

Transfert de style avec CycleGAN

Maxime GOURCEYRAUD , Eric HUARD

Polytechnique Montréal

{maxime.gourceyraud, eric-2.huard}@polymtl.ca

Abstract

Pour réaliser le transfert de style sans images appariées, nous avons utilisé l'architecture CycleGAN de [Zhu *et al.*, 2017]. Celle-ci est composée de deux GAN. Nous avons étudié l'impact des différentes composantes de la fonction de perte ainsi que différentes architectures pour les générateurs. Il semble que le générateur proposé initialement est plus efficace que les alternatives que nous avons étudiées. De même, nous avons confirmé l'utilité de la fonction de perte complète proposée par [Zhu *et al.*, 2017] face à des versions diminuées. Le code est disponible sur github.com/ehuard/CycleGAN.

1 Introduction

Le transfert de style est un problème d'apprentissage machine qui vise à modifier des images d'un "style" vers un autre. La transformation de tableaux du peintre impressionniste Claude Monet en photographies réalistes est un exemple célèbre de transfert de style. Plus formellement, à partir d'images d'un ensemble X , il faut générer des images qui seraient vraisemblables dans un ensemble Y tout en gardant la sémantique des images d'origine. Différentes techniques neuronales ont vu le jour à partir du milieu des années 2010 pour effectuer cette tâche, avec plus ou moins de succès [Jing *et al.*, 2018]. En 2017, [Zhu *et al.*, 2017] ont obtenus des résultats impressionnants et ont proposé CycleGAN, une architecture utilisant deux GAN entraînés conjointement et permettant d'effectuer le transfert de style entre deux domaines dans les deux sens, avec à la fois une bonne qualité d'image et un respect de la sémantique des images d'origine. Le but de ce projet est de reproduire ces résultats et de tester des variations du modèle présentées dans l'article de référence.

1.1 Travaux similaires

GAN Les Generative Adversarial Nets inventés par [Goodfellow *et al.*, 2014] permettent de générer des données à partir d'une distribution gaussienne. Pour cela, l'architecture est séparée en deux modèles. Le générateur $G : Z \mapsto X$ et un discriminateur $D : X \mapsto [0; 1]$. Le discriminateur s'entraîne en même temps que le générateur. Son rôle est de

faire la différence entre des données légitimes issues du jeu de données initial et les données produites par G .

Pix2pix et conditionnal GAN Le modèle sous-jacent à *pix2pix* proposé par [Isola *et al.*, 2016] réalise un transfert de style avec objectif, *i.e.* l'apprentissage est supervisé. Contrairement au problème décrit dans l'introduction, *pix2pix* avait accès à l'entrée dont il fallait transférer le style et le résultat attendu. Dans les problèmes abordés, on trouve par exemple le passage d'une image satellite d'une ville à une carte de Google Maps. Pour cela, [Isola *et al.*, 2016] ont utilisé des GANs conditionnels et non les GANs de [Goodfellow *et al.*, 2014]. Le générateur conditionnel $G : \{X, Z\} \mapsto Y$ utilise une image à transformer de X et le bruit Z gaussien pour générer une image dans le style de Y . De la même manière, le discriminateur conditionnel $D : \{X, Y\} \mapsto [0; 1]$ compare une image du style de base X avec l'image du style Y pour vérifier si Y est la vraie paire de X . Ces modèles sont dits conditionnels car les sorties de G et D dépendent de l'image de X qui impose une condition.

CoGAN L'architecture des Coupled Generative Adversarial Nets (CoGAN) utilise une paire de GAN pour générer des données fortement corrélées [Liu and Tuzel, 2016]. Les générateurs G_1 et G_2 prennent la même entrée gaussienne pour générer des représentations fortement corrélées. Pour augmenter la corrélation entre les sorties des générateurs et les réponses des discriminateurs, certaines couches sont partagées. La coexistence de deux GAN pose les bases de l'architecture CycleGAN qui sera décrite dans la section suivante.

VAE Les auto-encodeurs variationnels [Kingma and Welling, 2013] permettent une modélisation de la distribution de l'espace latent d'un auto-encodeur. On peut ensuite générer des données en échantillonnant suivant la distribution déterminée et en décodant. Cette architecture a été utilisée par [Liu *et al.*, 2017] pour réaliser une tâche de traduction non supervisée d'images *i.e.* la même que [Zhu *et al.*, 2017]. Le VAE utilisé de [Liu *et al.*, 2017] considère qu'il existe un espace latent Z qui encode les images de X et Y . Il y a donc deux encodeurs $E_1 : X \mapsto Z$ et $E_2 : Y \mapsto Z$. De la représentation latente, on peut ensuite générer une image de X ou Y via un générateur/décodeur $G_1 : Z \mapsto X$ et $G_2 : Z \mapsto Y$. Pour entraîner le modèle et assurer que les générateurs créent des images vraisemblables, [Isola *et al.*,

2016] ont ajouté des discriminateurs $D_1 : X \mapsto [0; 1]$ et $D_2 \mapsto [0; 1]$. On a donc une architecture VAE-GAN pour accomplir la tâche de traduction d'images non appariées.

2 Présentation du modèle

2.1 Architecture

Un cycleGAN est composé de deux générateurs et de deux discriminateurs. Un premier générateur G transforme des images du domaine X vers le domaine Y (il prend par exemple une image de cheval et la transforme en image de zèbre) alors que le second générateur F transforme des images du domaine Y vers le domaine X . Le discriminateur D_X (respectivement D_Y) est nourri d'images réelles du domaine X (resp. Y) et d'images générées par F (resp. G) et doit déterminer pour chaque image s'il celle-ci appartient au domaine X (resp. Y).

Les discriminateurs et les générateurs sont tous des modèles *fully convolutional*, c'est à dire à que ce sont des architectures qui extraient des informations uniquement à l'aide de couches de convolutions. Le discriminateur (voir fig.1) est un PatchGAN tel que présenté par [Isola *et al.*, 2016]. Une image de dimensions 256 par 256 pixels sur 3 couches est passée à travers différentes couches de convolution avec fonction d'activation LeakyReLU jusqu'à atteindre une carte 30 par 30 sur une seule couche. La dernière fonction d'activation étant une sigmoïde, le discriminateur classe chaque zone de l'image comme étant une vraie partie d'image de l'espace (la valeur de sortie sur ce patch est alors proche de 1) ou une image n'appartenant pas à cet espace. Cette architecture de discriminateur permet de localiser les zones à pénaliser dans les images générées.

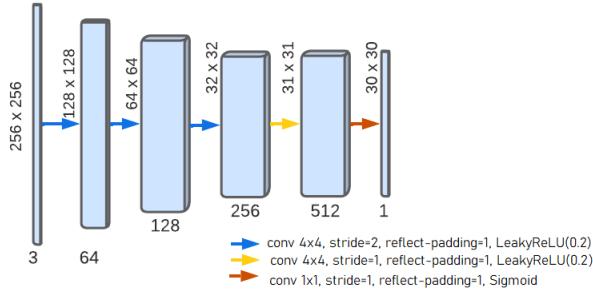


Figure 1: Architecture des discriminateurs dans CycleGAN

Les générateurs sont dérivés de l'architecture U-Net proposée par [Ronneberger *et al.*, 2015]. L'architecture de base présentée par [Zhu *et al.*, 2017] pour les CycleGAN (voir 2) est constituée tout d'abord d'un chemin contractant avec uniquement des convolutions, chaque convolution divisant la taille des côtés de l'image par 2 et doublant le nombre de couches. S'en suit un *bottleneck* composé de 9 blocs résiduels, chacun contenant deux couches de convolution 3 par 3 avec une seule d'entre elles ayant une fonction d'activation ReLU. Le chemin expansif est constitué de convolutions transposées pour augmenter la taille des

cartes. La dernière couche est une simple convolution 1 par 1 pour obtenir 3 canaux en sortie, un pour chaque couleur de l'image.

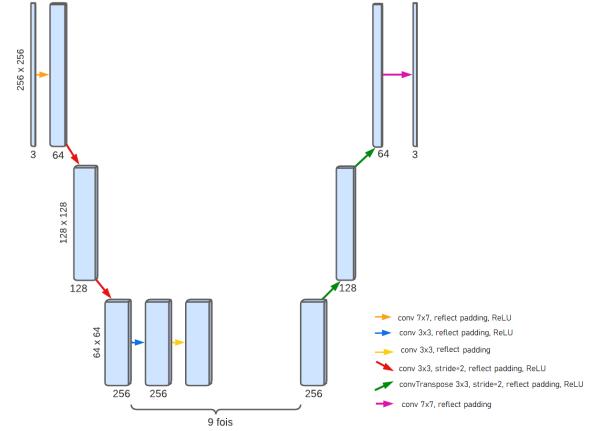


Figure 2: Architecture initiale des générateurs dans CycleGAN

Nous avons également utilisé des générateurs davantage similaires aux U-Net. Les implémentations de [Zhu *et al.*, 2017] contiennent également un générateur U-Net, mais ses résultats ne sont pas présentés dans l'article et le code est difficilement compréhensible. Nous proposons donc deux nouvelles architectures : la première correspond au modèle de base avec l'ajout de skip connections (ce qui semble avoir été implémenté par les auteurs de l'article) et est présentée en figure 3. La seconde correspond à une adaptation du modèle de [Ronneberger *et al.*, 2015], avec des opérations de *max pooling* pour réduire les dimensions et un nombre supérieur de convolutions à chaque étage. Cette architecture est présentée en figure 5.

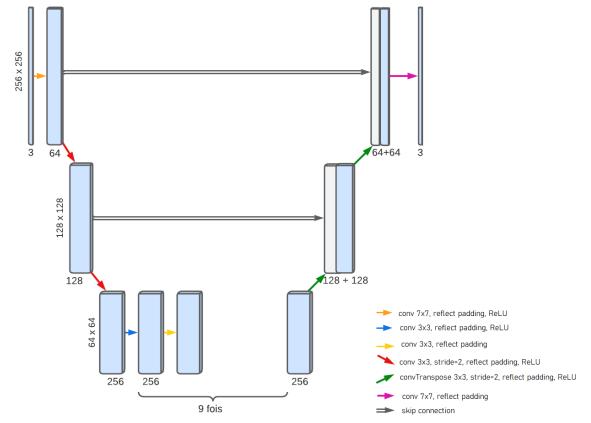


Figure 3: Générateurs de base avec l'ajout de skip-connections

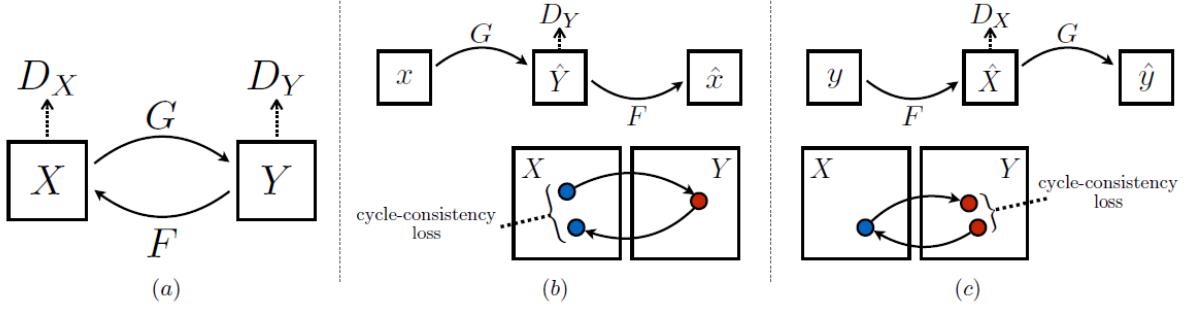


Figure 4: Schéma global de Cycle GAN issu de [Zhu *et al.*, 2017]

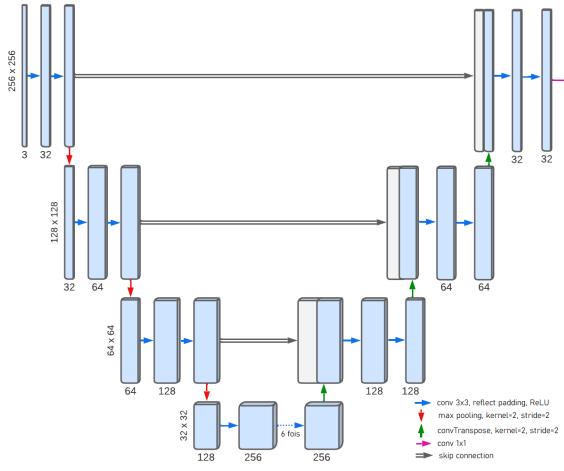


Figure 5: Architecture des générateurs U-Net

2.2 Fonction de perte

Perte GAN

Pour entraîner les GAN, [Zhu *et al.*, 2017] ont d'abord utilisé la fonction de perte classique de [Goodfellow *et al.*, 2014] :

$$\begin{aligned} \mathcal{L}_{GAN}(G, D_Y, X, Y) = & \mathbb{E}_{y \sim p_{data}(y)}[\log D_Y(y)] + \\ & \mathbb{E}_{x \sim p_{data}(x)}[\log(1 - D_Y(G(x)))] \end{aligned} \quad (1)$$

La première partie de la perte entraîne D_Y à reconnaître les vraies données avec une entropie croisée. À l'inverse la seconde partie entraîne le générateur G à tromper le discriminateur D_Y et ce dernier à reconnaître les fausses images. Dans la pratique de l'entraînement, [Zhu *et al.*, 2017] ont remplacé le \log par $x \mapsto x^2$. Cela transforme l'entropie croisée en *mean squared error*. Nous allons donc faire de même et la perte devient :

$$\begin{aligned} \mathcal{L}_{GAN}(G, D_Y, X, Y) = & \mathbb{E}_{y \sim p_{data}(y)}[D_Y(y)^2] + \\ & \mathbb{E}_{x \sim p_{data}(x)}[(1 - D_Y(G(x)))^2] \end{aligned} \quad (2)$$

Perte de cycle

Pour assurer la cohérence de l'architecture à deux GAN, [Zhu *et al.*, 2017] ont ajouté une perte de cycle. Cela correspond à la figure 4 sous-figures b) et c). Cette perte impose aux générateurs G et F d'être en bijection. En effet, pour $x \in X$ et $y \in Y$ on doit avoir $F(G(x)) \simeq x$ et $G(F(y)) \simeq y$. La fonction de perte s'écrit :

$$\begin{aligned} \mathcal{L}_{cycle}(G, F) = & \mathbb{E}_{x \sim p_{data}(x)}[\|F(G(x)) - x\|_1] + \\ & \mathbb{E}_{y \sim p_{data}(y)}[\|G(F(y)) - y\|_1] \end{aligned} \quad (3)$$

Perte identité

Pour le transfert de style entre des photographies et des peintures, [Zhu *et al.*, 2017] ont eu besoin d'ajouter une perte dite identité de [Taigman *et al.*, 2016] pour préserver les couleurs. En effet, sans cette perte, les générateurs pouvaient fortement changer les teintes des images. Cette fonction de perte encourage les générateurs à être des fonctions identités s'ils prennent en entrée une image issue de leur objectif. Par exemple, pour $y \in Y$, on veut $G(y) = y$. La perte s'écrit comme suit :

$$\begin{aligned} \mathcal{L}_{id}(G, F) = & \mathbb{E}_{y \sim p_{data}(y)}[\|G(y) - y\|_1] + \\ & \mathbb{E}_{x \sim p_{data}(x)}[\|F(x) - x\|_1] \end{aligned} \quad (4)$$

Perte totale

Soient $\lambda_{cycle}, \lambda_{id} \in \mathbb{R}_+$ des coefficients qui mesurent l'importance qu'on donne à la qualité de reconstruction et à la conservation des tons dans une image. Dans le cas où on n'utilise pas la fonction de perte identité, $\lambda_{id} = 0$. De manière générale, nous utilisons les coefficients préconisés par [Zhu *et al.*, 2017]: $\lambda_{id} = 5$ et $\lambda_{cycle} = 10$. La fonction de perte avec laquelle CycleGAN est entraîné vaut :

$$\begin{aligned} \mathcal{L}(G, F, D_X, D_Y, X, Y) = & \mathcal{L}_{GAN}(G, D_Y, X, Y) + \\ & \mathcal{L}_{GAN}(F, D_X, X, Y) + \\ & \lambda_{cycle}\mathcal{L}_{cycle}(G, F) + \\ & \lambda_{id}\mathcal{L}(G, F) \end{aligned} \quad (5)$$

Le problème à résoudre est un problème de mini-max comme pour un GAN [Goodfellow *et al.*, 2014] :

$$G^*, F^* = \arg \min_{G, F} \max_{D_X, D_Y} \mathcal{L}(G, F, D_X, D_Y, X, Y) \quad (6)$$

3 Méthodologie

Notre étude du modèle CycleGAN porte sur les jeux de données comprenant le transfert de style entre zèbres et chevaux et les données Monet-photographies. Ces jeux de données sont disponibles sur ce **site internet** de UC Berkley. Le jeu d’images Monet-photographies nous sert seulement lors de l’étude d’impact de la perte identité. Nous n’utilisons pas la perte identité sur le jeu de données zèbre-cheval car elle est utilisée sur les problèmes avec peinture dans [Zhu *et al.*, 2017]. Le reste des expérimentations se font sur le jeu de données zèbres-chevaux.

Nous étudions d’abord l’impact des fonctions de pertes sur la qualité des modèles. L’architecture utilisée est la même que celle dans l’article [Zhu *et al.*, 2017] (cf figure 2). Les fonctions de pertes sont les suivantes :

- \mathcal{L}
- \mathcal{L} sans \mathcal{L}_{cycle}
- \mathcal{L} sans $\mathbb{E}_{z \sim p_{data}(z\text{èbre})} [\|G(F(z))\|_1]$
- \mathcal{L} sans \mathcal{L}_{id} sur le jeu de données Monet-photographie

Nous étudions aussi l’impact de l’architecture des générateurs sur les performances :

- générateur de base (cf figure 2)
- générateur de base avec skip-connections (cf figure 3)
- générateur U-net (cf figure 5)

Pour comparer les performances des différents modèles mentionnés ci-dessus, nous allons utiliser le score FID. Ce dernier a été introduit par [Heusel *et al.*, 2018] pour mesurer quantitativement les performances des GAN. De plus, nous observeront qualitativement la qualité des images.

Pour l’entraînement, nous avons utilisé les cartes graphiques disponibles sur Colab pro ainsi que la Nvidia RTX3060 6GB de RAM de l’un de nos ordinateurs personnels.

4 Résultats

L’entraînement des modèles a été réalisé sur 120 epochs pour le problème zèbre-cheval et sur 20 epochs pour Monet-photographie. Cela correspond environ à 8h de calculs. Le premier jeu de données était plus petit (environ 1000 images de chaque classe) que le second (environ 6000 photographies), ce qui nous a poussé à augmenter le nombre d’epochs pour obtenir une durée d’entraînement équivalente.

4.1 Analyse quantitative - FID

Le score FID (Fréchet inception distance) mesure la distance entre les distributions de features d’images réelles et générées et est souvent utilisé pour mesurer la qualité de génération des GAN depuis [Heusel *et al.*, 2018]. Les scores FID obtenus à la fin des entraînements se trouvent dans le tableau 1. Les colonnes représentent les jeux de données à comparer. Par exemple, la colonne ”faux zèbres/zèbres” correspond aux images générées de zèbres à partir de chevaux et comparées avec les images de zèbres. La colonne ”faux zèbres/chevaux”

Modèle	faux zèbres/ zèbres	faux zèbres/ chevaux	faux chevaux/ chevaux	chevaux/ zèbres
Jeu de données	244.52	102.75	223.15	32.96
Base	141.88	157.86	162.21	246.38
Base avec skip-connections	373.37	170.85	211.01	246.38
Unet	239.40	161.27	193.78	51.63
Sans la perte cycle zèbre	369.38	372.07	269.11	314.04
Sans perte cycle	219.25	338.63	229.86	324.25

Table 1: Tableau des scores FID des fausses images comparées aux jeux de données cibles selon le modèle

représente la distance entre les images générées de zèbres et les images de chevaux qui ont permis de les générer. Les lignes représentent les différents modèles comme décrits dans la section 3 Méthodologie. La ligne ”Jeu de données” correspond à la comparaison entre le jeu de données de départ en test (d’où proviennent les générations) et le jeu de données cible d’entraînement (avec lequel on calcule les distances des colonnes 1 et 3). Par exemple le FID score de ”faux zèbres/zèbres” à la ligne ”Jeu de données” est la comparaison entre les chevaux de test (qui serviront à générer des zèbres) et les zèbres d’entraînement. Ceci nous donne une base pour comparer les scores. Le but est d’atteindre des distances les plus petites possibles sur les colonnes 1 et 3. On ne s’attend pas à faire mieux que les nombres présentés dans les colonnes 2 et 4 de la première ligne qui présentent donc des distances entre 2 jeux de données de photos de chevaux (colonne 2) et entre 2 jeux de données de photos zèbres (colonne 4).

Nous remarquons sur la table 1 que seul le modèle de base parvient à réellement diminuer le FID score entre les images générées et les images de référence des espaces objectifs. Le FID entre les générations de zèbres et les vrais zèbres (141.88) est assez proche de ce qu’on peut attendre en comparant deux ensembles d’images de zèbres (102.75). Le FID est même plus faible entre les images générées et les vrais zèbres qu’entre les images générées et les images d’origine (157.86). Si on se fie uniquement au FID, il semblerait que la génération de zèbres donne des résultats cohérents. La génération de chevaux est quant à elle assez mauvaise avec un FID de 162.21 entre les chevaux générés et de vrais chevaux.

Les autres modèles ne permettent pas d’obtenir des résultats décents. On observe une légère amélioration du FID score pour la génération de chevaux avec U-Net et pour la génération de zèbres sans perte de cycle, mais les différences restent élevées. Le générateur U-Net semble ne changer que légèrement les images qui lui sont données (les FID sont plus faibles dans les colonnes 2 et 4 que dans les colonnes 1 et 3). Enfin, remarquons des nombres supérieurs à 250: certains modèles se sont même éloignés de ce qu’ils devaient générer. Par exemple, les chevaux d’origine sont plus proches de zèbres que les zèbres générés par le modèle de base avec skip-connections.

4.2 Analyse qualitative

L’analyse quantitative avec FID donne une indication la qualité des résultats, mais il est nécessaire de regarder les images générées pour savoir si la tâche est réalisée correctement.



Figure 6: Comparaison de générations de zèbres et de chevaux avec différentes variations du modèle. Les images d'origine à transformer sont présentées à la première ligne.



Figure 7: Comparaison de générations de photos et de tableaux de Monet. Les images d'origine à transformer sont présentées à la première ligne.

Pour synthétiser nos résultats, nous avons sélectionné trois images de zèbres, trois images de chevaux, trois peintures de Monet et trois photographies. Nous avons ensuite réalisé le transfert de style. Les images sont rassemblées dans la figure 6 pour le problème zèbre-cheval et en figure 7 pour l'impact de la perte identité sur le problème Monet-photographie. Les images d'origine sont situés en première ligne de chaque figure.

Sans la perte de cycle zèbre La deuxième ligne de la figure 6 correspond modèle auquel on a retiré la perte de cycle des zèbres, *i.e.* on ne vérifie pas que la reconstruction d'un zèbre à partir d'une fausse image de cheval donne bien le zèbre de départ. On remarque que ce modèle a des difficultés à générer des zèbres à partir de chevaux. Dans ce cas, nous avons dû utiliser les images issues de l'epoch 61 car les images suivantes étaient noires. Pour ce qui est de la génération de chevaux à partir de zèbres, on retrouve bien la texture marron mais les formes ne sont pas conservées. Ceci correspond bien au fait qu'une perte de cycle manque et donc qu'on demande moins au modèle de conserver les formes.

Sans la perte de cycle La troisième ligne présente les résultats d'un modèle entraîné sans la perte de cycle. On remarque que les formes sont présentes mais les couleurs ne correspondent pas à ce qui est demandé.

Base La quatrième ligne affiche les résultats du modèle avec le générateur de base ainsi que la perte complète. C'est clairement le modèle avec les meilleurs résultats. La première et la troisième image sont plutôt réussies à quelques artefacts visuels près. La deuxième image ne correspond pas vraiment à ce qui était attendu, il semblerait que le modèle ait plus de mal à effectuer la transformation sur de sujets éloignés.

La quatrième image est réussie. Dans la cinquième les zones plus complexes comme le tour du rétroviseur et les babines du zèbre ne sont pas transformées. La sixième image est assez complexe puisqu'il y a plusieurs individus et les transformations ne sont pas réalisées.

Base avec skip-connections La cinquième ligne correspond à des images traitées par le modèle de base avec un générateur à skip-connections et la perte complète. Les transformations dans les deux sens ne sont pas réussies. Les formes sont conservés mais les couleurs ne correspondent pas du tout. De plus, très peu de zébrures ont été appliquées aux chevaux pour les transformer en zèbres. A l'inverse, les zèbres ont gardé une partie importante de leurs zébrures.

Unet La sixième et dernière ligne correspond aux résultats du modèle ayant un générateur Unet et la fonction de perte complète. Les formes et les paysages sont globalement respectés. La même analyse que pour le modèle précédent s'applique: les formes sont respectées mais les couleurs sont incohérentes. Le transfert de style semble encore moins présent.

Finalement, il semble que pour l'oeil humain, le modèle proposé par [Zhu *et al.*, 2017] avec la perte complète donne les meilleurs résultats. En effet, il surpasse toutes les alternatives que nous avons essayées à la fois en termes de couleurs, de formes et de réussite du transfert de style. Nous comprenons alors pourquoi ils ont évoqué ces modèles alternatifs sans présenter leurs résultats qui étaient tout simplement insuffisants. Il est tout de même plus étonnant de voir les mauvais résultats avec le modèle de base et les skip-connections ou le avec le générateur U-Net. Nous pourrons tester par la suite l'implémentation du générateur U-Net des auteurs de

l'article en clonant leur *repository* Github, mais ceci n'entre pas dans le cadre de ce projet.

La figure 8 montre un cycle de transfert de style. On remarque que le résultat est satisfaisant. En effet, le cheval reconstruit à partir de son pendant zèbre a la bonne couleur et la bonne forme. Cependant, son pelage n'est pas tout à fait net : le transfert de style s'applique correctement tout en gardant la sémantique de l'image, ce qui permet de la reconstruire, mais la qualité des images générées n'est pas celle d'une photo réaliste.



Figure 8: Exemple de cycle cheval → zèbre puis zèbre → cheval

La figure 7 présente nos résultats sur le jeu de données Monet-photographie. Nous avons observé l'influence de la perte identité comparée au modèle avec la perte complète. Le générateur utilisé est celui proposé par [Zhu *et al.*, 2017].

Sans perte identité La deuxième ligne présente des résultats du modèle entraîné sans perte identité. Le transfert de Monet vers photographie est raté. En effet, les images ne ressemblent pas à des photographies. On observe tout de même une amélioration de la netteté mais ce n'est pas suffisant. Les couleurs et les formes sont conservées. Pour le transfert de photographie vers Monet, le résultat est insuffisant. Quelques éléments qu'on peut trouver sur des peintures de Monet se trouvent dans les images générées, mais elles ressemblent encore à des photos (surtout sur des photos avec peu de ciel ou de mer qui ne sont pas présentées ici). Il existe aussi des artefacts qui pourraient peut être disparaître avec un entraînement plus long.

Base Des résultats du modèle de base sont présentés dans la troisième ligne. On remarque que les résultats sont comparables à ceux de la deuxième ligne. Nous remarquons une plus grande présence d'artefacts, mais ceux si sont en réalité réapparus à la dernière epoch dont on présente les images. Les résultats sont très similaires sur l'ensemble des 20 epochs.

Finalement, le transfert de style entre tableaux de Monet et photographies a donné des résultats assez décevants par rapport à ce qui est présenté dans l'article. Les tableaux de Monet ne sont pas transformés en photos, et les photos ne sont que légèrement modifiées pour ressembler à des peintures. Nous avions initialement choisi ce jeu de données pour mener notre étude, mais l'absence de résultats probants nous a forcé à nous reporter sur les images de zèbres et de chevaux. Il est possible que l'absence de résultats soient due à un entraînement trop court, mais les générations ne changeait que très peu après 5 epochs. Notons tout de même que l'utilisation de la perte d'identité ne semble pas nécessaire

même lorsque nous travaillons avec des photographies: nous avons remarqué que les couleurs pouvaient ne pas être respectées dans les premières epochs, mais que le générateur apprenait rapidement à conserver le ton de l'image initiale même sans perte d'identité.

5 Limites et problèmes rencontrés de notre étude

Au cours de la réalisaiton de notre projet, nous nous sommes rendus compte que la qualité des images générées atteignait rapidement un plateau, après environ une à deux heures d'apprentissage sur huit heures environ au total. En regardant les courbes de pertes, par exemple en figure 9, nous remarquons que le discriminateur parvient à atteindre une perte près de 0 alors que la perte des générateurs augmente légèrement. En même temps, la perte de cycle d'un générateur continue de diminuer: il semble qu'il soit plus facile pour les générateurs d'apprendre à générer des images qui permettent une bonne reconstruction que des images qui peuvent tromper le discriminateur. Il se pose alors la question de la valeur à donner à λ_{cycle} : des valeurs plus petites permettraient d'accorder plus d'importance au fait de tromper le discriminateur. Après de rapides tests, des valeurs plus petites semblaient donner des résultats encore moins bons. Nous avons donc gardé les coefficients proposés par [Zhu *et al.*, 2017] comme nous sommes dans l'impossibilité de mener une recherche intensive des hyperparamètres.

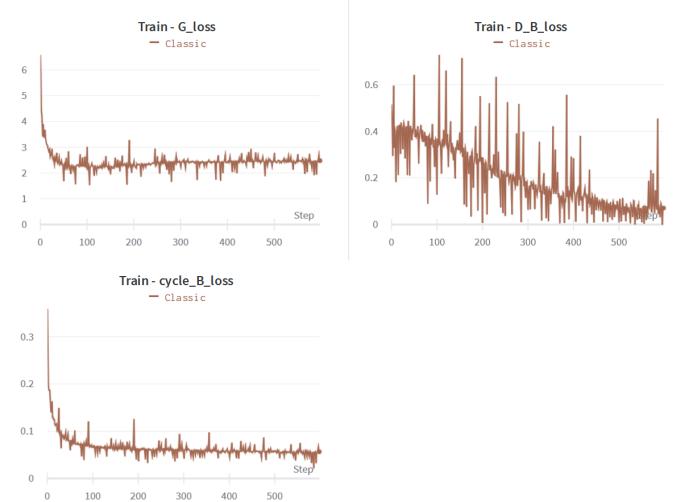


Figure 9: Fonctions de perte lors d'un entraînement

Une autre piste d'amélioration serait une meilleure gestion du taux d'apprentissage. Nous avons testé plusieurs valeurs fixes différentes, que ce soit pour les générateurs ou les discriminateurs, sans voir de grande amélioration. Des taux d'apprentissages variables pourraient perfectionner l'entraînement. Un autre problème est que les discriminateurs apprennent beaucoup plus rapidement que les générateurs. Il faudrait donc réduire en conséquence le taux d'apprentissage des discriminateurs pour maintenir une compétition équilibrée.

Lors de nos expérimentations, nous n'avons pu entraîner nos modèles que sur 120 epochs pour les problèmes zèbre-cheval et une vingtaine d'epochs pour Monet-photographie. Ceci est bien loin des 200 epochs de [Zhu *et al.*, 2017]. En effet, les contraintes de temps et de matériel nous ont poussé à limiter l'entraînement pour réaliser notre étude.

6 Conclusion

Pour aborder ce sujet, nous avons lu l'article sur CycleGAN et quelques articles sur des sujets similaires comme présenté en introduction 1.1. Nous avons réalisé nous-même une implémentation en se basant sur celle [Zhu *et al.*, 2017] ainsi que celle en accès libre d' Aladdin Persson. Ce projet nous a permis d'aborder plus en profondeur les GANs et de travailler sur une tâche non supervisée très complexe.

Lors de cette étude, nous avons déterminé que le modèle de base proposé par [Zhu *et al.*, 2017] est bien la meilleure variante de CycleGAN parmi celles que nous avons essayées. Les pertes utilisées sont utiles à l'apprentissage du modèle. De plus, les alternatives de générateurs ne sont pas concluantes.

La tâche du transfert de style reste un domaine actif de la recherche. Ces dernières années ont vu le jour plusieurs modèles permettant d'effectuer des transferts de style plus personnalisés. [Liu *et al.*, 2023] ont proposé une méthode utilisant[Liu *et al.*, 2021] (architecture basée sur VGG le mécanisme d'attention) et une technique de segmentation afin de transférer le style sur certains objets de l'image uniquement. Les modèles de diffusion ont été également largement appliqués au transfert de style selon une entrée textuelle d'un utilisateur. Nous pouvons notamment citer les travaux de [Zhang *et al.*, 2023] ou encore les résultats impressionnantes obtenus avec StableDiffusion. Les modèles de diffusion, de par leur nature stochastique, ont cependant tendances à moins respecter la sémantique de l'image d'origine. Cependant des travaux récents cherchent à régler ce problème [Yang *et al.*, 2023].

References

- [Goodfellow *et al.*, 2014] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 27. Curran Associates, Inc., 2014.
- [Heusel *et al.*, 2018] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.
- [Isola *et al.*, 2016] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. 2016.
- [Jing *et al.*, 2018] Yongcheng Jing, Yezhou Yang, Zunlei Feng, Jingwen Ye, Yizhou Yu, and Mingli Song. Neural style transfer: A review, 2018.
- [Kingma and Welling, 2013] Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. 2013.
- [Liu and Tuzel, 2016] Ming-Yu Liu and Oncel Tuzel. Coupled generative adversarial networks. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.
- [Liu *et al.*, 2017] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [Liu *et al.*, 2021] Songhua Liu, Tianwei Lin, Dongliang He, Fu Li, Meiling Wang, Xin Li, Zhengxing Sun, Qian Li, and Errui Ding. Adaatttn: Revisit attention mechanism in arbitrary neural style transfer, 2021.
- [Liu *et al.*, 2023] Songhua Liu, Jingwen Ye, and Xinchao Wang. Any-to-any style transfer: Making picasso and da vinci collaborate, 2023.
- [Ronneberger *et al.*, 2015] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation, 2015.
- [Taigman *et al.*, 2016] Yaniv Taigman, Adam Polyak, and Lior Wolf. Unsupervised Cross-Domain image generation. 2016.
- [Yang *et al.*, 2023] Serin Yang, Hyunmin Hwang, and Jong Chul Ye. Zero-shot contrastive loss for text-guided diffusion image style transfer, 2023.
- [Zhang *et al.*, 2023] Yuxin Zhang, Nisha Huang, Fan Tang, Haibin Huang, Chongyang Ma, Weiming Dong, and Changsheng Xu. Inversion-based style transfer with diffusion models, 2023.
- [Zhu *et al.*, 2017] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. March 2017.