Étude des algorithmes pour le Transfert de Style

Thomas ISLA
thomas.isla@student-cs.fr

Abstract

Le Transfert de Style est une tâche qui consiste à appliquer le style d'une image sur une seconde image, tout en conservant le contenu de cette dernière. Dans ce projet, on se concentre sur la production de peintures, semblables à celles d'artistes célèbres, à partir de photographies.
La première approche que l'on étudie est basée sur un réseau convolutif. Elle permet d'obtenir de très bons résultats rapidement sur un couple d'images. La seconde méthode s'appuie sur les réseaux génératifs afin d'apprendre un mapping entre le domaine de nos peintures et celui de nos photographies. Elle permet donc de travailler sur des jeux de données plus volumineux, mais ses performances sont moindres.

1. Introduction

Dans ce rapport, on présente les travaux effectués sur la thématique du transfert de style. De nos jours, il s'agit d'un sujet de recherche actif, car ses applications sont variées.
Dans la littérature, le premier article qui traite d'une architecture profonde pour réaliser cette tâche a été publié en 2015. Les auteurs, Gatys *et al*. [1], emploient notamment un réseau convolutif. Ces derniers sont principalement reconnus pour leurs performances dans les opérations de classification, de reconnaissance ou encore de segmentation.
Les travaux précédents ont connu un certain nombre d'extensions. Néanmoins, il faut attendre l'émergence des réseaux antagonistes génératifs pour voir apparaître des approches résolument différentes pour la tâche que l'on étudie. C'est le cas notamment de Zhu *et al*. [2] qui présentent, en 2017, l'algorithme *CycleGan*.

2. Le jeu de données

Dans le but de transposer le style d'une peinture à une photographie, on choisit d'utiliser le jeu de données *monet2photo*. Il a été conçu dans le cadre de la publication des recherches sur le *CycleGan*. Il dispose de 1072 peintures de Monet et de 6287 photographies de paysages pour son entraînement. Cependant, afin d'économiser du temps de calcul, on ne conserve que les 1100 premières. Il dispose également de 751 photographies et de 121 œuvres de l'artiste pour son évaluation. Au cours de nos expérimentations, on sera amené à utiliser d'autres ensembles de données. Notamment, dans la première partie de ce rapport, on va travailler à partir d'une peinture de Vangogh. Dans la seconde partie, on va considérer *ukiyoe2photo* et *vangogh2photo*, deux jeux de données similaires à *monet2photo*.
L'ensemble des liens est disponible sur ma page GitHub.

3. Approche I - Réseau Convolutif

En 2015, Gatys [1] introduisent un algorithme de transfert de style basé sur des réseaux convolutifs.
Ils proposent de considérer le transfert de style comme une tâche de transfert de texture. Ils s'appuient sur les méthodes non paramétriques préexistantes ainsi que sur les avancées des architectures profondes.

3.1. Description de la méthode

Dans l'article [1], Gatys *et al*. suggèrent d'utiliser les représentations produites par les couches convolutives d'un réseau de neurones. Ils utilisent un réseau VGG-19 préentrainé sur le jeu de données *ImageNet*. L'objectif est de récupérer, de manière indépendante, des informations sur le contenu d'une photographie $\vec{p}$ et le style d'une peinture $\vec{a}$, puis de les manipuler afin de générer une nouvelle image $\vec{x}$. Ainsi, la tâche de transfert de style se résume à un problème d'optimisation modélisé par la fonction de coût suivante :

$$L_{totale}(\vec{p}, \vec{a}, \vec{x}) = \beta L_{content}(\vec{p}, \vec{x}) + \alpha L_{style}(\vec{a}, \vec{x}) \quad (1)$$

Avec α et β, deux hyperparamètres qui contrôlent le niveau de ressemblance entre l'image générée et la peinture, respectivement la photographie.

3.1.1 Fonction de coût - Contenu

Le premier élément de la fonction de coût permet de quantifier la différence sémantique entre la photographie et l'image générée. Elle est calculée à partir des cartes d'activations produites lors de la propagation. On définit $F^l_{i,j}$ comme l'activation, dans la couche l, du filtre i à la

position j pour l'image générée. Respectivement, on a $P^l_{i,j}$ pour la photographie :

$$L_{content}(\vec{p}, \vec{x}) = \frac{1}{2} \sum_{i,j} (F^l_{i,j} - P^l_{i,j})^2 \quad (2)$$

Par descente de gradient, on modifie alors les pixels de l'image générée afin d'obtenir une réponse similaire à celle produite par la photographie. En effet, des cartes d'activations similaires signifient un contenu semblable.

3.1.2 Fonction de coût - Style

Le second élément de la fonction de coût permet de quantifier la différence de style entre la peinture et l'image générée. Pour cela, on calcule une matrice de Gram afin de capturer l'information textuelle encodée par les couches convolutives du réseau de neurones. En reprenant les notations précédentes, on a :

$$G^{layer}_{i,j} = \sum_k F^{layer}_{i,k} F^{layer}_{j,k} \quad (3)$$

Soit $G^{layer}_{i,j}$ la matrice de Gram de l'image générée et $A^{layer}_{i,j}$ celle de la peinture, on définit alors :

$$L_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l \frac{1}{4N_l M l} \sum_{i,j} (G^l_{i,j} - A^l_{i,j})^2 \quad (4)$$

3.2. Implémentation classique

Dans un premier temps, on cherche à reproduire les résultats présentés dans l'article de 2015 [1].
Les auteurs utilisent un réseau VGG-19 pré-entrainé issu de la bibliothèque *Caffe*. Pour nos travaux, on privilégie *pyTorch* car son usage s'est popularisé depuis la publication de l'article.
On respecte les choix de conception réalisés, c'est-à-dire, un optimiseur *L-BFGS*, la sortie de la deuxième couche convolutive du quatrième bloc du réseau pour $L_{content}$ et celle de la première couche convolutive de chaque bloc pour L_{style}.

3.2.1 Quelles différences entre *Caffe* et *PyTorch*

Il existe des différences selon la bibliothèque que l'on considère. Tout d'abord, *Caffe* propose une version de l'architecture dans laquelle les poids du réseau sont normalisés afin que l'activation moyenne des filtres soit égale à 1. De plus, chaque réseau nécessite un prétraitement particulier. Ce dernier impacte l'échelle de valeurs des pixels des images que l'on fournit aux réseaux. La figure 1 montre que le prétraitement *Caffe* permet de conserver des valeurs plus élevées d'intensité.

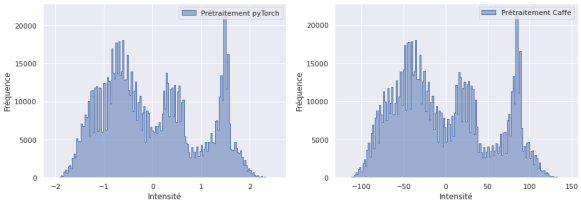

Figure 1. Comparaison de l'intensité des pixels de la photographie selon le prétraitement. À gauche, on utilise un prétraitement classique pour un réseau entrainé sur *pyTorch*. À droite, un autre spécifique au réseau *Caffe*.

3.2.2 Premiers résultats

La figure 2 présente les images générées. Le réseau *Caffe* produit un résultat très satisfaisant, contrairement au réseau *pyTorch*. L'étude des courbes d'apprentissages apporte une explication. Dans le second cas, les valeurs de la fonction de coût sont faibles. Ainsi, à chaque propagation arrière, les correctifs apportés sont minimes et l'apprentissage est lent : la situation s'apparente à un problème de gradient évanescent.
On met à l'épreuve deux correctifs. On entreprend d'augmenter les valeurs des hyperparamètres α et β (Figure 2, en bas à gauche). Puis, on applique le prétraitement *Caffe* aux images que l'on fournit au réseau *pyTorch* (Figure 2, en bas à droite). L'idée est d'augmenter la valeur des pixels des images manipulés par l'algorithme. Les résultats les plus

Figure 2. Résultats obtenus selon la méthode employée

attrayants sont obtenus grâce à la seconde approche proposée, bien qu'il soit généralement conseillé de conserver le même prétraitement que celui appliqué sur les données d'entraînement.
Enfin, malgré nos améliorations, le VGG-19 employé dans l'article [1] a les meilleures performances. On en conclut que la normalisation des poids du réseau, discuté dans l'article, présente un réel avantage pour la tâche de transfert de style.

3.3. Expérimentations

Dans cette partie, on présente les expérimentations que l'on a menées afin d'adapter le réseau à notre jeu de données. On considère uniquement le réseau VGG-19 issu de la bibliothèque *pyTorch*.

3.3.1 Étude de l'initialisation de l'image

Avant de démarrer l'entraînement de notre algorithme, on se pose la question de l'initialisation des pixels de l'image qui va être générée par le réseau. Plusieurs approches sont possibles, on en considère quatre : initialisation à partir de la peinture, initialisation à partir de la photographie, initialisation à partir d'un bruit blanc ou une somme pondérée entre les deux précédentes méthodes.
Théoriquement, on s'attend à ce que les deux premières méthodes produisent des résultats plus convaincants, car elles possèdent déjà une des composantes souhaitées, le contenu ou le style. Les deux dernières méthodes sont susceptibles de fournir des résultats plus intéressants, car elles sont moins contraintes. Néanmoins, pour ces dernières, il est plus difficile d'établir un compromis satisfaisant entre les hyperparamètres α et β.

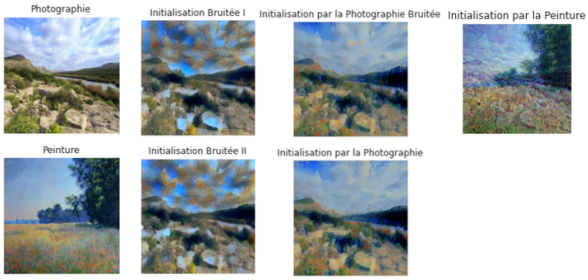

Figure 3. Résultats obtenus sur notre jeu de données selon l'initialisation de l'image

Selon l'approche choisit, la figure 3 illustre les différences observées. Il est intéressant de remarque que, même si les peintures de la deuxième colonne présentent des défauts, l'initialisation par un bruit blanc est la seule solution qui permet d'obtenir de la diversité dans les résultats. Enfin, dans le cas de nos données, la meilleure génération est obtenue avec l'initialisation par la photographie.

3.3.2 Étude de l'hyperparamètre α

Grâce à l'initialisation proposée précédemment, on s'assure de conserver la sémantique de la photographie au cours de l'entraînement. Avec cette garantie, on décide de fixer $\beta = 1$. Puis, on étudie plusieurs valeurs pour α car, c'est cet hyperparamètre qui va conditionner le niveau de ressemblance entre la peinture et l'image produite par le réseau.

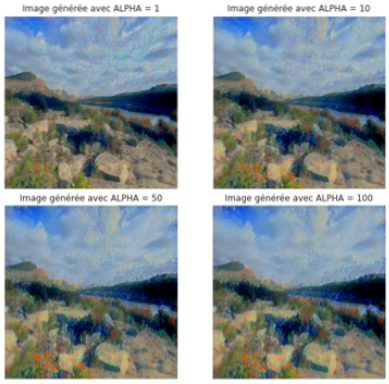

Figure 4. Image générée selon la valeur de l'hyperparamètre α. Des illustrations supplémentaires, pour des valeurs α plus élevée, sont disponibles dans le notebook.

La figure 4 atteste que, lorsque le rapport entre α et β augmente, l'aspect de l'image s'apparente aux œuvres de Monet. Avec des valeurs faibles pour α, les modifications concernent uniquement les tonalités de couleurs. Dès $\alpha = 50$, des éléments présents dans la peinture (les coquelicots) sont transposés au sein de l'image. Les lignes de contours se brouillent également afin de donner l'illusion d'un travail au pinceau. À partir de $\alpha > 100$, une limite semble être atteinte et les améliorations sont moins flagrantes. Ainsi, pour des valeurs supérieures, on devrait considérer une initialisation avec une proportion de bruit blanc afin de s'affranchir de certaines contraintes sémantiques pour augmenter le niveau de style de la génération.
Pour la suite des expérimentations, on choisit d'utiliser $\alpha = 100$ et $\beta = 1$ car ce choix produit le résultat le plus attrayant visuellement.

3.3.3 Étude des couches convolutives pour le style

Dans l'article de Gatys *et al.* [1], la représentation stylistique de la peinture est calculée à partir de la sortie de la première couche convolutive de chaque bloc du réseau VGG-19. Ils précisent que le nombre et la position des couches sélectionnées conduisent à des résultats différents, on décide donc de les faire varier. Parallèlement, pour comprendre l'information qui est encodée selon les couches considérées, on génère une image à partir d'un bruit blanc et sans contraintes de contenu $\beta = 0$.

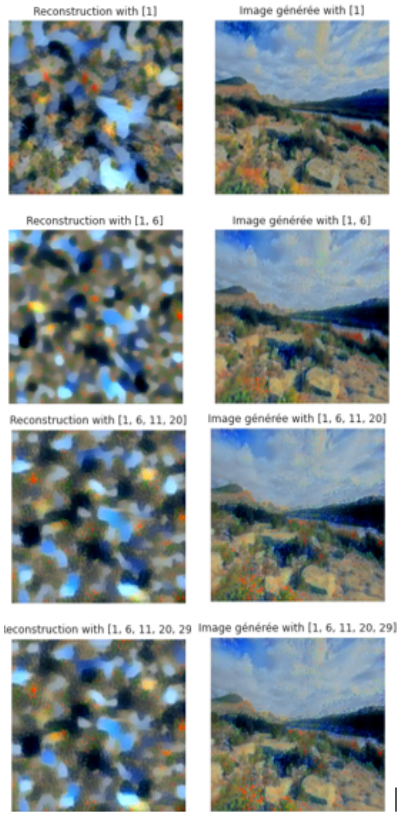

Figure 5. Information textuelle encodée (gauche) par le réseau et image générée (droite) selon les couches convolutives sélectionnées. Des illustrations supplémentaires, pour des combinaisons de couches différentes, sont disponibles dans le notebook.

Grâce à la figure 5, on remarque que la texture extraite dans les couches peu profondes possède une granularité plus fine en comparaison à celle des couches plus profondes. C'est un résultat qui nous paraît correct, car dans ces dernières, le champ récepteur des filtres est plus large et capture une information, ici une texture, globale.

3.3.4 Étude de l'architecture

Les auteurs de l'étude conseillent l'emploi d'un réseau de neurones VGG-19, mais ne motive pas leur choix. Dans un premier temps, on s'intéresse à des versions moins profondes de cette architecture. Les images générées (6) sont toutes satisfaisantes, ainsi on serait amené à privilégier une architecture plus légère, car moins gourmande en ressources de calculs.
Par la suite, on décide d'étudier d'autres architectures convolutives. En effet, de nouveaux algorithmes plus performants, ont émergés depuis la publication. Les images produites à l'aide de *ResNet* ou *Inception* sont identiques à la photographie. Ces mauvais résultats résultent potentiellement d'un mauvais réglage des hyperparamètres. Cependant, la littérature suggère que ces architectures sont moins adaptées pour le transfert de style. Elles sont capables de réaliser cette tâche après un nombre important de modifications et les performances sont très en deçà de celles d'un VGG. Les explications de ces phénomènes sont peu documentés, les articles avancent des hypothèses.

Figure 6. Image générée selon l'architecture employée.

3.4. Conclusion

Deux autres expérimentations, l'une sur le choix de l'optimiseur et l'autre sur les espaces de couleurs, ont étés menés. Par souci de concision, on ne présentera pas les conclusions dans ce rapport, mais l'ensemble de nos travaux est disponible sur ce GitHub. En guise de conclusion, on génère une dizaine d'images à partir de photographie et de peinture sélectionnées aléatoirement dans le jeu de données. Une nouvelle fois, les résultats sont disponibles sur le GitHub. On constate que les résultats sont très dépendants de ces dernières et que, finalement, il faudrait ajuster nos hyperparamètres pour chaque couple d'images. Il s'agit de l'un des inconvénients de cette méthode, elle n'est pas applicable à un jeu de données volumineux contrairement à la prochaine architecture étudiée.

4. Approche II - Réseau Antagoniste Génératif

En 2017, Zhu *et al.* [2] proposent une modification des réseaux génératifs existants pour répondre à la problématique de la traduction image à image avec des jeux de données non appariés. Soit X et Y deux domaines, leur objectif est de convertir une image du premier domaine vers le second, et inversement, sans correspondance préalablement établit entre eux. Pour ce faire, ils intègrent la notion de cycle à leur architecture. Cette méthode a de nombreuses applications, dont le transfert de style.

4.1. Description de la méthode

Soit un générateur G, responsable de produire des images à partir de données provenant de X, et un discriminateur D_Y. L'entraînement de ces deux réseaux est réalisé simultanément grâce à la fonction de coût classique des *GAN* :

$$L_{GAN}(G, D_Y, X, Y) = \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y G(x))] \quad (5)$$

L'objectif du générateur est de minimiser la valeur de cette expression. Pour cela, il doit produire des images similaires à celles du domaine Y. Inversement, le discriminateur veut la maximiser. Il doit donc faire la distinction entre les images réelles y et générées $G(x)$.
La première amélioration suggérée par Zhu *et al.* [2] est d'ajouter un second couple de réseaux adversaires afin d'apprendre la fonction inverse de G. On introduit donc le générateur F, le discriminateur D_X et la fonction de coût $L_{GAN}(F, D_X, Y, X)$ comme précédemment. Par la suite, ils définissent la fonction de coût suivante :

$$\begin{aligned} L_{GAN}(G, F, D_X, D_Y) &= L_{GAN}(G, D_Y, X, Y) \\ &+ L_{GAN}(F, D_X, Y, X) \\ &+ \lambda L_{cycle}(G, F) \end{aligned} \tag{6}$$

Le dernier terme de cette expression est au centre de l'architecture que l'on étudie, son importance est discuté dans la prochaine section.

4.1.1 Fonction de coût - Cycle

Jusqu'à présent, il n'y avait aucune contraine sur les fonctions apprises par les générateurs. De ce fait, il existe une multitude de transformation possibles entre les domaines X et Y. Le terme $L_{cycle}(G, F)$ introduit dans la fonction de coût 6 permet de restreindre le nombre de possibilités en imposant une condition forte. En effet, les transformations effectuées par les générateurs doivent être réversible. C'est-à-dire que, si l'on considère une image x appartenant au domaine X ainsi que la sortie $G(x)$ du générateur G alors $F(G(x)) \approx x$.

$$\begin{aligned} L_{cycle}(G, F) &= \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \end{aligned} \tag{7}$$

4.2. Implémentation classique

Dans un premier temps, on s'évertue à s'approcher des résultats présentés dans l'article de 2017 [2]. On reproduit les éléments mis en place par les auteurs pour favoriser l'apprentissage des réseaux : une initialisation des poids à partir d'une distribution gaussienne $\mathcal{N}(0, 0.02)$, un taux d'apprentissage linéairement décroissant et un buffer des précédentes images générées. On respecte également les valeurs des hyperparamètres.

4.2.1 Premiers résultats

Grâce à la figure 7, on s'aperçoit que les discriminateurs partagent la même dynamique. On s'attendait à ce que leur courbe se stabilise au cours de l'entraînement car, cela aurait signifié qu'ils rencontraient des difficultés pour différencier les fausses images des vraies. Ce phénomène n'est pas observable dans notre cas. Concernant les

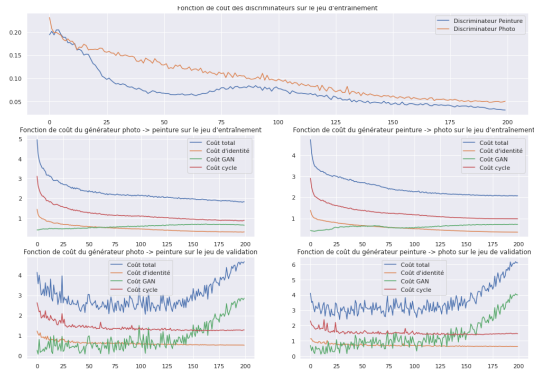

Figure 7. Courbes de l'apprentissage des réseaux sur les jeux d'entraînement et de validation pendant 200 epochs.

générateurs, notre principale préoccupation concerne la forte augmentation de L_{GAN} pour l'ensemble de validation. Cela suggère que les générateurs échouent à produire des résultats satisfaisants. Il est difficile de diagnostiquer avec précision les raisons de ces phénomènes, car les *GAN* sont difficiles à entraîner par nature. Néanmoins, on émet l'hypothèse que le volume réduit de notre jeu de données explique ces performances en demi-teintes.

Figure 8. Images générées à partir du jeu de validation au bout de 100 epochs. Les résultats au bout de 200 epochs sont disponibles sur le GitHub. La première ligne présente des photographies transformées en peintures et la seconde ligne, l'inverse.

Désormais, on s'intéresse aux images générées à partir du jeu de test. Les images produites au cours de l'apprentissage sont disponibles sur le GitHub. Les résultats sont très encourageants.

4.3. Expérimentations

L'architecture que l'on étudie est complexe, c'est pourquoi nos expérimentations ont vocation à la décomposer afin de comprendre l'impact de chacun de ses éléments. On réalise les prochains entraînements sur 100 epochs.

4.3.1 Étude du terme L_{cycle} dans la fonction de coût

La notion de cycle est primordiale au sein du réseau discuté. Afin de comprendre son rôle plus en détails, on réalise un entraînement où l'on retire ce terme de la fonction de coût.

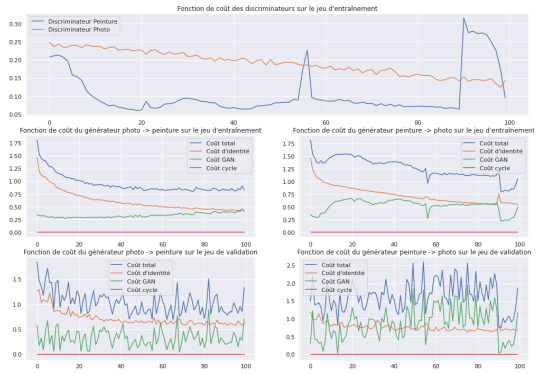

Figure 9. Courbes de l'apprentissage des réseaux sur les jeux d'entraînement et de validation pendant 100 epochs sans notion de cycle.

Le graphique 9 présentent des courbes très différentes de celles obtenus dans la section précédente, on remarque davantage d'oscillations notamment. Ainsi, sans la restriction imposée par la notion de cycle, il semble que l'apprentissage soit laborieux.

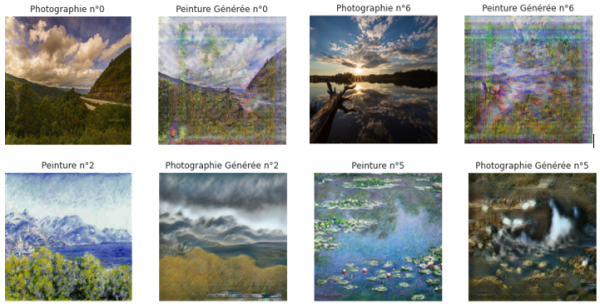

Figure 10. Images générées à partir du jeu de validation au bout de 100 epochs sans notion de cycle. La première ligne présente des photographies transformées en peintures et la seconde ligne, l'inverse.

L'observation des images en sorties des réseaux génératifs confirment l'impression conférée par les courbes d'apprentissages. Les transformations entre les domaines ne sont pas correctement apprises car pas assez restreindres. Les peintures générées sont les plus impactées, on note la présence de nombreux artefacts. Les photographies sont mieux dans l'ensemble mais leurs textures et leurs couleurs ne paraissent pas naturels.

4.3.2 Étude du terme $L_{identity}$ dans la fonction de coût

Dans le cadre de la tâche de transfert de style, les auteurs de l'article conseillent d'ajouter un terme d'identité à la fonction de coût 7. Par ce moyen, ils souhaitent encourager les générateurs à ne trop modifier les couleurs de l'image au cours de la propagation.On laisse de côter le graphique qui illustre les courbes d'apprentissages car, dans ce cas, elles ne révèlent aucun phénomène particulier. En revanche, on s'attarde sur les images produites par l'architecture.

Figure 11. Images générées à partir du jeu de validation au bout de 100 epochs sans terme d'identité. La première ligne présente des photographies transformées en peintures et la seconde ligne, l'inverse.

Les peintures produites sont parasités par une sorte de filtre de couleur vert. La qualité de ces dernières est fortement dégradée par rapport à l'implémentation classique. Les photographies générées semblent être moins impactées par la situation. Ainsi, on suppose que l'apprentissage de la fonction PHOTOGRAPHIES $\rightarrow$ PEINTURES est plus difficile que PEINTURES $\rightarrow$ PHOTOGRAPHIES. Il semblerait que, en l'absence du terme $L_{identity}$, les générateurs ont la possibilité de changer la couleur des images. Les résultats générées sont alors de moins bonne qualité.

4.3.3 Conclusion

Une dernière expérimentation a été menée à propos de le taille du buffer. Par souci de concision, les détails ne sont pas présentés dans ce rapport, mais sont disponibles sur mon GitHub. En guise de conclusion, on réalise deux autres entraînements à partir des peintures de deux autres artistes, Vangogh et Ukiyoe. On choisit ces artistes car, contrairement à Monet, leurs oeuvres sont moins réalistes et les résultats du transfert de style est plus évident. Cependant, les conclusions sont similaires pour l'ensemble des artistes, les résultats dépendent de l'image considérée. Certaines transformations sont plus évidentes tandis que pour d'autres, les générateurs échouent.

5. Conclusion

Pour conclure, on aurait aimé intégrer une métrique numérique afin de pouvoir quantifier les différences entre les stratégies employées.
Les deux algorithmes implémentés au cours de ce projets permettent de transformer des photographies en peintures. La première approche discutée privilégie la précision à la quantité. En effet, les travaux de Gatys [1] permettent de travailler sur un couple d'images uniquement. Cela permet de contrôler en détails l'image générée grâce aux réglages des hyperparamètres. La seconde approche fait le choix inverse. La méthode *CycleGAN* [2] permet d'apprendre les traductions entre deux domaines. On peut alors générér un nombre important d'images mais on a un contrôle très faible sur les actions des générateurs. Enfin, de manière globale, les résultats de la méthode convolutives sont plus impressionnants puisqu'elle arrive à donner l'illusion des coups de pinceau de l'artiste. Les réseaux génératifs n'atteignent pas ce niveau de réalisme.

References

[1] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2414–2423, 2016. 1, 2, 3, 7

[2] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017. 1, 4, 5, 7