Cet exemple a été généré en utilisant pour le style et le contenu les couches classiquement utilisées, c'est-à-dire la sortie de la dernière convolution pour le contenu et la sortie de chacun des blocs de même nombre de *feature* pour le style. Néanmoins on peut voir sur la figure 2 un autre exemple utilisé en ne prenant en compte que les sorties des 5 premières convolutions qui convergent plus rapidement et semblent de qualité semblable, hormis la tendance à ramener les couleurs vers le blanc.



(a) Entrée contenu (b) Résultat (c) Entrée style

Figure 1: Transfert de style

(a) Entrée contenu (b) Résultat (c) Entrée style

Figure 2: Transfert de style en choisissant les sorties qui semblent donner le meilleur résultat

3 Apprentissage d'un style - Réseau de convolution pour la transformation d'image

La deuxième approche étudiée est celle proposée par l'article *Perceptual Losses for Real-Time Style Transferand Super-Resolution* [2]. Cet article expose l'utilisation et l'optimisation de réseaux *feed-forward* pour répondre à deux problématiques de transfert d'image : l'augmentation de résolution (produire une reproduction 4 à 8 fois plus grande d'une image d'origine, en gardant la même résolution) et le transfert de style (produire une image semblable à une image d'entrée en lui appliquant le style d'une image de style).

La première partie du modèle de Johnson *et al.* consiste à implémenter un réseau de transformation d'image, qui réalise sur une image une transformation qu'il va falloir par la suite optimiser afin de faire correspondre cette transformation au style souhaité.

Nous avons codé ce réseau sous le nom de ImgTNet en Python, en utilisant la librairie PyTorch, qui nous fournit tous les outils nécessaires à cette implémentation. Son architecture repose sur plusieurs couches de convolution, ainsi que sur des "blocs résiduels de Gross et Wilber" [5].

L'architecture détaillée du réseau de transformation d'image est la suivante :

Couche	Fonction d'activation
Reflection Padding (40×40)	-
Convolution (9×9), stride 1	ReLU
Convolution (3×3), stride 2	ReLU
Convolution (3×3), stride 2	ReLU
Residual	-
Residual	-
Residual	-
Residual	-
Residual	-
Deconvolution (3×3), stride 2	ReLU
Deconvolution (3×3), stride 2	ReLU
Deconvolution (9×9), stride 1	Scaled tanh [0 ; 255]

Table 1 : Architecure du réseau ImgTNet

La fonction d'activation *Scaled tanh* permet de ramener les valeurs obtenues après la *forward-propagation* entre 0 et 255, afin de garder des valeurs correspondantes aux informations RGB du pixel.
Par ailleurs, le bloc résiduel peut être ramené à deux convolutions (3×3) successives [6] :

Couche	Fonction d'activation
Convolution (3×3), stride 1	ReLU
Convolution (3×3), stride 1	ReLU

Table 2 : Architecture d'un bloc *Residual*

Cette architecture pour un réseau de transfert de style modifie aléatoirement les pixels de l'image, et le résultat obtenu est pour l'instant loin de ce que l'on souhaite. Il faut alors entraîner ce réseau ; c'est ainsi que va intervenir le réseau de reconnaissance d'image VGG19 présenté précédemment.

4 L'*instance normalization* pour accélérer les calculs

Cet article propose des améliorations de la méthode précédente, que l'on va surnommer "stylisation rapide" à cause de sa vitesse élevée de convergence.

D'après nos résultats et ceux de l'article, la méthode de stylisation rapide de Johnson et al.(2016) permet d'obtenir un résultat beaucoup plus rapide mais moins bon que celle de Gatys et al.(2016).

Pour cette raison, Ulyanov et al. (2016) proposent une légère modification de la méthode de stylisation rapide pour améliorer grandement les résultats. L'article propose d'utiliser des normalisations de contraste plutôt que des normalisation de batchs. Cette amélioration permet au réseau d'être plus spécifique et donne de meilleurs résultats.

Nous avons donc implémenté cette légère modification : nous avons gardé la même architecture de réseau que celle utilisée précédemment à la différence près que

nous avons réalisé des normalisations pour chaque instance.

5 Utilisation de styles arbitraires

Comme énoncé dans les sections précédentes, les CNN sont utilisés pour le transfert de style. Des améliorations ont ensuite été apportées pour obtenir des résultats équivalents plus rapidement. Cependant, l'accélération de la vitesse pour obtenir les résultats a eu pour conséquence de rendre les réseaux de neurones utilisés spécifiques à certains styles. Cette méthode est donc peu généralisable et donne de mauvais résultats sur de nouveaux styles. Dans l'article *Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization* [4], Xun Huang et Serge Belongie ont présenté une nouvelle approche permettant d'obtenir des résultats satisfaisants, en un temps court et avec n'importe quel style.

Les premiers articles traitant du transfert de style utilisaient une couche de *batch normalization* après chaque couche de convolution. Il a été montré qu'utiliser des couches d'*instance normalization* améliorait grandement la qualité des résultats. Cette forme de normalisation dépend des dimensions spatiales. Avec la BN, les images d'un même *batch* sont normalisées de la même façon. L'IN normalise chaque image indépendemment, avec des paramètres affines. Cependant, bien que les résultats étaient meilleurs sur un même style, chaque réseau était spécifique à un style ou à un nombre fini et limité de styles.

L'objectif de ce papier est de proposer des paramètres qui pourraient s'adapter au style en entrée. Ils appellent cette méthode l'*adaptive instance normalization* (AdaIN) :

$$\text{AdaIN}(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \qquad (1)$$

Les paramètres $\sigma(y)$ et $\mu(y)$ sont appris des données et dépendent du style y en entrée.

Ci-dessous, on retrouve le principe de leur algorithme.

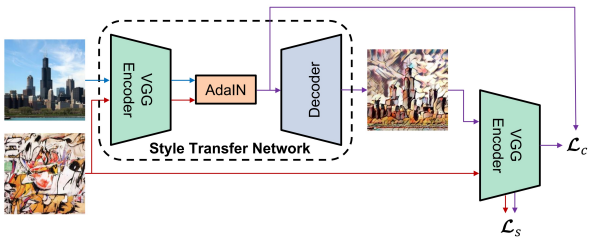


Figure 3: Principe de l'algorithme de transfert de style (image issue de l'article)

Il s'agit de donner en entrée à l'algorithme une image "contenu" et une image "style". Ces images sont données à un réseau pré-entraîné sur des images de la base de données ImageNet appelé VGG19. Ce réseau permet d'encoder les images d'entrée dans le bon espace de paramètres. La couche AdaIN permet d'aligner les moyennes et variances des images contenu et style. Le décodeur donne une sortie dans l'espace des images. Cette sortie est l'image stylisée voulue.

Comme dans les articles présentés dans les sections précédentes, l'objectif est de réduire la valeur de la fonction de perte entre l'image de sortie et les deux images en entrée. Intuitivement, il s'agit de trouver un compromis entre perdre du contenu, tout en se rapprochant du style demandé.

La méthode présentée dans cet article permet d'obtenir des résultats satisfaisants. Qualitativement, ils sont parfois moins bons qu'avec les méthodes des articles précédents. Cependant, il faut garder à l'esprit que le modèle n'a jamais vu le style donné en entrée pendant son entraînement. De plus, les temps de calculs sont équivalents aux meilleurs de la littérature, mais applicables sur un nombre infini de styles.

6 Résultats

Après notre travail de bibliographie, nous avions pour objectif d'implémenter ces modèles et de les entraîner pour comparer leurs performances. Nous avons réussi à implémenter les modèles, disponibles sur Github à l'adresse https://github.com/Achleole/projetML. Pour l'article [4], nous nous sommes servis d'une implémentation en PyTorch, disponible à l'adresse https://github.com/naoto0804/pytorch-AdaIN.

Pour entraîner les modèles, nous avons utilisé des images issus de l'API visualgenome.org. Les images de ce jeu de données étaient variées, mais notre GPU était trop limité pour utiliser des *batchs* de grande taille. Nous manquions de place pour stocker tous les gradients. Nous avons donc utilisé des *batchs* de taille 2 et obtenu une *loss* moyenne de 600. Pour avoir de bons résultats, nous aurions dû descendre sous une *loss* moyenne de 200 car 200 correspond a un résultat correct avec le premier modèle.

Nous n'avons donc pas obtenu de résultats avec les modèles nécessitant une phase d'entraînement mais ce travail nous a permis de réaliser une importante étude bibliographique sur le transfert de style, ainsi qu'une ébauche de code. Ce domaine de recherche est très récent et continue encore aujourd'hui de présenter des avancées majeures. Les dernières avancées permettent d'utiliser le transfert de style sur des vidéos [7].

7 Références

1. Leon A. Gatys, Alexander S. Ecker, Matthias Bethge : A Neural Algorithm of Artistic Style. (2015)
2. Justin Johnson, Alexandre Alahi, Li Fei-Fei : Perceptual Losses for Real-Time Style Transfer and Super-Resolution. (2016)
3. Dmitry Ulyanov, Andrea Vedaldi, Victor Lempitsky : Instance Normalization : The Missing In-gredient for Fast Stylization. (2016)
4. Xun Huang, Serge Belongie : Arbitrary Style Transfer in Real-time with Adaptive Instance Normalization. (2017)
5. Sam Gross, Michael Wilber : Training and investigating residual nets. (2016)
6. Justin Johnson, Alexandre Alahi, Li Fei-Fei : Perceptual Losses for Real-Time Style Transferand Super-Resolution: Supplementary Material. (2016)
7. Ondrej Jamriska *et al.* : Stylizing Video by Example. (2019)