Injection de style par blanchissage et coloration dans un réseau génératif profond

Mémoire

Antoine Dufour

Maîtrise en génie électrique - avec mémoire
Maître ès sciences (M. Sc.)

Québec, Canada

Injection de style par blanchissage et coloration dans un réseau génératif profond

Mémoire

Antoine Dufour

Sous la direction de:

Jean-François Lalonde

Résumé

Dans la génération et la manipulation d'images basées sur les *GAN*s, l'injection de style par *Adaptive Instance Normalization* (*AdaIN*) est devenue la norme pour paramétrer la génération avec une représentation latente du domaine des images.

AdaIN fonctionne en modulant les statistiques des caractéristiques de l'image : il normalise d'abord les caractéristiques en soustrayant leur moyenne et en divisant par leur écart type puis injecte un vecteur de style par l'inverse de cette opération. Bien que cette méthode ait été utilisée avec succès dans une variété de scénarios de traduction d'image à image, la représentation statistique d'*AdaIN* est limitée en ce qu'elle ne tient pas compte des corrélations entre les caractéristiques. Cependant, dans la littérature du transfert de style, la transformation par blanchiment et coloration (*Whitening & Coloring Transformation WCT*) est devenue l'approche privilégiée, car elle prend compte de l'existence de ces corrélations. Toutefois, malgré ses bonnes performances en matière de transfert de style, l'utilisation du *WCT* n'a jusqu'à présent pas été explorée de manière approfondie dans le contexte de l'injection de style.

Dans ce travail, nous comblons cette lacune en remplaçant *AdaIN* par une opération de *WCT* explicite pour l'injection de style dans les *GAN*. Plus précisément, nous introduisons un module qui peut être utilisé en remplacement des blocs *AdaIN* (sans changement additionnel) dans les architectures *GAN* populaires existantes et présentons son impact sur les tâches de génération. Effectivement, dans la génération d'images conditionnelles, où l'espace latent est destiné à représenter le style des images, nous constatons que le blanchiment aide à s'assurer que l'espace n'encode que des informations stylistiques, ce qui permet au contenu de l'image conditionnelle d'être plus visible. Nous démontrons les performances de notre méthode dans deux scénarios : **1)** dans un context d'entraînement supervisé à l'aide du jeu de données *Google Maps* et **2)** en ayant recours à l'architecture StarGANv2 multi-domaine et multi-modale dans une situation d'entraînement non-supervisé et ce en utilisant le jeu de données *Animal Faces-HQ* (*AFHQ*).

Abstract

In the *GAN*-based images generation and manipulation domain, style injection by *Adaptive Instance Normalization* (*AdaIN*) has become the standard method to allow the generation with a latent representation of the image domain.

AdaIN works by modulating the statistics of the characteristics of the image: it first normalizes the characteristics by subtracting their mean and dividing by their standard deviation then it injects a style vector by the reverse of this operation. Although this method has been used successfully in a variety of image-to-image translation scenarios, the statistical representation of *AdaIN* is limited in that it does not take into account the existing correlations between the features. However, in the style transfer literature, the transformation by whitening and coloring (*Whitening & Coloring Transformation WCT*) has become the preferred approach because it takes into account the existence of these correlations. Yet, despite its good performance in terms of style transfer, the use of *WCT* has so far not been explored in depth in the style injection literature.

In this work, we fill this gap by replacing *AdaIN* by an explicit operation of *WCT* for style injection in *GAN*. More specifically, we introduce a module that can be used as a replacement for the *AdaIN* blocks (without any additional change) in the existing popular *GAN* architectures and we present its impact on generation tasks. Indeed, in the conditional image generation task, where the latent space is intended to represent the style of the images, we find that whitening helps ensure that the space encodes only stylistic information which allows the content of the input image to be more visible. We demonstrate the performance of our method in two scenarios: **1)** in a supervised training context using the *Google Maps* dataset and **2)** using multi-domain and multi-modal StarGANv2 architecture in an unsupervised training setup using the *Animal Faces-HQ* (*AFHQ*) dataset.

Table des matières

Liste des tableaux

Liste des figures

Introduction

La génération d'images est un domaine de la vision numérique dont les applications ne cessent de se multiplier. Que ce soit l'édition d'images, l'augmentation de la résolution (super-résolution) ou le transfert de domaine, toutes ces tâches s'avèrent être des applications auxquelles s'adresse la génération d'image. La figure 0.1 présente des exemples de ces méthodes de traitement d'images. Toutefois, malgré le fait que les transformations d'images décrites plus haut peuvent sembler simples à conceptualiser par l'Homme, celles-ci s'avèrent en réalité difficile à représenter à l'aide des principes mathématiques connus. Il va sans dire que la formalisation mathématique de ces concepts se présente comme une tâche ardue à réaliser. Effectivement, l'ensemble de ces procédés de traitement d'images nécessitent l'élaboration d'algorithmes complexes devant prendre en compte une multitude de facteurs tels que la forme, la pose, les textures et les couleurs.

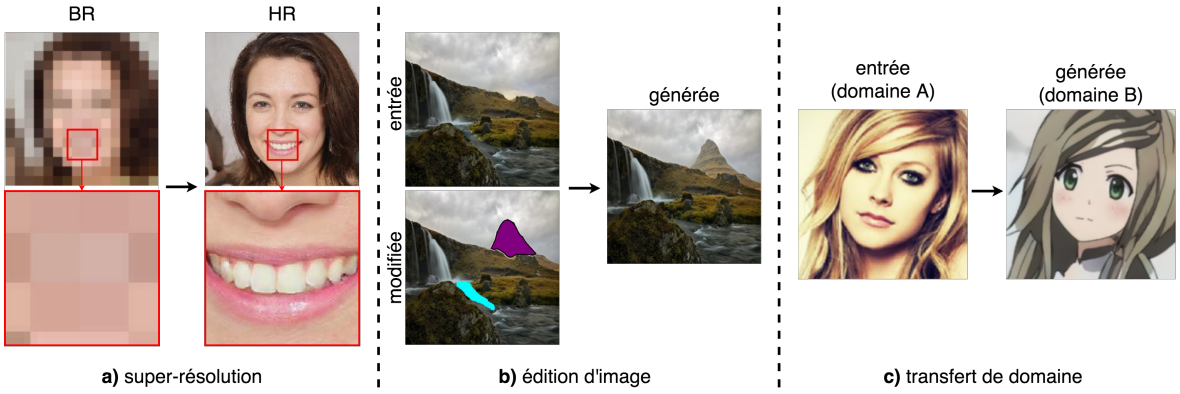

FIGURE 0.1 – **a)** Exemple d'application de super-résolution tiré de *PULSE* (Menon et al., 2020). **b)** Application d'édition d'image tel que vu dans l'article *SESAME* (Ntavelis et al., 2020). **c)** Transfert de domaine d'un visage réel vers un manga. Images tirées de *U-GAT-IT* (Kim et al., 2019)

Ainsi, l'utilisation des réseaux génératifs antagonistes (Goodfellow et al., 2014; Radford et al., 2015) a donc permis de réaliser de grandes avancées dans le domaine de la génération d'images en allouant l'élaboration d'algorithmes se basant sur les réseaux de neurones par apprentissage profond capable de produire des images de haute qualité. Plus particulièrement, les architec-

tures effectuant un transfert de domaine (traduction d'image-à-image) (Isola et al., 2017; Zhu et al., 2017a; Wang et al., 2017; Liu et al., 2017; Choi et al., 2017; Kim et al., 2019) se sont présentées comme une avenue de solutions prometteuses afin de réaliser la traduction de la représentation d'une image vers son expression dans autre domaine où les traits stylistiques (ex. couleurs, textures) peuvent s'avérer être différents. Et ce, tout en préservant les caractéristiques structurelles (ex. pose, orientation, structure) de l'image initiale. L'exemple à la figure 0.2 présente l'application du processus basé sur les architectures effectuant le transfert de domaine.

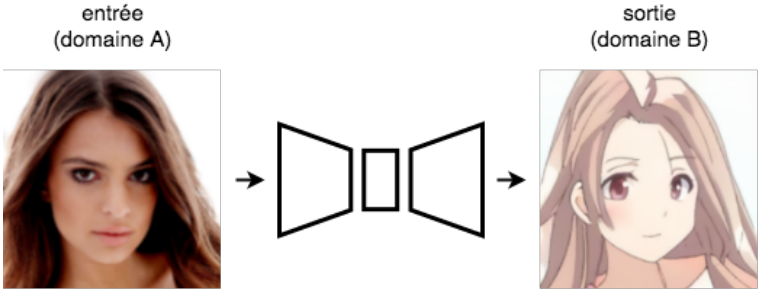

FIGURE 0.2 – Application de la traduction de domaine d'un visage réel (domaine A) vers un dessin de style manga (domaine B). Images tirées de l'article : *Stable, Controllable, Diverse Image to Image Translation* (Chong and Forsyth, 2021)

L'application de la traduction d'images est une méthode qui souffre du problème de *mode collapse* (Salimans et al., 2016; Srivastava et al., 2017), ce qui engendre peu de diversité dans les résultats produits pour une même image (*one-to-one mapping*). La figure 0.3 illustre le phénomène engendré par le manque de diversité dans les résultats produits pour une même entrée. Plusieurs méthodes ont ainsi été développées afin de permettre la production résultats diversifiés pour une même image en entrée (*one-to-many mapping*) (Zhu et al., 2017b). Les solutions développées les plus récentes utilisent une représentation dénouée de l'image (Lee et al., 2018; Gonzalez-Garcia et al., 2018; Huang et al., 2018b; Park et al., 2019; Zhu et al., 2019; Park et al., 2020) qui dissocie la représentation de la structure de l'image, nommée « contenu structurel » (c.-à-d. la pose, les formes ainsi que les traits physique), des caractéristiques stylistiques de celle-ci (couleurs et textures) propres au domaine d'intérêt. La partie considérée comme étant le contenu de l'image est alors préservé (on dit qu'elle est invariante au domaine) et ce ne sont que les caractéristiques stylistiques qui sont modifiées lors de la traduction de domaine. Il devient alors possible de modifier indépendamment le contenu et style lors de l'application du procédé de transfert de domaine.

FIGURE 0.3 – Les résultats générés présentent peu de diversité ce qui rend les sorties presque toutes identiques. **a)** Génération de résultats avec en entrée une carte et en sortie plusieurs résultats de l'image satellite correspondante. **b)** Exemple de plusieurs images de souliers générées à partir du même croquis en entrée. Résultats tirés de l'article *DSGAN* (Yang et al., 2019).

Cette dissociation s'opère avec l'encodage et la séparation, d'une part, des traits invariants caractérisant l'image (contenu) et, d'autre part, des caractéristiques spécifiques au domaine (style). Ainsi, lors de l'opération de traduction d'image-à-image, seules les caractéristiques de style sont modifiées. Le transfert de domaine peut alors être réalisée en modifiant uniquement les traits encodés de l'image source se rapportant au style afin de remplacer ceux-ci par les caractéristique de style du domaine cible. L'utilisation d'une représentation latente des caractéristiques de style propre à un domaine, généralement représenté sous la forme d'une distribution gaussienne $\mathcal{N}(\vec{0}, \mathbf{I})$, a menée au développement d'une méthode de régularisation de la diversité (Mao et al., 2019; Yang et al., 2019) poussant le réseau générateur à explorer les différents modes présents dans le jeu de données afin d'accentuer la production de résultats diversifiés.

Cependant, les architectures basées sur cette méthode nécessitent un processus d'insertion d'information au sein du réseau générateur en vue de modifier les composantes de style. Ainsi, inspiré des travaux réalisés dans le transfert de style, l'utilisation d'*Adaptive Instance Normalization (AdaIN)* (Huang and Belongie, 2017) a été préconisée comme méthodes d'injection de style dans le générateur (Choi et al., 2020; Saito et al., 2020; Baek et al., 2020).

Toutefois, l'injection de style avec *AdaIN* comporte des limitations dans sa représentation du style. En effet, c'est une méthode qui ignore les corrélations pouvant exister au sein des caractéristiques du générateur. Aucune transformation n'est donc appliquée de sorte à modifier ces corrélations. La représentation du style est donc limitée en ce qu'elle ne peut être exprimée et modifiée par les interactions existantes entres les caractéristiques du générateur.

Afin de palier à cette limitation, ce mémoire présente une méthode d'injection d'information par blanchissage et coloration au sein d'un réseau de neurones pro-

fonds génératif, ce qui a pour objectif d'améliorer la qualité ainsi que la diversité des résultats générés. Pour à atteindre ce but, nous proposons les trois contributions suivantes :

1. La première contribution correspond à l'exploration de l'utilisation du blanchissage et du coloriage dans un contexte d'injection de style au sein d'une architecture de réseau antagoniste génératif. L'utilisation du principe de blanchissage des paramètres d'un ensemble de donnée ainsi que la production de matrices de colorations et son application sont intégrés au sein du processus de génération des images.
2. La seconde contribution correspond à l'utilisation de cette méthode dans deux architectures de réseaux antagonistes génératifs tirées de la littérature. De manière plus concrète, cela est fait dans le contexte d'un entraînement supervisé basé sur un jeu de données appariées, et dans un scénario d'entraînement non-supervisé où les jeux de données d'entrée et de cible ne contiennent pas de correspondances.
3. La dernière contribution est une série d'expérimentations qui démontre que l'utilisation du blanchissage et de la coloration, à des fins d'injection d'information dans le cadre de la traduction d'image à image, engendre la génération de résultats de qualité supérieure à ceux produit par l'état de l'art actuel.

L'ensemble de ces contributions est présenté en trois chapitres. Le chapitre 1 met en contexte le problème auquel répond ce mémoire à travers une revue des travaux antérieurs ayant été réalisés dans le domaine de la génération d'images et de manière plus spécifique, dans le domaine de la traduction d'image à image. Le chapitre 2 introduit la méthode d'injection de style par blanchissage et coloration ainsi que le module développé pour en faire l'application dans un réseau générateur. Finalement, afin de consolider l'utilisation de la méthode, le chapitre 3 présente une série d'expériences réalisée de façon à analyser l'effet de l'injection de style par blanchissage et coloration sur les résultats produits.

Chapitre 1

Contexte et travaux antérieurs

Le chapitre suivant présente les techniques et les méthodes qui forment les bases des travaux effectuées dans le contexte du développement de l'injection de style par blanchiment et coloration.

1.1 Réseaux antagonistes génératifs (GAN)

1.1.1 GAN

Le domaine de la vision numérique, plus particulièrement le domaine de la génération d'images a grandement été transformé et influencé au cours des dernières années par l'avènement des réseaux antagonistes génératifs (Goodfellow et al., 2014). En effet, les réseaux antagonistes génératifs, communément appelés *GAN* (*Generative Adversarial Networks*), correspondent à une classe d'algorithmes d'apprentissage machine non-supervisés se basant sur l'utilisation des réseaux de neurones profonds comme point central de la méthode afin de produire des données. Ceux-ci permettent par exemple de générer des images réalistes en ayant recours à un modèle génératif qui tente de reproduire de la manière la plus fidèle la distribution des données avec laquelle il a été entraîné.

Plus concrètement, les *GAN*s utilisent deux fonctions différentiables : un réseau génératif G qui tente de reproduire la distribution des données p_{data} dans le but de générer des échantillons réalistes et un réseau discriminant D dont le rôle est d'estimer la probabilité qu'un échantillon proviennent de la distribution des vraies données p_{data} ou de la distribution des données produites par le générateur, p_g. Les modèles utilisés comme générateur et discriminateur étaient à l'origine des perceptrons multicouche (MLP) (Rumelhart et al., 1986). Ceux-ci ont toutefois rapidement été remplacés par des réseaux convolutifs (CNN) (LeCun et al., 1989) puisque ces derniers se sont avérés être plus performant dans le contexte de la génération d'images (Radford et al., 2015).

Afin de permettre au générateur d'apprendre une représentation du domaine des données uti-

lisées à l'entraînement p_g selon $x \sim p_{data}(x)$, l'utilisation d'une variable d'entrée généralement échantillonnée d'une gaussienne multivariée $z \sim p_z(z)$ est préconisée. Cet espace de complexité moindre est appelé espace latent. Le générateur agit alors comme une fonction qui effectue la traduction d'un espace ayant une moins grande complexité (espace latent à n dimensions) vers un espace de complexité beaucoup plus grande, c'est-à-dire l'espace des données $G(z) \rightarrow x_{gen}$.

Pour sa part, $D(x)$ détermine la probabilité qu'un échantillon x appartienne à p_{data}. Lors de l'entraînement, G et D s'opposent l'un à l'autre dans un jeu à deux joueurs minimax dont l'objectif est le suivant :

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}\,(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]. \tag{1.1}$$

Ici, l'objectif du générateur est de tromper le discriminateur en produisant des résultats indiscernables de ceux provenant des vraies données de façon à minimiser $\log(1 - D(G(\boldsymbol{z})))$. Le discriminateur est entraîné simultanément afin de discerner les vraies des fausses données afin de maximiser $\log D(\boldsymbol{x}) + \log(1 - D(G(\boldsymbol{z})))$. La figure 1.1 illustre la structure générale qui forme un *GAN*.

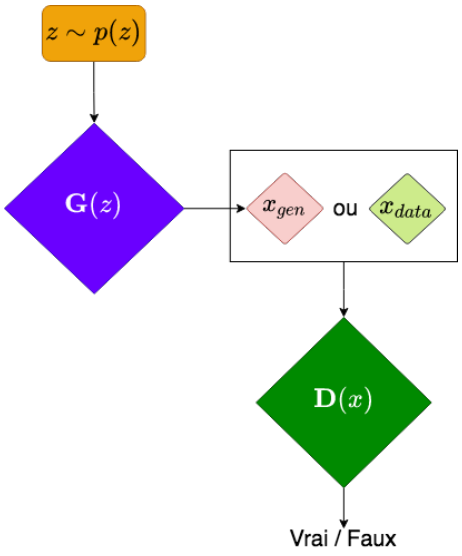

FIGURE 1.1 – Représentation de la structure conventionnelle d'un *GAN* où le générateur reçoit en entrée un vecteur latent z afin de produire une donnée x_{gen} et le discriminateur tente de discriminer les vraies x_{data}, des fausses données x_{gen}.

Cependant, malgré la capacité des *GANs* à produire des résultats très similaires à ceux provenant des jeux de données avec lesquels ils ont été entraînés, ceux-ci n'offrent aucun contrôle sur les résultats générés. L'entrée en provenance de l'espace latent ne donnant pas de contrôle sur la structure du résultat produit par le générateur.

1.1.2 Réseaux antagonistes génératifs conditionnels ($cGAN$)

Dans le but de palier à cette limitation, une extension a été apportée aux réseaux antagonistes génératifs afin de permettre l'ajout d'un meilleur contrôle des images générées. En effet, les réseaux antagonistes génératifs conditionnels ($cGAN$) (Mirza and Osindero, 2014) donnent la possibilité de passer de l'information additionnelle au générateur et au discriminateur lors du processus de génération des fausses données. Pour ce faire, le générateur et le discriminateur sont conditionnés sur des informations supplémentaires en entrée y de manière à modifier l'objectif du jeu minimax $V(D, G)$ de la manière suivante :

$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))]. \tag{1.2}$$

La figure 1.2 montre de quelle manière le générateur ainsi que le discriminateur sont menés à prendre en considération les informations supplémentaires fournies par la condition lors du processus d'entraînement. Dans l'exemple présenté, la condition $y = 3$ force le générateur à produire le chiffre trois puisque celui-ci est fourni en entrée. Afin s'assurer le respect de la condition, cette dernière est aussi passée au discriminateur qui détient alors l'information nécessaire pour déterminer si en plus d'être vrai, le résultat généré correspond à la condition.

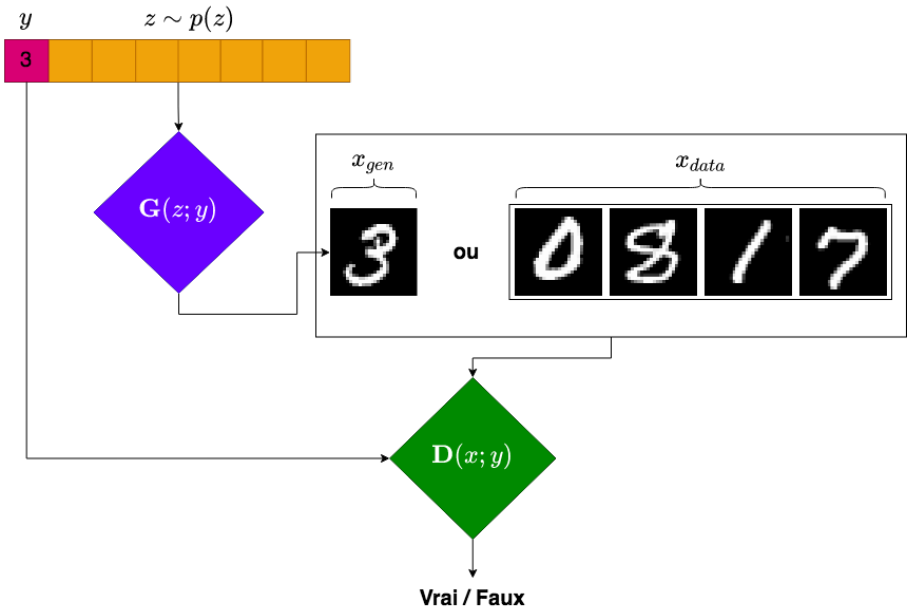

FIGURE 1.2 – Représentation d'un $cGAN$ sur le jeu de données $MNIST$ (LeCun et al., 2010) avec à l'entrée un vecteur latent z auquel est ajouté une condition $y = 3$ dans le but d'indiquer au générateur de produire une image qui correspond au chiffre trois. Le discriminateur reçoit aussi l'information conditionnelle afin de discriminer correctement le chiffre trois.

1.1.3 Traduction appariée d'*image-to-image* (I2I)

Les *cGANs* ont permis une grande avancée en vision numérique dans le développement de solutions afin de répondre au problème de transfert de domaine d'une image en entrée vers une image provenant d'un autre domaine en sortie. Cette tâche de transfert de domaine, s'étant vu attribuer le nom de "image-to-image translation" ou de manière abréviative I2I, utilise une fonction génératrice conditionné sur une image en entrée afin de produire un résultat d'une correspondance spécifique en sortie. Cette méthode peut être utilisée dans différents type d'applications tels que la coloration d'images (Zhang et al., 2016; He et al., 2018; Xu et al., 2020; Su et al., 2020), la super-résolution d'images (Sun and Chen, 2019; Anwar and Barnes, 2019; Dai et al., 2019), la retouche d'image (*image inpainting*) (Yang et al., 2016; Ulyanov et al., 2017a; Yu et al., 2018) ainsi que le transfert de style. De manière plus précise, la traduction *image-to-image* peut être définie comme étant la réalisation de la transformation d'une image provenant d'un domaine source $x_s \in \mathcal{X}_s$ vers un autre domaine en ayant recours à une fonction de traduction $G_{s\to c}$ agissant sur l'image en entrée afin de déterminer une correspondance dans le domaine cible $x_c \in \mathcal{X}_c$.

Les algorithmes les plus récents d'I2I (Isola et al., 2017; Wang et al., 2017; Park et al., 2019; Zhu et al., 2019; Zhang et al., 2020) se basent directement sur l'utilisation de réseaux antagonistes génératifs conditionnels afin de résoudre une multitude de tâches de transfert de domaine de manière supervisées. De fait, les techniques d'*image-to-image* utilisent les réseaux convolutifs profonds comme fonction de traduction de représentation $G_{s\to c}$ généralement basés sur les fondements des architectures U-Net (Ronneberger et al., 2015) et ResNet (He et al., 2015). Considérant un jeu de donnée apparié où chaque donnée du domaine source x_s détient une correspondance x_c (étiquette) dans le domaine cible, l'objectif d'apprentissage dans ce contexte peut alors être décrit comme étant

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G), \tag{1.3}$$

où la fonction de perte du réseau antagoniste peut être exprimée de la manière suivante :

$$\mathcal{L}_{GAN}(G, D) = \mathbb{E}_{x_c}[\log D(x_c)] + \mathbb{E}_{x_s,z}[\log(1 - D(G(x_s, z))], \tag{1.4}$$

avec G la fonction générative de translation et D le discriminateur. La fonction de perte $\mathcal{L}_{L1}$ permet quant à elle d'assurer un résultat généré $G(x_s, z)$ fidèle à la cible x_c. Le paramètre λ est une pondération appliquée afin de déterminer le poids de la fonction de perte $\mathcal{L}_{L1}$ par rapport à celui de la fonction de perte $\mathcal{L}_{GAN}(G, D)$.

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x_s,x_c,z}\left[\|x_c - G(x_s, z)\|_1\right]. \tag{1.5}$$

La figure 1.3 présente la structure générale d'une architecture d'*image-to-image* de base. Le générateur G reçoit l'image de condition en entrée afin de produire une image dans le domaine cible le plus réalise possible. Le discriminateur D détermine alors la véracité de l'image produite. Notons que lors du processus d'entraînement, le discriminateur reçoit en alternance des images réelles et des images générées afin d'être en mesure de discerner leur provenance.

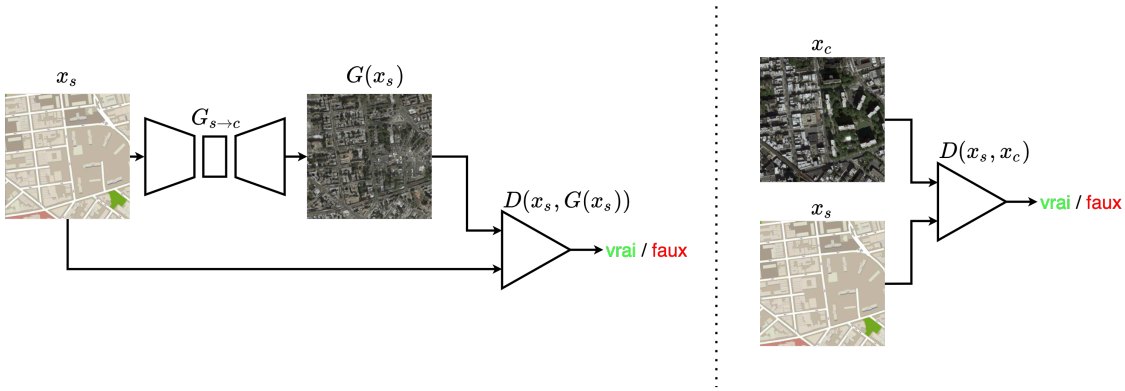

FIGURE 1.3 – Schématisation d'une architecture classique d'*image-to-image* où le générateur G réalise la traduction de l'image source x_s vers une image du domaine cible x_c. Le discriminateur D détermine si l'images est réelles ou générées.

1.1.4 Traduction non-appariée d'*image-to-image* (UI2I)

Bien que la méthode supervisée d'I2I permet de générer des résultats très prometteurs, dans la plupart des cas, un jeu de données appariées peut être difficile voire impossible à obtenir. Dans le but de surmonter cette limitation, plusieurs méthodes basée sur une contrainte cyclique (*cycle-consistency loss*) ont été proposées dans différents articles (Zhu et al., 2017a; Yi et al., 2017; Kim et al., 2017; Liu et al., 2017) afin de permettre la traduction de domaine *I2I* tout en ne nécessitant aucune donnée appariée. La figure 1.4 présente un exemple d'application tiré de l'article Zhu et al. (2017a) où deux images provenant de différents domaines sans appariement sont utilisées afin de réaliser le transfert d'une domaine à l'autre.

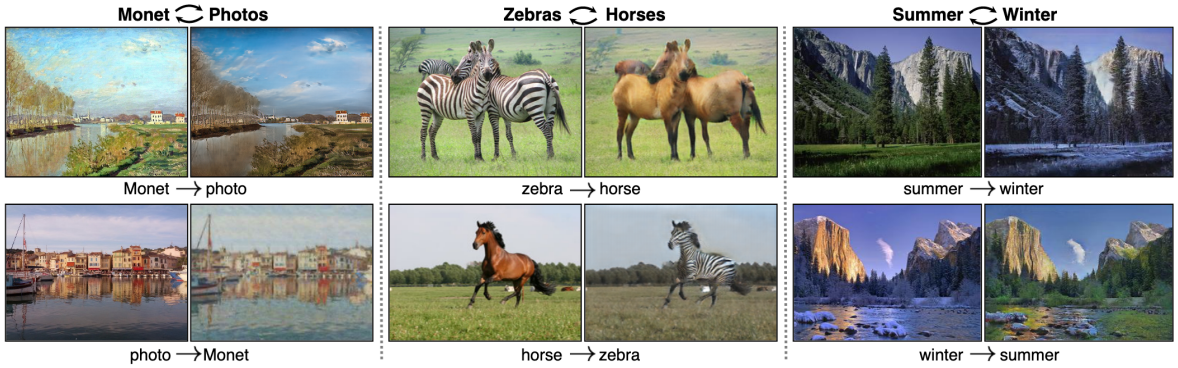

FIGURE 1.4 – Paires d'images provenants de deux domaines différents traduites d'un domaine vers l'autre à l'aide de et ce sans appariement entre les images. À gauche, peinture de Monet et scène réelle, au centre, zebre et cheval et à droite, des scène d'hiver et d'été. Figure tirée du travail réalisé par Zhu et al. (2017a).

Considérons deux ensembles d'images non appariées provenant des domaines A et B. L'objectif de l'entraînement peut être défini comme étant une fonction $G_{AB} : A \to B$ qui traduit une image $x_a \in A$ vers une image $x_b \in B$ ainsi qu'une fonction inverse à G_{AB} qui réalise la traduction de retour au domaine initial, $F_{BA} : B \to A$. La fonction de perte cyclique à optimiser afin de permettre l'ajustement des deux générateurs peut alors être définie comme suit :

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G_{AB}, F_{BA}) = \mathbb{E}_{x_a \sim p_{data}(x_a)} \left[\|F_{BA}(G_{AB}(x_a)) - x_a\|_1\right] \\ + \mathbb{E}_{x_b \sim p_{data}(x_b)} \left[\|G_{AB}(F_{BA}(x_b)) - x_b\|_1\right].\end{aligned} \tag{1.6}$$

La cohérence cyclique correspond concrètement à l'évaluation de deux fonctions de perte de reconstruction afin d'assurer que le passage du domaine $A \to B$ avec $G_{AB}(x_a) = x_{ab}$ et le retour au point d'origine $B \to A$ avec la fonction inverse $F_{BA}(x_{ab}) = x_{aba}$ soit garantie à l'aide de $\|x_{aba} - x_a\|_1$. La seconde partie de l'équation, symétrique à la première, s'assure que le cycle $B \to A \to B$ soit lui aussi respecté. La figure 1.5 illustre de manière schématique la procédure effectuée afin d'évaluer la cohérence cyclique entre deux domaines A et B.

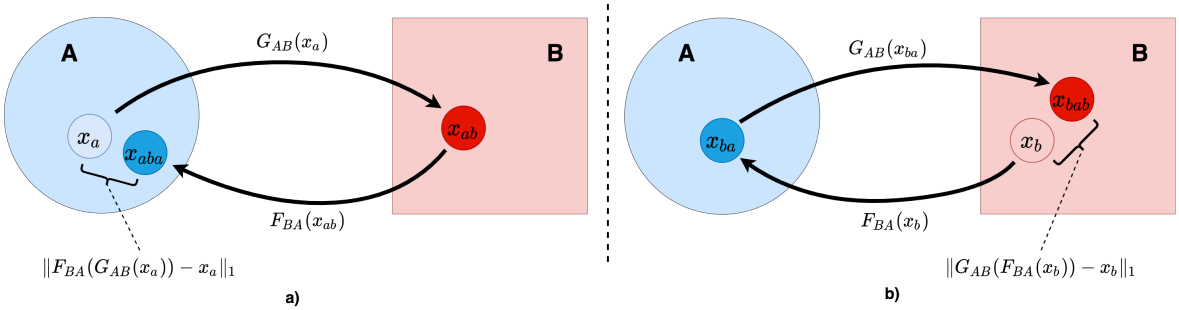

FIGURE 1.5 – La fonction de perte cyclique permet d'assurer une cohérence des résultats produits lors du retour dans le domaine d'origine à la suite d'une cycle complet effectué entre les deux domaines. Cet objectif permet de guider l'entraînement à produire deux générateurs G et F qui sont des fonctions inverse l'une à l'autre. Cela permet de réaliser un retour à l'endroit exact de départ suite à un cycle. **a)** Cycle effectué $A \rightarrow B \rightarrow A$ assurant la reconstruction $F_{BA}(G_{AB}(x_a)) \approx x_a$. **b)** Cycle effectué $B \rightarrow A \rightarrow B$ assurant la reconstruction $G_{AB}(F_{BA}(x_b)) \approx x_b$.

En plus de l'objectif de cohérence du cycle, une fonction de perte antagoniste pour chaque domaine est ajoutée à l'entraînement. Celles-ci, à l'aide d'un discriminateur par domaine, assurent la qualité des résultats produits par le générateur de chaque domaine. L'objectif qui assure la qualité des résultats produit dans le domaine B est le suivant :

$$\begin{aligned}\mathcal{L}_{\mathrm{GAN_B}}\left(G_{AB}, D_B\right) = \mathbb{E}_{x_b \sim p_{\mathrm{data}}\,(x_b)}\left[\log D_B\left(x_b\right)\right] \\ + \mathbb{E}_{x_a \sim p_{\mathrm{data}}\,(x_a)}\left[\log\left(1 - D_B\left(G_{AB}\left(x_a\right)\right)\right)\right],\end{aligned} \tag{1.7}$$

avec D_B, fonction discriminant les données réelles des données générées provenant du domaine B. Dans manière similaire, une fonction de perte antagoniste est aussi appliquée pour le domaine A. Celle-ci n'est cependant pas exprimé ici puisqu'elle correspond de façon symétrique à l'équation 1.7.

1.1.5 Traduction d'*image-to-image* multi-domaine

Bien que les avancées dans le cadre de la traduction d'images entre deux domaines aient permis de produire une multitude de résultats de haute qualité, cette méthode est contrainte à l'apprentissage de la relation exclusive entre deux domaines. Il devient donc difficile, lorsqu'il est désiré de réaliser la traduction entre plusieurs domaines, de développer un algorithme ne nécessitant pas une architecture dont la complexité croît de manière significative à l'ajout de chaque domaine. En effet, considérant n domaines, le nombre de générateurs nécessaire à la conception de l'architecture devient ainsi $n \times (n-1)$.

À cette fin, la traduction d'image-to-image multi-domaine à été conçue afin de permettre le changement de domaines multiples en ayant recours à un seul modèle unifié plutôt qu'à un ensemble de générateurs. Certains travaux (Zhao et al., 2018; Wu et al., 2019; Tang et al., 2019)

s'intéressent au développement de méthodes permettant la génération multi-domaine d'images. Cependant, une méthode s'est avérée plus simple et efficace afin de résoudre ce problème. Effectivement, le travail réalisé par Choi et al. (2017), nommé StarGAN, propose une nouvelle approche qui permet le développement d'un modèle unique entraîné sur n domaines de manière simultanée. Cette méthode permet de capturer certaines informations qui transcendent les limites de la représentation à deux domaines dans un processus d'apprentissage multi-domaines en plus de réduire la complexité des architectures utilisées. La figure 1.6 permet la visualisation de la réduction de la complexité du modèle utilisé afin de produire des résultats provenant d'une multitude de domaines.

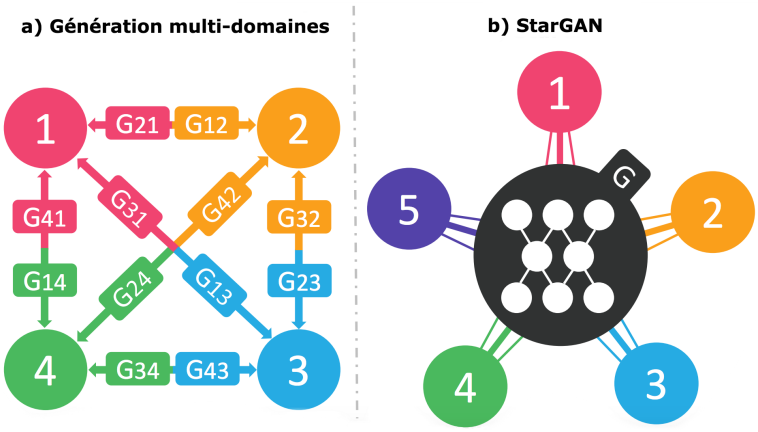

FIGURE 1.6 – a) Entraînement de $n \times (n-1)$ générateurs afin de réaliser la traduction dans n domaines. b) Schématisation de l'architecture de StarGAN où un seul générateur permet la traduction vers une multitude de domaine. Figure tirée de Choi et al. (2017).

1.2 Diversité des résultats générés

Un problème récurrent qui est observé dans les tâches de génération utilisant des modèles d'I2I est la mauvaise généralisation du modèle de génération qui produit de manière systématique des résultats provenant d'un faible sous-ensemble de la représentation réelle du domaine. Ce symptôme bien connu des réseaux antagonistes génératifs (Goodfellow et al., 2014; Salimans et al., 2016) mène le générateur à ne représenter qu'un faible sous-ensemble de la distribution des données en sortie. De manière concrète, cela peut s'exprimer par de la traduction un-pour-un (*one-to-one mapping*) où pour une entrée particulière au générateur la même sortie sera toujours produit et donc l'absence de génération stochastique est observée. La figure 1.7 illustre le phénomène déterministe qui survient lors du processus de traduction d'une image.

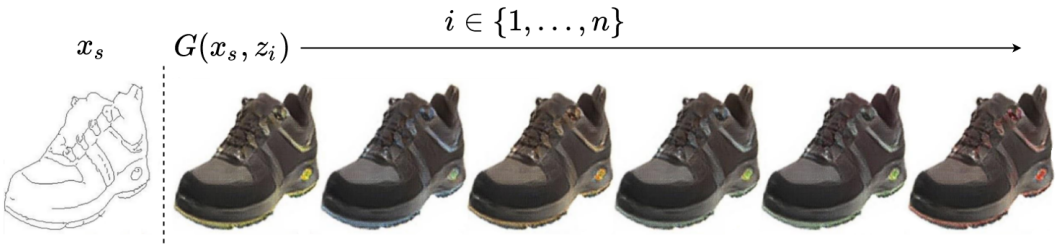

FIGURE 1.7 – Exemple de traduction d'une image en entrée x_s vers un résultat déterministe (*one-to-one mapping*). Il est possible de remarquer que pour la même entrée, malgré la variation du vecteur latent z_i, le résultat est toujours le même à quelques variations non-significatives près.

1.2.1 Traduction d'image-to-image multi-mode

Une multitude de travaux (Metz et al., 2016; Che et al., 2016; Durugkar et al., 2016; Zhu et al., 2017b; Huang et al., 2018b; Lee et al., 2018; Wang et al., 2019b) a été effectuée dans le domaine de la traduction d'image afin d'améliorer la diversité des images générées en permettant une meilleure représentation du domaine de génération appris. La figure 1.8 présente un exemple d'application. Toutefois, ces méthodes ont généralement plusieurs contraintes. Par exemple, celles-ci peuvent nécessiter une lourde charge de calcul à l'entraînement ou doivent être développées de manière à effectuer une tâche spécifique ce qui rend leur généralisation difficile. Il est donc complexe voire impossible d'étendre ces méthodes à la résolution d'autres problèmes.

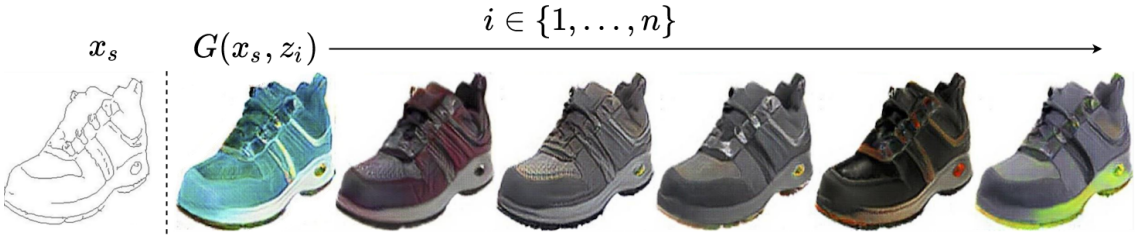

FIGURE 1.8 – Exemple de traduction d'une image en entrée x_s vers plusieurs résultats $G(x_s, z_i)$ pour $i \in \{1, \cdots, n\}$ (*one-to-many mapping*).

1.2.2 Régularisation de diversité

Afin de pallier aux difficultés que ces méthodes apportent, Mao et al. (2019) et Yang et al. (2019) proposent une fonction de perte qui permet une régularisation de l'entraînement. Cette régularisation, incluse dans l'objectif global d'entraînement, encourage le générateur à explorer les différents modes présents dans l'espace des données afin de produire des résultats diversifiés qui dépendent de l'échantillonnage d'un code latent.

Notons le générateur conditionnel G qui prend en entrée une donnée $x \in \mathcal{X}$ et un code latent $z \in \mathcal{Z}$ afin de générer une donnée. L'objectif à maximiser suivant est intégré à l'entraînement

du générateur :

$$\max_{G} \mathcal{L}_{\boldsymbol{z}}(G) = \mathbb{E}_{\boldsymbol{z}_1,\boldsymbol{z}_2}\left[\min\left(\frac{\|G(\boldsymbol{x},\boldsymbol{z}_1) - G(\boldsymbol{x},\boldsymbol{z}_2)\|}{\|\boldsymbol{z}_1 - \boldsymbol{z}_2\|}, \tau\right)\right], \tag{1.8}$$

où z_1 et z_2, couramment appelés code latent, correspondent à deux vecteurs échantillonnés d'une distribution statistique généralement gaussienne, τ est une borne qui assure la stabilité numérique et $\|\cdot\|$ correspond à une norme quelconque par souci de généralité bien qu'en pratique la norme L1 soit préconisée. Ainsi, lorsque le générateur produit des résultats déterministes dans un seul mode en se basant uniquement sur la variable d'entrée $\boldsymbol{x}$, l'objectif atteint son minimum puisque $G(\boldsymbol{x},\boldsymbol{z}_1) \approx G(\boldsymbol{x},\boldsymbol{z}_2)$. La régularisation force alors le générateur à se baser sur le code latent $z \sim \mathcal{N}(\mathbf{0},\mathbf{1})$ afin de produire des résultats diversifiés dans plusieurs modes.

1.2.3 Adaptive Instance Normalization (AdaIN)

Dans le but de réaliser l'injection de l'information provenant du code latent dans le réseau générateur, une méthode développée préalablement dans le domaine du transfert de style s'est avérée être efficace à cette fin. Inspiré des travaux de Ioffe and Szegedy (2015); Ulyanov et al. (2017b); Dumoulin et al. (2016), Adaptive Instance Normalization (AdaIN), développée par Huang and Belongie (2017), s'est présenté comme une méthode très efficace afin de produire des résultats dans le domaine du transfert de style de haute qualité tout en étant conceptuellement simple. Influencées par l'utilisation de cette méthode afin de transférer les statistiques de style d'une image vers une autre image, les récentes architectures de travaux marquants dans le domaine des réseaux génératifs (Huang et al., 2018b; Karras et al., 2019; Choi et al., 2020) ont maintenant recours à cette méthode pour réaliser l'injection de code latent (code de style).

Pour se faire, AdaIN prend les cartes d'activations d'une couche du réseau générateur $\mathbf{x}_i$ ainsi qu'un style $\mathbf{y}$ afin de modifier les statistiques de moyenne $\mu(\mathbf{x}_i)$ et de variance $\sigma(\mathbf{x}_i)$ des paramètres de $\mathbf{x}_i$ dans le but de les faire correspondre à ceux de $\mathbf{y}$. L'opération d'AdaIN peut être définie comme étant

$$\text{AdaIN}(\mathbf{x}_i,\mathbf{y}) = \mathbf{y}_{s,i}\frac{\mathbf{x}_i - \mu(\mathbf{x}_i)}{\sigma(\mathbf{x}_i)} + \mathbf{y}_{b,i}, \tag{1.9}$$

où $\mathbf{y}_{s,i}$ et $\mathbf{y}_{b,i}$ correspondent respectivement au facteur d'échelle ainsi qu'au biais provenant du style pour la couche i. L'obtention du vecteur de style $\mathbf{y}$, telle que décrite à la figure 1.9, est effectuée à l'aide d'un code latent $\mathbf{z} \in \mathcal{Z}$ qui est traduit dans une fonction de traduction (*mapping network*) non-linéaire $f : \mathcal{Z} \to \mathcal{W}$ afin de produire un vecteur latent $\mathbf{w} \in \mathcal{W}$. Une couche linéaire apprise transforme finalement $\mathbf{w}$ vers le style $\mathbf{y} = (\mathbf{y}_s, \mathbf{y}_b)$ qui est utilisé dans l'opération que produit AdaIN. Il est important de mentionner que l'utilisation du terme « style » est justifiée par le désir de préserver le concept de style introduit par les auteurs y

ayant recours dans un objectif de transfert de style. Toutefois, dans le cas actuel, le style $\mathbf{y}$ correspond plutôt au résultat de la transformation affine apprise permettant le transfert du code latent $\mathcal{W}$ vers l'espace $\mathcal{Y}$.

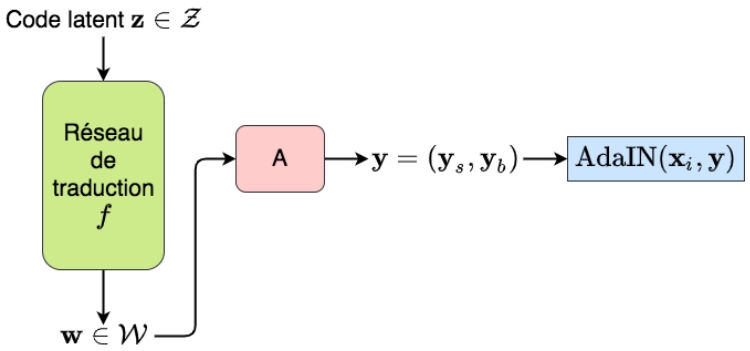

FIGURE 1.9 – Le code latent $\mathbf{z}$ est premièrement traduit vers un espace intermédiaire $\mathcal{W}$ pour ensuite être transformé en un vecteur de style $\mathbf{y}$. La boite A correspond à la transformation linéaire apprise entre $\mathbf{w}$ et $\mathbf{y}$. Figure inspirée de Karras et al. (2019).

1.3 Métriques d'évaluation des images générées

La génération d'images, en plus d'être une tâche complexe, nécessite un processus d'évaluation de la qualité des résultats de manière quantitative qui requiert un processus ardu. Bien que cette tâche puisse sembler être relativement simple pour l'œil humain, celle-ci devient très complexe dans le cadre d'un processus d'évaluation automatisé basé sur une métrique. Ainsi, afin de répondre au besoin d'évaluation des modèles génératifs, des métriques d'évaluations ont été développés telles que Inception Score (Salimans et al., 2016), LPIPS de diversité (Zhang et al., 2018b; Zhu et al., 2017b; Huang et al., 2018b), GAN-train/GAN-test(Shmelkov et al., 2018), Fréchet Inception Distance (Heusel et al., 2017) et plusieurs autres tel que mentionné par Borji (2019). Malgré les diverses possibilités, ce travail base ses évaluations quantitatives sur deux de ces métriques qui sont parmi les plus couramment utilisées à cette fin actuellement. FID (Heusel et al., 2017) comme métrique de qualité des résultats et LPIPS de diversité (Zhang et al., 2018b; Huang et al., 2018b) pour mesurer le niveau de diversité des résultats générés. Les sections suivantes décrivent leur fonctionnement de manière plus précise.

1.3.1 *Learned Perceptual Image Patch Similarity* (*LPIPS*) pour diversité

Inspirés par le développement de la fonction de perte perceptuelle (Johnson et al., 2016), Zhang et al. (2018a) développent un métrique de comparaison des images. Ce métrique peut être considéré comme une manière d'évaluer la distance entre deux images selon leur représentation conceptuelle dans l'espace des caractéristiques d'un réseau de neurones. Cette approche tente de s'inspirer de la manière dont un humain réaliserait son évaluation de la similarité entre deux images. Cette distance entre deux images I_0 et I peut ainsi être exprimée comme étant

$$d\left(I, I_0\right) = \sum_l \frac{1}{H_l W_l} \sum_{h,w} \left\| w_l \odot \left(\hat{y}^l_{hw} - \hat{y}^l_{0hw} \right) \right\|_2^2, \tag{1.10}$$

où $\hat{y}^l_{hw}, \hat{y}^l_{0hw} \in \mathbb{R}^{H_l \times W_l \times C_l}$ correspondent aux activations d'un réseau $\mathcal{F}$ à la couche $l \in L$ (L étant l'ensemble des couches appartenant au réseau $\mathcal{F}$). Le réseau $\mathcal{F}$ est généralement un réseau classique de classement tel que VGG-19 (Simonyan and Zisserman, 2014), Inception (Szegedy et al., 2014) ou AlexNet (Krizhevsky et al., 2012). $w_l \in \mathbb{R}^C_l$ est un vecteur de poids appris qui fait une mise à l'échelle des activations par canal. Cette mise à l'échelle peut être vue comme une « calibration perceptuelle » des activations.

Afin d'évaluer la diversité des résultats produit par un réseau générateur, Zhu et al. (2017b) proposent une méthode qui a recours à la métrique LPIPS afin d'évaluer la similarité entre les images générées par la même entrée puisque celle-ci est corrélée avec la perception humaine de la similarité (Zhang et al., 2018a). Ce score de diversité est calculé en mesurant la distance LPIPS moyenne obtenue entre toutes les paires de données générées aléatoirement pour la même entrée. Par la suite, une moyenne sur l'ensemble des résultats obtenues pour toutes les images en entrée est calculée afin d'obtenir la valeur finale. Suivant Zhu et al. (2017b); Huang et al. (2018b), 100 images en entrée sont utilisées afin de produire 19 paires d'images pour chacune d'entre elles. Cela donne un total de 1900 paires d'images sur lesquelles la distance LPIPS peut être mesurée afin de déterminer la distance moyenne entre ces paires d'images.

1.3.2 Fréchet Inception distance (FID)

Une méthode couramment utilisée afin de mesurer la qualité visuelle des images produites par les GAN est la Fréchet Inception Distance (Heusel et al., 2017). Les images réelles et générées sont tout d'abord passées dans un modèle Inception (Szegedy et al., 2014) afin d'obtenir des caractéristiques basées sur des concepts visuels qui ont été appris par le réseau. En assumant que ces caractéristiques suivent une loi normale multivariée, les statistiques de moyenne et de covariance de ces celles-ci sont alors comparées entre les données réelles et synthétiques. Plus précisément, une mesure de la différence des deux distributions gaussiennes multivariées est calculée en ayant recours à la distance de Fréchet (Dowson and Landau, 1982) aussi connue comme étant la distance de Wasserstein-2 (Vaserstein, 1969).

La FID peut alors être calculée avec l'équation

$$\text{FID}(\mathcal{N}_v, \mathcal{N}_f) = \|\vec{\mu_v} - \vec{\mu_f}\|^2 + \text{Tr}(\mathbf{\Sigma}_v + \mathbf{\Sigma}_f - 2(\mathbf{\Sigma}_v \mathbf{\Sigma}_f)^{\frac{1}{2}}), \tag{1.11}$$

où Tr correspond à la trace de la matrice résultante, $\mathcal{X}_v \sim \mathcal{N}(\vec{\mu_v}, \mathbf{\Sigma}_v))$ et $\mathcal{X}_f \sim \mathcal{N}(\vec{\mu_f}, \mathbf{\Sigma}_f))$, sont les 2048 activations dimensionnelles de la couche *pool3* du réseau Inception-v3 (Szegedy et al., 2014) pour des données réelles et générées respectivement.

Sommaire des travaux antérieurs

En somme, cette section présente certains concepts fondamentaux qui permettent de poser les bases du traitement d'images. Plus particulièrement, celle-ci réalise une revue des méthodes de générations d'images qui ont mené au développement des notions du transfert de domaine à l'aide des réseaux antagonistes génératifs. Ce sont donc les principes introduits dans ce chapitre qui mettent en place les fondements sur lesquels repose l'ensemble de la méthode présentée dans ce mémoire.

Dans un même ordre d'idées, deux méthodes d'évaluations traitant d'une part de la qualité des images la **FID** (1.11) et d'autre part de la diversité des résultats découlant de la génération d'images **LPIPS diversité** (1.3.1) sont aussi présentées afin de permettre l'évaluation des performances résultantes des changements apportées par la méthode proposée à la section 2.

Chapitre 2

Méthodologie

Approche

Ce chapitre est consacré à la description de la méthode d'injection de style *Adaptive Whitening and Coloring Style Injection* (*AWCSI*) ainsi qu'au module qui permet son application dans le contexte des réseaux génératifs. Les choix de designs y sont présentés afin de montrer la forte compatibilité de la méthode avec les architectures déjà existantes. L'objectif de cette méthode vise à remplacer, dans un réseau générateur quelconque, l'injection de style utilisant *AdaIN* par notre module *AWCSI* et ce, sans avoir à apporter d'autres modifications à l'architecture du réseau ou en ayant recours à aucune fonction de perte additionnelle pour réguler l'entraînement. Bien que les techniques de blanchissage et de coloriage aient déjà été utilisées dans le contexte du transfert de style (Li et al., 2017b) ainsi que dans les *GANs* (Cho et al., 2018; Siarohin et al., 2019; Wang et al., 2019a), à notre connaissance, nous sommes les premiers à utiliser cette méthode dans un contexte propre à l'injection de style.

2.1 Transformation blanchissante

La transformation blanchissante est un processus de transformation linéaire visant à décorréler et rendre de variance unitaire les paramètres d'un jeu de données de façon à rendre ceux-ci linéairement indépendants.

Supposons un ensemble de données $\mathcal{X}$ suivant une loi gaussienne multivariée. Une transformation linéaire dite blanchissante est appliquée de sorte à modifier la distribution de la manière suivante

$$\mathcal{X} \sim \mathcal{N}(\vec{\mu}, \mathbf{\Sigma}) \rightarrow \mathcal{Z} \sim \mathcal{N}(\vec{0}, \mathbf{I}). \tag{2.1}$$

Il existe différentes méthodes de blanchissage telles que le blanchissage par analyse des composantes principales (ACP) (Friedman, 1987), le blanchissage de Cholesky ainsi que le blanchissage ZCA (Zero Components Analysis ou blanchissage de Mahalanobis). L'utilisation du

blanchissage ZCA est cependant ici préconisée puisque qu'elle permet de maximiser la similarité entres les variables de l'espace transformé et de l'espace original (Kessy et al., 2018). Dans le cadre du présent travail, l'objectif d'utilisation du blanchissage vise à réaliser la décorrélation des cartes d'activations du réseau de neurones et ainsi rendre de variance unitaire la dimension des canaux C des activations de celui-ci.

Considérons $\mathbf{X} \in \mathbb{R}^{C \times H \times W}$, un tenseur caractérisant les cartes d'activations d'une couche du réseau sur lequel nous appliquons le processus de blanchissage. C est le nombre de canaux et H et W correspondent aux dimensions spatiales des cartes d'activations. On applique une opération de réorganisation des dimensions du tenseur (ψ) afin d'obtenir un vecteur aplati combinant les dimensions spatiales H et W pour chaque canal C. La figure 2.1 illustre le processus de la transformation spatiale ψ.

$$\psi(\mathbf{X}) : \mathbf{X} \in \mathbb{R}^{C \times H \times W} \rightarrow \mathbf{X} \in \mathbb{R}^{C \times HW}. \tag{2.2}$$

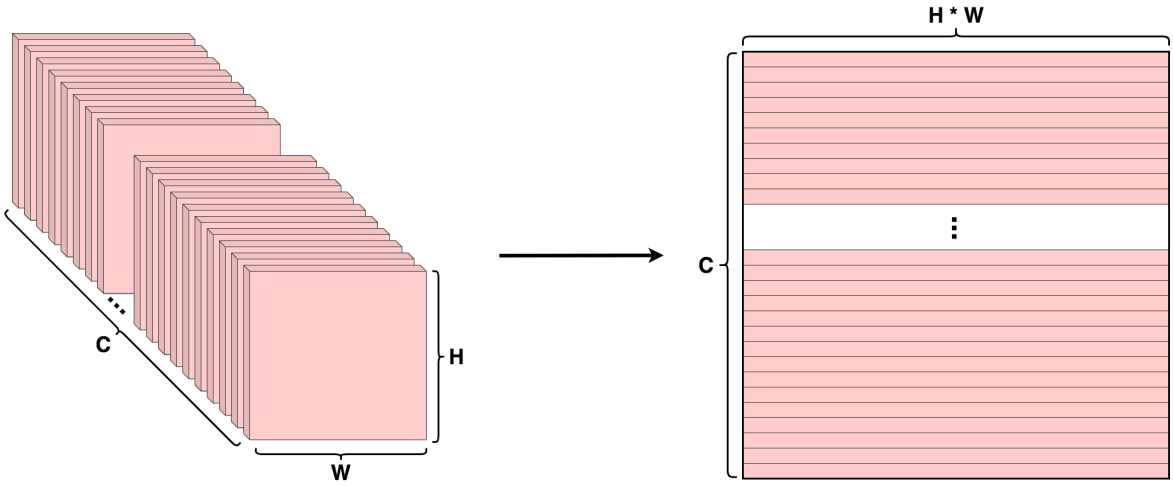

FIGURE 2.1 – Exemple de transformation des dimensions d'un tenseur afin de produire une matrice dont les lignes sont le résultat de la vectorisation des dimensions H et W.

Considérons pour la suite de ce chapitre que les activations ont la forme $\mathbf{X} \in \mathbb{R}^{C \times HW}$ suite à l'application de l'opération ψ. Il devient alors possible d'obtenir la matrice de corrélation-croisée entre les canaux des cartes d'activations en produisant la matrice de covariance que nous notons ici $\text{Cov}(\bar{\mathbf{X}}, \bar{\mathbf{X}}^\intercal) = \mathbf{\Sigma} \in \mathbb{R}^{C \times C}$. La matrice de covariance entre les canaux des cartes d'activations correspond alors à

$$\mathbf{\Sigma} = \frac{1}{HW - 1} \bar{\mathbf{X}} \bar{\mathbf{X}}^\intercal, \tag{2.3}$$

où les valeurs de $\mathbf{X}$ sont centrées $\bar{\mathbf{X}} = \mathbf{X} - \vec{\boldsymbol{\mu}}\, \mathbf{1}_{\text{HW}}^\intercal$ avec $\vec{\boldsymbol{\mu}} = [\mu_1, \mu_2, \ldots, \mu_C]$, qui correspond au vecteur des moyennes par canal.

En ayant recours à la factorisation par décomposition en valeurs singulières, la matrice de covariance $\mathbf{\Sigma}$ peut alors être exprimée de la manière suivante (celle-ci équivaut à la décomposition en valeurs propres puisque la matrice de covariance $\mathbf{\Sigma}$ est symétrique définie semi-positive)

$$\mathbf{\Sigma} = \mathbf{E}\mathbf{D}\mathbf{E}^{*}. \tag{2.4}$$

$\mathbf{E} = [\vec{\boldsymbol{e_1}}, \vec{\boldsymbol{e_2}}, \ldots, \vec{\boldsymbol{e_C}}]$, $\mathbf{D} = diag(\lambda_1, \lambda_2, \ldots, \lambda_C)$ sont respectivement les vecteurs propres et la matrice diagonale des valeurs propres de $\mathbf{\Sigma}$. La matrice de covariance $\mathbf{\Sigma}$ étant symétrique semi-définie positive, les vecteurs propres résultants sont orthogonaux et ainsi $\mathbf{E}^{*} = \mathbf{E}^{\intercal}$.

La transformation blanchissante de Mahalanobis correspond à appliquer successivement une transformation par rotation $\mathbf{E}$ suivi d'un changement d'échelle $\mathbf{D}^{-\frac{1}{2}}$ et finalement une rotation inverse $\mathbf{E}^{\intercal}$ afin d'effectuer un retour dans l'espace original dans lequel résidait l'information contenue dans $\mathbf{X}$.

$$\mathbf{W}_{\text{ZCA}} = \mathbf{\Sigma}^{-\frac{1}{2}} = \mathbf{E}\mathbf{D}^{-\frac{1}{2}}\mathbf{E}^{\intercal} \tag{2.5}$$

L'équation suivante d'écrit l'application du blanchissage sur $\mathbf{X}$ afin d'obtenir des canaux non corrélés.

$$\mathbf{Z} = \mathbf{W}_{\text{ZCA}}\bar{\mathbf{X}}. \tag{2.6}$$

2.2 Blanchissage par méthode itérative

Comme il a été montré à la section 2.1, le processus de blanchissage de Mahalanobis requiert une décomposition en valeurs propres (2.4). Toutefois, dans le contexte où cette méthode est utilisée au sein d'un processus d'optimisation par descente du gradient, le calcul de la dérivée de cette opération de décomposition est essentiel.

Tel que démontré par Ionescu et al. (2015), considérant une fonction de perte quelconque $\mathcal{L}$, la dérivée de la fonction de perte par rapport à la décomposition en valeurs singulières de la matrice de covariance (2.4) correspond à

$$\frac{\partial \mathcal{L}}{\partial \mathbf{\Sigma}} = \mathbf{E}\left\{2\mathbf{D}\left(\mathbf{K}^{\top} \circ \left(\mathbf{E}^{\top}\frac{\partial \mathcal{L}}{\partial \mathbf{E}}\right)\right)_{sym} + \left(\frac{\partial \mathcal{L}}{\partial \mathbf{D}}\right)_{diag}\right\}\mathbf{E}^{\top}, \tag{2.7}$$

où les éléments de la matrice $\mathbf{K}$ sont

$$\mathbf{K}_{ij} = \begin{cases} \frac{1}{\sigma_i - \sigma_j}, & i \neq j \\ 0, & i = j \end{cases}. \tag{2.8}$$

Ici, σ_i et σ_j correspondent à deux valeurs singulières distinctes de la matrice de covariance.

Dans le contexte de la rétro-propagation de l'erreur, chacun des éléments (i, j) de la matrice $\mathbf{K}$ est inversement proportionnel à la différence entre les valeurs singulières i et j de la matrice de covariance $\mathbf{\Sigma}$. Ceci a pour effet, lorsque deux valeurs singulières ont des valeurs très près l'une de l'autre voir même identiques, de rendre le calcul de la matrice $\mathbf{K}$ très sensible à de faibles variations de la différence entre deux valeurs singulières

$$\lim_{\sigma_i - \sigma_j \to 0} \frac{1}{\sigma_i - \sigma_j} = \infty. \tag{2.9}$$

Le gradient peut ainsi devenir numériquement instable et entraîner un échec du processus d'optimisation. Cela a généralement pour effet de détériorer de manière drastique la qualité de l'entraînement du réseau. En effet, lors de l'entraînement du réseau, un seul échec du calcul numérique du gradient de la décomposition en valeurs singulières mène à l'arrêt de l'entraînement puisque des valeurs indéterminées (***NaN***) sont propagées au moment de l'ajustement des poids. Dans un entraînement typique, l'opération de décomposition peut être répétée plusieurs centaines de milliers de fois, il est donc fortement probable que le gradient de l'une de ces opérations s'avère mal conditionné et puisse ainsi mener à l'échec de l'entraînement.

Afin de surmonter cette limitation, il est possible de calculer la matrice de blanchissage $\mathbf{W}_{\text{ZCA}}$ en ayant recours à la méthode itérative couplée de Newton-Schulz (Higham, 2008) en suivant l'approche de Li et al. (2017a). Newton-Schulz est une méthode itérative permettant de calculer la racine carrée ainsi que la racine carrée inverse d'une matrice.

$$\text{Newton-Schulz}(\mathbf{\Sigma}) \to (\mathbf{\Sigma}^{-\frac{1}{2}}, \mathbf{\Sigma}^{\frac{1}{2}}) \tag{2.10}$$

Puisque $\mathbf{W}_{\text{ZCA}} = \mathbf{\Sigma}^{-\frac{1}{2}}$, en appliquant la méthode itérative sur $\mathbf{\Sigma}$, il est possible d'obtenir $\mathbf{W}_{\text{ZCA}}$ de manière directe et ce sans avoir recours à la méthode de décomposition en valeurs propres. L'instabilité numérique décrite plus haut est ainsi évitée à l'aide de la méthode itérative. L'algorithme 1 décrit les étapes du processus itératif de Newton-Schulz.

Algorithme 1 : Méthode itérative de Newton-Schulz

Input : Matrice de covariance $\mathbf{\Sigma}$, nombre d'itérations n, ϵ

Output : Matrice racine carré inverse $\mathbf{W}_{\text{ZCA}}$

$\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} + \epsilon\mathbf{I}$;

$\mathbf{Y_0} \leftarrow \frac{\mathbf{\Sigma}}{\|\mathbf{\Sigma}\|_F}$;

$\mathbf{Z_0} \leftarrow \mathbf{I}$;

for $i = 1 : n$ **do**

  $\mathbf{Y}_{n+1} \leftarrow \frac{1}{2}\mathbf{Y}_n(3\mathbf{I} - \mathbf{Z}_n\mathbf{Y}_n)$;

  $\mathbf{Z}_{n+1} \leftarrow \frac{1}{2}(3\mathbf{I} - \mathbf{Z}_n\mathbf{Y}_n)\mathbf{Z}_n$;

end

$\mathbf{W}_{\text{ZCA}} \leftarrow \frac{\mathbf{Z}_{n+1}}{\sqrt{\|\Sigma\|_F}}$;

Dans le but d'améliorer la stabilité de l'algorithme 1, un estimateur de contraction (Schäfer and Strimmer, 2005) se basant sur l'ajout d'une matrice de régularisation à la matrice de covariance empirique est ajouté afin d'en améliorer le conditionnement. Cet estimateur correspond à une matrice identité pondérée d'un facteur $\epsilon > 0$, une faible valeur positive : $\mathbf{\Sigma} \leftarrow \mathbf{\Sigma} + \epsilon\mathbf{I}$.

Pré-normalisation De plus, il est important de s'assurer de la convergence de la méthode de Netwon-Schulz puisque celle-ci ne convergent que localement selon le critère de convergence $\|\mathbf{I} - \mathbf{\Sigma}\|_p < 1$ pour $p = 1, 2$ ou ∞ (Higham, 2008). Afin de respecter ce critère, une mise à l'échelle de la matrice de covariance est préalablement appliquée (Huang et al., 2018a). Cette pré-normalisation réduit l'échelle de $\mathbf{\Sigma}$ selon sa norme de Frobenius $\|\mathbf{\Sigma}\|_F = \sqrt{\sum_j \sigma_j^2}$.

La démonstration suivante consolide la justification de cette pré-normalisation. Notons ici $\mathbf{\Sigma} \in \mathbb{R}^{a\times a}$ la matrice d'intérêt et $\mathbf{I} \in \mathbb{R}^{a\times a}$ une matrice identité alors $\mathbf{A} = \mathbf{I} - \frac{\mathbf{\Sigma}}{\|\mathbf{\Sigma}\|_F}$ à des fins de simplification. La condition de convergence dans le cas $p = 2$ correspond alors à $\|\mathbf{A}\|_2 < 1$.

Pour toute matrice $\mathbf{B} \in \mathbb{R}^{b\times b}$ ayant des valeurs propres λ_i ainsi que des vecteurs propres ν_i pour $i \in \{1, \ldots, b\}$, pour tout scalaire c alors $\lambda_b + c$ correspond aux valeurs propres de la matrice résultantes de la somme des matrices $\mathbf{B} + c\mathbf{I}$.

Suivant ce résultat, il est possible de conclure que les valeurs propres de $\mathbf{A}$ correspondent à

$$\lambda_i(\mathbf{A}) = 1 - \frac{\sigma_i(\mathbf{\Sigma})}{\sqrt{\sum_j \sigma_j(\mathbf{\Sigma})^2}}, \tag{2.11}$$

pour $i \in \{1, \ldots, a\}$.

La matrice $\mathbf{A}$ étant symétrique, ses valeurs singulières correspondent à $\sigma_i = |\lambda_i|$ et donc

$$\sigma_i = \left| 1 - \frac{\sigma_i}{\sqrt{\sum_j \sigma_j^2}} \right|. \tag{2.12}$$

Considérant la norme matricielle $\|\mathbf{A}\|_2 = \sigma_{max}(\mathbf{A})$, le critère de convergence $\|\mathbf{A}\|_2 < 1$ peut être exprimé $\sigma_{max}(\mathbf{A}) < 1$.

Étant donné que l'on cherche la valeur singulière maximale de la matrice $\mathbf{A}$ correspondant à la différence de $1 - \frac{\sigma_a}{\sqrt{\sum_j \sigma_j^2}}$, σ_a sera par conséquent la plus petite valeur singulière (a-ième valeur singulière) de la matrice $\frac{\mathbf{\Sigma}}{\|\mathbf{\Sigma}\|_F}$.

Il est possible d'observer que la plus grande valeur singulière de la matrice $\mathbf{A}$ correspond à

$$\left| 1 - \frac{\sigma_a}{\sqrt{\sum_j \sigma_j^2}} \right| < 1. \tag{2.13}$$

Le résultat de cette équation est toujours inférieur à 1 puisque la division d'une des valeurs singulière par la somme de toutes les valeurs singulière produit un résultat inférieur à 1. Cette conclusion permet d'assurer un respect de la condition de convergence. De plus, l'ajout de l'estimateur de contraction ϵ garantie que le résultat ne puisse égaler 1 et donc briser la condition de convergence.

Post-compensation L'utilisation de la pré-compensation nécessite une post-compensation afin de compenser le résultat obtenu suite à la normalisation. Effectivement, le résultat obtenu à l'aide de la méthode itérative ne correspond pas à $\sqrt{\mathbf{\Sigma}}$ mais plutôt à $\sqrt{\frac{\mathbf{\Sigma}}{\|\mathbf{\Sigma}\|_F}}$. Il est donc essentiel de post-compenser en multipliant le résultat obtenu à l'aide de la racine de la norme de la façon suivante

$$\sqrt{\|\mathbf{\Sigma}\|_F}\sqrt{\frac{\mathbf{\Sigma}}{\|\mathbf{\Sigma}\|_F}}. \tag{2.14}$$

L'utilisation de la norme de Frobenius a été ici préconisée puisque celle-ci est moins encline à causer une surestimation de l'erreur à la post-compensation avec des matrices de plus grandes tailles. Effectivement, considérant une erreur induite par la méthode itérative, celle-ci sera accentuée par la post-compensation. Celle-ci dépend de l'échelle des valeurs singulières plutôt que de la taille des matrices et donc même avec de grandes matrices, le résultat de la norme n'est pas nécessairement élevé, ce qui réduit l'accentuation de l'erreur provenant de la méthode itérative.

2.3 Processus de coloration et échantillonnage de matrice de coloration

Le processus de coloration correspond à l'opération inverse du blanchissage. L'application d'une matrice de coloration permet d'affecter de nouvelles statistiques à un ensemble de données blanchies. Soit $i \in I$, une couche particulière de l'ensemble des couches où l'injection de style est réalisée, il est possible d'appliquer la coloration en utilisant une matrice de coloration notée $\mathbf{\Gamma}_i$ ainsi qu'un vecteur de moyennes $\vec{\boldsymbol{\mu}}_i$. L'application du blanchissage suivi de la coloration pour la couche $i \in I$ peut être décrite comme étant

$$\tilde{\mathbf{X}}_i = \mathbf{\Gamma}_i \mathbf{Z}_i + \vec{\boldsymbol{\mu}}_i \mathbf{1}_{\mathrm{HW}}^{\intercal}. \tag{2.15}$$

$\tilde{\mathbf{X}}_i$ sont les cartes d'activation recolorées et $\mathbf{Z}_i = \mathbf{W}_{\mathrm{ZCA}i}(\mathbf{X}_i - \boldsymbol{\mu}(\mathbf{X}_i)\ \mathbf{1}_{\mathrm{HW}}^{\intercal})$ sont les cartes d'activations blanchies pour la couche i. La figure 2.2 a) illustre le fonctionnement du module d'injection de style tel qu'utilisé à l'intérieur d'un réseau générateur.

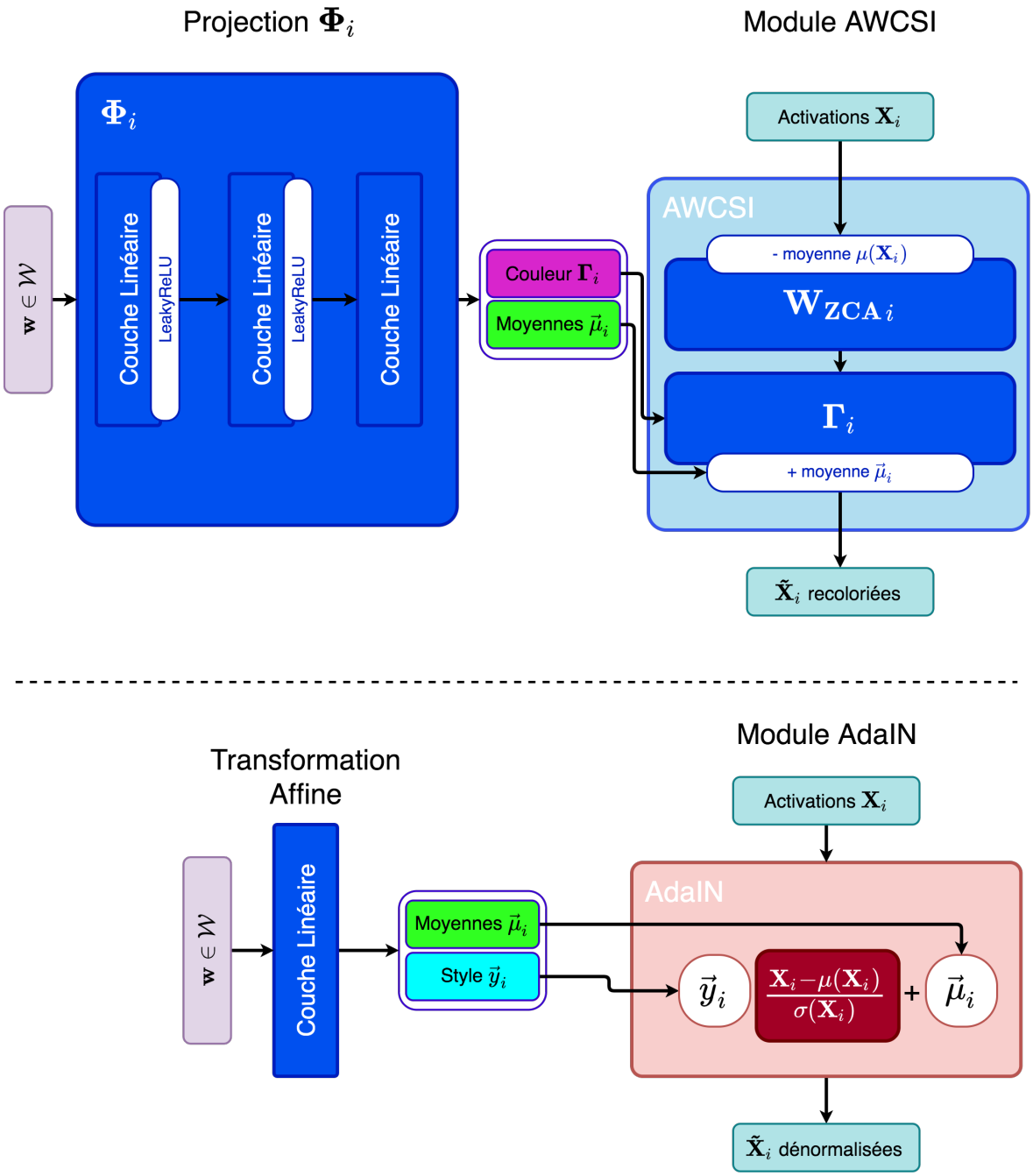

FIGURE 2.2 – **a)** Injection de style à l'aide du module *AWCSI*. Une projection du vecteur latent $\mathbf{w}$ vers une matrice de coloration est effectué afin de permettre la coloration des activations blanchies. **b)** Module *AdaIN* permettant la normalisation et la dénormalisation des activations à l'aide d'un vecteur de style et d'un vecteur de moyennes. Les processus de normalisation et de dénormalisation sont appliqués par canal.

Dans le but d'appliquer une matrice de coloration provenant du style aux activations du réseau générateur, il faut tout d'abord permettre l'échantillonnage de matrices de coloration

et de moyennes au cours de l'entraînement. Afin de minimiser les modifications apportées aux architectures existantes, un vecteur latent $\mathbf{z} \in \mathcal{Z}$ est échantillonné puis traduit à l'aide d'un réseau de traduction (*mapping network*) $f : \mathcal{Z} \rightarrow \mathcal{W}$ afin de produire un vecteur de latent $\mathbf{w} \in \mathcal{W}$. Celui-ci est ensuite utilisé à chaque couche d'injection afin d'être transformé vers une matrice de coloration $\mathbf{\Gamma}_i$ et un vecteur de moyennes $\vec{\boldsymbol{\mu}}_i$ au moyen d'un réseau de projection Φ_i. La projection peut être définie comme suit

$$\mathbf{\Phi}_i(\mathbf{w}) = \mathbf{\Gamma}_i, \vec{\boldsymbol{\mu}}_i, \tag{2.16}$$

où $\mathbf{\Gamma}_i \in \mathbb{R}^{C \times C}$ et $\vec{\boldsymbol{\mu}}_i \in \mathbb{R}^C$ sont respectivement la matrice de coloration (*style*) et le vecteur de moyennes pour la couche i. Il est important de mentionner que le vecteur latent $\mathbf{w}$ est partagé par chaque couche d'injection. Celui-ci est toutefois traduit vers des matrices de coloration et des vecteurs de moyennes différents pour chaque couche i. La figure 2.3 montre un exemple général d'injection de style avec la méthode *AWCSI*.

Cette approche permet de réduire les modifications à apporter aux architectures déjà existantes en laissant intact le réseau de traduction f et ne modifiant que l'injection de style du réseau générateur G. L'intégration du module d'injection de style aux travaux existants est alors grandement simplifiée.

Comparaison avec *AdaIN* La figure 2.2 présente les différences que comporte notre méthode (illustrée à la figure 2.2 a)) avec le module *AdaIN* qui est actuellement la méthode de base utilisée afin de réaliser l'injection de style (présentée à la figure 2.2 b)). *AWCSI* utilise une matrice de coloration $\mathbf{\Gamma}_i$ produite à l'aide d'un réseau de projection afin d'introduire de nouvelles corrélations inter-canaux dans les cartes d'activations en comparaison à *AdaIN* qui ignore l'impact des corrélations entres les canaux et n'applique qu'une simple standardisation.

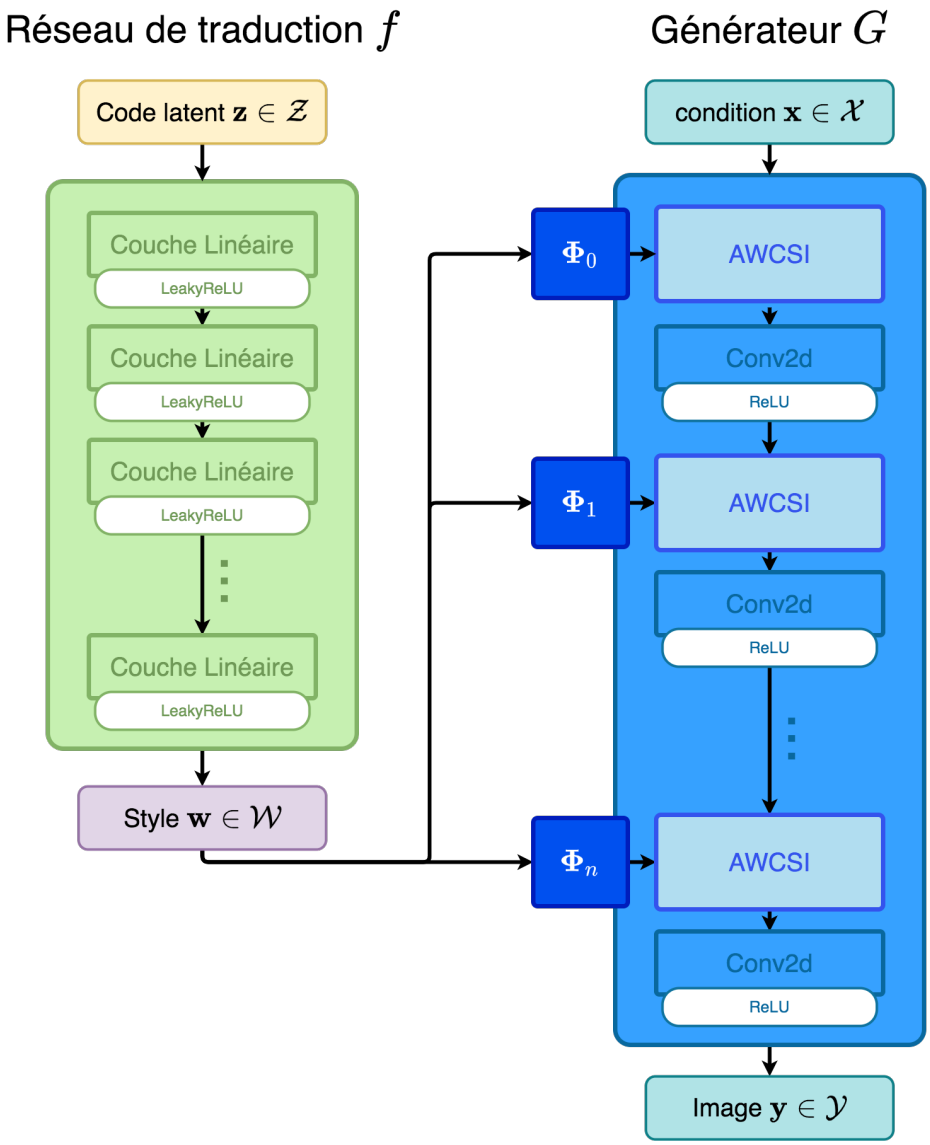

FIGURE 2.3 – Injection du *style* dans un générateur standard à l'aide du module d'injection AWCSI. Un réseau de projection $\mathbf{\Phi}_i(\mathbf{w})$ est utilisé pour chaque module d'injection afin transférer le vecteur $\mathbf{w}$ vers l'espace de la matrice de coloration $\mathbf{\Gamma}_i$ et du vecteur de moyennes $\vec{\boldsymbol{\mu}}_i$. Le vecteur de moyennes est de longueur égale au nombre de canaux de la couche d'activations.

2.4 Blanchissage et coloration par groupe

La génération d'une matrice de coloration à l'aide de la projection du vecteur de style peut s'avérer être une tâche de calcul difficile nécessitant un réseau de projection ayant un grand nombre de paramètres entraînables. Dans le cas de l'injection de style dans un couche du générateur où le nombre de canaux des cartes d'activations s'élève à 256, la production de la matrice de coloration 256×256 requiert à elle seule que la dernière couche du réseau de projection possède $256 \times 256 \times 512 = 33\ 554\ 432$ paramètres entraînables. Cela se base sur le

fait que la couche cachée qui précède la sortie comporte 512 neurones. Ainsi, considérant que ce nombre de paramètres soit nécessaire pour chacun des modules d'injection et que certains générateurs peuvent nécessiter plus d'une dizaine de modules d'injection de style (Karras et al., 2019), il devient alors inconcevable de requérir à un nombre aussi grand de paramètres. Nous proposons donc de procéder par groupe de canaux (Huang et al., 2018a, 2017) plutôt que d'effectuer le processus de blanchissage et de coloration sur l'ensemble des canaux. Cette méthode vise à réduire la charge de calcul nécessaire à l'entraînement du générateur.

L'objectif est de regrouper les canaux en sous-ensemble afin de produire des sous-groupes de canaux qui seront blanchis et coloriés en faisant abstraction des corrélations pouvant exister entre les canaux ne faisant pas parties d'un même groupe. Cela permet de produire des matrices de coloration beaucoup plus petites et ainsi réduire le nombre de paramètres à entraîner pour chaque réseau de projection $\mathbf{\Phi}_i$. La matrice de coloration $\mathbf{\Gamma}_i \in \mathbb{R}^{C \times C}$ devient alors une matrice par blocs composée de $\boldsymbol{n}$ blocs (ou groupes) de sous-matrices de coloration $\mathbf{\Gamma}_{ji}$ sur la diagonale où $j = 1...n$ correspond à l'indice du groupe. La taille des sous-matrices de coloration peut alors être exprimée comme étant $G = \frac{C}{n}$.

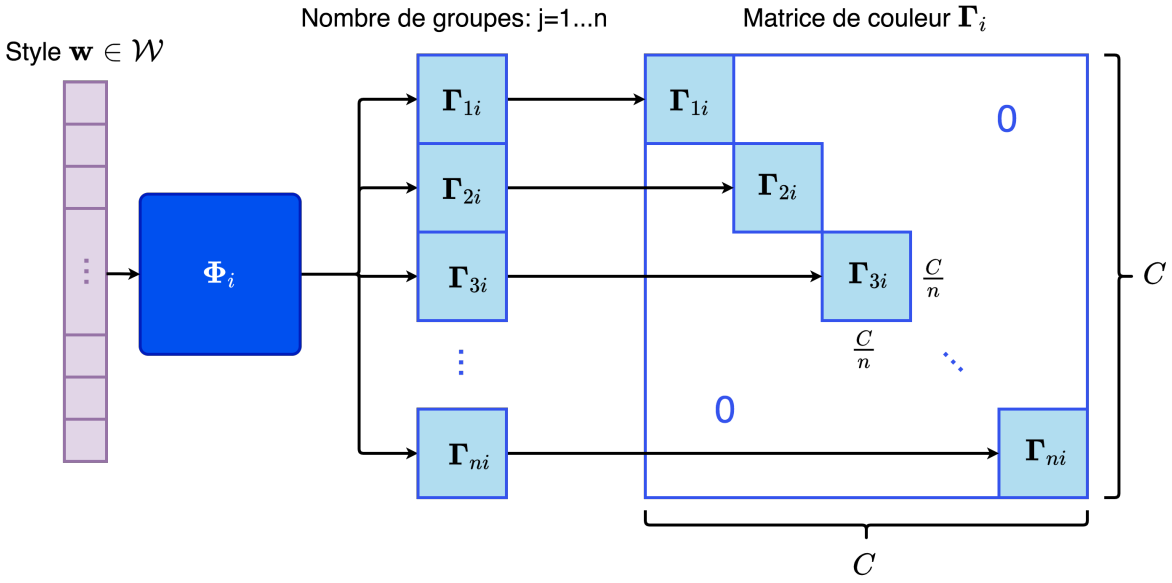

FIGURE 2.4 – Projection $\mathbf{\Phi}_i$ du vecteur latent de *style* $\mathbf{w}$ vers $\mathbf{n}$ sous-matrices $\mathbf{\Gamma}_{ji}$ de taille $\frac{C}{n} \times \frac{C}{n}$ afin de produire la matrice de coloration par blocs $\mathbf{\Gamma}_i$ de taille $C \times C$.

Afin d'appliquer le blanchissage et la coloration des cartes d'activations du générateur par groupe, il est essentiel de procéder à une opération de regroupement des cartes d'activations en n groupes de taille G. Ainsi, en se basant sur la taille des canaux C des cartes d'activations, il est possible de déterminer le nombre de groupes $n = \frac{C}{G}$ et de réorganiser les dimensions des cartes d'activations de la manière suivante.

Notons l'opération de regroupement Ω, alors la réorganisation des dimensions peut être décrite

comme étant,

$$\Omega(\mathbf{X}) : \mathbf{X} \in \mathbb{R}^{C\times H\times W} \rightarrow \mathbf{X} \in \mathbb{R}^{n\times G\times H\times W}. \tag{2.17}$$

Cela permet de diviser les cartes d'activation en $\boldsymbol{n}$ groupes de cartes d'activations que nous pouvons noter $g_j \in \mathbb{R}^{G\times H\times W}$ avec $j = 1 \ ... \ n$. Il est alors possible de déterminer les sous-matrices de corrélations

$$\psi(g_j)\psi(g_j)^\intercal = \mathbf{\Sigma}_j \in \mathbb{R}^{G\times G}, \tag{2.18}$$

formant la matrice de corrélation par blocs des cartes d'activations $\mathbf{\Sigma} \in \mathbb{R}^{C\times C}$. L'opération ψ correspond à la vectorisation des dimensions spatiales. La figure 2.5 décrit l'application du processus de construction de la matrice de corrélation par blocs. Une fois la matrice de corrélation par blocs des cartes d'activations $\mathbf{\Sigma}$ construite, il est possible d'appliquer le processus de blanchissage aux cartes d'activations telle que décrit aux sections 2.1 et 2.2. La coloration par bloc peut ensuite être effectuée à l'aide de la matrice de coloration par blocs $\mathbf{\Gamma}$ en suivant la méthode décrite à la section 2.3. L'algorithme 2 présente la séquence d'opérations nécessaire à l'application de la méthode *AWCSI* par groupe.

Algorithme 2 : Grouped Adaptive Whitening Coloring Style Injection

Input : Carte d'activations $\mathbf{X}$, vecteur de style $\mathbf{w}$, $n = \frac{C}{G}$ groupes

Output : Carte d'activations coloriées $\tilde{\mathbf{X}}$

$\mathbf{\Gamma}, \vec{\boldsymbol{\mu}} \leftarrow \mathbf{\Phi}(\mathbf{w})$;

g_j pour $j \in \{1, \ldots, n\} \leftarrow \Omega(\mathbf{X})$;

for $j = 1 : n$ **do**

  $\mathbf{\Sigma}_j \in \mathbb{R}^{G\times G} \leftarrow \psi(g_j)\psi(g_j)^\intercal$;

end

Construction de la matrice par blocs à l'aide des sous matrices $\mathbf{\Sigma}_j$ pour $j \in \{1, \ldots, n\}$.

$$\mathbf{\Sigma} = \begin{bmatrix} \mathbf{\Sigma}_1 & & 0 \\ & \ddots & \\ 0 & & \mathbf{\Sigma}_n \end{bmatrix} \in \mathbb{R}^{C\times C};$$

$\mathbf{W}_{\text{ZCA}} \leftarrow$ Newton-Schulz($\mathbf{\Sigma}$);

$\mathbf{Z} \leftarrow \mathbf{W}_{\text{ZCA}}(\mathbf{X} - \boldsymbol{\mu}(\mathbf{X})\mathbf{1}_{\text{HW}}^\intercal)$;

$\tilde{\mathbf{X}} \leftarrow \mathbf{\Gamma Z} + \vec{\boldsymbol{\mu}}\mathbf{1}_{\text{HW}}^\intercal$;

Il est important de mentionner qu'en procédant par groupe de canaux, les corrélations hors du groupe sont ignorées puisque le processus de blanchissage et de coloration n'est appliqué que sur les corrélations existantes au sein d'un même groupe d'activations. Certaines corrélations présentes entre des canaux n'appartenant pas au même groupe peuvent donc persister dans le

processus puisque aucun traitement sur ces corrélations n'est effectué. Ainsi, en faisant varier la taille des groupes (G), il est possible d'obtenir différents niveaux d'application du blanchissage et du coloriage. Par exemple, si la taille des groupes est égale au nombre de canaux des cartes d'activations ($G = C$) alors l'application de la méthode correspond à une coloration et un blanchissage complet des canaux sans blocs. En opposition, dans le cas où $G = 1$, cela revient à appliquer la méthode d'*AdaIN* où la corrélation entre les canaux est entièrement ignorée. La section 3.2.4 des expérimentations fait l'analyse de l'impact de la taille des groupes sur les résultats obtenus.

Sommaire de la méthode

Ce chapitre fait la description de la méthode proposée d'injection de style au sein d'un réseau générateur. Plus particulièrement, on y décrit l'application d'un processus de blanchissage ZCA itératif (basé sur Newton-Schulz) des canaux des cartes d'activations suivie d'un processus de coloration des cartes d'activations par groupes afin de réduire la charge de calculs étant attribués à la génération des matrices de coloration. La méthode proposée peut être vue comme une généralisation de *AdaIN* (Huang and Belongie, 2017), qui est une méthode populaire d'injection de style au sein des réseaux génératifs, puisqu'elle produit un changement de la covariance des activations en appliquant une transformation linéaire lors des processus de blanchissage et de coloration en comparaison à *AdaIN* qui ne s'intéresse qu'à la modification de la variance.

La section suivante présente une analyse de la méthode à travers une série d'expériences qui permettent de mettre en lumière l'impact de l'*AWCSI* et de ses composantes sur la génération d'images dans un contexte de transfert de domaine.

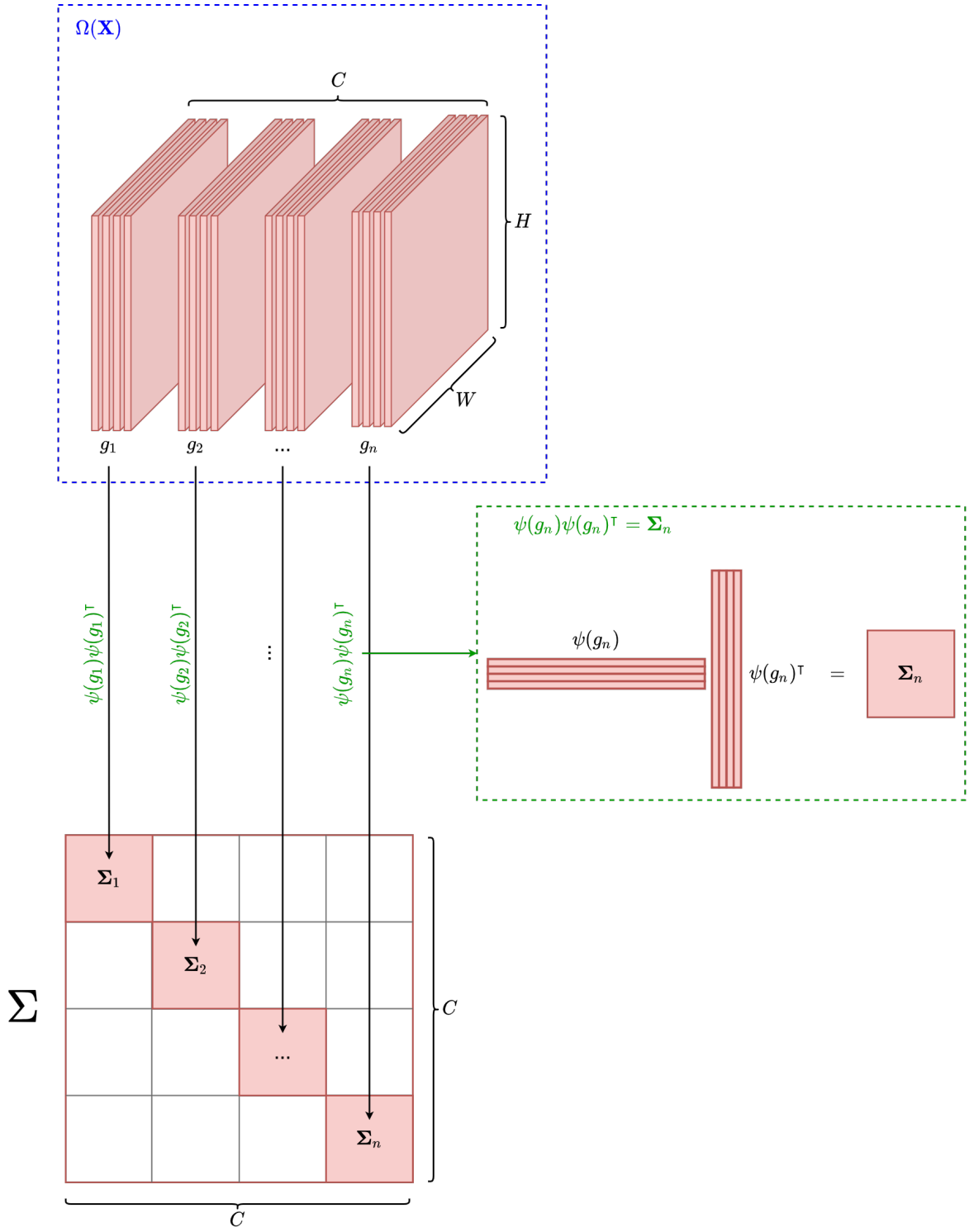

FIGURE 2.5 – Construction de la matrice de corrélation des activations $\mathbf{X}$ par blocs en appliquant l'opération de regroupement $\Omega(\mathbf{X})$ suivie de l'application de l'opération de vectorisation ψ sur chaque groupe pour construire les $\boldsymbol{n}$ ($j = 1 \ ... \ n$) sous-matrices de corrélations $\psi(g_j)\psi(g_j)^\intercal = \mathbf{\Sigma}_j$.

Chapitre 3

Analyse des expériences et résultats

Dans le but d'évaluer les effets de la méthode décrite au chapitre 2, une série d'expérimentations est réalisé dans différentes configurations. L'analyse des résultats qui en découle est ensuite présentée afin de permettre de tirer des conclusions sur les bénéfices apportés par notre approche.

3.1 Entraînement en configuration d'images appariées

L'objectif de cette section est d'étudier l'impact de la méthode développée et décrite au chapitre 2 dans une configuration d'entraînement d'un modèle d'*image-to-image* supervisé. Concrètement, l'entraînement est dit supervisé puisque chacune des données provenant de l'ensemble d'entraînement $(x_i, y_i) \in \{(x_1, y_1), \ldots, (x_n, y_n)\}$ est une paire d'images (entrée, cible) qui permet la construction de la fonction de traduction entre les images du domaine en entrée $x \in \mathcal{X}$ et leur correspondance $y \in \mathcal{Y}$ dans l'ensemble du domaine cible.

Nous présentons d'abord le jeu de données utilisé et la configuration de l'architecture employée afin de produire l'ensemble des résultats. Nous exposons et discutons finalement l'ensemble des résultats quantitatif et qualitatifs obtenus afin réaliser une comparaison de notre méthode (*AWCSI*) à la méthode d'injection de style actuellement utilisée : *AdaIN*.

3.1.1 Jeu de données

Le modèle utilisé pour faire la génération d'images appariées est entraîné sur un jeu de données provenant de Isola et al. (2017). Ce jeu de données effectue des correspondances entre des photos satellites obtenues de *Google Maps* et les cartes correspondantes à ces images satellites. Les images originales ont une taille de 512×512 pixels mais sont toutefois redimensionnées à une taille de 256×256 pixels pour l'entraînement. La figure 3.1 montre des exemples des images correspondantes utilisées pour l'entraînement. Le jeu de données est composé de 1096 images pour réaliser l'entraînement ainsi que de 1098 images afin de réaliser la phase de

validation. L'ensemble des images de validation est nécessaires puisqu'il permet de réaliser une évaluation quantitative et qualitative qui est indépendante des exemples utilisés lors de la phase d'entraînement. Effectivement, l'appariement des images en entraînement peut amener à un sur-apprentissage et donc à des résultats de faible qualité lorsque le modèle doit traduire des exemples qu'il n'a jamais « vu ». Cela cause généralement une dégradation des résultats pour de nouveaux exemple. Il est donc essentiel de valider la méthode à l'aide de données qui n'ont pas été utilisés lors du processus d'entraînement. Ainsi, l'ensemble des résultats présentés dans les sections suivantes proviennent de données du jeu de validation.

FIGURE 3.1 – Jeu de données des photos aériennes (première rangée) et de leurs représentations cartographiques correspondantes (deuxième rangée).

3.1.2 Détails de l'implémentation de l'architecture

Cette section présente l'architecture utilisée afin de réaliser les expérimentations sur le jeu de données des cartes satellite.

Générateur L'architecture du générateur peut être exprimée sous la forme d'un encodeur-décodeur qui compresse l'information en entrée vers un goulot d'étranglement pour ensuite effectuer un décodage de l'information compressée. Ce modèle comprend une série de blocs résiduels qui effectues une augmentation du nombre de canaux tout en diminuant la taille spatiale (sous-échantillonnage) des cartes d'activations. Seule, la section des blocs résiduels qui augmente la dimension spatiale (sur-échantillonnage) des cartes d'activations effectue l'in-

jection de style. La figure 3.2 illustre l'architecture du générateur, qui est également détaillée dans la table 3.1.

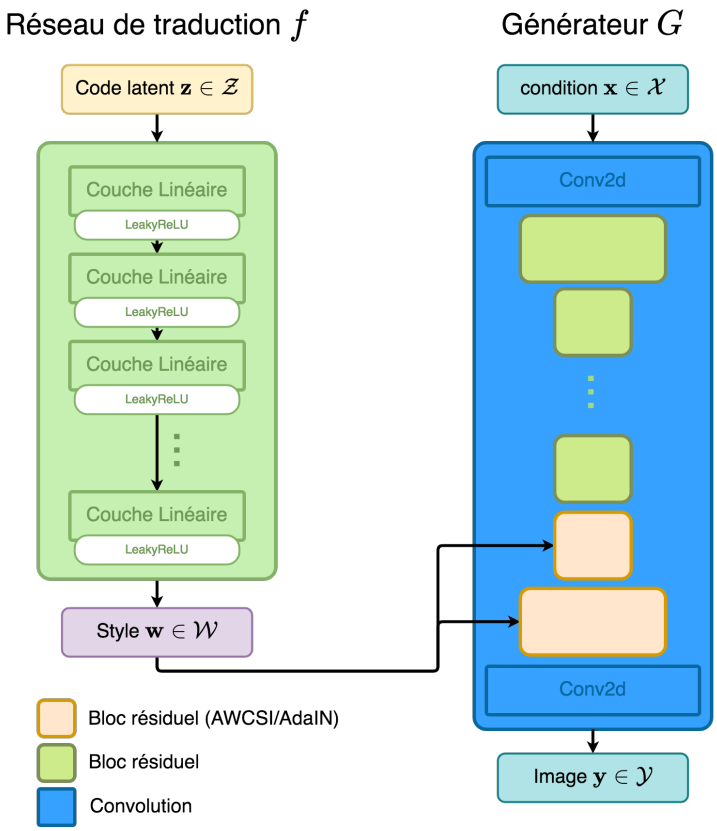

FIGURE 3.2 – Le réseau générateur est composé d'une série de blocs résiduels qui encode et décode l'information entrée sous la forme d'un sablier. L'injection du style dans le réseau se fait à l'aide d'un module d'injection (*AWCSI* ou *AdaIN*) dont l'information du vecteur de style $w \in \mathcal{W}$ provient du *mapping network* f.

Couche	**Échantillonnage**	**Normalisation**	**Taille de sortie** $[H \times W \times C]$
Image	-	-	$256 \times 256 \times 3$
Conv 3×3	-	-	$256 \times 256 \times 64$
Bloc Résiduel	Average Pool	IN	$128 \times 128 \times 128$
Bloc Résiduel	Average Pool	IN	$64 \times 64 \times 256$
Bloc Résiduel	-	IN	$64 \times 64 \times 256$
Bloc Résiduel	-	IN	$64 \times 64 \times 256$
Bloc Résiduel	-	IN	$64 \times 64 \times 256$
Bloc Résiduel	-	IN	$64 \times 64 \times 256$
Bloc Résiduel	Upsample(bilinéaire)	*AWCSI/AdaIN*	$128 \times 128 \times 128$
Bloc Résiduel	Upsample(bilinéaire)	*AWCSI/AdaIN*	$256 \times 256 \times 64$
Conv 1×1	-	IN	$256 \times 256 \times 3$

TABLE 3.1 – Architecture détaillée du réseau générateur. L'échantillonnage correspond à la méthode utilisée afin de réaliser une modification de la dimension spatiale des cartes d'activations (sur ou sous-échantillonnage). La normalisation correspond à l'*Instance Normalization* (IN) (Ulyanov et al., 2016) ou à une des méthodes d'injection de style (*AWCSI/AdaIN*) dans le générateur.

Réseau de traduction (*mapping network*) Le *mapping network*, basé sur le travail de Karras et al. (2019), correspond à réseau composé de couches linéaires pleinement connectées. Celui-ci transforme un code latent $\mathbf{z} \in \mathcal{Z}$ (64 dimensions) vers un vecteur latent w (128 dimensions). Ce réseau correspond à une fonction non linéaire qui peut être définie comme suit, $f : \mathcal{Z} \rightarrow \mathcal{W}$, où $\mathbf{w} \in \mathcal{W}$. Le bénéfice majeur de l'utilisation du *mapping network* est qu'il permet une flexibilité de la représentation de la distribution $\mathcal{W}$. Effectivement, puisque $f(\mathbf{z}) \rightarrow \mathbf{w}$ est une fonction non-linéaire apprise, cela rend les facteurs de variation des caractéristiques davantage linéaires. Cela est dû au fait que l'espace $\mathcal{W}$ n'est pas échantillonné d'une distribution fixe comme $\mathcal{Z}$. La figure 3.3 illustre l'impact de la fonction de traduction sur la représentation des caractéristiques d'un jeu de données quelconque. Le réseau est décrit de manière exhaustive à la table 3.2.

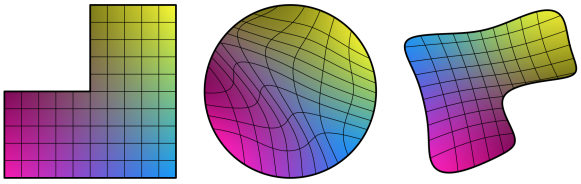

FIGURE 3.3 – À la gauche, la distribution des caractéristiques du jeu d'entraînement, au milieu, la traduction de $\mathcal{Z}$ vers ces mêmes caractéristiques et à droite, la traduction de $\mathcal{W}$ vers les caractéristiques. Figure tirée de Karras et al. (2019).

Couche	Activation	Taille de sortie
z Latent	-	64
7× Linéaire	7× LeakyReLU	512
⋮	⋮	⋮
Linéaire	-	128

TABLE 3.2 – Architecture détaillée du "mapping network". Composée de 7 couches linéaires de 512 neurones et d'une couche de 128 sorties.

Discriminateur Le discriminateur suit l'architecture standard *PatchGAN* telle que développée par Isola et al. (2017). Le champ réceptif de chaque activation en sortie correspond à un carré de 70×70 pixels dans l'image d'entrée. Cela permet ainsi au discriminateur d'évaluer le réalisme d'une image selon des zones de taille 70×70 pixels. La table 3.3 décrit de manière exhaustive l'architecture utilisée.

Couche [pad, stride]	**Normalisation**	**Activation**	**Taille de sortie** $[H \times W \times C]$
Image	-	-	$256 \times 256 \times 3$
Conv 4×4 [1, 2]	-	LeakyReLU	$128 \times 128 \times 64$
Conv 4×4 [1, 2]	IN	LeakyReLU	$64 \times 64 \times 128$
Conv 4×4 [1, 2]	IN	LeakyReLU	$32 \times 32 \times 256$
Conv 4×4 [3, 1]	IN	LeakyReLU	$32 \times 32 \times 512$
Conv 4×4 [3, 1]	-	-	$32 \times 32 \times 1$

TABLE 3.3 – Architecture détaillée du réseau discriminateur. La normalisation correspond à l'*Instance Normalization* (IN) (Ulyanov et al., 2016)

3.1.3 Objectif d'entraînement

Lors de l'entraînement du réseau, un ensemble de fonctions de pertes ayant pour but de guider l'objectif final à atteindre sont utilisées. La section suivante fait un bref survol de chacune d'entre elles.

Fonction de supervision L'objectif de cette fonction est d'assurer une bonne correspondance entre l'image générée par le réseau générateur $G(\mathbf{x}, \mathbf{z})$ et l'image cible $\mathbf{y}$

$$\mathcal{L}_{\text{L1}} = \mathbb{E}_{\mathbf{x},\mathbf{y},\mathbf{z}} \left[\|\mathbf{y} - G(\mathbf{x}, \mathbf{z})\|_1 \right]. \tag{3.1}$$

Cela correspond à une différence absolue de la valeur par pixel entre les deux images.

Fonction de perte antagoniste Cette fonction de perte a pour objectif de réaliser l'entraînement du discriminateur D et du générateur G dans un scénario de compétition de la

théorie des jeux tel que décrit à la section 1.1.2

$$\mathcal{L}_{\text{cGAN}} = \mathbb{E}_{\mathbf{x}}[\log D(\mathbf{x}, \mathbf{y})] + \mathbb{E}_{\mathbf{x},\mathbf{z}}[\log(1 - D(\mathbf{x}, G(\mathbf{x}, \mathbf{z})))], \tag{3.2}$$

où $\mathbf{x}$, $\mathbf{y}$ et $\mathbf{z}$ sont respectivement l'image en entrée (condition), l'étiquette associée à cette dernière et le vecteur latent.

Régularisation de la diversité des résultats Cette régularisation force le générateur à produire des résultats diversifiés pour une même image en entrée du générateur ayant cependant des codes latents distincts z_i pour $i \in \{1, 2\}$. Ainsi, maximiser cette régularisation pousse le générateur à explorer le domaine cible pour différents des codes latents

$$\mathcal{L}_{ds} = \mathbb{E}_{\mathbf{x},\mathbf{z}_1,\mathbf{z}_2} \left[\|G(\mathbf{x}, \mathbf{z}_1) - G(\mathbf{x}, \mathbf{z}_2)\|_1 \right]. \tag{3.3}$$

Objectif global L'objectif d'entraînement du réseau peut être exprimé comme étant

$$\min_{G,f} \max_{D} \mathcal{L}_{cGAN} + \lambda_{L1}\mathcal{L}_{L1} - \lambda_{ds}\mathcal{L}_{ds}, \tag{3.4}$$

avec λ_{L1} et λ_{ds}, les pondérations des fonctions de pertes. Notons que la fonction de perte $\mathcal{L}_{ds}$ est négative. En effet, puisque l'ensemble du processus d'optimisation est une minimisation et que nous voulons maximiser $\mathcal{L}_{ds}$, il est nécessaire d'utiliser son inverse.

3.1.4 Comparaison quantitative et qualitative des résultats

Cette section présente une analyse comparative des résultats quantitatifs et qualitatifs obtenus à l'aide de l'architecture décrite en 3.1.2 sur le jeu de données (3.1.1) entre les méthodes d'injection de style *AdaIN* et *AWCSI*. Une évaluation des métriques est tout d'abord effectuée suite à laquelle une comparaison qualitative est réalisée dans le but de consolider visuellement les métriques obtenues.

Analyse comparative quantitative

La table 3.4 rapporte les valeurs de *FID* (définit selon l'équation 1.11) et de *LPIPS* (présenté à la section 1.3.1) tel que décrit dans le travail de Zhu et al. (2017b). Les résultats sont présentés pour la méthode AWCSI ainsi que pour la méthode de référence, AdaIN. Il est possible de remarquer que AWCSI obtient des valeurs supérieures pour les méthodes de *FID* et de *LPIPS*. En effet, avec inspection visuelle des résultats générés, la qualité des images produites en utilisant AWCSI pour l'injection de style préférable à celle obtenue avec AdaIN. Cela concorde avec la mesure de *FID* qui est inférieure de près de 6% par rapport à la méthode de référence.

	FID ↓	**LPIPS ↑**
AWCSI	**51.17**	**0.32**
AdaIN	54.46	0.30

TABLE 3.4 – Mesure de la **FID** (1.11) ainsi que de la **LPIPS diversité** (1.3.1)sur le jeu de données des cartes satellites. La meilleure valeur pour chaque métrique est indiquée en gras.

Analyse comparative qualitative

La figure 3.4 montre l'impact de la méthode sur l'amélioration de la diversité dans les résultats générés. En effet, en échantillonnant des vecteurs latents différents z_1 et z_2, il est possible de remarquer que la méthode *AWCSI* génère des résultats plus variés en comparaison avec l'injection par *AdaIN* et ce, tout en préservant la structure de l'image en entrée. Cela permet d'autant plus de corroborer les résultats quantitatifs obtenus à la table 3.4 où la mesure *LPIPS* indique une plus grande variété dans les résultats générés lorsque le style est injecté en ayant recours à la méthode d'*AWCSI*.

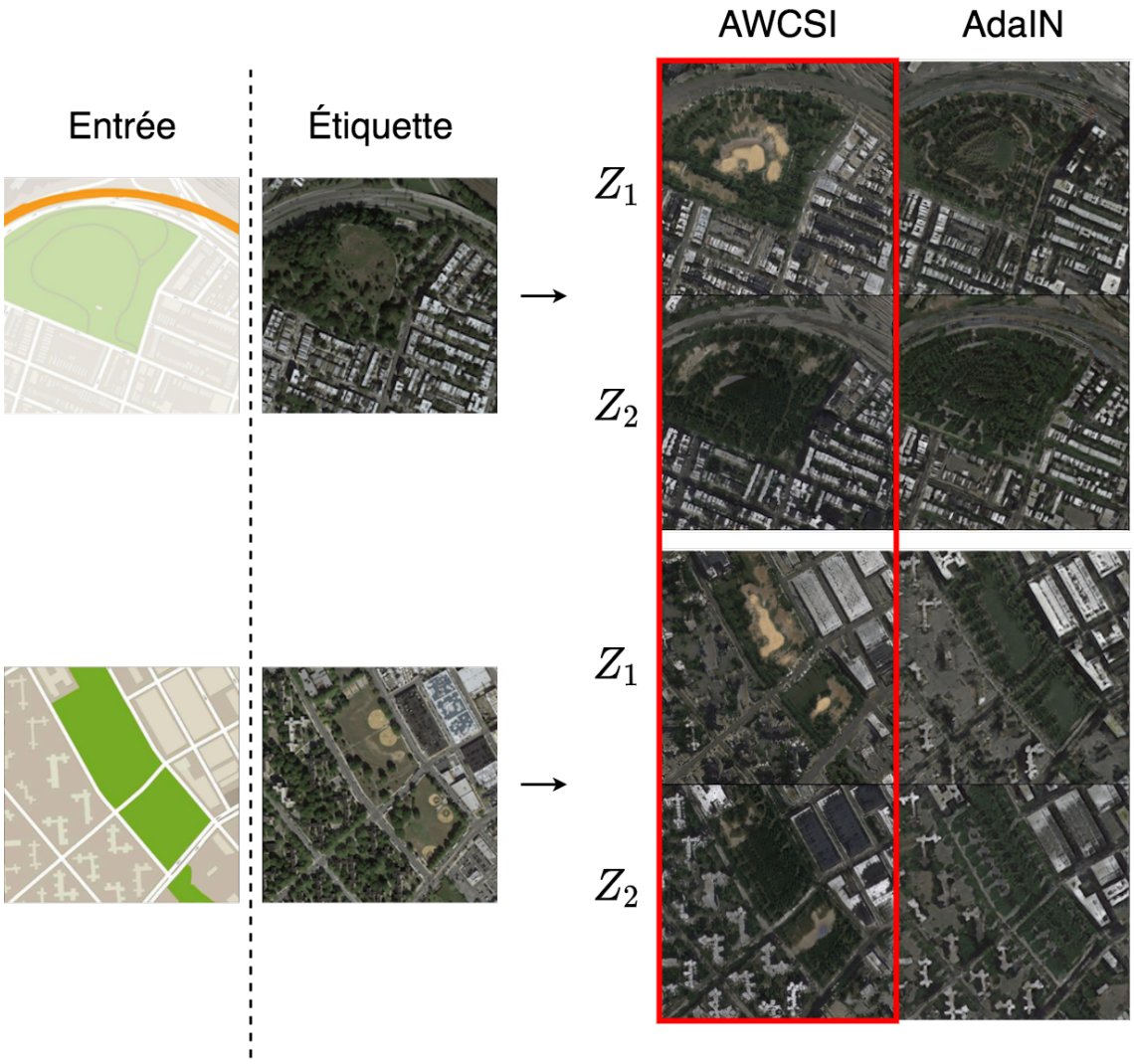

FIGURE 3.4 – Représentation de la variété dans les images avec changement du code latent z injecté. La figure présente, pour chacune des méthodes, deux résultats obtenus pour la même entrée en utilisant deux codes latents z_1 et z_2 différents.

FIGURE 3.5 – Comparaison qualitative entre la méthode d'injection AdaIN et notre méthode (*AWCSI*) sur le jeu de données des cartes satellite. Chaque méthode est présentée pour la génération d'une image selon l'image de condition en entrée (indiquée à la gauche) et un vecteur de style w injecté à l'aide des méthodes présentées. L'étiquette de chaque associé à chaque image en entrée est présentée à titre indicatif dans le but d'offrir un comparatif entre le résultat ciblé et ceux produits par le générateur.

Interpolation du vecteur latent En effectuant l'interpolation entre deux vecteurs latents dans l'espace $\mathcal{W}$, il est possible de remarquer l'impact qu'a l'injection de style par blanchissage et coloration sur l'amélioration de la variété des résultats. Effectivement, tel qu'illustré à la figure 3.6, en procédant à la génération d'images à l'aide d'un vecteur latent produit par le résultat de l'interpolation entre deux vecteurs latents z_1 et z_2 fixes, il est possible d'observer une variation progressive du résultat obtenu en faisant varier linéairement un paramètre α

$$z_{inter} = (1 - \alpha)z_1 + \alpha z_2. \tag{3.5}$$

Dans le cas présent, cela applique une variation de la densité de la végétation dans les espaces verts des images produites. On constate que cette variation est nettement plus marquée avec la méthode *AWCSI* où l'image de gauche, avec très peu de végétation, passe à l'extrême droite à une carte satellite avec une forte présence de végétation. Ceci est, en comparaison à la méthode d'injection *AdaIN* où la variation est moins distinctement prononcée.

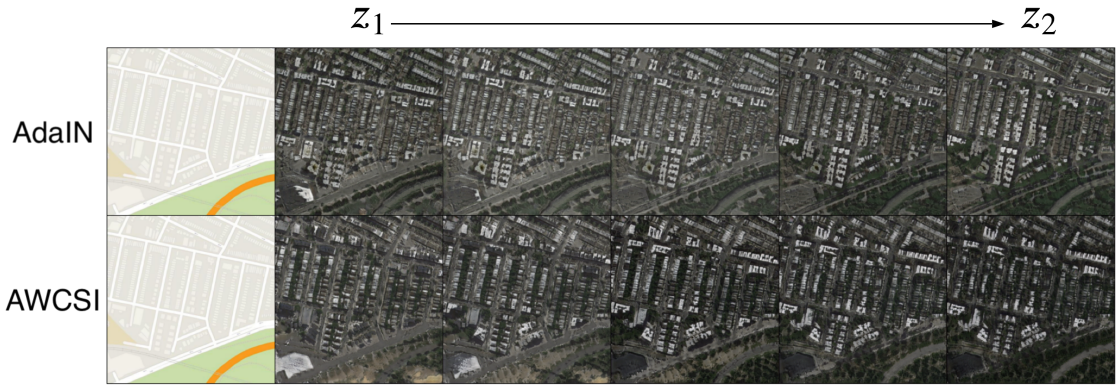

FIGURE 3.6 – Comparaison de l'impact de la diversité des images générées sur l'interpolation du code latent de style injecté. De gauche à droite, l'image d'entrée, le résultat produit avec $z1$ uniquement, trois résultats provenant d'un vecteur intermédiaire $z_{inter} = (1 - \alpha)z_1 + \alpha z_2$ et finalement le résultat obtenu de z_2.

3.2 Entraînement en configuration d'images non-appariées

L'intérêt grandissant pour l'entraînement des *GANs* en configuration non-appariée se justifie par la difficulté d'obtention de jeu de données appariées. Ainsi, afin de valider notre méthode dans un contexte d'entraînement non-supervisé, nous réalisons une série d'expérimentations en se basant sur l'architecture de *StarGAN v2* (Choi et al., 2020). Dans le travail effectué par Choi et al. (2020), le processus d'injection de l'information de style avec *AdaIN* est un concept central sur lequel repose l'ensemble de la méthode. Ainsi, l'architecture développée par les auteurs de cet article s'est avérée être un banc d'essai prometteur afin de démontrer

l'efficacité de notre méthode au sein de l'architecture proposée par StarGAN-v2. L'objectif de cette section est de réaliser la comparaison de l'injection de style à l'aide de notre méthode décrite dans la section 2 à celle actuellement utilisée (*AdaIN*) afin d'en observer les impacts sur les performances de génération d'images dans un contexte d'entraînement non-supervisé. L'entraînement non-supervisé peut-être définit comme étant l'utilisation d'un ensemble de données provenant d'un domaine source $\{x_i\}$ pour $i = 1, \ldots, N$ pour réaliser la traduction vers un domaine cible étant constitué d'un jeu de données $\{y_j\}$ pour $j = 1, \ldots, M$.

Dans cette section, le jeu de données utilisé pour l'entraînement est premièrement décrit. Suit une description de la configuration de l'architecture utilisée ainsi que de l'objectif d'entraînement. Finalement les résultats quantitatifs et qualitatifs obtenus suite à nos expérimentations sont analyser.

3.2.1 Jeu de données *AFHQ*

L'entraînement multi-domaines non-apparié est effectué sur le jeu de données Animal Faces-HQ (*AFHQ*) produit par Choi et al. (2020). Ce dernier comprend des images provenant de trois domaines visuels différents : chats, chiens et animaux sauvages comprenant diverses espèces (lions, renards, tigres, guépards, loups, jaguars, léopards). Chaque domaine regroupe 4500 images en entraînement et 500 images en validation de résolution 512×512 pixels pour un total de 15 000 images. Il est important de mentionner que l'ensemble des entraînements a été effectué sur les images redimensionnées aux tailles 128×128 et 256×256 dans le but d'accélérer le temps de calcul. La figure 3.7 montre des exemples aléatoirement échantillonnés du jeu de données.

FIGURE 3.7 – Exemples d'images tirées du jeu de données *AFHQ* pour chacun des trois domaines. Vert → chats, bleu → chiens, magenta → animaux sauvages.

3.2.2 Détails de l'implémentation de l'architecture

Cette section fait état de l'ensemble de l'architecture utilisée dans le contexte des expérimentations effectuées en se basant sur *StarGAN v2*. L'objectif est ainsi de remplacer le module d'injection *AdaIN* par notre module *AWCSI* sans altérer l'architecture *StarGAN v2* outre

la méthode d'injection du vecteur de style. L'ensemble des réseaux ainsi que les fonctions de perte utilisés reste inchangées dans le but d'observer l'impact attribuable au module d'injection de style *AWCSI*. La structure globale de chacune des composantes de l'architecture est schématisée à la figure 3.8.

Aperçu de l'architecture Tel que présenté à la figure 3.8, l'architecture de *StarGAN v2* (Choi et al., 2020) comporte différentes composantes qui jouent un rôle essentiel dans le processus de génération. Le générateur G réalise la traduction d'une image x_{source} vers un autre domaine en se basant sur un vecteur de style s qui lui peut provenir de deux sources distinctes. La première, étant celle du réseau de traduction F qui produit un vecteur de style s à partir d'un code latent échantillonné z et la deuxième étant un vecteur de style qui correspond au résultat produit à partir de l'image de référence x_{ref} une fois celle-ci passée dans l'encodeur de style E. Le discriminateur pour sa part, indépendemment de la provenance du vecteur de style, tente de déterminer la véracité des images produites par G.

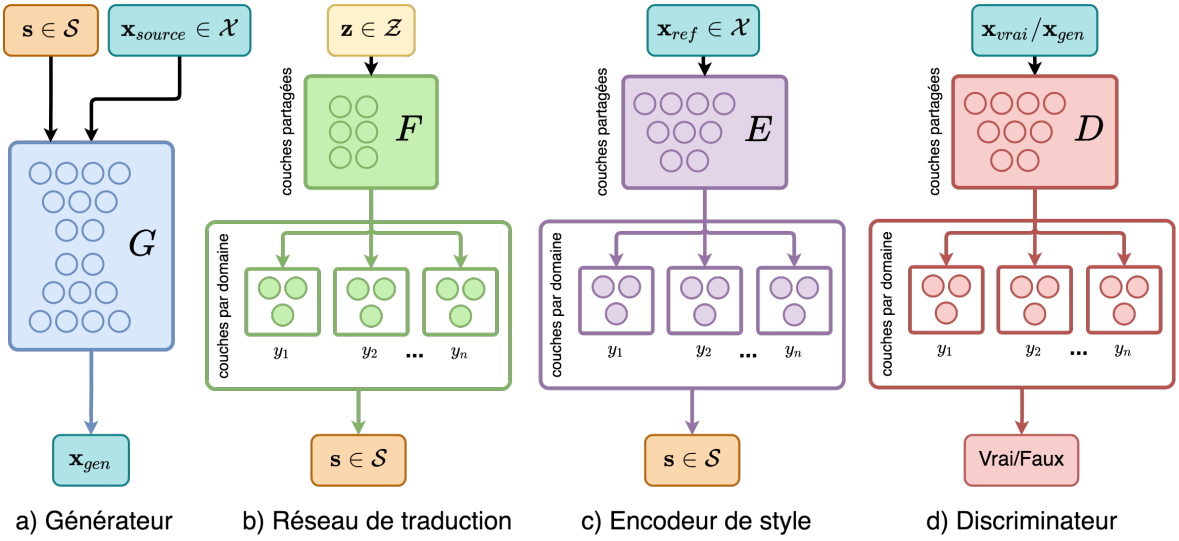

FIGURE 3.8 – Représentation des composantes formant l'architecture *StarGAN v2* utilisée afin de réaliser les expérimentations de la section 3.2. **a)** Générateur G produisant la traduction d'une image x_{source} vers une image d'un autre domaine $y \in \mathcal{Y}$ en se basant sur un style $\mathbf{s} \in \mathcal{S}$. **b)** Réseau de traduction (*mapping network*) F réalisant la projection d'un vecteur latent $\mathbf{z} \in \mathcal{Z}$ vers un vecteur de style $\mathbf{s} \in \mathcal{S}$ appartenant à un domaine spécifique y_i pour $i \in \{1, \ldots, n\}$(pour n domaines). **c)** Encodeur de style E, prend une image de référence afin d'en obtenir le vecteur de style spécifique au domaine choisi. **d)** Réseau discriminant D qui détermine la véracité d'une image selon le domaine. Les réseaux F, E et D ont une partie qui est partagée entre les domaines ainsi qu'une partie propre à chaque domaine. Figure inspirée du travail de Choi et al. (2020).

Générateur Le générateur G effectue la traduction d'une image $\mathbf{x}_{source} \in \mathcal{X}$ vers une image de sortie $G(\mathbf{x}_{source}, \mathbf{s}) \rightarrow \mathbf{x}_{gen}$ en se basant sur un vecteur de style $\mathbf{s}$ propre à un domaine spécifique y_i pour $i \in \{1, \ldots, n\}$, où n est le nombre total de domaines contenus dans le

jeu de données. Le vecteur de style est injecté à l'aide du module *AdaIN* qui dans le cas de cette expérimentation sera remplacée par notre méthode *AWCSI* afin d'en comparer les performances. Le vecteur de style est échantillonné de deux manières : la première, correspond à un échantillonnage d'un code latent $\mathbf{z}$ traduit en vecteur de style à l'aide d'un réseau de traduction et la seconde, une image de référence $\mathbf{x}_{ref}$ encodé vers un vecteur de style. La table 3.5 présente de manière exhaustive l'architecture du générateur. Considérant que l'objectif est d'évaluer l'impact du module *AWCSI* sur l'injection de style au sein de l'architecture *StarGAN v2* (Choi et al., 2020), aucune modification a été réalisée à l'architecture déjà existante afin de ne pas altérer le positionnement des modules d'injections et ainsi possiblement créer des variations aux résultats qui ne seraient pas uniquement attribuables au remplacement du module d'injection de style *AdaIN* par le module *AWCSI*. Cette manière de procéder assure que les variations dans les résultats sont exclusivement liées à la méthode *AWCSI*.

Couche	**Échantillonnage**	**Normalisation**	**Taille de sortie** $[H \times W \times C]$
Image	-	-	$256 \times 256 \times 3$
Conv 1×1	-	-	$256 \times 256 \times 64$
Bloc Résiduel	Average Pool	IN	$128 \times 128 \times 128$
Bloc Résiduel	Average Pool	IN	$64 \times 64 \times 256$
Bloc Résiduel	Average Pool	IN	$32 \times 32 \times 512$
Bloc Résiduel	Average Pool	IN	$16 \times 16 \times 512$
Bloc Résiduel	-	IN	$16 \times 16 \times 512$
Bloc Résiduel	-	IN	$16 \times 16 \times 512$
Bloc Résiduel	-	AWCSI/AdaIN	$16 \times 16 \times 512$
Bloc Résiduel	-	AWCSI/AdaIN	$16 \times 16 \times 512$
Bloc Résiduel	Upsample(bilinéaire)	*AWCSI/AdaIN*	$32 \times 32 \times 512$
Bloc Résiduel	Upsample(bilinéaire)	*AWCSI/AdaIN*	$64 \times 64 \times 256$
Bloc Résiduel	Upsample(bilinéaire)	*AWCSI/AdaIN*	$128 \times 128 \times 128$
Bloc Résiduel	Upsample(bilinéaire)	*AWCSI/AdaIN*	$256 \times 256 \times 64$
Conv 1×1	-	-	$256 \times 256 \times 3$

TABLE 3.5 – Architecture détaillée du réseau générateur *StarGAN v2*. L'échantillonnage correspond à la méthode utilisée afin de réaliser une modification de la dimension spatiale des cartes d'activations (sur ou sous-échantillonnage). La normalisation indique si la méthode de normalisation *Instance Normalisation* (Ulyanov et al., 2016) ou l'injection de style (*AWCSI/AdaIN*) a été utilisé en plus de situer l'endroit où elle est effectuée dans le générateur. Table tirée de Choi et al. (2020).

Réseau de traduction (*Mapping network*) Le réseau de traduction produit un vecteur de style en se basant sur code latent $\mathbf{z}$ échantillonné d'une fonction gaussienne multivariée ainsi qu'un domaine $y \in \mathcal{Y}$ auquel le vecteur de style doit appartenir. Cela peut être résumé comme étant $F_y(\mathbf{z}) = \mathbf{s}$. Le réseau de traduction est noté F_y puisque les quatre dernières couches du réseau dépendent du domaine $\mathbf{y}$ vers lequel le code latent doit être traduit et ne

sont donc pas partagées entre tous les domaines. L'architecture est présentée à la table 3.6.

Couche	**Type**	**Activation**	**Taille de sortie**
Code latent $\mathbf{z}$	-	-	128
Linéaire	partagée	ReLU	512
Linéaire	partagée	ReLU	512
Linéaire	partagée	ReLU	512
Linéaire	partagée	ReLU	512
Linéaire	spécifique au domaine	ReLU	512
Linéaire	spécifique au domaine	ReLU	512
Linéaire	spécifique au domaine	ReLU	512
Linéaire	spécifique au domaine	-	64

TABLE 3.6 – Le réseau de traduction partage les quatre premières couches pleinement connectées alors qu'il y a K groupes de quatre couches connectées pour les couches suivantes (un groupe par domaine). Table tirée de Choi et al. (2020).

Encodeur de style L'encodeur de style produit un vecteur de style en se basant sur une image de référence $\mathbf{x}_{ref}$ ainsi que sur le domaine de correspondance de cette dernière y. Le style de l'image est prélevé $E_y(\mathbf{x}_{ref}) = \mathbf{s}$. L'ensemble des couches de l'encodeur est partagé pour tous les domaines mis à part la dernière couche qui elle dépend du domaine vers lequel l'image doit être encodée. Il y a donc $K = n$ couches linéaires en sortie dont la taille de sortie dépend de la taille du vecteur de style $\mathbf{s}$.

Couche	**Échantillonnage**	**Normalisation**	**Taille de sortie** $[H \times W \times C]$
Image	-	-	$256 \times 256 \times 3$
Conv 1×1	-	-	$256 \times 256 \times 64$
Bloc Résiduel	Average Pool	-	$128 \times 128 \times 128$
Bloc Résiduel	Average Pool	-	$64 \times 64 \times 256$
Bloc Résiduel	Average Pool	-	$32 \times 32 \times 512$
Bloc Résiduel	Average Pool	-	$16 \times 16 \times 512$
Bloc Résiduel	Average Pool	-	$8 \times 8 \times 512$
Bloc Résiduel	Average Pool	-	$4 \times 4 \times 512$
Conv 4×4	-	-	$1 \times 1 \times 512$
Vectorisé	-	-	512
Linéaire $\times$ K	-	-	K $\times$ taille de $\mathbf{s}$

TABLE 3.7 – Encodeur de style composé d'une série de blocs résiduel qui effectuent le sous-échantillonnage de l'image de référence vers un vecteur de style. Les couches sont partagées pour tous les domaines excepté la dernière couche pleinement connectée qui elle est dépendante du domaine. Il y a donc une couche de sortie différente pour chaque domaine. Table tirée de Choi et al. (2020).

.

3.2.3 Objectif d'entraînement

Afin d'assurer un bon entraînement du générateur, les fonctions de perte suivantes sont utilisées afin de guider le générateur dans l'accomplissement de sa tâche. En opposition à la section 3.1.3, aucune fonction de perte supervisée n'est utilisée puisque l'entraînement se fait de manière non-supervisée considérant le jeu de données qui est composé d'images sans correspondances.

Fonction de perte cyclique Afin d'assurer une cohérence structurelle de l'image produite par le générateur avec les caractéristiques de l'image source (p. ex. pose), une fonction de perte cyclique (Zhu et al., 2017a) est ajoutée à l'entraînement. Cette dernière assure la cohérence du générateur face aux données traduites d'un domaine vers un autre. C'est aussi celle-ci qui permet l'entraînement de manière non-supervisée puisque c'est le cycle qui assure la cohérence de la traduction entre les domaines.

$$\mathcal{L}_{cyc} = \mathbb{E}_{\mathbf{x},y,\tilde{y},\mathbf{z}} \left[\|\mathbf{x} - G(G(\mathbf{x}, \widetilde{\mathbf{s}}), \hat{\mathbf{s}})\|_1 \right] \tag{3.6}$$

Fonction de perte antagoniste Soit deux domaines compris dans le jeu de données $\{y, \tilde{y}\} \subseteq \mathcal{Y}$ ainsi qu'un code latent $\mathbf{z} \in \mathcal{Z}$. En produisant le vecteur de style à l'aide du *mapping network* $F_{\tilde{y}}(z) = \tilde{\mathbf{s}}$, la fonction de perte associée à l'objectif antagoniste (Goodfellow et al., 2014) correspond à

$$\mathcal{L}_{adv} = \mathbb{E}_{\mathbf{x}_{source},y} \left[\log D_y(\mathbf{x}_{source}) \right] + \mathbb{E}_{\mathbf{x},\tilde{y},\mathbf{z}} \left[\log \left(1 - D_{\tilde{y}}(G(\mathbf{x}_{source}, \widetilde{\mathbf{s}})) \right) \right], \tag{3.7}$$

où $D_y(\mathbf{x}_{source})$ est la fonction discriminante pour le domaine y et l'image source et $D_{\tilde{y}}(G(\mathbf{x}_{source}), \widetilde{\mathbf{s}})$ est la fonction discriminante sur l'image générée pour le domaine cible $\tilde{y}$.

Reconstruction du style L'objectif de cette fonction est d'assurer un bon encodage par l'encodeur de l'image qui sera utilisé comme référence par le générateur. Cela force le générateur à utiliser le style de l'image encodée afin de produire la traduction de l'image source vers l'image de référence.

$$\mathcal{L}_{\text{sty}} = \mathbb{E}_{\mathbf{x}_{source},\tilde{y},\mathbf{z}} \left[\|\widetilde{\mathbf{s}} - E_{\tilde{y}}(G(\mathbf{x}_{source}, \widetilde{\mathbf{s}}))\|_1 \right]. \tag{3.8}$$

Cela peut être interprété par une fonction de perte assurant la reconstruction du style encodé puis décodé.

Régularisation de la diversité des résultats Afin d'assurer une diversification des résultats générés pour une même image $\mathbf{x}_{source}$ en entrée du générateur, une régularisation (Mao

et al., 2019; Yang et al., 2019) qui force le générateur à produire des résultats différents pour une même image source ayant cependant des styles en provenance de codes latents distincts $\tilde{\mathbf{s}}_i = F_{\tilde{y}}(\mathbf{z}_i)$ pour $i \in \{1, 2\}$ est utilisée. Ainsi, maximiser cette régularisation pousse le générateur à explorer le domaine cible pour des vecteurs de style différents.

$$\mathcal{L}_{ds} = \mathbb{E}_{\mathbf{x}_{source}, \tilde{y}, \mathbf{z}_1, \mathbf{z}_2} \left[\| G\left(\mathbf{x}_{source}, \widetilde{\mathbf{s}}_1\right) - G\left(\mathbf{x}_{source}, \widetilde{\mathbf{s}}_2\right) \|_1 \right] \tag{3.9}$$

Objectif global L'objectif d'entraînement du réseau peut alors être décrit comme étant

$$\min_{G,F,E} \max_{D} \mathcal{L}_{adv} + \lambda_{sty}\mathcal{L}_{sty} - \lambda_{ds}\mathcal{L}_{ds} + \lambda_{cyc}\mathcal{L}_{cyc} \tag{3.10}$$

avec la pondération de chacune des composantes de l'objectif $\lambda_{cyc} = 1$, $\lambda_{sty} = 1$ et $\lambda_{ds} = 2$.

3.2.4 Comparaison qualitative et qualitative des résultats

Référence latente et référence guidée par image Dans l'architecture utilisée, le style injecté au générateur peut provenir de deux sources. La première est un code latent échantillonné aléatoirement d'une gaussienne $\mathbf{z} \in \mathcal{Z}$ traduit à l'aide d'un *mapping network* F vers un vecteur de style $F_{\tilde{y}}(\mathbf{z}) \rightarrow \tilde{\mathbf{s}}$. La seconde, par référence, utilise une image de référence qui est encodée à l'aide d'un réseau encodeur $E_{\tilde{y}}$ permettant d'obtenir le vecteur de style de cette dernière $E_{\tilde{y}}(\mathbf{x}_{ref}) \rightarrow \tilde{\mathbf{s}}$. L'ensemble des tableaux des résultats présenté différencie la méthode utilisée pour générer le vecteur de style selon si le vecteur de style provient d'une référence ou d'un code latent.

Analyse comparative qualitative La figure 3.9 illustre la comparaison qualitative des résultats obtenus avec l'utilisation d'une image de référence pour obtenir le vecteur de style $\tilde{s}$. Il est possible d'observer que les résultats produits par la méthode *AWCSI* sont plus réalistes et résultent en une meilleure représentation des animaux générées en fonction de leur pose et de la structure de l'image source (image d'entrée) tout en traduisant avec fidélité le style ainsi que le domaine de l'image de référence. En effet, sur l'ensemble des résultats produits, *AWCSI* permet la génération d'animaux dont les couleurs, les caractéristiques du pelage ainsi que celles des yeux sont bien transférées à l'image source. De plus, l'aspect structurel de l'image source (orientation de la pose et forme générale) est préservé dans le résultat tout en produisant des résultats dont la structure respecte celle du domaine cible (celui de l'image de référence). En contrepartie, certaines images produites avec la méthode *AdaIN* détruisent la structure naturelle de l'image de référence en préservant trop fortement les caractéristiques physionomiques de l'image source ce qui entraîne des résultats invraisemblables ou dont la qualité visuelle est grandement dégradée.

Le domaine des animaux sauvages regroupe différents animaux. Il est possible de remarquer que *AdaIN* échoue à produire des résultats qui respectent l'animal "intra-domaine" alors que *AWCSI* semble mieux transférer la représentation intra-domaine provenant des images de références.

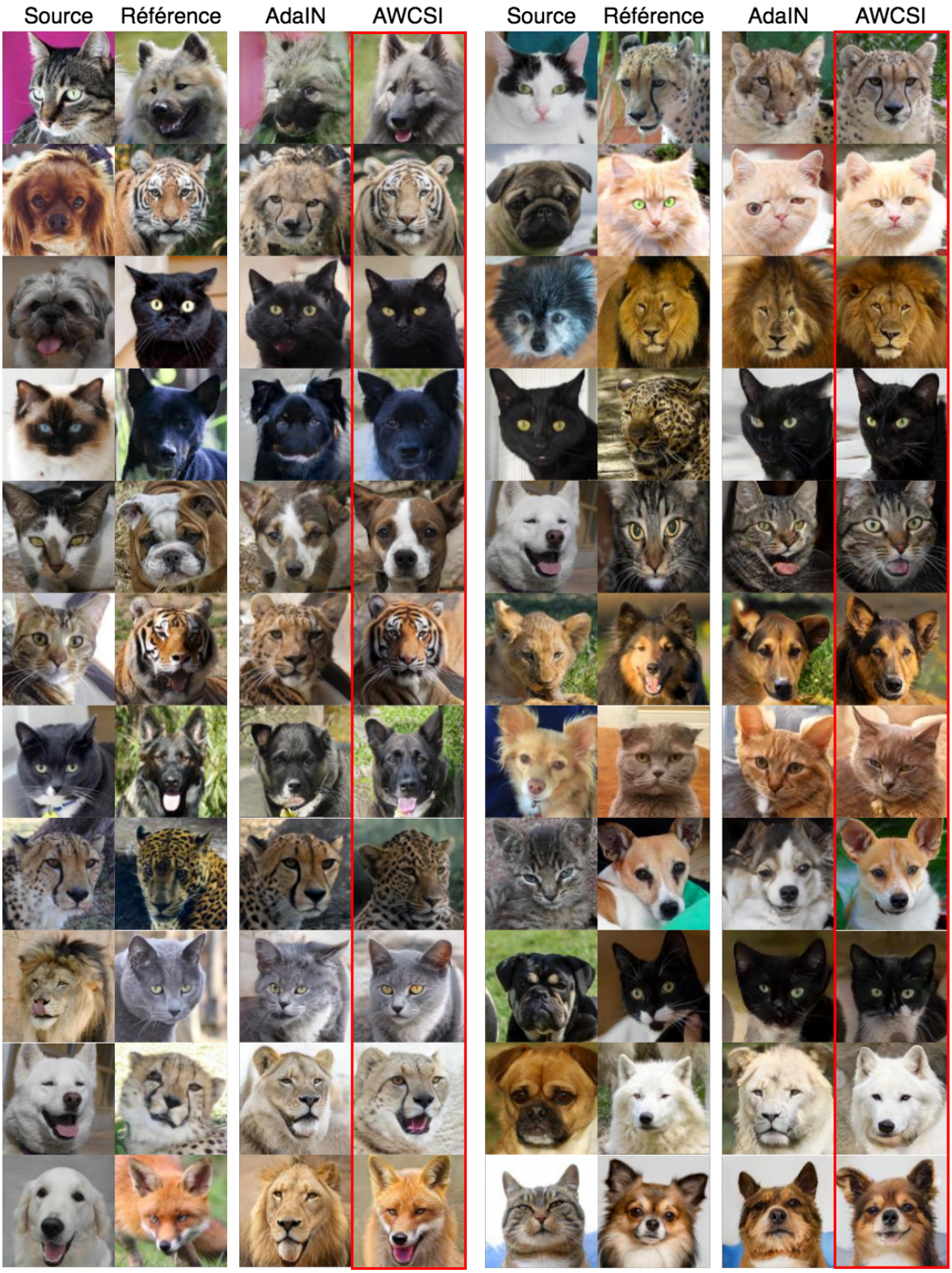

FIGURE 3.9 – Comparaison qualitative sur le jeu de données *AFHQ* entre notre méthode (*AWCSI*) et *AdaIN* sur des images générées avec une référence (style). La première colonne à partir de la gauche de chaque ensemble correspond à l'image source (orientation et structure), la deuxième colonne correspond à la référence de style alors que les deux autres colonnes sont respectivement *AdaIN* et *AWCSI*. Chaque méthode transfère l'image source vers le domaine cible tout en suivant le style de l'image de référence du domaine cible.

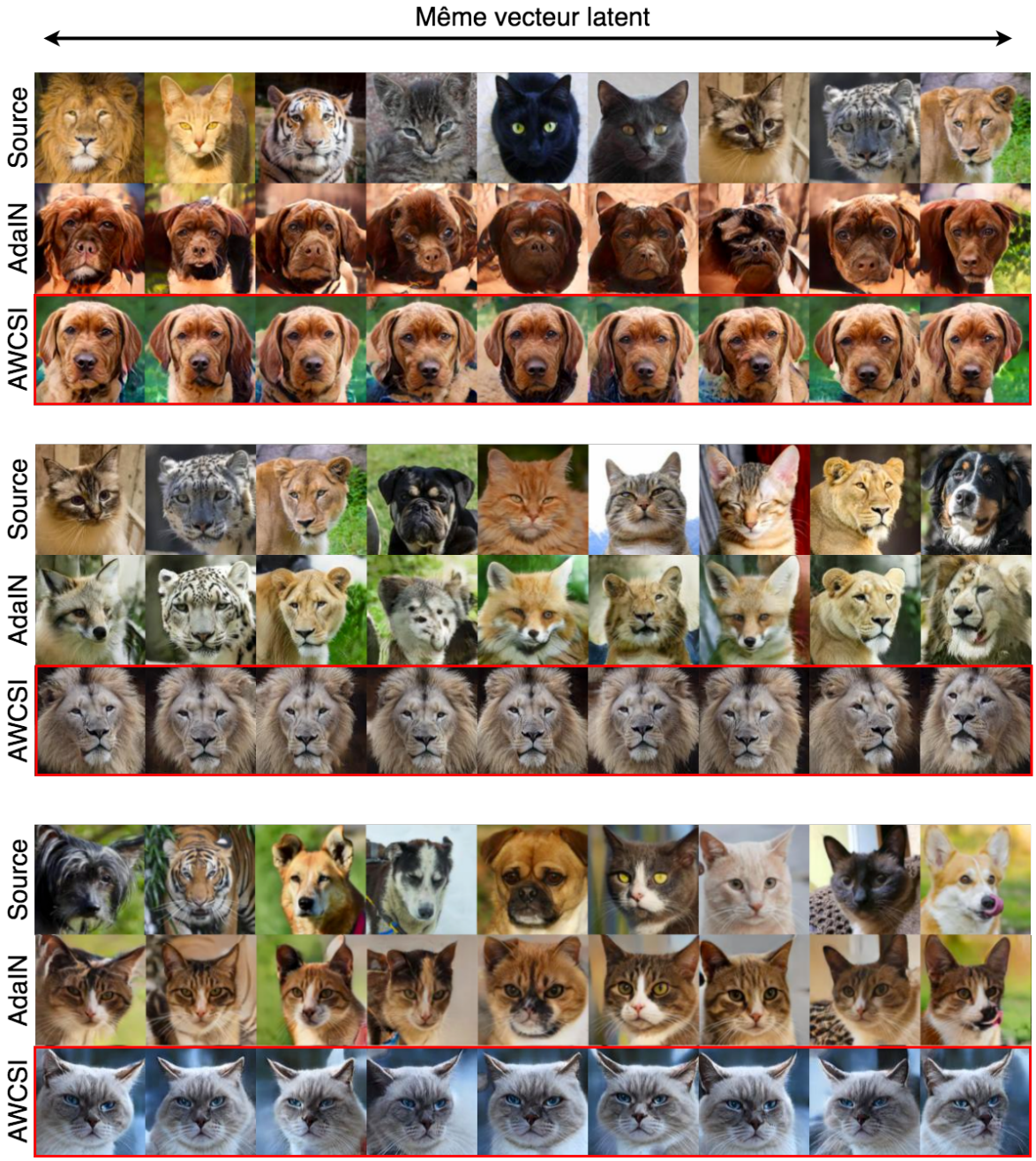

FIGURE 3.10 – Comparaison qualitative sur le jeu de données *AFHQ* de résultats générés avec vecteur de style latent. Chacune des deux méthodes traduit l'image source (première rangée de chaque groupe d'image) vers un même code latent échantillonné aléatoirement. Chaque groupe de trois lignes correspondent à un ensemble d'images sources sur lesquelles est appliqué le même vecteur latent.

	AWCSI	AdaIN
dog → cat	**0.330**	0.302
wild → cat	**0.329**	0.300
cat → dog	**0.378**	0.313
wild → dog	**0.377**	0.298
cat → wild	**0.378**	0.315
dog → wild	**0.378**	0.281
Moyenne	**0.362**	0.301

(a) Guidé par image de référence.

	AWCSI	AdaIN
dog → cat	**0.357**	0.298
wild → cat	**0.357**	0.295
cat → dog	**0.407**	0.316
wild → dog	**0.407**	0.303
cat → wild	**0.405**	0.338
dog → wild	**0.405**	0.302
Moyenne	**0.390**	0.309

(b) Guidé par code latent

TABLE 3.8 – Mesure de la **LPIPS** ↑(1.3.1) en diversité sur les résultats obtenus à l'aide du jeu de données *AFHQ*. La meilleure valeur pour chaque métrique est indiquée en gras. L'ensemble des résultats est généré à l'aide de l'architecture StarGANv2 (Choi et al., 2020) décrite à la section 3.2.2.

Analyse comparative quantitative La table 3.8 présente la comparaison quantitative de la métrique de diversité des résultats obtenus pour chacune des méthodes. La table 3.8 a) fait la comparaison de la diversité lors de l'injection d'un style provenant d'une image de référence alors que la section b) de la table 3.8 présente la diversité obtenue lorsque le style injecté est échantillonné d'un vecteur latent. Il est possible de voir que notre méthode surpasse par une large marge l'injection de style par AdaIN pour le transfert entre tous les domaines et ce autant lorsque le style injecté est basé sur une image de référence ($\sim 21\%$) que lorsque le style provient d'un vecteur latent ($\sim 17\%$). Ces résultats indiquent que pour une image en entrée, notre méthode produit des résultats plus diversifiés en sortie. Ainsi, l'injection de style par AWCSI favorise l'exploration et la découverte des modes du jeu de données par le générateur tout en préservant la qualité visuelle des résultats (voir table 3.9).

	AWCSI	AdaIN
dog → cat	9.34	**7.35**
wild → cat	10.66	**9.55**
cat → dog	**30.27**	45.76
wild → dog	**30.38**	42.05
cat → wild	**10.06**	22.77
dog → wild	**9.81**	34.31
Moyenne	**16.76**	26.97

(a) Guidé par image de référence.

	AWCSI	AdaIN
dog → cat	8.57	**8.18**
wild → cat	**9.87**	12.19
cat → dog	**25.96**	42.23
wild → dog	**26.06**	39.76
cat → wild	**7.82**	28.18
dog → wild	**7.64**	30.84
Moyenne	**14.32**	26.90

(b) Guidé par code latent.

TABLE 3.9 – Mesure de la **FID** ↓(1.3.2) sur les résultats obtenus à l'aide du jeu de données *AFHQ*. La valeur optimale pour chaque métrique est indiquée en gras. L'ensemble des résultats est généré à l'aide de l'architecture StarGANv2 (Choi et al., 2020) décrit à la section 3.2.2.

Afin d'évaluer la qualité visuelle des résultats de manière quantitative, une analyse de la FID (1.3.2) a été effectuée sur les images générées pour chaque méthode. La table 3.9 montre la comparaison de la métrique sur chaque méthode et ce pour l'ensemble des transferts de domaines réalisés. Il est possible de constater que la méthode AWCSI performe mieux dans la majorité des cas et ce autant avec référence latente que référence par image. Cela correspond aux observations réalisées aux figures 3.9 et 3.10 où la qualité des résultats obtenus est nettement supérieure.

Résultats de l'évaluation sur différentes tailles de blocs. Dans le but d'évaluer l'impact de la taille des blocs sur la qualité des résultats produit par le générateur, une série d'entraînements ayant recours à la méthode AWCSI et faisant intervenir des groupes de différentes dimensions ont été effectuées. La section suivante rapporte les résultats obtenus suite à ces expérimentations.

La table 3.10 présente les résultats obtenus lors de l'évaluation de la qualité des résultats produits à l'aide de générateur dont la taille des groupes diffère. Pour les résultats guidés par image de référence, une tendance semble être observable lors de l'augmentation de la taille des blocs utilisés. En effet, les métriques de qualité (*FID*) et diversité (*LPIPS*) tendent à s'améliorer avec l'utilisation de blocs de plus grande taille. Cela tend à montrer que l'application du blanchiment et de la coloration sur un plus grand nombre de canaux permet de réduire les restrictions sur la méthode de sorte à allouer un plus grand contrôle sur les corrélations inter-canaux lors du processus de génération d'images impliquant l'injection de style.

(a) Guidé par image de référence.

	FID ↓	**LPIPS** ↑
1	24.98	0.309
4	20.79	0.328
8	20.48	0.338
16	18.22	0.375
32	17.80	0.370
64	16.63	0.382

(b) Guidé par code latent.

	FID ↓	**LPIPS** ↑
1	18.23	0.342
4	16.65	0.359
8	20.48	0.368
16	14.02	0.410
32	14.64	0.399
64	14.24	0.408

TABLE 3.10 – Mesure de la **FID** ↓(1.3.2) et de la **LPIPS** ↑(1.3.1) pour différentes tailles de bloc utilisées lors du processus d'injection de style. La dimension des blocs progresse de haut en bas de chacune des tables. Les résultats présentés correspondent à la moyennes obtenue sur tous les domaines et ce pour chaque métrique.

Résultats sur images 256×256 pixels. Afin de généraliser l'impact de la méthode à des images de plus grande taille, nous avons procédé à un entraînement du réseau permettant la génération d'images 256×256 pixels. L'objectif étant ainsi de démontrer l'invariabilité de la méthode à l'augmentation de la dimensionnalité et donc de la complexité de la tâche à effectuer par le générateur afin de produire des résultats de bonne qualité. La section suivante

présente de manière sommaire les images générées ainsi que les métriques qui ont été obtenues suite à cette expérimentation.

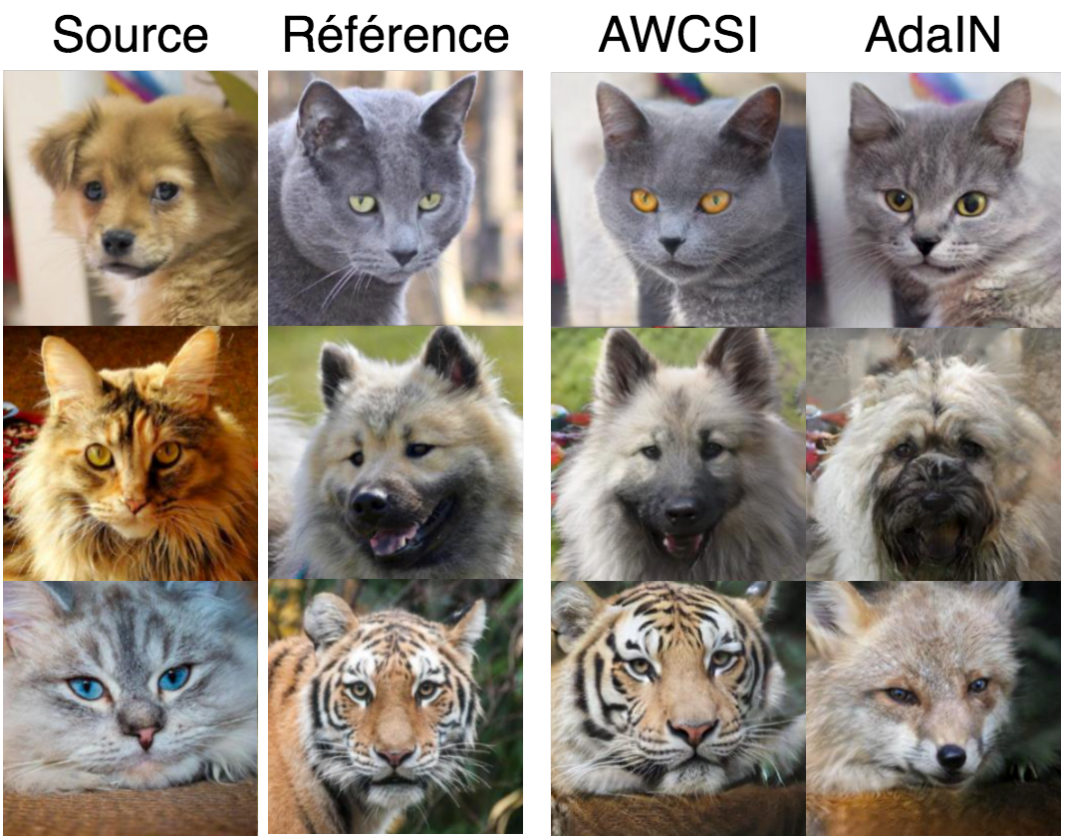

FIGURE 3.11 – Comparaison des résultats obtenus avec *AdaIN* et *AWCSI* pour la génération d'images de taille 256×256. De gauche à droite : l'image de source qui correspond à la structure, l'image référence de style, le résultat obtenu avec *AWCSI* et le résultat provenant d'*AdaIN*.

Les résultats qualitatifs présentés à la figure 3.11 permettent bien de voir que malgré l'augmentation de la dimension des résultats, la méthode est toujours aussi performante et permet la production d'images visuellement réalistes dont la structure et le style respectent les contraintes appliquées par les images de source et de référence. Il est effectivement possible de constater que les résultats produits à l'aide de la méthode d'injection *AWCSI* préservent correctement l'orientation (la pose de l'animal) et majoritairement la forme de l'image source en plus d'offrir un excellent transfert du *style* de l'image de référence tant au niveau des couleurs qui caractérisent l'animal qui s'y retrouve que du domaine auquel appartient celui-ci.

(a) Guidé par image de référence.

	AWCSI	AdaIN
FID ↓	**16.20**	19.78
LPIPS ↑	**0.434**	**0.430**

(b) Guidé par code latent.

	AWCSI	AdaIN
FID ↓	**13.07**	16.18
LPIPS ↑	**0.476**	0.450

TABLE 3.11 – Mesure de la **FID** ↓(1.3.2) et de la **LPIPS** ↑(1.3.1) sur les résultats obtenus à l'aide du jeu de données *AFHQ*. Les résultats présentés correspondent à la moyennes obtenue sur tous les domaines et ce pour chaque métrique pour la génération d'images de taille 256×256. La meilleure valeur pour chaque métrique est indiquée en gras. L'ensemble des résultats est généré à l'aide de l'architecture StarGANv2 (Choi et al., 2020) décrit à la section 3.2.2.

La table 3.11 présentent les résultats obtenus pour chaque métrique en moyenne sur chacun des domaines. Il est possible de conclure que la méthode d'injection du style AWCSI surpasse AdaIN pour la qualité des résultats (FID) de manière significative et ce indépendamment du fait que le style soit guidé par une image de référence ou par un vecteur de style latent. En ce qui a trait à la variété (LPIPS) des résultats produits, la diversité générée semble nettement moins marquante entre les deux méthodes. Cela indique que les deux méthodes permettent de produire des résultats d'une diversité similaire.

Conclusion

Malgré les améliorations récentes apportées aux algorithmes génératifs d'images à images, les tâches de transfert de domaine que tentent d'accomplir les réseaux génératifs antagonistes restent difficiles à réaliser. Plus particulièrement, la méthode d'injection d'information actuellement utilisée dans la littérature restreint la capacité de représentation du style injecté par le générateur puisqu'elle repose sur la supposition qu'il n'existe pas de corrélations entres les canaux des cartes d'activation du réseau. Il est donc possible de constater, comme nous l'avons démontré au chapitre 3, que la méthode *AdaIN* échoue dans certains cas à produire des résultats de qualité.

Ainsi, ce mémoire présente une méthode d'injection d'information dans le réseau générateur des architectures d'un réseau antagoniste génératif qui permet une amélioration de la qualité des résultats générés. Pour ce faire, le travail réalisé dans cet ouvrage propose d'introduire de l'information dans le générateur à l'aide d'une technique de blanchissage et de coloration inter-canaux des cartes d'activations afin d'en modifier les statistiques. À cette fin, un module d'injection est développé dans le but de permettre l'utilisation de la méthode sans modifications significatives à l'architecture du générateur. Cela rend l'application de ce module d'injection d'information très attrayante considérant l'impact minimal que celui-ci représente sur la structure des réseaux déjà existants. Le développement du travail effectué au sein de ce mémoire peut-être considéré comme étant une généralisation de la méthode *AdaIN* déjà existante dans le but de produire des résultats de plus grande qualité.

Les expérimentations réalisées et présentées dans le chapitre 3 permettent de démontrer que l'approche développée apporte une amélioration substantielle à la qualité de résultats produits et ce autant quantitativement que qualitativement. En effet, l'analyse des résultats qualitatifs présent une amélioration visuelle des résultats produit ainsi qu'un meilleur respect de la structure et du style des images générées dans un contexte de transfert de domaine. De plus, les métriques d'évaluation utilisées démontrent que les performances obtenues par la méthode proposée dans ce mémoire surpassent celles correspondantes à ce qui est obtenu à l'aide de l'état de l'art (*AdaIN*). Les résultats obtenus montrent que la généralisation apportée par notre méthode augmente la capacité de représentation du réseau générateur en donnant la flexibilité à ce dernier de manipuler les corrélations existantes au sein des cartes d'activations

en opposition à *AdaIN*.

Cependant, l'algorithme développé dans ce mémoire comporte malgré tout certaines limitations. Effectivement, dû à la lourde charge de calculs engendrée par la génération des matrices de coloration, l'utilisation de l'application du blanchiment et de la coloration par blocs sur les cartes d'activations est préconisé afin d'amoindrir le temps d'entraînement de la méthode. Ce compromis a pour effet d'ignorer certaines corrélations pouvant exister entres des canaux n'appartenant pas au même groupe. Cela réduit donc la capacité de la méthode à permettre l'utilisation de l'information contenue entre les canaux. Il serait ainsi intéressant dans des travaux futurs d'apporter une amélioration à la méthode de sorte à être en mesure de produire des matrices de coloration complètes. Il serait alors possible d'appliquer le processus sur l'ensemble des canaux et donc permettre la pleine utilisation du concept des corrélations entres les canaux. Dans un même ordre d'idées, notre méthode repose sur l'hypothèse que les corrélations existantes sont linéaires, il pourrait donc être intéressant d'explorer la manipulation des cartes d'activations à l'aide de méthodes non-linéaires. Cela pourrait peut-être permettre d'autres avancées dans l'amélioration des techniques d'introduction d'information au sein des réseaux de neurones ainsi que d'ouvrir la porte à de nouvelles avancées en lien avec la modification des statistiques des cartes d'activations.

En somme, il est possible de conclure que l'utilisation de la technique de blanchiment et de coloration dans le contexte d'injection d'information dans réseau antagoniste génératif permet une augmentation de la représentation de l'information introduite dans les réseaux générateurs ce qui engendre une amélioration de la qualité visuelle des résultats produits.

Bibliographie

Saeed Anwar and Nick Barnes. Densely residual laplacian super-resolution, 2019.

Kyungjune Baek, Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Hyunjung Shim. Rethinking the truly unsupervised image-to-image translation. *CoRR*, abs/2006.06500, 2020.

Ali Borji. Pros and cons of gan evaluation measures. *Computer Vision and Image Understanding*, 179 :41–65, 2019. ISSN 1077-3142. doi : https://doi.org/10.1016/j.cviu.2018.10.009.

Tong Che, Yanran Li, Athul Paul Jacob, Yoshua Bengio, and Wenjie Li. Mode regularized generative adversarial networks. *CoRR*, abs/1612.02136, 2016.

Wonwoong Cho, Sungha Choi, David Keetae Park, Inkyu Shin, and Jaegul Choo. Image-to-image translation via group-wise deep whitening and coloring transformation. *CoRR*, abs/1812.09912, 2018.

Yunjey Choi, Min-Je Choi, Munyoung Kim, Jung-Woo Ha, Sunghun Kim, and Jaegul Choo. Stargan : Unified generative adversarial networks for multi-domain image-to-image translation. *CoRR*, abs/1711.09020, 2017.

Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2 : Diverse image synthesis for multiple domains. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8185–8194, 2020. doi : 10.1109/CVPR42600.2020.00821.

Min Jin Chong and David A. Forsyth. Gans n' roses : Stable, controllable, diverse image to image translation (works for videos too !). *CoRR*, abs/2106.06561, 2021.

T. Dai, J. Cai, Y. Zhang, S. Xia, and L. Zhang. Second-order attention network for single image super-resolution. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11057–11066, 2019. doi : 10.1109/CVPR.2019.01132.

D.C Dowson and B.V Landau. The fréchet distance between multivariate normal distributions. *Journal of Multivariate Analysis*, 1982.

Vincent Dumoulin, Jonathon Shlens, and Manjunath Kudlur. A learned representation for artistic style. *CoRR*, abs/1610.07629, 2016.

Ishan P. Durugkar, Ian Gemp, and Sridhar Mahadevan. Generative multi-adversarial networks. *CoRR*, abs/1611.01673, 2016.

Jerome H. Friedman. Exploratory projection pursuit. *Journal of the American Statistical Association*, 82(397) :249–266, 1987. ISSN 01621459.

Abel Gonzalez-Garcia, Joost van de Weijer, and Yoshua Bengio. Image-to-image translation for cross-domain disentanglement. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

Mingming He, Dongdong Chen, Jing Liao, Pedro V. Sander, and Lu Yuan. Deep exemplar-based colorization. *CoRR*, abs/1807.06587, 2018.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Günter Klambauer, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a nash equilibrium. *CoRR*, abs/1706.08500, 2017.

Nicholas J. Higham. *Functions of Matrices : Theory and Computation*. Society for Industrial and Applied Mathematics, 2008.

Lei Huang, Xianglong Liu, Bo Lang, Adams Wei Yu, and Bo Li. Orthogonal weight normalization : Solution to optimization over multiple dependent stiefel manifolds in deep neural networks. *CoRR*, abs/1709.06079, 2017.

Lei Huang, Dawei Yang, Bo Lang, and Jia Deng. Decorrelated batch normalization. *CoRR*, abs/1804.08450, 2018a.

Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

Xun Huang, Ming-Yu Liu, Serge J. Belongie, and Jan Kautz. Multimodal unsupervised image-to-image translation. *CoRR*, 2018b.

Sergey Ioffe and Christian Szegedy. Batch normalization : Accelerating deep network training by reducing internal covariate shift. *CoRR*, abs/1502.03167, 2015.

C. Ionescu, O. Vantzos, and C. Sminchisescu. Matrix backpropagation for deep networks with structured layers. *CoRR*, 2015.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017.

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling, editors, *Computer Vision – ECCV 2016*, pages 694–711, Cham, 2016. Springer International Publishing.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

Agnan Kessy, Alex Lewin, and Korbinian Strimmer. Optimal whitening and decorrelation. *The American Statistician*, 72(4) :309–314, Jan 2018. ISSN 1537-2731. doi : 10.1080/00031305.2016.1277159.

Junho Kim, Minjae Kim, Hyeonwoo Kang, and Kwanghee Lee. U-GAT-IT : unsupervised generative attentional networks with adaptive layer-instance normalization for image-to-image translation. *CoRR*, abs/1907.10830, 2019.

Taeksoo Kim, Moonsu Cha, Hyunsoo Kim, Jung Kwon Lee, and Jiwon Kim. Learning to discover cross-domain relations with generative adversarial networks. *CoRR*, abs/1703.05192, 2017.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, 2012.

Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural Computation*, 1989.

Yann LeCun, Corinna Cortes, and CJ Burges. Mnist handwritten digit database. *ATT Labs [Online]. Available : http ://yann.lecun.com/exdb/mnist*, 2, 2010.

Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. Diverse image-to-image translation via disentangled representations. In *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

Peihua Li, Jiangtao Xie, Qilong Wang, and Zilin Gao. Towards faster training of global covariance pooling networks by iterative matrix square root normalization. *CoRR*, 2017a.

Yijun Li, Chen Fang, Jimei Yang, Zhaowen Wang, Xin Lu, and Ming-Hsuan Yang. Universal style transfer via feature transforms. 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., 2017b.

Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. *CoRR*, abs/1703.00848, 2017.

Qi Mao, Hsin-Ying Lee, Hung-Yu Tseng, Siwei Ma, and Ming-Hsuan Yang. Mode seeking generative adversarial networks for diverse image synthesis. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1429–1437, 06 2019. doi : 10.1109/CVPR.2019.00152.

Sachit Menon, Alexandru Damian, Shijia Hu, Nikhil Ravi, and Cynthia Rudin. Pulse : Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Luke Metz, Ben Poole, David Pfau, and Jascha Sohl-Dickstein. Unrolled generative adversarial networks. *CoRR*, abs/1611.02163, 2016.

Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets. *CoRR*, abs/1411.1784, 2014.

Evangelos Ntavelis, Andrés Romero, Iason Kastanis, Luc Van Gool, and Radu Timofte. SESAME : semantic editing of scenes by adding, manipulating or erasing objects. *CoRR*, abs/2004.04977, 2020.

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. *CoRR*, abs/1903.07291, 2019.

Taesung Park, Jun-Yan Zhu, Oliver Wang, Jingwan Lu, Eli Shechtman, Alexei A. Efros, and Richard Zhang. Swapping autoencoder for deep image manipulation, 2020.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks, 2015.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net : Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. In D. E. Rumelhart and J. L. McClelland, editors, *Parallel Distributed Processing*, chapter 8. MIT Press, Cambridge, MA, 1986.

Kuniaki Saito, Kate Saenko, and Ming-Yu Liu. COCO-FUNIT : few-shot unsupervised image translation with a content conditioned style encoder. *CoRR*, abs/2007.07431, 2020.

Tim Salimans, Ian J. Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *CoRR*, abs/1606.03498, 2016.

Juliane Schäfer and Korbinian Strimmer. A Shrinkage Approach to Large-Scale Covariance Matrix Estimation and Implications for Functional Genomics. *Statistical Applications in Genetics and Molecular Biology*, 2005.

Konstantin Shmelkov, Cordelia Schmid, and Karteek Alahari. How good is my gan ? *CoRR*, abs/1807.09499, 2018.

Aliaksandr Siarohin, Enver Sangineto, and Nicu Sebe. Whitening and coloring batch transform for gans, 2019.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

Akash Srivastava, Lazar Valkov, Chris Russell, Michael U. Gutmann, and Charles Sutton. Veegan : Reducing mode collapse in gans using implicit variational learning, 2017.

Jheng-Wei Su, Hung-Kuo Chu, and Jia-Bin Huang. Instance-aware image colorization, 2020.

Wanjie Sun and Zhenzhong Chen. Learned image downscaling for upscaling using content adaptive resampler. *CoRR*, abs/1907.12904, 2019.

Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

Hao Tang, Dan Xu, Wei Wang, Yan Yan, and Nicu Sebe. Dual generator generative adversarial networks for multi-domain image-to-image translation, 2019.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Instance normalization : The missing ingredient for fast stylization. *CoRR*, abs/1607.08022, 2016.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Deep image prior. *CoRR*, abs/1711.10925, 2017a.

Dmitry Ulyanov, Andrea Vedaldi, and Victor S. Lempitsky. Improved texture networks : Maximizing quality and diversity in feed-forward stylization and texture synthesis. *CoRR*, abs/1701.02096, 2017b.

L. N. Vaserstein. Markovian processes on countable space product describing large systems of automata. *Probl. Peredachi Inf.*, 5(3) :64–72, 1969. ISSN 0555-2923.

Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CoRR*, abs/1711.11585, 2017.

Wei Wang, Zheng Dang, Yinlin Hu, Pascal Fua, and Mathieu Salzmann. Backpropagation-friendly eigendecomposition. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019a.

Ze Wang, Xiuyuan Cheng, Guillermo Sapiro, and Qiang Qiu. Stochastic conditional generative networks with basis decomposition. *CoRR*, abs/1909.11286, 2019b.

Po-Wei Wu, Yu-Jing Lin, Che-Han Chang, Edward Y. Chang, and Shih-Wei Liao. Relgan : Multi-domain image-to-image translation via relative attributes, 2019.

Zhongyou Xu, Tingting Wang, Faming Fang, Yun Sheng, and Guixu Zhang. Stylization-based architecture for fast deep exemplar colorization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

Chao Yang, Xin Lu, Zhe Lin, Eli Shechtman, Oliver Wang, and Hao Li. High-resolution image inpainting using multi-scale neural patch synthesis. *CoRR*, abs/1611.09969, 2016.

Dingdong Yang, Seunghoon Hong, Yunseok Jang, Tiangchen Zhao, and Honglak Lee. Diversity-sensitive conditional generative adversarial networks. In *International Conference on Learning Representations*, 2019.

Zili Yi, Hao Zhang, Ping Tan, and Minglun Gong. Dualgan : Unsupervised dual learning for image-to-image translation. *CoRR*, abs/1704.02510, 2017.

Jiahui Yu, Zhe Lin, Jimei Yang, Xiaohui Shen, Xin Lu, and Thomas S. Huang. Free-form image inpainting with gated convolution. *CoRR*, abs/1806.03589, 2018.

Pan Zhang, Bo Zhang, Dong Chen, Lu Yuan, and Fang Wen. Cross-domain correspondence learning for exemplar-based image translation, 2020.

Richard Zhang, Phillip Isola, and Alexei A. Efros. Colorful image colorization. *CoRR*, abs/1603.08511, 2016.

Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018a.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018b.

Bo Zhao, Bo Chang, Zequn Jie, and Leonid Sigal. Modular generative adversarial networks. *CoRR*, abs/1804.03343, 2018.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A. Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. *CoRR*, abs/1703.10593, 2017a.

Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 465–476. Curran Associates, Inc., 2017b.

Peihao Zhu, Rameen Abdal, Yipeng Qin, and Peter Wonka. SEAN : image synthesis with semantic region-adaptive normalization. *CoRR*, abs/1911.12861, 2019.