

Projet Applicatif : Vision artificielle

Transfert de style

Simon Audrix
ENSC

Tarek Lammouchi
ENSEIRB

Quentin Lanneau
ENSEIRB

Gabriel Native-Fontaine
ENSC

Abstract

Durant ce projet, nous avions pour objectif d'explorer et d'implémenter des méthodologies de transfert de style. Nous nous sommes concentrés sur deux approches. La première utilise des réseaux préentraînés et constitue l'état de l'art dans le domaine. Pour la seconde, nous avons choisi de mettre en place une méthode non supervisée, Cycle GAN. Cette méthodologie donne également de très bons résultats et est l'une des pistes les plus prometteuses pour le transfert de style non supervisé. Une fois nos recherches effectuées, les architectures de base, ont été implémentées à l'aide de python et de la bibliothèque Tensorflow. Une fois ces deux algorithmes implémentés nous avons réalisé différentes expériences afin de tester leurs capacités et limites sur des problèmes de transfert de style. Nous avons également testé leurs capacités à réaliser des tâches différentes inspirées du transfert de style telles que la colorisation ou l'amélioration de qualité.

Ces algorithmes ont montré de très bonnes performances en transfert de style itératif. Cependant, qu'ils relèvent d'un apprentissage supervisé ou non, leurs capacités de généralisation à d'autres problèmes restent très limitées. En effet, nos deux implémentations ne permettent pas de répondre à des problèmes annexes au transfert de style.

1. Introduction

Le transfert de style est un domaine particulièrement exploré au cours de la dernière décennie. Certaines méthodes utilisant des réseaux de neurones comme celle proposée par Gatys [5] représentent aujourd'hui l'état de l'art dans le domaine. Cette méthode repose sur l'utilisation d'une fonction de coût "perceptuelle" (perceptual loss), fonction dont l'objectif est de minimiser à la fois la distance entre le style de l'image de style en entrée et l'image de sortie, mais également la distance entre le contenu sémantique de l'image à laquelle on souhaite appliquer le style et l'image de sortie.

En parallèle, les années 2000 ont vu apparaître des algorithmes totalement non supervisés n'utilisant pas de réseaux de neurones préentraînés. Frigo [4] propose une méthode de patch qui donne de bons résultats, notamment dans la reconstruction du contenu de l'image d'origine.

Notre travail s'est concentré sur l'étude de deux modèles de réseaux de neurones qui ont fait leurs preuves dans le domaine. Le modèle développé par Gatys est aujourd'hui encore considéré comme l'état de l'art dans le transfert de style. Nous avons cherché à étudier les différents éléments du modèle et principalement la fonction de coût en analysant comment ses termes influent sur les résultats.

Dans un premier temps, nous présentons les méthodes utilisées, les réseaux créés et leurs architectures. Dans un second temps, nous revenons sur les techniques d'entraînement mises en place ainsi que sur les données utilisées. Enfin, nous comparerons les résultats et nous discuterons sur les possibilités d'amélioration.

2. Matériel et méthodes

2.1. Neural style transfer (Gatys)

L'algorithme développé par Gatys en 2015 repose sur l'utilisation d'un réseau de neurones préentraîné à reconnaître des images. Dans l'article, les auteurs utilisent le modèle VGG-16 [7], préentraîné sur la base de donnée ImageNet [8], le projet ImageNet est une vaste base de données visuelles conçue pour être utilisée dans la recherche en reconnaissance d'image et tout particulièrement dans des applications de machine learning conçues pour la reconnaissance d'images. En 2016, plus de dix millions d'URL d'images ont été annotées à la main pour ImageNet indiquant quels objets sont représentés sur les images. La méthode proposée par Gatys ne consiste pas à réentraîner ce réseau sur un nouveau problème, mais plutôt à tirer profit du "savoir" acquis par le modèle déjà entraîné à la détection des caractéristiques dans les images.

Ces réseaux convolutionnels préentraînés à la classification contiennent donc des caractéristiques "toutes faites".

Ils sont capables d'extraire rapidement des informations pertinentes de nouvelles images. Les réseaux deviennent invariants au contexte, permettant ainsi de relever les informations perceptuelles d'images hors de leur jeu d'entraînement.

Les réseaux de convolution sont inspirés du fonctionnement biologique de la vision. Au sein du système visuel, chaque couche de neurones a accès à des caractéristiques de plus haut niveau que les couches précédentes. Ainsi, les premiers neurones détectent des primitives simples comme les courbes, les lignes droites orientées dans un sens spécifique, et en avançant dans les couches, les neurones détectent des primitives de plus "haut niveau" comme les formes, les couleurs...

Gaty se sert de ce postulat et utilise les caractéristiques issues du modèle VGG pour modifier de manière itérative une image et lui appliquer le style d'une autre image. Cette modification est faite par descente de gradient, en calculant un coût "perceptuel". L'objectif est donc de minimiser ce coût en modifiant l'image.

Cette fonction de coût cherche à la fois à minimiser la différence entre l'image modifiée et l'image de contenu et à minimiser la différence entre l'image modifiée et l'image de style. La fonction prend donc en compte deux éléments, un terme appelé "style loss" et un terme appelé "content loss". La "style loss" calcule la différence entre les styles de l'image en cours de modification et l'image de style. Pour ce faire, Gaty utilise les caractéristiques issues des premières couches du réseau de neurones. La "content loss" calcule la différence entre les contenus de l'image en cours de modification et l'image de style. Les auteurs utilisent cette fois les caractéristiques issues des dernières couches du réseau de neurones.

2.2. Cycle Gan

Le CycleGan, pour Cycle-Consistent GAN [9], est un réseau de neurones utilisant un réseau génératif adversarial (GAN). Un réseau génératif adversarial est un réseau constitué de deux sous réseaux [6]. Souvent utilisé dans le cadre d'algorithmes d'apprentissage non supervisé, un premier réseau générateur créé des candidats pour un second réseau discriminateur qui les évalue et détermine si un élément est "réel" ou généré par le générateur. L'entraînement consiste donc à entraîner le générateur à tromper le discriminateur et le discriminateur à ne pas être trompé. C'est un jeu adversarial.

Le CycleGan consiste à tirer profit du fonctionnement des GAN pour créer une fonction qui va mapper les images d'un style vers un autre style (et inversement). L'avantage du Cycle GAN est sa capacité à mapper de manière non supervisée n'importe quelle image de contenu avec n'importe quelle image de style des jeux d'entraînement.

L'algorithme du CycleGan consiste à entraîner en même

temps deux générateurs et deux discriminateurs. De manière croisé, l'un des générateurs apprend à générer des images d'un style en recevant des images de l'autre style et le second générateur fait l'inverse. Les discriminateurs continuent à évaluer la qualité des images générées. On parle de cohérence de cycle car l'algorithme est entraîné de sorte que l'image générée par l'un des générateurs entraîne la génération de son image d'origine lorsqu'elle est passée en entrée du second générateur.

Pour entraîner l'algorithme à conserver la cohérence de cycle, on ajoute un terme supplémentaire à la fonction de coût classique des réseaux génératifs adversariaux.

2.3. Protocole Expérimental

2.3.1 Perceptual loss (Gaty)



Figure 1. Image obtenue par l'algorithme initial après 5000 itérations

Les performances de l'algorithme de Gaty semblent complètement reposer sur les couches utilisées dans les deux composantes de la fonction de coût. Dans l'article, les auteurs utilisent les toutes premières couches du réseau dans le calcul de la "style loss" et une couche parmi les dernières du réseau dans la "content loss".

Nous allons donc expérimenter en suivant les résultats obtenus par l'algorithme lorsqu'on modifie les couches utilisées. Nous nous sommes également posé quelques questions:

- Que se passe-t-il si, en conservant la même image de style, on fourni à l'algorithme une image de bruit à la place du contenu ?
- Que se passe-t-il si on fourni une image de contenu en noir et blanc et une image de style légèrement similaire mais en couleur ?
- Que se passe-t-il si on fourni une image de contenu issue d'un film ancien et en style une image similaire issue d'un film récent ?

2.3.2 Cycle Gan

Pour effectuer des tests sur un modèle de Cycle Gan, nous avons tout d'abord essayé de réimplémenter l'algorithme par nous-mêmes, en nous inspirant de l'exemple tiré d'un livre [3]. Cette tentative s'étant soldée par un échec (le modèle superposant un filtre coloré par-dessus l'image), nous avons décidé de reprendre plusieurs versions sur internet pour mener à bien nos expérimentations [1] [2].

La loss du Cycle Gan est composée de plusieurs sous-loss: l’adversarial loss, mais également l’identity loss, sans oublier la cycle loss. Il pourrait être intéressant de faire varier les poids de ces différentes loss, et observer les répercussions sur les images produites.

Dans le modèle récupéré, les poids initiaux pour l’identity loss et la cycle loss étaient respectivement de 10 et 0.5. Nous avons effectué un premier test en inversant ces deux valeurs, puis nous avons testé, avec les valeurs de base, de multiplier l’adversarial loss par la somme des deux autres. L’idée sous-jacente était de signifier au réseau que cette adversarial loss était, quelque soit les poids donnés aux différentes loss, une mesure importante. Enfin, nous avons essayé, pour la même raison, de diviser la somme précédemment calculée par $(1 + \text{epsilon} - \text{adversarial loss})$, epsilon étant un réel positif assez petit, utilisé pour s’assurer que le dénominateur ne soit pas égal à 0. Tous ces modèles ont été entraînés sur 90 epochs, comme le modèle donné en exemple.

3. Résultats

3.1. Perceptual loss (Gaty)

Sur le premier algorithme créé, nous avons effectué différents tests afin de mieux comprendre son comportement, et les possibilités qu’offrait cette architecture.

Dans un premier temps, nous avons effectué une utilisation “classique” en transférant le style d’un tableau de Van Gogh (La nuit étoilée) sur une photographie de skyline comme présenté dans la partie **Protocole**.

Dans un second temps, nous avons également tenté de modifier les couches du réseau utilisées dans le calcul de l’une ou l’autre des loss. Enfin, nous avons testé des configurations avec différentes images pour tester les limites du modèle.

Modification des couches utilisées dans la loss

- Style premières couches, contenu dernières couches
- Contenu premières couches, style dernière couches

Lorsqu’on modifie les couches utilisées dans le calcul de la loss, on observe une forte diminution des performances du réseau qui ne semble plus capable d’appliquer le transfert de style sur des images.

Insertion de bruit à la place du contenu Lorsqu’on passe à l’algorithme une image de bruit (bruit gaussien) à la place d’une image de contenu, on observe que l’image résultante semble être constituée des éléments de base de l’image de style, ici sur la figure 2 on observe les formes plutôt arrondies de l’image de style utilisée.

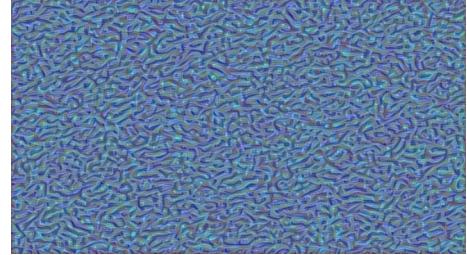


Figure 2. Résultat de l’algorithme après 5000 itérations



Figure 3. Résultat de l’algorithme après 5000 itérations

Insertion de bruit à la place du style Lorsqu’on passe à l’algorithme une image de bruit (bruit gaussien) à la place d’une image de style, comme on s’y attend, le contenu n’est pas vraiment altéré, en revanche, l’image est légèrement bruitée et semble avoir perdu ses caractéristiques colorées [figure 3].

Noir et blanc vers couleur Ici, les images ont été modifiées. Pour cette expérience, nous avons utilisé une image de film de guerre en couleur et en noir et blanc. Dans la mesure où l’algorithme n’est pas utilisé pour transformer des images en niveaux de gris en des images en couleur, nous avons choisi de commencer par une expérience simple en utilisant strictement la même image en contenu et en style (à la différence de l’échelle de couleur).

Ci-dessous on peut observer le résultat d’apprentissage de ce test après 5000 itérations, on observe bien une modification de l’image d’origine notamment proche des contours et des éléments saillants de l’image.

Cependant, ici les résultats ne sont pas aussi bons qu’on aurait pu l’espérer. Certains aspects de couleur semblent être transférés sur l’image mais plus sous une forme locale pixel à pixel. On peut conclure de ces tests que sans modifications, l’algorithme de Gatys n’est pas adapté à la colorisation d’image.

”Toonification” La figure 5 représente un essai amusant qui a particulièrement bien fonctionné à nos yeux. On peut observer que l’image de contenu a bien pris le style de l’image issue du jeu minecraft, notamment au niveau des arbres qui ont cet aspect cubique caractéristique.



Figure 4. Résultat de colorisation après 5000 epochs



Figure 5. Résultat de transfert de style après 5000 epochs

3.2. Cycle Gan

Modèle initial On présente ici les résultats obtenus grâce au modèle de base, fourni avec le code.

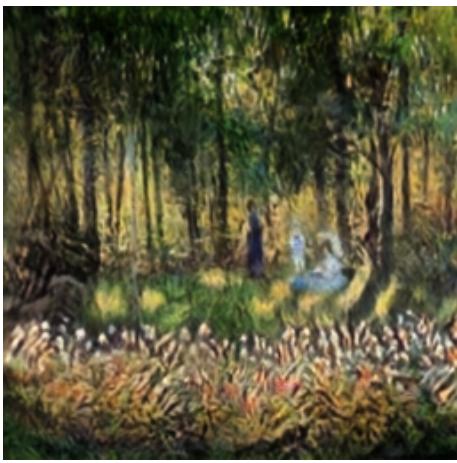


Figure 6. Résultats du modèle de base

La figure ci-dessus (6) servira de point de repère dans les tests suivants. On peut voir qu'elle combine tout à fait le style voulu, tout en conservant le contenu initial.

Inversion des deux poids initiaux L'inversion des deux poids donne des résultats corrects. Après les 90 epochs d'entraînement, on peut voir (7) qu'il parvient lui aussi à combiner le fond d'un espace, et la forme de l'autre. Cependant, le style transféré semble moins marqué, et on peut apercevoir, notamment à la droite de l'image, des distorsions de couleur, peut-être marquant un apprentissage non-complet.



Figure 7. Résultats du modèle aux poids inversés

Cependant, le style transféré semble moins marqué, et on peut apercevoir, notamment à droite de l'image, des distorsions de couleur, peut-être marquant un apprentissage non-complet. Ce modèle a sans doute besoin de quelques epochs de plus pour libérer son plein potentiel, et il reste plus proche des images d'entrée que ne l'était le modèle de base.

Cela n'est guère surprenant lorsqu'on prend en compte le fait qu'on l'a encouragé à bien reconstruire des images, cela l'a donc poussé à ne pas trop changer les images données en entrée, car la loss était trop pénalisante.

Multiplication des loss La multiplication des loss est censée pousser le réseau à équilibrer son apprentissage entre convaincre les discriminateurs, et améliorer ses images de sortie. En effet, à tout moment dans l'apprentissage, ces deux parties sont d'égale importance. Il en résulte que durant tout l'apprentissage, les réseaux discriminants ont eu une loss stable, à environ 0,25. Cela semble indiquer qu'ils arrivent à distinguer les images fausses une fois sur deux quand ils les croisent.

On pourrait potentiellement faire diminuer arbitrairement bas cette mesure en fixant un poids plus élevé à la loss issue de la discrimination.

Comme on peut le voir sur les images ci-dessus (fig 8 et 9), les images obtenues en sortie souffrent d'une instabilité marquée, et peuvent varier du tout au tout entre deux



Figure 8. Une image à l'epoch 88



Figure 9. Une image à l'epoch 89

epochs. Malgré la qualité plutôt bonne de ces images, ce calcul de loss est donc risqué.

Division des loss Travaillant sous Colab, je n'ai pas eu le temps d'entraîner ce dernier modèle avec le temps qui m'était attribué pour l'utilisation de GPU (l'utilisation de CPU étant hors de propos pour de telles quantités d'images). Cependant, on pourrait s'attendre à des résultats similaires au test précédent, car la loss au dénominateur serait inversement proportionnelle, au lieu de proportionnelle, ce qui ne change pas grand-chose.

3.3. Conclusion

4. Discussion

L'algorithme de Gaty présente le gros avantage de ne pas nécessiter d'apprentissage sur de grands jeux de données, en effet utilisant des poids de réseaux préentraînés, il peut transférer un style d'une image vers une autre image en peu de temps et avec des ressources limitées.

À l'inverse, le Cycle Gan nécessite une longue période d'entraînement, de plusieurs heures, allant jusqu'à plus d'une dizaine d'heures suivant l'architecture de la machine sur laquelle l'entraînement se déroule. Cependant, cette méthode a comme avantage de pouvoir transférer le style appris à une photo en dehors du set d'apprentissage, mais également de pouvoir faire la transformation inverse. Cela permet donc, une fois entraîné, une plus grande souplesse d'utilisation.

References

- [1] A. K. Nain. Cyclegan. <https://github.com/keras-team/keras-io/blob/master/examples/generative/cyclegan.py>, 2020. Last modified: 2020/08/12.
- [2] Brownlee J. How to develop a cycle-gan for image-to-image translation with keras. <https://machinelearningmastery.com/cyclegan-tutorial-with-keras/>, 2020. Last modified: 2020/09/11.
- [3] David Foster. *Generative Deep Learning*. O'Reilly Media Inc., Juin 2019.
- [4] Oriel Frigo, Neus Sabater, Julie Delon, and Pierre Hellier. Split and Match: Example-Based Adaptive Patch Sampling for Unsupervised Style Transfer. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 553–561, Las Vegas, NV, USA, June 2016. IEEE.
- [5] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. A Neural Algorithm of Artistic Style. *arXiv:1508.06576 [cs, q-bio]*, Sept. 2015. arXiv: 1508.06576.
- [6] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. *arXiv:1406.2661 [cs, stat]*, June 2014. arXiv: 1406.2661.
- [7] Karen Simonyan and Andrew Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition. *arXiv:1409.1556 [cs]*, Apr. 2015. arXiv: 1409.1556.
- [8] Stanford University Stanford Vision Lab. Imagenet. En ligne le 06/01/2021.
- [9] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. *arXiv:1703.10593 [cs]*, Aug. 2017. arXiv: 1703.10593.