REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : 003 /IVA/M2/2024

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et Vie Artificielle (IVA)**

Utilisation de Pix2Pix pour simuler l'occlusion ambiante dans l'infographie graphique

---

Par :
GUETTAL AYOUB

Soutenu le 11/06/2024 devant le jury composé de :

Cherif Foudil	Professeur	Président
Zerari Abdelmoumene	M.C.B	Rapporteur
Bentrah Ahlem	M.C.B	Examinateur

Année universitaire 2023-2024

Remerciements

Alhamdoulillah qui m'a facilité mon périple, et qui m'a fait sortir des ténèbres de l'ignorance à la lumière de la science et de la connaissance, merci mon dieu de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel, Tout le mérite revient à Dieu avant tout.

Le travail présenté dans cette mémoire a été réalisé au département d'informatique de la faculté **FSESNV** de l'Université de Mohamed kheider Biskra, sous la direction **Dr. ZERARI ABDELMOUMENE**, je voudrais saisir cette occasion et exprimer ma plus grande gratitude pour mon encadreur pour sa disponibilité et la confiance qu'il m'a accordée.

Je tiens également à remercier les membres du jury de m'avoir fait l'honneur de juger mon travail.

A mon très cher père de tous les pères, tu es le meilleur.

Grâce à toi papa j'ai appris le sens du travail et de la responsabilité, Tes conseils ont toujours guidé mes pas vers la réussite, Je tiens à honorer l'homme que tu es, En ce jour, j'espère réaliser l'un de tes rêves. . . .

Et particulièrement à ma très chère maman qui a toujours été là pour moi. Ta prière et ta bénédiction m'ont été d'un grand secours pour mener à bien mes études et tout au long de ma vie.

A mes frères **Oussama , Mohamed Anis**

A mes sœurs **Hadjer , Sara.**

A ma chère épouse **Imane**, qui a été très patiente avec moi et a pris beaucoup de son temps, surtout A mon fils **Mohamed Yanis**.

Un merci spécial à mon directeur au travail, **Mahrougui abdelaziz**, et merci spécial au chef de service informatique, **Sattafi Lakhdar** pour toutes les installations et le soutien qui m'ont permis de concilier travail et études, ainsi que pour les encouragements et le soutien.

Enfin, je remercie mes collègues de travail, en particulier au service informatique **Hamza, Salah,Ahmed et Mejda**, pour le soutien qu'ils m'ont apporté pendant mes études et la préparation de ce travail.

Résumé

L'occultation ambiante cherche à reproduire avec précision l'éclairage global en temps réel, permettant ainsi de rendre les détails de la scène 3D avec des ombres douces. Cependant, son utilisation requiert l'utilisation de G-Buffers, ce qui limite son application dans des scénarios nécessitant des changements interactifs dans la scène virtuelle.

Par ailleurs, l'occultation ambiante peut être simulée de manière réaliste grâce à l'utilisation de modèles génératifs tels que la traduction d'image à image, un réseau de neurones profonds. Ce modèle est entraîné sur un ensemble de données contenant des paires d'images avec et sans occultation ambiante, lui permettant ainsi d'apprendre à reproduire de façon convaincante les effets visuels de l'occultation.

Dans ce contexte, ce mémoire propose de calculer l'occultation ambiante à l'aide d'un modèle génératif. Nous avons ainsi créé un ensemble de données comportant 10 000 images de synthèse et développé un modèle capable de générer l'occultation ambiante.

La simulation de l'occultation ambiante par pix2pix marque une avancée importante dans le domaine de la modélisation des effets visuels, ouvrant ainsi de nouvelles perspectives pour créer des simulations plus réalistes et immersives. Cela permet également d'aborder les défis liés à l'entraînement et à l'optimisation des modèles pix2pix pour cette tâche spécifique. Les expérimentations montrent que notre technique peut générer des images de haute qualité et nettes.

Mots-clés : Occlusion ambiante dans l'espace d'écran (SSAO), deep learning, réseaux de neurones convolutifs(CNN), réseaux antagonistes génératifs (GAN), Traduction image à image (Pix2Pix).

Abstract

Ambient occlusion aims to accurately reproduce global lighting in real time, thus rendering the details of the 3D scene with soft shadows. However, its use requires the utilization of G-Buffers, limiting its application in scenarios requiring interactive changes in the virtual scene.

Additionally, ambient occlusion can be realistically simulated using generative models such as image-to-image translation, a deep neural network. This model is trained on a dataset containing pairs of images with and without ambient occlusion, allowing it to convincingly reproduce the visual effects of occlusion.

In this context, this thesis proposes to calculate ambient occlusion using a generative model. We thus created a dataset containing 10,000 synthetic images and developed a model capable of generating ambient occlusion. The simulation of ambient occlusion by pix2pix represents a significant advancement in the field of visual effects modeling, opening up new possibilities for creating more realistic and immersive simulations. This also addresses the challenges associated with training and optimizing pix2pix models for this specific task. Experiments show that our technique can generate high-quality and sharp images.

Keywords : Screen space ambient occlusion (SSAO), deep learning, convolutional neural networks (CNN), generative adversarial networks (GAN), image-to-image translation (Pix2Pix).

ملخص :

يسعى الإنسداد المحيطي إلى إعادة إنتاج الإضاءة العالمية بدقة في الوقت الفعلي، مما يسمح بعرض تفاصيل المشهد ثلاثي الأبعاد بظلال ناعمة. ومع ذلك، يتطلب استعماله استخدامG-Buffers ، مما يحد من تطبيقه في السيناريوهات التي تتطلب تغييرات تفاعلية في المشهد الافتراضي.

علاوة على ذلك، يمكن محاكاة الإنسداد المحيطي بشكل واقعي من خلال استخدام النماذج التوليدية مثل الترجمة من صورة إلى صورة، وهي شبكة عصبية عميقة. تم تدريب هذا النموذج على مجموعة بيانات تحتوي على أزواج من الصور مع أو بدون الإنسداد المحيطي، مما يسمح له بتعلم إعادة إنتاج التأثيرات المرئية للانسداد بشكل مقنع.

في هذا السياق، تقترح هذه الأطروحة حساب الانسداد المحيط باستخدام نموذج توليدي. وهكذا أنشأنا مجموعة بيانات تضم 10000 صورة اصطناعية وقمنا بتطوير نموذج قادر على توليد الإخفاء المحيط.

تمثل محاكاة الإنسداد المحيطي بواسطة pix2pix تقدمًا مهمًا في مجال نمذجة التأثيرات المرئية، وبالتالي فتح آفاق جديدة لإنشاء عمليات محاكاة أكثر واقعية وغامرة. ويساعد هذا أيضًا في مواجهة تحديات التدريب وتحسين نماذج pix2pix لهذه المهمة المحددة. تظهر التجارب أن تقنيتنا يمكنها إنتاج صور عالية الجودة وحادة.

الكلمات المفتاحية: انسداد مساحة الشاشة المحيطة(SSAO) ، التعلم العميق، الشبكات العصبية التلافيفية(CNN) ، شبكات الخصومة التوليدية(GAN) ، ترجمة الصورة إلى صورة(Pix2Pix).

Table des matières

Table des figures

Liste des tableaux

Liste des abréviations

AO Ambient occlusion

SSAO Screen space Ambient occlusion

ANN Artifitial Neural Network

CNN Convolutional Neural Network

DL Deep Learning

GAN Generative Adversarial Network

IA Intelligence Artificielle

ML Machine Learning

Introduction Générale

La simulation de l'occlusion ambiante est une tâche cruciale dans de nombreux domaines, de la vision par ordinateur à la réalité virtuelle. Elle consiste à modéliser les zones d'ombres ou d'obscurité résultant de l'interaction complexe entre la lumière et l'environnement. Cette simulation est essentielle pour diverses applications, telles que la conception architecturale, la navigation autonome des véhicules et la génération d'images photo réalistes. L'une des approches les plus récentes et prometteuses pour simuler l'occlusion ambiante est l'utilisation de réseaux de neurones génératifs, et plus spécifiquement, de modèles comme pix2pix, qui est un modèle de réseau de neurones profonds qui a été largement utilisé pour la traduction d'images entre différents domaines, notamment la traduction d'images de segmentation sémantique en images photo réalistes.

Dans le contexte de la simulation de l'occlusion ambiante, pix2pix est utilisé pour générer des images réalistes qui simulent les effets d'obscurité ou de perte de luminosité dans un environnement 3D donné. En entraînant le modèle sur un ensemble de données approprié, comprenant des paires d'images avec et sans occlusion ambiante, le modèle peut apprendre à reproduire de manière convaincante les effets visuels de l'occlusion dans des scènes variées. Cette approche offre plusieurs avantages, notamment une grande flexibilité dans la modélisation des différents types d'occlusion, une capacité à générer des images photo réalistes et la possibilité d'adaptation à divers scénarios et applications. Cependant, elle comporte également des défis, tels que la nécessité d'un ensemble de données de qualité et d'une architecture de réseau appropriée pour obtenir des résultats satisfaisants.

Pour présenter notre travail, nous avons divisé notre mémoire en trois chapitres principaux comme suit :

Dans le premier chapitre, nous explorons l'utilisation de l'occlusion ambiante dans la modélisation tridimensionnelle, qui est utilisée pour améliorer le réalisme d'un rendu en tenant compte des

ombres de contact. Cette technique permet d'assombrir les zones peu accessibles à la lumière et de mettre en valeur le relief des objets.

Dans le chapitre suivant, nous proposons un aperçu détaillé de l'apprentissage automatique, en nous concentrant spécifiquement sur les divers types de réseaux de neurones profonds. Nous examinons notamment les réseaux de neurones convolutifs (CNNs) et les réseaux antagonistes génératifs (GANs), en expliquant leurs principes de fonctionnement, leurs architectures distinctives, et leurs applications courantes dans des domaines tels que la reconnaissance d'images, la génération de contenu et la modélisation prédictive.

Dans le dernier chapitre, nous présentons notre technique et explorons la simulation d'occlusion ambiante en utilisant des modèles de type pix2pix. Nous analysons en profondeur les performances de ces modèles, leur capacité à reproduire fidèlement les nuances de l'occlusion ambiante, ainsi que les avantages qu'ils offrent par rapport aux méthodes traditionnelles.

Nous conclurons ce mémoire par une synthèse générale qui récapitule les principaux aspects de notre recherche. Cette conclusion mettra en lumière les objectifs initiaux, les méthodologies employées, les résultats obtenus, ainsi que les implications et les perspectives futures de notre travail. Elle offrira une vue d'ensemble des contributions apportées et des avancées réalisées, tout en proposant des pistes pour des recherches ultérieures.

Chapitre I

Occlusion Ambiante

I.1 Introduction

L'occlusion ambiante est un concept fondamental en vision par ordinateur et en infographie. Il fait référence à la diminution de la luminosité ou de la visibilité d'un objet lorsqu'il est partiellement caché par d'autres objets en d'autres termes, lorsque la lumière est bloquée ou partiellement bloquée par un objet, cela crée une zone d'ombre ou d'obscurité sur les surfaces environnantes, ce qui peut altérer la perception visuelle de ces surfaces, dans le domaine de la modélisation et du rendu 3D, la prise en compte de l'occlusion ambiante est cruciale pour obtenir des images réalistes et précises, elle permet de simuler les effets de la lumière indirecte et des interactions entre les objets dans une scène. En effet, même lorsque la lumière directe ne frappe pas directement une surface, elle peut être réfléchie ou diffusée par d'autres objets environnants, créant ainsi des zones d'ombre subtiles mais importantes.

I.2 Lumière ambiante

La lumière ambiante est la lumière qui provient de toutes les directions et qui éclaire uniformément la scène, elle n'a pas de source précise, elle est le résultat de la diffusion de la lumière par l'atmosphère, les nuages, les murs, etc.. la lumière ambiante est souvent utilisée pour créer un éclairage de base, qui peut ensuite être complété par des sources de lumière directes, comme le soleil, les lampes, les feux, etc[1](Figure I.1).

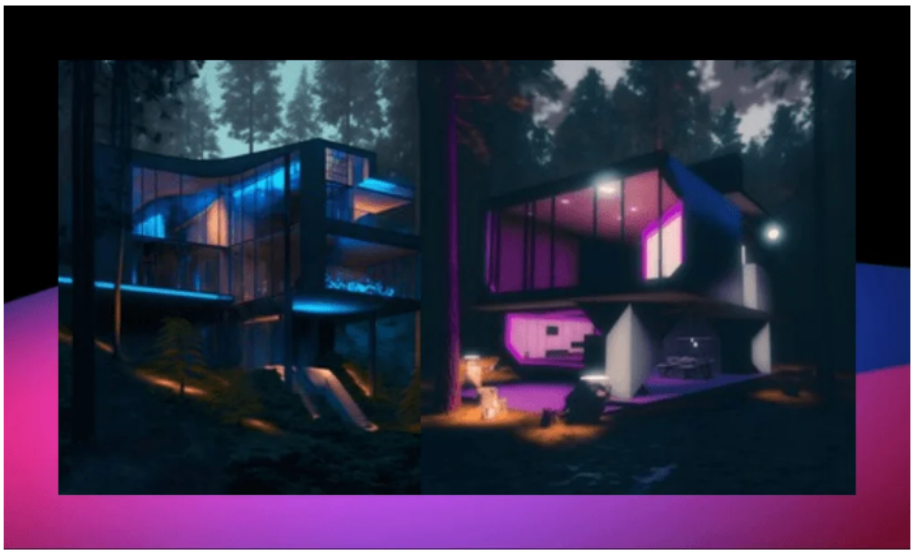

Figure I.1 – Lumière ambiante[1]

I.3 L'occlusion ambiante

L'occlusion ambiante est une technique de rendu graphique qui permet de simuler les effets de la lumière ambiante sur les objets 3D. Elle consiste à assombrir les zones où la lumière est bloquée ou réfléchie par d'autres objets, créant ainsi des ombres douces et des contrastes réalistes. L'occlusion ambiante est importante pour le réalisme car elle ajoute de la profondeur et du relief à la scène, en tenant compte de la géométrie et de l'orientation des objets.Il existe différents types d'occlusion ambiante, qui se basent sur des algorithmes et des paramètres différents, offrant des résultats plus ou moins fidèles et performants[1](Figure I.2).

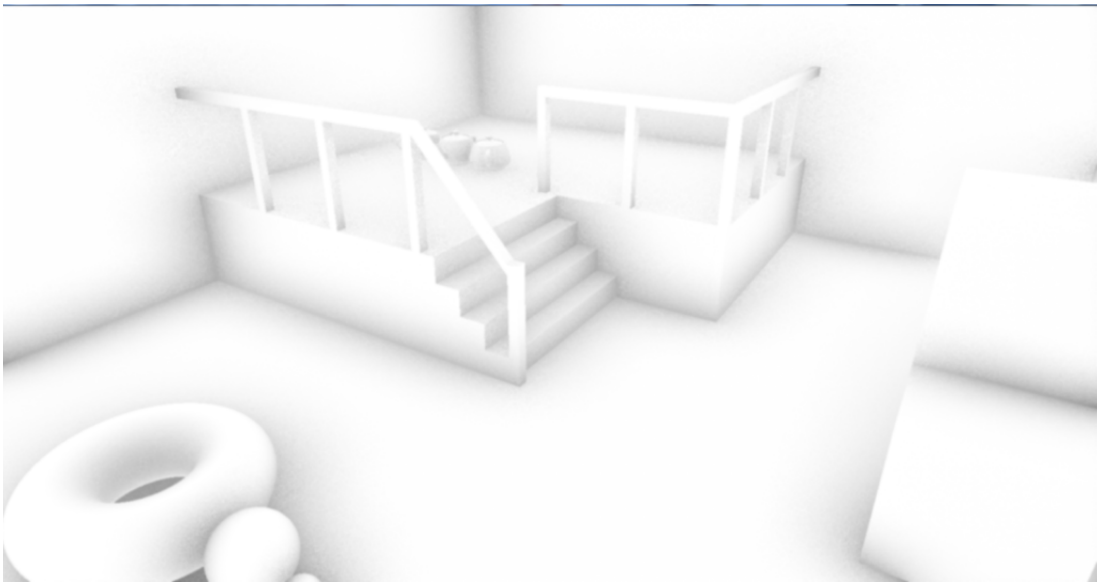

FIGURE I.2 – Occlusion ambiante[1]

Nous débutons par le dessin d'un quad en plein écran afin de calculer le terme AO pour chaque pixel [6], nous sélectionnons un nombre défini de triangles à partir du tampon triangulaire. Ces triangles sont ensuite intégrés dans l'algorithme AO normal.
Le calcul de l'occlusion ambiante peut être effectué de différentes manières comme l'approche couramment utilisée appelée "occlusion ambiante écran-espace".

Le principe de l'occlusion ambiante écran-espace est :

1-Préparation de la géométrie : Tout d'abord, vous devez préparer la géométrie de votre scène 3D. Cela inclut les modèles d'objets, les maillages, les normales des surfaces, etc(Figure I.3).

FIGURE I.3 – Placer la géométrie dans la scène[2]

2-Création d'une carte de profondeur : Vous devez générer une carte de profondeur de votre scène en utilisant la caméra virtuelle. Cette carte de profondeur attribue à chaque pixel de l'écran une valeur représentant la distance entre la caméra et l'objet le plus proche(Figure I.4).

FIGURE I.4 – Calculer la profondeur de la scène[2]

3-Échantillonnage des points : Ensuite, vous devez échantillonner un certain nombre de points dans un motif aléatoire sur l'écran. Ces points serviront de points de test pour calculer l'occlusion ambiante(Figure I.5).

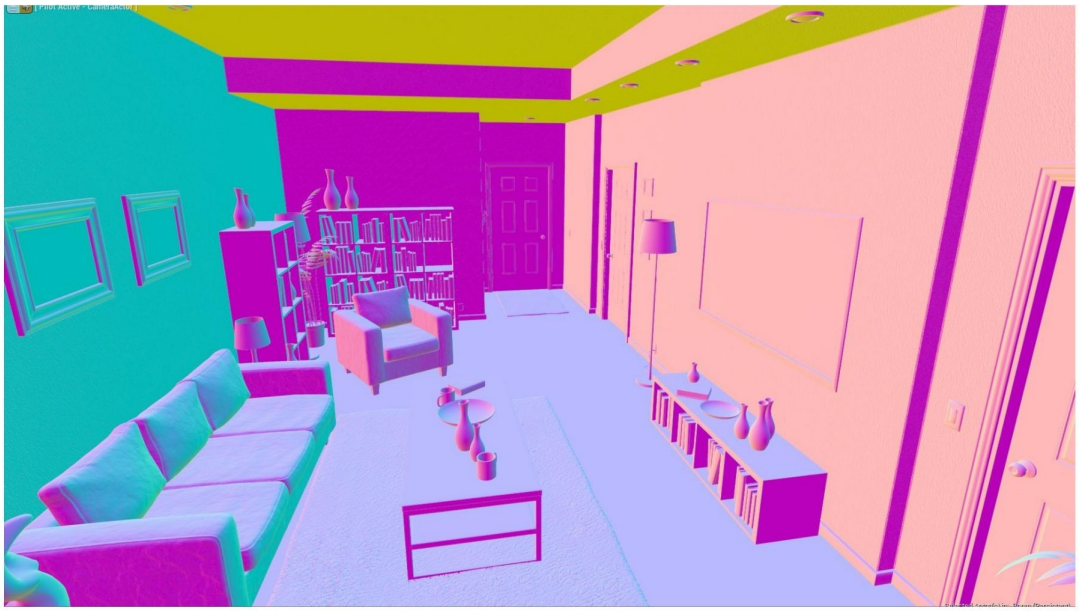

FIGURE I.5 – Calculer les valeurs normales[2]

4-Calcul de l'occlusion ambiante : Pour chaque point échantillonné, vous effectuez une vérification de visibilité. Vous lancez un rayon depuis le point de test vers la caméra et vérifiez s'il intersecte des objets dans la scène. Si le rayon est bloqué, cela signifie que le point est dans l'ombre et vous augmentez la valeur d'occlusion ambiante pour ce point(Figure I.6).

FIGURE I.6 – Le tampon d'occlusion[2]

5-Agrégation des résultats : Après avoir effectué le calcul pour tous les points échantillonnés, vous agrégez les résultats pour obtenir une valeur d'occlusion ambiante pour chaque pixel de l'écran. Cela peut être fait en moyennant les valeurs des points échantillonnés voisins(Figure I.7).

FIGURE I.7 – Combiner avec le reste des tampons[2]

6-Application d'occlusion ambiante : Enfin, vous pouvez utiliser la valeur d'occlusion ambiante calculée pour modifier l'éclairage de votre scène. Cela peut être fait en multipliant l'intensité de la lumière ambiante par la valeur d'occlusion ambiante, ou en utilisant d'autres techniques pour moduler l'éclairage(Figure I.8).

FIGURE I.8 – Effets post-traitement[2]

I.4 Différence entre l'occlusion ambiante et les ombres portées

Les ombres portées sont des ombres que font les objets lorsqu'ils sont éclairés par une source de lumière directe. Elles dépendent de la position et de la forme de l'objet, ainsi que de la direction et de la distance de la source de lumière. Les ombres portées sont souvent utilisées pour créer un contraste et une profondeur entre les objets et le fond. L'occlusion ambiante est l'ombre que font les objets lorsqu'ils sont éclairés par la lumière ambiante. Elle dépend de la proximité et de l'orientation des objets entre eux, ainsi que du rayon et de l'intensité de la lumière ambiante. L'occlusion ambiante est souvent utilisée pour créer un réalisme et une finesse dans les détails des objets[1].

I.5 Travaux connexes

Dans le domaine des jeux vidéo, parvenir à un rendu visuel réaliste implique une simulation complexe de l'éclairage. Cette simulation, basée sur la résolution de l'équation de rendu [35], vise à reproduire le transport de la lumière en calculant à la fois l'éclairage direct et indirect. Connue sous le nom d'illumination globale, cette méthode est cruciale pour recréer des effets d'éclairage réalistes. Les techniques traditionnelles, telles que le lancer de rayons, offrent une grande précision, mais leur utilisation est limitée dans les jeux vidéo en raison du temps de calcul nécessaire pour simuler tous les aspects de l'éclairage.

Pour remédier à cela, une méthode couramment utilisée est l'occlusion ambiante, qui estime la quantité de lumière obstruée par la géométrie d'un objet. Par exemple, le jeu Crysis a initialement adopté une technique d'occlusion ambiante basée sur l'espace écran [36]. Plusieurs variantes ont depuis été développées pour améliorer cette méthode, telles que HBAO (occlusion ambiante basée sur l'horizon) et VAO (occlusion ambiante volumétrique). Des approches plus récentes, comme NNAO (occlusion ambiante de réseau neuronal) [37], exploitent les capacités des réseaux de neurones pour générer des effets d'éclairage plus réalistes.

L'intégration de modules d'attention dans ces modèles génératifs [38] a également été explorée pour améliorer la modélisation des relations spatiales dans les images générées. Ces avancées promettent d'augmenter les performances et de produire des rendus de haute résolution dans les jeux vidéo.

Il existe différents types d'occlusion ambiante, qui se basent sur des méthodes et des paramètres différents :

I.5.1 Occlusion ambiante de l'écran-espace

Occlusion ambiante de l'écran-espace (en Anglais Screen Space Ambient Occlusion : SSAO) est le type d'occlusion ambiante le plus simple et le plus courant. Il se base sur une idée simple : plus deux objets sont proches l'un de l'autre, plus ils font de l'ombre l'un sur l'autre. Le SSAO mesure donc la distance entre les objets à l'écran[1], en utilisant un tampon nommé le z-buffer. Le z-buffer, c'est comme une carte qui indique la distance entre chaque point de l'écran et la caméra. Ensuite, le SSAO compare la distance entre les points voisins pour savoir s'ils font de l'ombre ou pas. Si c'est le cas, il assombrit le point en fonction du niveau d'ombre(Figure I.9).

— Espace d'écran, indépendant de la complexité de la scène.

— Quelle quantité de points d'échantillonnage aléatoires sont masqués, à l'aide de la carte de profondeur.

— Flou pour éliminer le bruit[2].

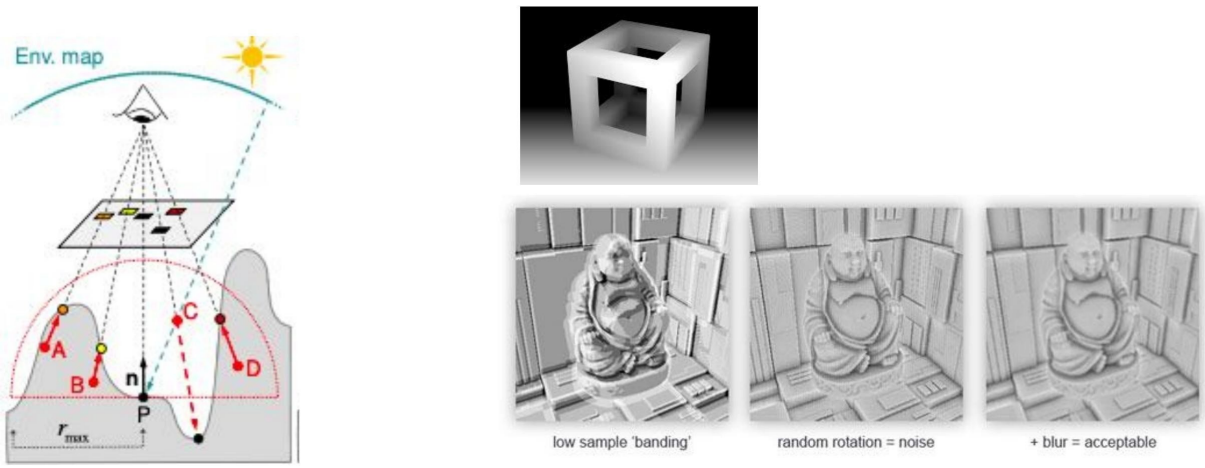

FIGURE I.9 – SSAO :Screen Space Ambient Occlusion[2]

I.5.1.1 Avantages

— Elle permet d'estimer l'illumination globale d'une image de synthèse, contribuant ainsi à un rendu plus réaliste.

— En calculant localement le degré d'exposition à la lumière ambiante de chaque point de l'image, elle crée des effets d'ombre et de lumière plus naturels.
— En simulant les interactions lumineuses entre les formes géométriques avoisinantes, elle renforce le réalisme visuel des scènes 3D.
— Simple et rapide à réaliser, il s'adapte aux mouvements de la caméra et des objets.

I.5.1.2 Inconvénients

— La complexité de calcul peut entraîner une augmentation du temps de rendu, surtout pour des scènes complexes.
— Une mauvaise configuration de l'occultation ambiante peut parfois conduire à des artefacts visuels indésirables, nécessitant une expertise pour son utilisation optimale.
— Dans certains cas, une occultation ambiante excessive peut assombrir excessivement certaines parties de l'image, affectant la lisibilité ou la qualité visuelle globale.
— Il ne tient pas compte de la direction ou de la couleur des objets, il ne fait pas la différence entre les objets proches et les objets lointains, et il ne gère pas bien les bords ou les transparences. Il peut donc faire des ombres trop sombres, trop floues ou mal placées, qui ne sont pas très réalistes.

I.5.2 HBAO :Horizon-Based Ambient Occlusion

Le HBAO [3] est un type d'occlusion ambiante plus avancé que le SSAO. Il se base sur une idée plus complexe : plus un objet a un horizon élevé, plus il reçoit de la lumière ambiante. L'horizon d'un objet, c'est comme sa ligne de vue vers le ciel.

— Espace d'écran
— Choisissez une direction. Tracez la hauteur pour approximer l'occlusion.
— Demi résolution en raison de la lenteur du calcul et des artefacts.
— HBAO+ résout ces problèmes[3](Figure I.10).

Par exemple :

— Si tu es au sommet d'une montagne, tu as un horizon très élevé, car tu vois beaucoup de ciel.
— Si tu es dans une vallée, tu as un horizon plus bas, car tu vois moins de ciel.

Le HBAO mesure donc l'horizon de chaque point à l'écran, en utilisant une image qui s'appelle

le normal buffer. Le normal buffer, c'est comme une carte qui indique la direction de la surface de chaque point. Ensuite, le HBAO compare l'horizon des points voisins pour savoir s'ils font de l'ombre ou pas. Si c'est le cas, il assombrit le point en fonction du niveau d'ombre.

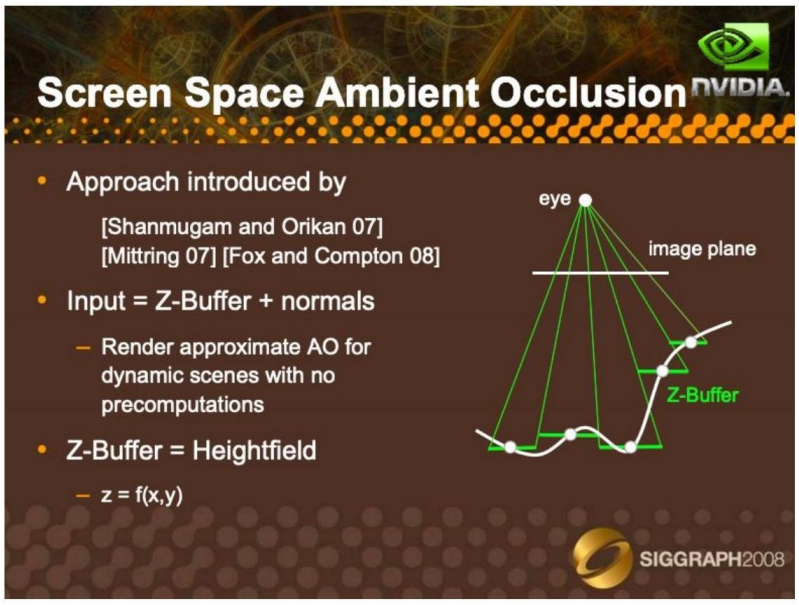

FIGURE I.10 – HBAO :Horizon-Based Ambient Occlusion[3]

I.5.2.1 Avantages

Il est plus précis et plus réaliste que le SSAO. Il tient compte de la direction et de la couleur des objets, il fait la différence entre les objets proches et les objets lointains, et il gère mieux les bords et les transparences. Il fait donc des ombres plus fines, plus nettes et mieux placées, qui renforcent le réalisme de la scène.

I.5.2.2 Inconvénients

— Il est plus coûteux et plus complexe que le SSAO. Il nécessite de mesurer l'horizon de chaque point, ce qui demande plus de ressources.
— Il dépend aussi de la qualité du normal buffer, qui peut être imprécis ou bruité.
— Il ne marche pas avec certain domaines, il faut qu'ils fournissent le normal buffer ou qu'ils utilisent un moteur graphique compatible.

I.5.3 HDAO :High Definition Ambient Occlusion

Le HDAO[1] est un type d'occlusion ambiante plus performant que le HBAO,il se base sur une idée plus adaptative : plus un point a besoin d'échantillons, plus il a de chances d'être occlus par les objets environnants. Les échantillons sont comme des points de vue qu'on prend autour d'un point pour voir s'il est éclairé ou pas (Figure I.11). Par exemple :

— Si tu es dans une pièce sombre, tu as besoin de beaucoup d'échantillons pour voir si tu es occlus par un meuble ou un mur.

— Si tu es dans une pièce claire, tu as besoin de moins d'échantillons pour voir si tu es occlus par une fenêtre ou une lampe.

Le HDAO calcule donc le nombre d'échantillons de chaque point à l'écran, en utilisant une méthode qui varie le nombre d'échantillons en fonction de la distance et de l'angle des points. Ensuite, il compare le nombre d'échantillons des points voisins pour savoir s'ils font de l'ombre ou pas. Si c'est le cas, il assombrit le point en fonction du niveau d'ombre.

FIGURE I.11 – SSAO, HBAO+ et VXAO[1]

I.5.3.1 Avantages

Il est de qualité et performant. Il utilise un nombre optimal d'échantillons pour chaque point, ce qui évite les pertes de qualité ou les gaspillages de ressources. Il fait donc des ombres plus détaillées et plus profondes, qui améliorent la qualité de la scène.

I.5.3.2 Inconvénients

Il est incompatible et difficile à régler, qui doit choisir le nombre d'échantillons minimum et maximum, ainsi que le rayon et l'intensité de l'occlusion.

I.5.4 VXAO :Voxel Accelerated Ambient Occlusion

Le VXAO[1] est un type d'occlusion ambiante plus avancé que le HDAO. Il se base sur une idée plus complète : plus un point a une représentation voxelisée précise de la scène, plus il a de chances d'être occlus par les objets environnants. La représentation voxelisée, c'est comme une maquette en 3D qui reproduit la scène avec des petits cubes appelés voxels. Par exemple : si tu as une maison en voxels, tu peux voir tous les détails de sa structure et de son éclairage. Le VXAO crée donc une représentation voxelisée de la scène à l'écran, en utilisant une méthode qui s'appelle le voxel cone tracing. Le voxel cone tracing, c'est comme si on envoyait des cônes virtuels dans toutes les directions depuis chaque point pour voir quels voxels bloquent la lumière, Ensuite, il compare la représentation voxelisée des points voisins pour savoir s'ils font de l'ombre ou pas. Si c'est le cas, il assombrit le point en fonction du niveau d'ombre(Figure I.12).

— Espace voxel (comme une structure d'accélération de partitionnement spatial).

— Diviser la scène en petits voxels[4].

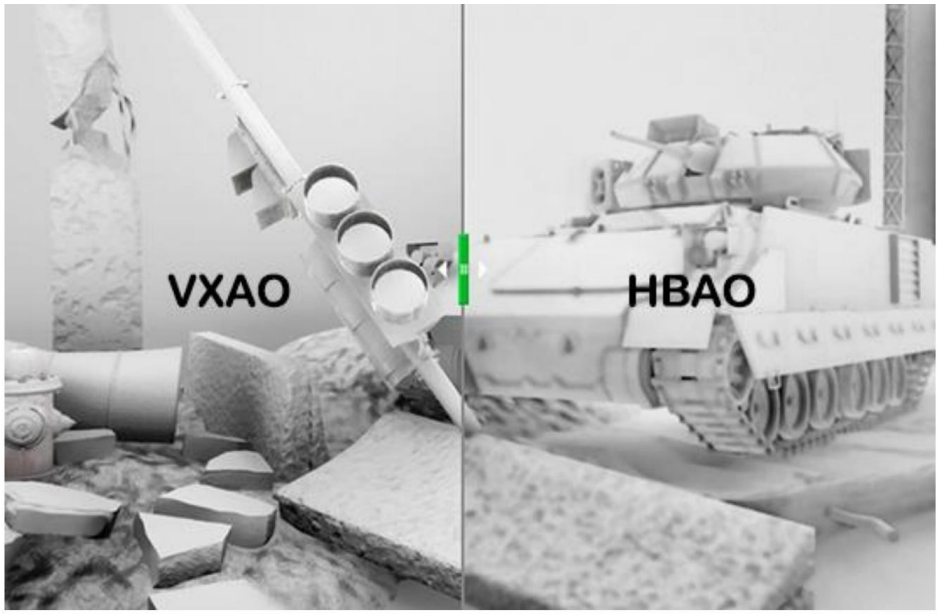

FIGURE I.12 – VXAO :Voxel Accelerated Ambient Occlusion[4]

I.5.4.1 Avantages

Il est détaillé et profond. Il utilise une représentation voxelisée complète et précise de la scène, ce qui permet de capturer tous les effets d'occlusion possibles, y compris ceux dus aux objets dynamiques ou aux sources lumineuses. multiples. Il crée donc des ombres très fines et très profondes, qui augmentent la profondeur de la scène.

I.5.4.2 Inconvénients

Il est gourmand et limité. Il nécessite de créer une représentation voxelisée de la scène, ce qui demande beaucoup de ressources, notamment de la mémoire vidéo. Il n'est pas compatible, il faut qu'ils utilisent un moteur graphique spécifique.

I.6 Ray tracing

Le ray tracing est une technique de rendu graphique qui consiste à simuler le trajet des rayons lumineux dans la scène 3D, en tenant compte des interactions avec les objets et les matériaux. Le ray tracing permet donc de calculer l'occlusion ambiante de manière plus réaliste et plus précise que les techniques actuelles, mais il demande aussi beaucoup plus de ressources(Figure I.13).

— SSAO n'est qu'une simple approximation.
— SSAO ne peut pas gérer l'occlusion hors écran.
— SSAO rend l'image floue pour éliminer le bruit[5].

FIGURE I.13 – Ray tracing[1]

I.7 L'intelligence artificielle au Service de l'Occlusion Ambiante

L'intelligence artificielle est une technique qui consiste à utiliser des algorithmes capables d'apprendre et d'évoluer en fonction des données fournies. L'intelligence artificielle peut donc être utilisée pour optimiser le calcul de l'occlusion ambiante, en adaptant le nombre d'échantillons ou le rayon d'occlusion en fonction de la scène ou du mouvement, elle peut aussi être utilisée pour corriger les artefacts ou les imprécisions causés par les techniques actuelles, en utilisant des réseaux de neurones ou des modèles pré-entraînés. Mais cette utilisation de l'intelligence artificielle soulève aussi des questions sociales et économiques. Rien qu'en France, l'utilisation de l'IA tue 217 emplois[1]. Elle peut aussi être utilisée pour corriger les artefacts ou les imprécisions causés par les techniques actuelles, en utilisant des réseaux de neurones ou des modèles pré-entraînés.

I.8 Exploration de la Réalité Virtuelle pour l'Occlusion Ambiante

La réalité virtuelle est une technique qui consiste à plonger le joueur dans un environnement 3D immersif, en utilisant un casque ou des lunettes spéciales. La réalité virtuelle peut donc bénéficier de l'occlusion ambiante, qui renforce le réalisme et la profondeur de la scène. Mais elle pose aussi des contraintes, comme la nécessité d'avoir un nombre d'images par seconde élevé et constant, pour éviter les nausées ou les maux de tête. Il faut donc utiliser des techniques d'occlusion ambiante adaptées à la réalité virtuelle, comme le Foveated Rendering, qui consiste à réduire la qualité graphique des zones périphériques du champ de vision[1].

I.9 Bilan

Chaque méthode d'occlusion ambiante présente des avantages et des inconvénients spécifiques, adaptés à différentes exigences de performance et de qualité visuelle :
SSAO pour une implémentation rapide et performante.
HBAO pour une meilleure qualité visuelle sans trop de surcoût.
HDAO pour un compromis qualité/performance optimisé pour les GPU AMD.
VXAO pour des projets haut de gamme nécessitant une qualité visuelle supérieure.

Ray tracing pour une précision et une qualité exceptionnelles, mais à un coût élevé.
IA pour des améliorations potentielles de la qualité et de la performance grâce à l'apprentissage automatique.
VR pour des applications nécessitant une immersion et des performances élevées spécifiques à la réalité virtuelle.

I.10 Conclusion

L'occlusion ambiante est un élément crucial de l'éclairage en infographie, contribuant significativement au réalisme des rendus d'images 3D. En simulant de manière approximative les ombres douces et le transfert de lumière indirecte dans les zones obscurcies, elle apporte du contraste, des détails et une profondeur visuelle aux scènes 3D. C'est pourquoi elle est largement intégrée dans la plupart des moteurs de rendu 3D modernes, aussi bien pour des applications en temps réel comme les jeux vidéo que pour la production d'images photoréalistes hors ligne. Des approches émergentes basées sur l'apprentissage profond, telles que l'utilisation de l'architecture Pix2Pix, pourraient permettre de simuler l'occlusion ambiante de manière entièrement basée sur les données, sans nécessiter de calcul explicite. Bien que des défis subsistent, cette piste représente un moyen prometteur d'accélérer considérablement les rendus tout en maintenant une qualité visuelle élevée.

Dans ce chapitre, nous explorons l'utilisation d'occlusion ambiante dans la modélisation tridimensionnelle afin d'améliorer le réalisme d'un rendu en rendant compte des ombres de contact. Cette technique permet d'assombrir les zones peu accessibles à la lumière et de mettre en relief le relief des objets. Diverses techniques, telles que le ray tracing et l'utilisation de l'intelligence artificielle, peuvent être employées pour améliorer le calcul de l'occlusion ambiante.

Chapitre II

Apprentissage automatique

II.1 Introduction

L'apprentissage automatique, et en particulier l'apprentissage profond, a révolutionné de nombreux domaines en permettant aux machines d'apprendre à partir de données et de réaliser des tâches complexes sans être explicitement programmées pour les effecteurs cette approche est devenue cruciale dans de nombreux secteurs, de la reconnaissance vocale à la conduite autonome, l'apprentissage profond, une sous-catégorie de l'apprentissage automatique, est caractérisé par l'utilisation de réseaux de neurones profonds, qui sont capables d'apprendre des représentations de données hiérarchiques ces réseaux de neurones, composés de nombreuses couches interconnectées, sont capables de capturer des modèles complexes et abstraits à partir de données brutes. Parmi les modèles les plus fascinants issus de l'apprentissage profond, on trouve les modèles génératifs, ces modèles ont la capacité de générer de nouvelles données réalistes, telles que des images, du texte ou même de la musique, à partir d'un ensemble d'apprentissage donné. Ils sont souvent utilisés dans des domaines tels que la création d'œuvres d'art, la synthèse de données et la génération de contenus multimédias, les modèles génératifs fonctionnent souvent selon le principe des génératif adversaire networks (GAN), où deux réseaux neuronaux, le générateur et le discriminateur, sont mis en compétition pour produire des données réalistes. Cette approche a donné lieu à des avancées significatives dans la génération de contenu réaliste et diversifié.
Dans ce chapitre, nous explorerons le paysage de l'apprentissage automatique, en mettant particulièrement l'accent sur l'apprentissage profond et les modèles génératifs, et en examinant comment ces techniques transforment de nombreux aspects de notre vie quotidienne et de nos industries.

II.2 Concept d'apprentissage automatique

L'apprentissage automatique (machine learning en anglais) est un domaine de l'intelligence artificielle qui permet aux systèmes informatiques d'apprendre et de s'améliorer à partir de données, sans être explicitement programmés au lieu d'écrire du code avec des instructions spécifiques pour effectuer une tâche, les algorithmes d'apprentissage automatique construisent un modèle mathématique basé sur des données d'entraînement, afin de faire des prédictions ou des décisions sans avoir été programmés avec des règles statiques, il s'agiter d'entraîner le

système informatique un reconnaître des modèles, des relations et des corrélations dans de vastes ensembles de données, puis d'utiliser ces connaissances pour faire des prédictions ou prendre des décisions concernant de nouvelles données [7].

Au cœur de la ML se trouve le concept d'apprentissage piloté par les données, qui consiste un alimenter ONU système informatique avec de grandes quantités de données et à utiliser des techniques statistiques pour identifier des schémas et des relations dans ces données(Figure II.1) [8].

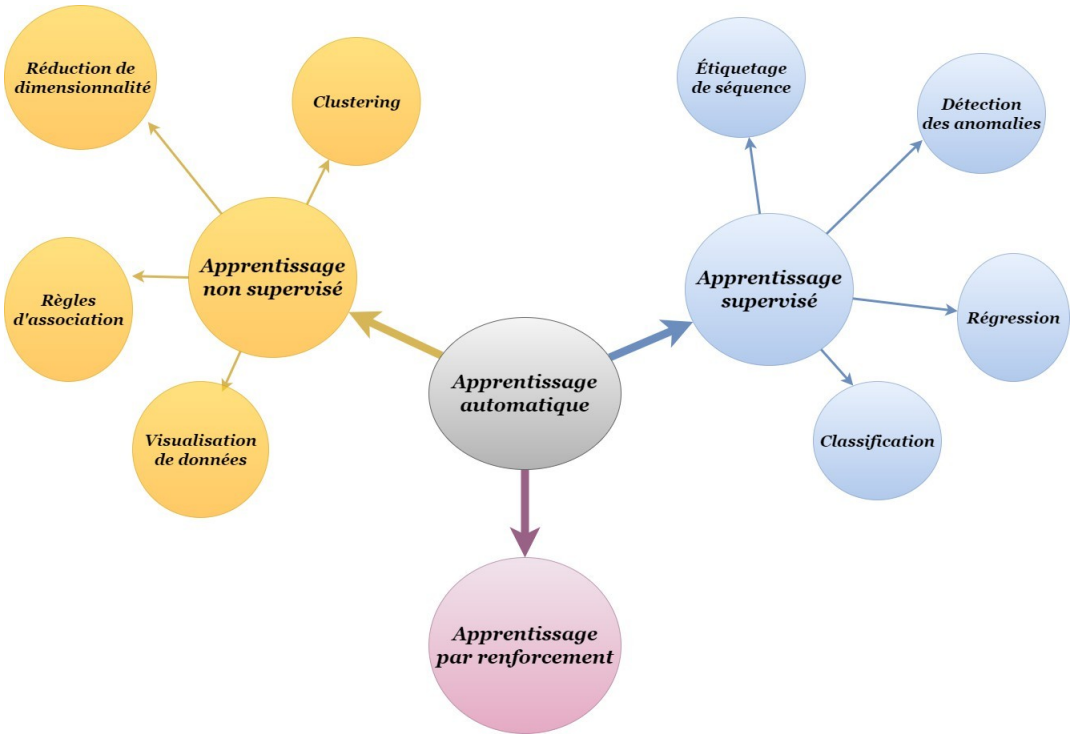

FIGURE II.1 – Les techniques de l'apprentissage automatique

II.3 Types d'algorithmes d'apprentissage automatique

L'apprentissage automatique est un vaste domaine qui regroupe diverses techniques permettant aux systèmes informatiques d'acquérir automatiquement des connaissances à partir de données, sans être explicitement programmés, on distingue principalement trois grands types d'apprentissage automatique selon la nature des données et des tâches à résoudre, peuvent être classés en fonction du mode d'apprentissage qu'ils emploient. Voici une liste des principaux types d'algorithmes d'apprentissage automatique :

II.3.1 Apprentissage supervisé

L'apprentissage supervisé est un type d'apprentissage automatique dans lequel on fournit à l'algorithme un ensemble de données d'entraînement composé d'exemples annotés ou labellisés, le but est que l'algorithme apprenne à établir une correspondance entre les données d'entrée (features) et les sorties désirées (labels ou valeurs cibles), dans cette approche, un modèle est réalisé à partir de données d'entrée (caractéristiques) et des étiquettes correspondants (résultats souhaités). Le modèle apprend à prédire les résultats pour de nouvelles données en généralisant à partir des exemples fournis(Figure II.2) [9].

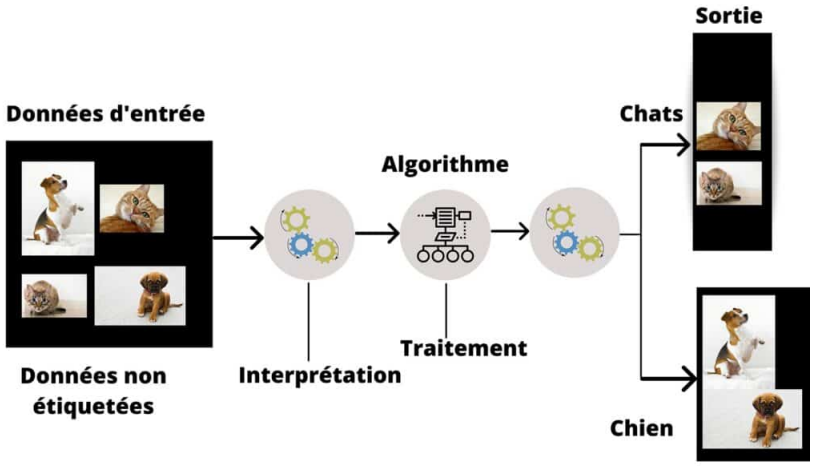

FIGURE II.2 – Apprentissage supervisé[10]

II.3.2 Apprentissage non supervisé

L'apprentissage non supervisé est un type d'apprentissage automatique où l'on ne fournit pas d'exemples labellisés ou de sorties désirées à l'algorithme. Celui-ci doit découvrir par lui-même les structures, motifs et relations présents dans les données d'entrée non annotées, le but est d'extraire des informations pertinentes à partir de données brutes, sans connaissance préalable des résultats attendus. On cherche à faire émerger une représentation interne organisée et informative des données(Figure II.3)[11].

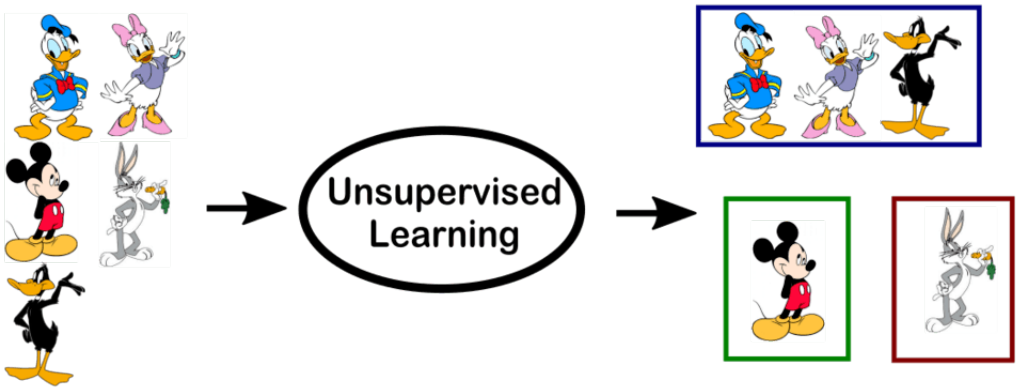

FIGURE II.3 – Apprentissage non supervisé[12]

II.3.3 Apprentissage par renforcement

L'apprentissage par renforcement est une branche de l'apprentissage automatique où un agent apprend à interagir avec un environnement de manière à maximiser une récompense cumulative à long terme. Contrairement à l'apprentissage supervisé qui utilise des données étiquetées, ou non supervisé sur données brutes, l'apprentissage par renforcement se base sur la rétroaction des actions d'un agent dans un environnement. Cette technique implique un agent qui interagit avec un environnement dynamique, l'agent prend des actions et reçoit des récompenses ou des punitions fonction de ses actions[13].

II.3.4 Apprentissage profond

Le Deep Learning est actuellement un concept très en vogue dans le domaine de l'informatique, devenant de plus en plus répandu dans de nombreuses applications en temps réel et représentant une composante essentielle du machine learning, il s'agit d'un domaine où les données sont analysées pour prendre des décisions concernant de nouvelles données, comme illustré dans la Figure II.4 Cette analyse est réalisée à travers l'utilisation de réseaux de neurones, notamment les réseaux de neurones profonds de nombreuses techniques d'apprentissage profond se basent sur l'utilisation de ces réseaux de neurones, qui sont devenus extrêmement populaires, notamment les réseaux neuronaux profonds dans le domaine de l'apprentissage profond, on trouve des architectures de réseaux spécifiques telles que les Convolutional Neural Networks [14], les Recurrent Neural Networks (RNN) et les Generative Adversarial Networks (GAN), qui sont parmi les plus utilisées et les plus populaires.

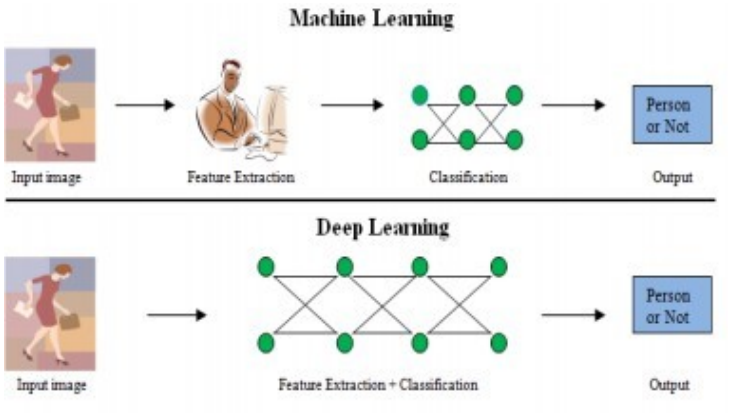

FIGURE II.4 – Machine Apprentissage contre Profond Apprentissage [14]

II.4 Réseau de neurones artificiels (ANN)

L'étude du cerveau humain remonte à des milliers d'années. Avec l'avènement de l'électronique moderne, il est devenu naturel d'explorer ce processus de réflexion, le premier pas vers les réseaux neuronaux artificiels a été franchi en 1943, lorsque Garenne McCulloch, un neurophysiologiste, et un jeune mathématicien, Walter Pitts, ont publié un article sur le fonctionnement potentiel des neurones. Ils ont créé un modèle simplifié de réseau neuronal à l'aide de circuits électriques les réseaux neuronaux, grâce à leur remarquable capacité à extraire des significations à partir de données complexes ou imprécises, peuvent être utilisés pour identifier des modèles et détecter des tendances qui seraient trop subtiles pour être remarquées par les humains ou d'autres techniques informatiques (Figure II.5)[15].

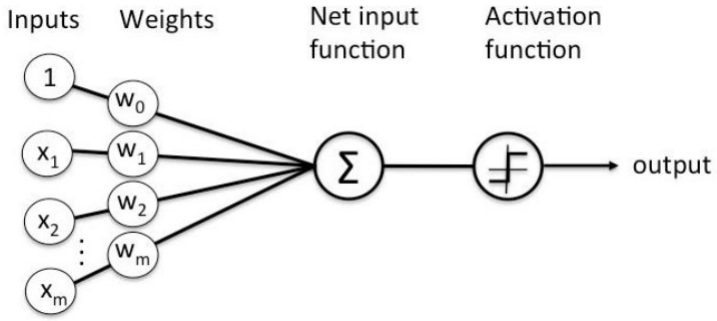

FIGURE II.5 – Schéma représentée le modèle mathématique d'un neurone artificiel[16]

II.5 Convolution Neural Network

Les réseaux de neurones convolutifs (CNN ou ConvNet) sont une variante spécialisée des réseaux de neurones à propagation avant, largement employée dans le traitement d'images, inspirés par l'organisation du cortex visuel des animaux, ils ont été initialement conçus par Fukushima en 1980 pour la reconnaissance de formes, avant de devenir essentiels dans la reconnaissance de caractères. Aujourd'hui, leur utilisation s'étend à divers domaines de l'intelligence artificielle, notamment le traitement automatique des langues, la classification de texte et de musique, ainsi que le traitement du signal.

II.5.1 Principe du CNN

Les réseaux de neurones convolutionnels sont des types de réseaux neuronaux profonds qui comprennent des couches de convolution et des couches entièrement connectées. La convolution est une opération mathématique largement utilisée dans le traitement du signal. Dans les CNN, les couches de convolution utilisent une technique appelée corrélation croisée, qui est très similaire à la convolution du point de vue technique. Les CNN sont composés de deux types de couches principales : les couches d'extraction de caractéristiques et les couches de classification [17].

II.5.2 Les dimensions du CNN

Il existe 3 types d'opérations de convolution avec différentes dimensions 1D, 2D et 3D. La différence entre ces types de convolution s'exprime essentiellement au niveau des tailles des filtres, du nombre de direction de mouvement des filtres sur lentrée A ainsi que de la dimension de la carte de caractéristiques produite en sortie [17] comme suite :

— CNN 1 dimensionnel — Conv1D

— CNN à 2 dimensions — Conv2D

— CNN tridimensionnel — Conv3D

.

II.5.3 Les principaux composants du CNN

Trois types de couches (en anglais Layers) composent le CNN : convolutional layers, pooling layers and fully-connected layers (FC). Lorsque ces couches sont empilées, une architecture CNN est formée. En plus de ces trois couches, il y' a trois paramètres plus importants qui sont dropout layers,the activation function et loss function [18].

II.5.3.1 1D Convolutional layers

La couche de convolution est la première couche utilisée pour extraire différentes caractéristiques des données d'entrée. Elle effectue une opération mathématique de convolution entre les données d'entrée et un filtre de taille particulière M*M. En faisant glisser le filtre sur les données, un produit scalaire est calculé entre le filtre et les parties des données d'entrée correspondant à la taille du filtre (M*M)(Figure II.6)[19]

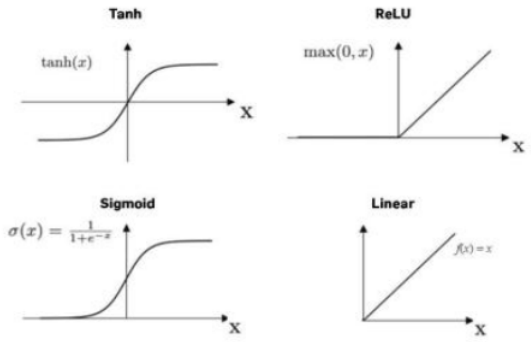

FIGURE II.6 – Les fonctions D'activation [19]

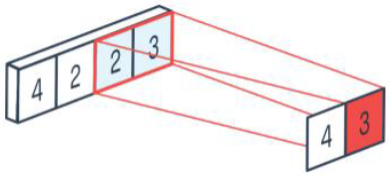

FIGURE II.7 – Exemple sur le Maxpooling [19]

II.5.3.2 Fully connected layers (FC)

Dans cette couche, la carte de caractéristiques des couches précédentes est aplatie et transmise à la couche FC. Le vecteur aplati passe ensuite par plusieurs couches FC supplémentaires où des opérations mathématiques sont généralement effectuées. C'est à ce stade que le processus de classification commence à avoir lieu [19](Figure II.8).

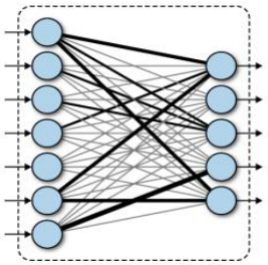

FIGURE II.8 – Représentation des fully-connected Layers [19]

II.5.3.3 Dropout layers

Lorsque toutes les caractéristiques sont reliées à la couche FC, cela peut souvent entraîner un surapprentissage dans l'ensemble de données d'apprentissage, où un modèle s'adapte si bien aux données d'apprentissage qu'il affecte négativement ses performances sur de nouvelles données (Figure II.9) [19].

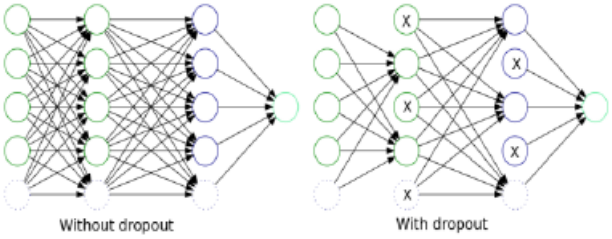

FIGURE II.9 – Exemple sur le fonctionnement des Dropout Layers[19]

II.5.4 Architecture

Les réseaux de neurones convolutifs sont basés sur le perceptron multicouche (MLP) et s'inspirent du fonctionnement du cortex visuel des vertébrés. Bien que les MLP soient efficaces pour le traitement d'images, ils rencontrent des difficultés avec les images de grande taille en raison de l'augmentation exponentielle du nombre de connexions en fonction de la taille de

l'image. Pour résoudre ce problème, les réseaux de neurones convolutifs sont constitués de plusieurs couches, comme illustré dans la (Figure II.10) [19].

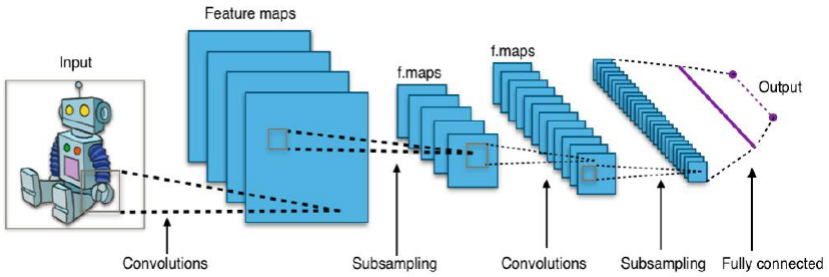

FIGURE II.10 – réseaux de neurone convolutif [19]

II.5.5 Paramètres du CNN

Dans les CNNs, il est crucial de faire des choix éclairés quant au nombre des couches de convolution, de correction ReLU, de pooling et de couches entièrement connectées. Il est également nécessaire de définir les paramètres spécifiques pour chaque couche de convolution et de pooling.

Pour dimensionner une couche de convolution, trois paramètres doivent être déterminés : le nombre de noyaux de convolution, le pas de convolution (stride) et le zéro-padding (zero-padding), qui détermine le nombre de zéros ajoutés aux frontières des cartes. Parfois, il est préférable de conserver la même taille que les cartes d'entrée.

Quant à la couche de pooling, elle est définie par la taille de la fenêtre de traitement et le pas de chevauchement[19].

II.6 Modèles génératifs

Les modèles génératifs sont des algorithmes qui permettent de générer de nouveaux exemples à partir de données existantes. Ils sont utilisés dans divers domaines, notamment dans les réseaux de neurones profonds, les réseaux adversaires génératifs (GAN) et les auto-encodeurs variationnels. Les modèles génératifs peuvent être utilisés pour générer des données de synthèse, comme des images, des vidéos ou des textes, ou pour améliorer le traitement de données existantes en les complétant ou en les augmentant. Les GAN sont un type de modèle génératif qui utilise un modèle générateur et un modèle discriminateur pour générer des données qui ressemblent à celles de l'ensemble d'apprentissage. Les auto-encodeurs variationnels sont une variante des auto-encodeurs qui permettent de générer de nouvelles données en décodant une distribution gaussienne aléatoire, ce qui permet de produire des données plus variées[20].

II.6.1 Auto-encodeur variationnel

En apprentissage automatique, un auto-encodeur variationnel (ou VAE de l'anglais variational auto encoder), est une architecture de réseau de neurones artificiels introduite en 2013 par D. Kingma et M. Welling, appartenant aux familles des modèles graphiques probabilistes et des méthodes bayésiennes variationnelles. Les VAE sont souvent rapprochés des auto-encodeurs[21] en raison de leur architectures similaires. Leur utilisation et leur formulation mathématiques sont cependant différentes. Les auto-encodeurs variationnels permettent de formuler un problème d'inférence statistique (par exemple, déduire la valeur d'une variable aléatoire à partir d'une autre variable aléatoire) en un problème d'optimisation statistique (c'est-à-dire trouver les valeurs de paramètres qui minimisent une fonction objectif). Ils représentent une fonction associant à une valeur d'entrée une distribution latente multivariée, qui n'est pas directement observée mais déduite depuis un modèle mathématique à partir de la distribution d'autres variables. Bien que ce type de modèle a été initialement conçu pour l'apprentissage non supervisé, son efficacité a été prouvée pour l'apprentissage semi-supervisé et l'apprentissage supervisé[22] (Figure II.11).

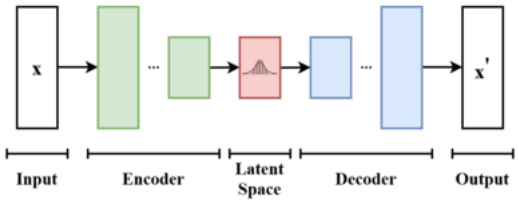

FIGURE II.11 – Le schéma d'un auto-encodeur variationnel[23]

II.6.2 Réseaux contradictoires génératifs

Un modèle d'apprentissage automatique appelé réseau génératif antagoniste (GAN) met en compétition deux réseaux neuronaux pour améliorer la précision de leurs prédictions. Les GAN fonctionnent généralement de manière non supervisée et utilisent un cadre de jeu à somme nulle coopératif pour l'apprentissage. Ces deux réseaux, appelés générateur et discriminateur, sont les composants clés d'un GAN. Le générateur est un réseau neuronal convolutif chargé de produire des sorties qui ressemblent autant que possible à des données réelles, tandis que le discriminateur est un réseau neuronal déconvolutif qui vise à distinguer les sorties générées artificiellement des données réelles. En essence, les GAN créent leur propre ensemble de données d'entraînement. Au fur et à mesure que les réseaux adverses continuent à s'affronter, le générateur améliore la qualité de ses résultats et le discriminateur devient plus efficace pour détecter les données artificiellement générées [24]. Pour atteindre cet objectif, les GAN se composent de deux sous-réseaux en compétition :

— **Le premier réseau**, appelé générateur, est chargé de capturer la distribution des données d'entrée et de produire de nouveaux exemples.

— **Le deuxième réseau**, appelé discriminateur, tente de distinguer les exemples réels issus des données d'entraînement de ceux générés artificiellement par le générateur.

Les deux réseaux sont formés simultanément dans un jeu à somme nulle non coopératif, où le gain d'un réseau se traduit par la perte de l'autre, jusqu'à ce que le discriminateur ne soit plus capable de distinguer entre les deux types d'échantillons(Figure II.12).

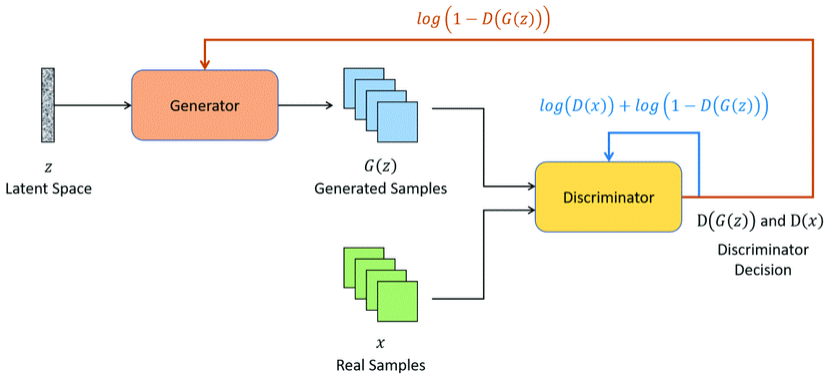

FIGURE II.12 – Architecture typique d'un GAN [25]

II.6.2.1 Architecture d'un réseau GAN

Les GAN sont composés de deux réseaux neuronaux : le générateur (G) et le discriminateur (D). En essence, G et D incarnent des fonctions implicites souvent réalisées à l'aide de réseaux de neurones profonds, illustrant ainsi la structure du modèle GAN. Plus précisément, G capture la distribution des données des échantillons réels et les projettes dans un nouvel espace. Les données générées sont représentées par G(z), dont la distribution est notée pg(z). L'objectif des GAN est de rendre pg(z) aussi similaire que possible à la distribution des échantillons d'entraînement pr(x). Le discriminateur D prend en entrée soit des données réelles x, soit des données générées G(z). Le résultat de D est une probabilité ou un scalaire qui prédit si l'entrée de D provient d'une distribution réelle(Figure II.13)[26].

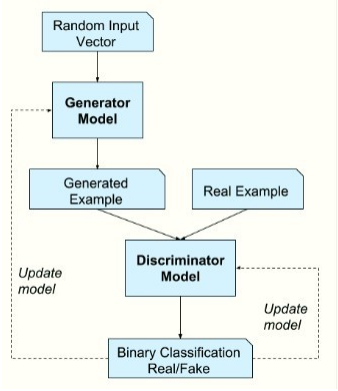

FIGURE II.13 – Architecture d'un Réseau Génératif Adversaire contradictoire[27]

II.6.2.1.a Le générateur

Le générateur est défini par une fonction différentiable G. Il recueille des variables aléatoires z issues d'une distribution antérieure et les transforme à travers un réseau de neurones en distributions de pseudo-échantillons G(z), dans un processus de suréchantillonnage. La variable d'entrée z est généralement un bruit gaussien, qui est une variable aléatoire dans l'espace latent. Pendant l'entraînement du GAN, les paramètres de G et de D sont mis à jour de manière itérative. Lorsque G est entraîné, les paramètres de D restent fixes. Les données générées par G sont considérées comme fausses et sont fournies en entrée à D. L'erreur entre la sortie du discriminateur (G(z)) et l'étiquette de l'échantillon est calculée, et l'algorithme de rétropropagation de l'erreur est utilisé pour mettre à jour les paramètres de G. Le générateur impose peu de contraintes sur les variables d'entrée, qui peuvent être injectées non seulement dans la première couche, mais également dans la dernière couche du réseau. De plus, du bruit peut être ajouté aux couches cachées sous forme de somme pondérée ou de superposition. Les GAN ne restreignent pas la dimension d'entrée de z, qui est généralement un vecteur de 100 dimensions. En outre, G doit être différentiable car les gradients sont propagés à travers le discriminateur pour mettre à jour les paramètres de G et de D[26](Figure II.14).

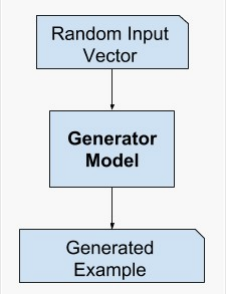

Figure II.14 – Générateur de GAN [27]

II.6.2.1.b Le discriminateur

L'objectif du discriminateur D est de déterminer si l'entrée provient d'un échantillon réel et de fournir un mécanisme de rétroaction qui affine les paramètres de poids de G. Lorsque l'entrée est un échantillon réel X, la sortie de D approche 1. Sinon, la sortie de D approche 0. Pendant l'entraînement du discriminateur, G est maintenu fixe. D reçoit à la fois l'échantillon positif x provenant de l'ensemble de données réel et l'échantillon négatif G(z) généré par le

générateur. Les deux sont fournis en entrée au discriminateur, et la sortie du discriminateur et les étiquettes d'échantillon sont utilisées pour calculer l'erreur. Enfin, l'algorithme de rétropropagation de l'erreur est utilisé pour mettre à jour les paramètres du discriminateur(Figure II.15)[26].

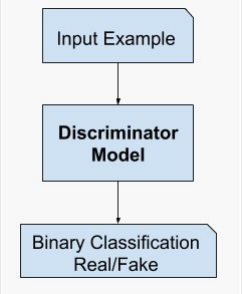

FIGURE II.15 – Discriminateur de GAN [27]

II.6.3 Fonctions de perte dans les GAN

La fonction de perte du GAN est basée sur un jeu à somme nulle entre deux acteurs, représentés par deux réseaux neuronaux qui rivalisent l'un contre l'autre. La fonction discriminatrice est notée D, avec des entrées et des paramètres x et (D), et sa fonction de perte est définie comme suit $[\log D(x)]$:

$$V(D, \theta_D) = -\mathbb{E}_{x \sim p_r(x)}[\log D(x)] - \mathbb{E}_{z \sim p_g(z)}[\log(1 - D(g(z)))]$$

Où p_r représente la distribution des données réelles et p_g représente la distribution des données générées. La fonction génératrice est notée G, avec des entrées et des paramètres z et (G), et sa fonction de perte est définie comme suit :

$$V(G, \theta_G) = \mathbb{E}_{z \sim p_g}[\log(1 - D(g(z)))]$$

Les deux acteurs ont leurs propres fonctions de perte. D doit maximiser $V(D, \theta_D, \theta_G)$ en mettant à jour _D , tandis que G doit minimiser $V(G, \theta_D, \theta_G)$ en mettant à jour _G

Les fonctions de perte des deux acteurs dépendent des paramètres de l'autre. Ils ne peuvent pas mettre à jour les paramètres de l'autre et ne cesseront pas l'entraînement tant qu'un équilibre n'aura pas été atteint. Le GAN est en fait un problème d'optimisation minimax, dont la fonction de perte est définie comme suit :

$$\min_G \max_D V(D,G) = \mathbb{E}_{x\sim p_r(x)}[\log D(x)] + \mathbb{E}_{z\sim p_g(z)}[\log(1 - D(g(z)))]$$

La première partie de cette formule représente le fait que D maximise son objectif lorsque des données réelles sont fournies en entrée. La dernière partie représente que lorsque des données générées sont fournies au discriminateur, ce dernier tente de faire approcher la sortie D(g(z)) de 0, tandis que G tente de faire approcher la sortie D de 1.

Une fois que les deux modèles ont été suffisamment entraînés, le jeu finit par atteindre un équilibre de Nash. Idéalement, D ne peut pas distinguer les échantillons réels des échantillons générés, c'est-à-dire que lorsque la probabilité est équivalente à 1/2, le modèle est optimal. Cela signifie que D ne peut pas distinguer les échantillons réels des échantillons générés.

II.6.4 Les avantages de GAN

Depuis l'émergence des GAN, ils ont été appliqués dans différents domaines avec des améliorations structurelles et des développements théoriques. Leurs avantages comprennent les aspects suivants [26] :

1) Il existe peu d'hypothèses préalables et pratiquement aucune restriction sur les ensembles de données, ce qui permet à GAN d'être appliqué à presque toutes les distributions, comme dans le GAN original proposé par Goodfellow[26].

2) L'objectif final est que les GAN offrent un pouvoir de modélisation infini et peuvent ajuster toutes les distributions, offrant ainsi une grande flexibilité dans la génération de données.

3) La conception du modèle GAN est simple et ne nécessite pas de modèles de fonctions complexes préconçus, ce qui facilite leur utilisation et leur adaptation à différentes tâches.

4) Les GAN fournissent une méthode puissante pour l'apprentissage en profondeur non supervisé, en contournant les limitations des algorithmes traditionnels d'intelligence artificielle qui sont souvent limités par la pensée humaine.

5) Les GAN permettent aux machines d'interagir entre elles à travers un processus de confrontation continue, ce qui leur permet d'apprendre les lois inhérentes du monde réel après une formation adéquate sur les données.

II.6.5 Les inconvénients de GAN

1.Difficulté d'entraînement : Les GAN peuvent être plus complexes à entraîner car ils nécessitent une variété de données continues pour vérifier leur bon fonctionnement, ce qui peut rendre le processus d'entraînement plus laborieux et exigeant en termes de ressources.

2.Sensibilité aux hyperparamètres : Les GAN sont connus pour être sensibles aux hyperparamètres, ce qui signifie qu'ils nécessitent un réglage minutieux et des expérimentations approfondies pour obtenir des performances optimales, ce qui peut rendre leur mise en œuvre plus délicate[28].

3.Vulnérabilité à certaines attaques : Les GAN peuvent être sujets à des attaques, notamment celles exploitant la nature en temps réel du processus d'apprentissage pour générer des échantillons prototypiques de jeux de données d'entraînement sensibles, soulevant ainsi des préoccupations en matière de sécurité et de confidentialité[29].

II.6.6 L'évolution des modèles GAN

Pour résoudre les problèmes du GAN d'origine, tels que la disparition du gradient, la formation instable et la faible diversité, de nombreux nouveaux modèles GAN ont été proposés pour augmenter la stabilité et améliorer la qualité des résultats générés. Parmi ces modèles, nous pouvons citer :

— Réseaux contradictoires génératifs à convolution profonde (DCGAN) [54]
— Conditionnel GAN (CGAN) [55]
— Wasserstein GAN (WGAN) [56]
— WGAN avec pénalité de gradient (WGAN-GP) [57]
— Basé sur l'énergie GAN (EBGAN) [58]
— Frontière équilibre GAN (BEGAN) [59]
— Information GAN (InfoGAN) [60]
— Moins Carrés GAN (LSGAN) [61]
— Auxiliaire Classificateur GAN (ACGAN) [55]
— Dégénérer évité GAN (DRAGAN) [62]
— Spectral Normalisation GAN (SNGAN) [63]

— Jacobien Régularisation GAN (JR-GAN) [64]

— CasquettesGAN [55]

— Banach Wasserstein GAN (BWGAN) [65]

— Décodeur-Encodeur GAN (DEGAN) [66]

Pour être plus précis, ces GAN sont classés en différents types en fonction de leurs fonctions objectives, de leurs structures et de leurs conditions, comme illustré dans la figure II.16 , Les fonctions objectives correspondent aux améliorations de la fonction de perte, tandis que les structures et les conditions correspondent aux développements de l'architecture. Le temps de chaque modèle proposé est également indiqué dans la figure pour montrer les relations entre eux[26].

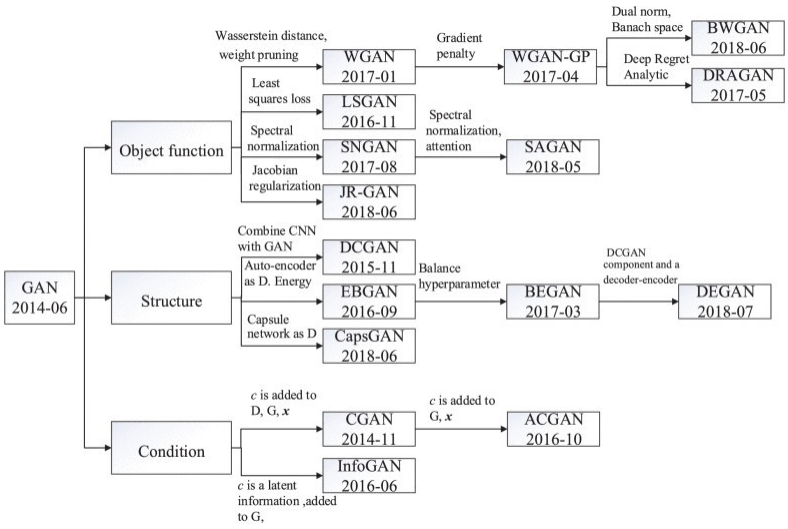

FIGURE II.16 – Schéma classification les modèles des GAN[26]

II.6.7 L'évolution de la structure des GAN

L'amélioration de la structure des GAN vise principalement à stabiliser le modèle et à réduire le retard de convergence. Pour mieux contrôler les modèles, différentes variantes de GAN capables de transporter plus d'informations (telles que CGAN, InfoGAN, ACGAN) ont été proposées[26].

II.6.7.1 Conditionnel Génératif Réseau (CGAN)

En tant que méthode d'apprentissage non supervisé, les GAN apprennent la loi de distribution de probabilité à partir d'ensembles de données non étiquettés et l'expriment au

cours d'un processus lent et non contraint,illustré la figure II.17 Cependant, lorsqu'il s'agit d'ensembles de données complexes ou de grande échelle, il devient difficile pour les GAN de contrôler les résultats générés. Pour résoudre ce problème, une idée naturelle est d'ajouter des contraintes et des cibles spécifiques pour le générateur, formant ainsi le CGAN. Celui-ci prend une variable aléatoire z et des données réelles x, ainsi qu'une variable conditionnelle c pour guider le processus de génération de données. En conséquence, la vitesse de convergence a été considérablement accélérée. La structure du CGAN est illustrée dans la figure II.17 [26].

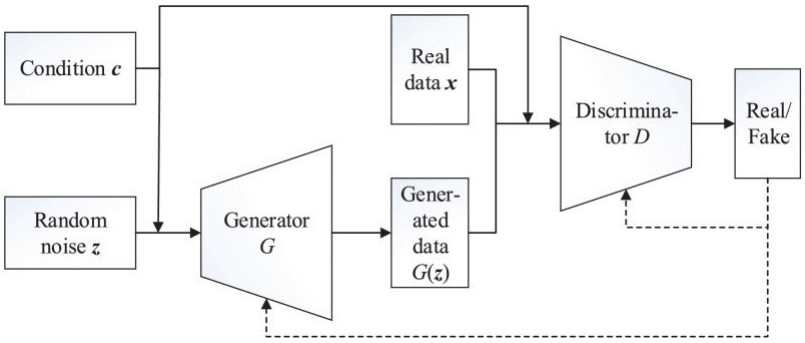

FIGURE II.17 – La structure de base du CGAN[26]

La variable conditionnelle c du CGAN peut être une étiquette de catégorie, transformant ainsi le GAN non supervisé en un modèle supervisé. Le variable C peut également être du texte, telles que des phrases décrivant les images correspondantes, ou une cible générée particulière, proposant un objectif à apprendre. Le CGAN peut non seulement générer des images avec des catégories et des étiquettes spécifiées, mais il peut également utiliser les caractéristiques de l'image comme variable conditionnelle pour générer un vecteur de mots pour l'image[26]. Cette amélioration simple s'avère très efficace et est largement utilisée dans les travaux ultérieurs.

II.6.7.2 Réseaux contradictoires génératifs à convolution profonde

Une étape majeure dans l'histoire des GAN est les Réseaux contradictoires génératifs à convolution profonde DCGAN, dont la structure du générateur est présentée dans la figure II.18, DCGAN combine les GAN avec les CNN, qui sont efficaces dans le domaine de la vision par ordinateur. DCGAN impose des restrictions topologiques au réseau CNN afin de garantir une formation stable et d'utiliser des représentations de fonctionnalités apprises pour clas-

sifier les images. Pour améliorer la qualité des images générées, DCGAN introduit plusieurs améliorations.

Premièrement, il utilise des convolutions à pas fractionnaires sur le discriminateur et le générateur pour remplacer les couches de pooling. Alors que les CNN sont généralement utilisés pour extraire des fonctionnalités, la structure CNN dans DCGAN doit générer des échantillons, ce qui est différent de l'extraction de fonctionnalités. Les convolutions à pas fractionnaires assurent la transmission efficace de l'information à la couche suivante pour garantir l'exhaustivité et la netteté des échantillons générés.

Deuxièmement, DCGAN utilise l'algorithme de normalisation par lots pour résoudre le problème de la disparition du gradient. Ce dernier résout les problèmes liés aux initialisations inadéquates, transmet le gradient à chaque couche et empêche le générateur de converger tous les échantillons vers le même point.

Troisièmement, différentes fonctions d'activation sont utilisées dans DCGAN, telles que la fonction d'activation ReLU, la fonction d'activation fuite ReLU, ainsi que l'optimisation par Adam.

Les résultats démontrent les performances satisfaisantes de DCGAN dans la pratique et confirment la capacité de la structure GAN à générer des échantillons. DCGAN est généralement considéré comme la norme par rapport à d'autres modèles de GAN[26].

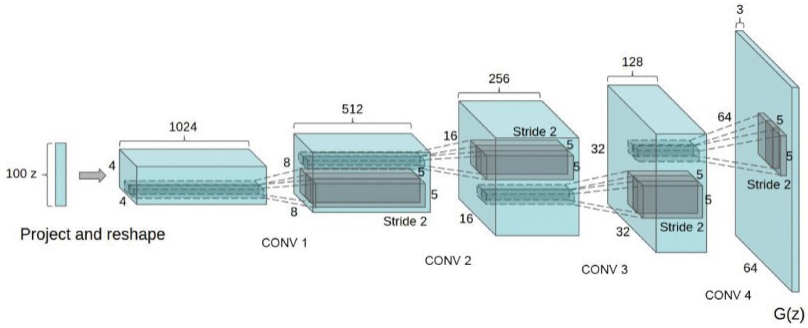

FIGURE II.18 – La structure du générateur de DCGAN[26]

II.6.8 Applications de (GAN)

Les réseaux antagonistes génératifs (GAN) offrent une méthode pour apprendre des représentations profondes sans nécessiter de larges ensembles de données annotées.

ces réseaux réalisent l'apprentissage en propageant des signaux de rétropropagation à travers

un processus compétitif impliquant une paire de réseaux. Les représentations apprises par les GAN peuvent être utilisées dans diverses applications. Les GAN ont fait d'importants progrès et ont montré des performances remarquables dans de nombreux domaines d'application, notamment l'édition d'images sémantiques, le transfert de style, la synthèse d'images, la super-résolution d'images et la classification [30].

II.6.8.1 Applications

Les GAN ont une multitude d'applications dans la vie réelle, parmi lesquelles :

1. CycleGAN
2. Traduction d'images à images

II.6.8.2 CycleGAN

Le transfert inter-domaines par le biais des GAN pourrait bien être l'une des premières applications commerciales majeures. Ces GAN permettent de transformer des images d'un domaine (comme des paysages réels) en un autre domaine (telles que des peintures de Monet ou de Van Gogh)illustrée dans la figure II.19.

FIGURE II.19 – Exemple de CycleGAN [31]

Par exemple, il peut transformer des photos de zèbres en photos de chevaux (Figure II.20)

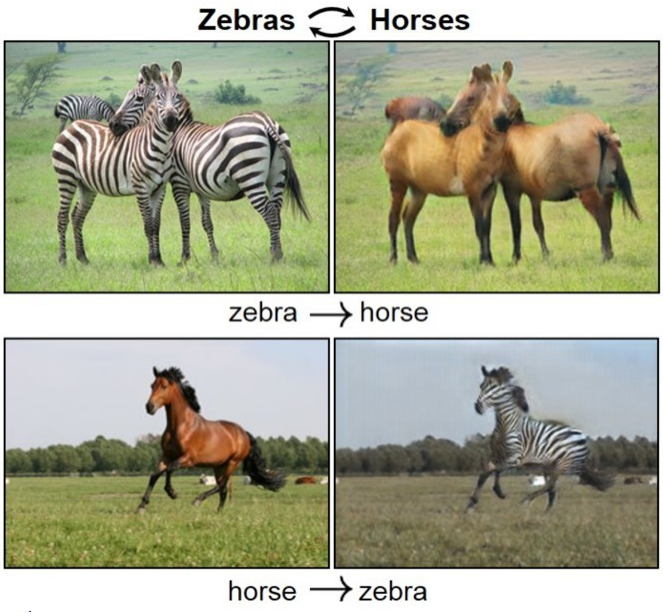

FIGURE II.20 – Exemple de CycleGAN [31]

CycleGAN construit deux réseaux G et F pour transformer des images d'un domaine à un autre et vice versa. Il utilise des discriminateurs D pour évaluer la qualité des images générées et distinguer si l'image est réelle ou générée. Ce processus est décrit dans la Figure II.21, avec la première direction étant du domaine A au domaine B.

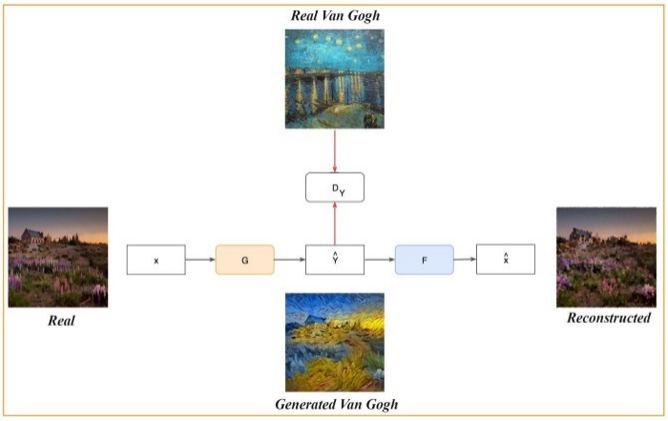

FIGURE II.21 – DomaineA → DomaineB [31]

Nous répétons le processus dans le sens inverse, du domaine B vers le domaine A, comme illustré dans la figure II.22.

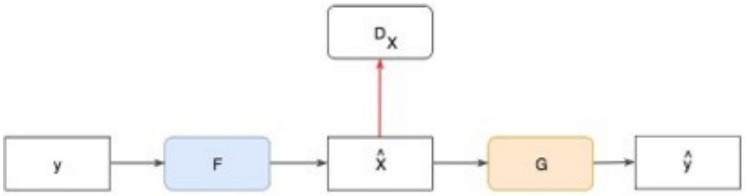

FIGURE II.22 – DomaineB → DomaineA [31]

II.6.9 Traduction d'image à image

Les GAN Pix2Pix (Generative Adversarial Networks) sont utilisés pour la traduction d'images en images, permettant de convertir une image d'une distribution d'entrée en une image d'une distribution de sortie. Cette approche repose sur un générateur qui crée des images synthétiques plausibles et un discriminateur qui distingue les images réelles des images générées. Le générateur est conditionné par une image source et est entraîné à produire des images plausibles dans le domaine cible, tandis que le modèle est également mis à jour via une perte L1 mesurée entre l'image générée et l'image attendue. Les GAN Pix2Pix ont été utilisés avec succès pour diverses tâches de traduction d'images, telles que la conversion de cartes en photos satellite ou de croquis en photographies de produits(Figure II.23)[32].

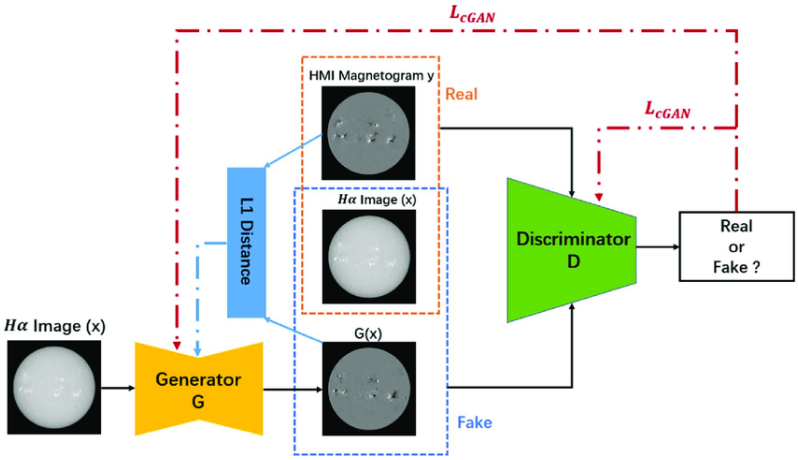

FIGURE II.23 – Architucture Pix2pix[33]

II.6.9.1 Principe de Pix2Pix

Vous avez entendu parler des GAN (Réseaux Génératifs Antagonistes) qui génèrent des images synthétiques réalistes de même, Pix2pix appartient à ce type appelé GAN conditionnel ou CGAN. Ils ont des paramètres conditionnels et apprennent la correspondance d'image à image sous cette condition[34]. Alors que les GAN de base génèrent des images à partir d'un vecteur de distribution aléatoire sans condition appliquée. Étapes impliquées :

- (1)Paires de données d'entraînement (x et y où x : image d'entrée et y : image de sortie)
- (2)Pix2Pix utilise le GAN conditionnel (CGAN) → G : x, z → y. (z → vecteur de bruit, x → image d'entrée, y → image de sortie).
- (3)Réseau de générateur (architecture encode-décode) car une image est l'entrée, nous voulons apprendre la représentation profonde et la décoder et un réseau de discriminateur (PatchGAN).
- (4)Fonction de perte CGAN et distance L1 ou L2.

II.6.9.2 Générateur U-net

Comme nous le savons, le générateur est un réseau encodeur-décodeur (d'abord une série de couches de réduction de l'échantillonnage, puis une couche de goulot d'étranglement, puis une série de couches de suréchantillonnage)[34]. Les auteurs ont utilisé l'architecture "U-Net" avec des connexions sautées comme réseau E-D. Les connexions sautées U-Net sont également intéressantes car elles ne nécessitent aucun redimensionnement, projection, etc., puisque la résolution spatiale des couches connectées correspond déjà entre elles(Figure II.24).

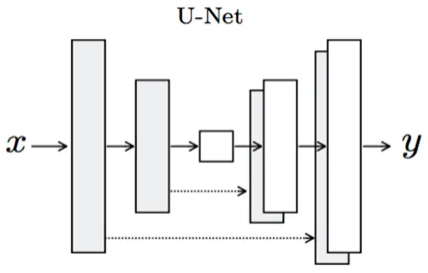

FIGURE II.24 – U-Net architecture[34]

II.6.9.3 Discriminateur PatchGAN

Le discriminateur utilise le réseau PatchGAN. Au lieu de prédire si l'image entière est fausse ou réelle, le modèle prend une image de patch N*N et prédit chaque pixel de ce patch s'il est réel ou faux. Les auteurs soutiennent que cela impose d'avantage de contraintes qui encouragent les détails haute fréquence nets. De plus[34], le PatchGAN a moins de paramètres et s'exécute plus rapidement que la classification de l'image entière.

II.6.9.4 Générateur Loss

Le document inclut également une perte L1 qui est l'erreur absolue moyenne (MAE) entre l'image générée et l'image cible. Cependant, le problème était que cela permettait à l'image générée de devenir structurellement similaire à l'image cible[34].
La fonction de perte du réseau générateur donne comme résultat (Figure II.25) :

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

FIGURE II.25 – Generator loss[34]

Dans les expériences, les auteurs rapportent qu'ils ont trouvé le plus de succès avec le paramètre lambda égal à 100[34].

II.7 Synthèse des travaux

Ce chapitre offre un aperçu complet des avancées récentes dans le domaine de l'apprentissage automatique, en mettant en lumière l'importance croissante des réseaux neuronaux et des modèles génératifs dans la résolution de problèmes complexes et dans la création de nouvelles technologies innovantes. Voici une synthèse des travaux présentés :
L'introduction expose les bases de l'apprentissage automatique, mettant en avant son rôle crucial dans l'analyse de données et la prise de décision automatisée.
Le concept d'apprentissage automatique est détaillé, soulignant son processus d'acquisition de connaissances à partir des données, avec une emphase sur l'apprentissage supervisé, non supervisé, par renforcement, et profond.

Les réseaux de neurones artificiels (ANN) sont explorés comme une méthode fondamentale d'apprentissage automatique, offrant la capacité d'apprendre des modèles complexes à partir de données.
Les réseaux neuronaux convolutifs (CNN) sont présentés comme une approche puissante pour la vision par ordinateur, avec un examen approfondi de leur architecture, de leurs composants et de leurs paramètres.
Les modèles génératifs sont introduits comme une classe de modèles capables de générer de nouvelles données réalistes, avec un accent sur les auto-encodeurs variationnels (VAE) et les réseaux antagonistes génératifs (GAN).
Les GAN sont examinés en détail, avec une discussion sur leur structure, leurs fonctions de perte, leurs avantages et leurs inconvénients, ainsi que leur évolution au fil du temps.
Différentes applications des GAN sont explorées, notamment dans le domaine de la traduction d'images, avec des exemples tels que Pix2Pix GAN et CycleGAN.
Enfin, des techniques spécifiques telles que le générateur U-net et le discriminateur PatchGAN sont présentées pour illustrer la façon dont les GAN peuvent être utilisés dans des tâches spécifiques de traitement d'images.

II.8 Les travaux antérieurs

Les méthodes récentes pour la génération de l'occlusion ambiante reposent sur des réseaux de neurones pour créer des modèles capables d'apprendre à partir de données, produisant ainsi des images photoréalistes [67-68]. Par exemple, [69] présentent une approche basée sur un réseau de neurones pour approximer l'occlusion ambiante. Ce modèle prend en entrée la profondeur et la profondeur autour d'un pixel, et prédit l'occlusion ambiante autour de ce pixel. D'autres modèles, comme celui de [70], utilisent des réseaux de neurones convolutifs pour déduire l'apparence à partir des attributs en se basant sur un grand ensemble de données d'images. Ces modèles peuvent reproduire divers effets, tels que l'occlusion ambiante, le flou, l'éclairage indirect, la diffusion de la lumière et la profondeur de champ, le tout en un temps réduit.
Une autre approche, mentionnée dans [71], utilise un perceptron multicouche pour capturer la non-linéarité des valeurs de l'occlusion ambiante. Ce modèle procède en deux étapes : une phase d'apprentissage sur un jeu de données pour former le réseau de neurones, et une phase

de rendu où le modèle est utilisé pour générer les images finales. L'utilisation d'un réseau de neurones implémenté via un shader permet de calculer l'occlusion ambiante en temps réel, améliorant ainsi l'efficacité du processus de rendu.
Pour les images RVB, [72] utilisent un réseau de neurones génératif pour générer une approximation de l'occlusion ambiante. Ce type de réseau est capable de produire des résultats de haute qualité en apprenant les caractéristiques complexes des données d'entrée et en générant des sorties réalistes.
Fayçal Abbas et al. [73] introduisent une méthode novatrice qui utilise un réseau génératif conditionnel avec un mécanisme d'attention intégré au générateur et au discriminateur pour estimer les ombres douces. Cette approche tire parti du mécanisme d'attention pour permettre au réseau de neurones de se concentrer sur les zones pertinentes de l'image, améliorant ainsi la précision de l'estimation des ombres. L'architecture du modèle permet de générer des images de qualité comparable à la méthode de référence sans nécessiter de prétraitement 3D, ce qui simplifie le processus et réduit le temps de calcul. Le modèle de Fayçal Abbas et al. offre de bons résultats en termes de précision et de performance, démontrant son efficacité dans la génération d'occlusion ambiante.

II.9 Bilan des Méthodes de Génération d'Occlusion Ambiante Basées sur les Réseaux de Neurones

Les techniques modernes pour la génération d'occlusion ambiante (AO) s'appuient largement sur les réseaux de neurones, qui permettent de créer des modèles apprenant à partir de données et produisant des images photoréalistes. Ces méthodes se démarquent par leur capacité à simuler des effets de lumière réalistes, cruciales pour l'amélioration du réalisme dans les images de synthèse.
1. Approximation de l'AO avec Profondeur : Une méthode [69] propose d'utiliser un réseau de neurones prenant en entrée la profondeur et la profondeur environnante d'un pixel pour prédire l'occlusion ambiante autour de ce pixel. Cette approche se montre efficace pour générer des ombres douces et réalistes.
2. Réseaux Convolutifs pour Divers Effets : Une autre méthode [70] utilise des réseaux de neurones convolutifs (CNN) pour déduire l'apparence d'une scène à partir d'attributs dérivés

d'un grand ensemble de données d'images. En plus de l'occlusion ambiante, cette méthode peut reproduire d'autres effets comme le flou, l'éclairage indirect, et la profondeur de champ, le tout en un temps réduit.

3. Perceptron Multicouche pour Non-Linéarité : Une approche mentionnée dans [71] emploie un perceptron multicouche pour capturer la non-linéarité des valeurs d'occlusion ambiante. Ce modèle fonctionne en deux étapes : une phase d'apprentissage sur un jeu de données et une phase de rendu. Cette technique permet un calcul en temps réel de l'occlusion ambiante via un shader.

4. Réseau Génératif pour Images RVB : Dans le cas des images RVB, [72] utilisent un réseau de neurones génératif pour approximations l'occlusion ambiante. Cette approche s'avère particulièrement efficace pour générer des résultats de haute qualité en apprenant les caractéristiques complexes des données d'entrée.

5. Contribution de Fayçal Abbas et al. ont introduit une méthode innovante utilisant un réseau génératif conditionnel avec un mécanisme d'attention intégré au générateur et au discriminateur. Leur approche permet d'estimer les ombres douces de manière précise en se concentrant sur les zones pertinentes de l'image. Cette méthode se distingue par les points suivants :

- Mécanisme d'Attention : Intégré au générateur et au discriminateur, il améliore la précision de l'estimation des ombres en permettant au modèle de focaliser son attention sur les zones pertinentes.
- Pas de Prétraitement 3D : Contrairement à d'autres méthodes, cette approche ne nécessite aucun prétraitement 3D, simplifiant ainsi le processus et réduisant le temps de calcul.
- Qualité et Performance : Les images générées sont de qualité comparable à celles produites par les méthodes de référence, avec une bonne précision et des performances élevées.

II.10 Conclusion

Dans ce chapitre, nous explorons l'apprentissage automatique, un pan essentiel de l'intelligence artificielle qui a récemment permis des progrès considérables. En dotant les systèmes de la capacité d'apprendre par eux-mêmes à partir de données, sans nécessiter de programmation explicite, cette discipline a ouvert la voie à d'innombrables applications innovantes. Les réseaux de neurones artificiels, inspirés du fonctionnement cérébral biologique, sont devenus un outil phare de l'apprentissage automatique. Leur puissance remarquable pour traiter des données complexes telles que les images, le son ou le texte en fait des modèles incontournables. Plus particulièrement, l'avènement en 2014 des réseaux adversaires génératifs (GAN) et de leurs variantes comme les CGAN et Pix2Pix a révolutionné le domaine de l'apprentissage profond génératif. Basés sur des réseaux de neurones en compétition, ces modèles ont rendu possible la génération d'échantillons synthétiques (images, sons, textes) d'un réalisme stupéfiant, ouvrant des perspectives passionnantes mais aussi des défis à relever. Nous prévoyons d'utiliser Pix2Pix dans notre projet, capitalisant ainsi sur sa capacité à générer des images réalistes à partir de données d'entrée.

Chapitre III

Conception, implémentation et résultats

III.1 Introduction

Ce chapitre détaille le développement d'un système novateur utilisant un réseau génératif conditionnel Pix2Pix (CGAN) pour simuler l'occlusion ambiante. Notre contribution repose sur l'utilisation d'une approche simplifiée, différente de celle de Fayçal Abbas et al.[73], qui se base sur l'exploitation d'une unique carte de normale pour générer l'occlusion ambiante, en opposition à l'utilisation de trois cartes distinctes : la carte de normale, la carte de profondeur et la carte de position. Nous débutons en exposant l'environnement de développement ainsi que les bibliothèques utilisées, fournissant ainsi une base technique solide pour notre approche. Par la suite, nous détaillons le fonctionnement du système, décomposant chaque étape cruciale, de la préparation des données à la formation du modèle et à l'évaluation des résultats. Cette exploration mettra en lumière les techniques et méthodologies utilisées pour générer des images réalistes et précises dans des conditions d'occlusion, tout en soulignant l'efficacité de notre approche basée sur l'utilisation d'une unique carte de normale dans le contexte du réseau Pix2Pix.

III.2 Pix2pix GAN

Pix2Pix est un modèle de réseau antagoniste génératif (GAN) conçu pour la traduction d'images à usage général. Ce GAN a été utilisé pour convertir des cartes en photographies satellites, des photographies en noir et blanc en images en couleur, ainsi que des croquis de produits en photographies de produits. Dans Pix2Pix, le générateur est un réseau convolutif avec une architecture de type U-Net, et le discriminateur est un PatchGAN (Figure III.1) [40].

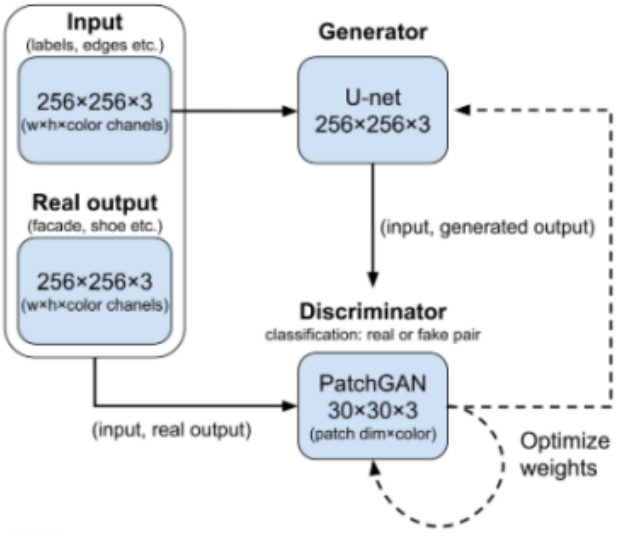

FIGURE III.1 – Architecture de Pix2pix GAN[39]

III.3 Environnement de développement

III.3.1 Google Colab

Colab, développé par Google Research, est un environnement Jupyter gratuit et portable fonctionnant entièrement dans le cloud. Conçu spécifiquement pour le Machine Learning, il offre un accès gratuit aux GPU et permet aux utilisateurs de développer leurs compétences en Python [41][42]. Une caractéristique clé est la possibilité de collaborer en temps réel sur les notebooks, et Colab intègre toutes les bibliothèques essentielles pour le Machine Learning. De plus, il permet d'importer et d'enregistrer des notebooks depuis Google Drive, ainsi que d'importer ou de publier des programmes et des bases de données depuis GitHub ou Kaggle [43].

III.3.2 Le processeur graphique

Les GPU (unités de traitement graphique) sont essentiels pour le traitement des données massives dans les programmes, notamment ceux du Deep Learning, car leur exécution sur un CPU peut prendre des mois. Colab propose gratuitement des GPU Nvidia Tesla K80, ce qui accélère considérablement les calculs et facilite le travail des développeurs, notamment dans les domaines de l'Intelligence Artificielle[41][43].

III.3.3 Python

Python, créé par Guido van Rossum en 1991[44][45], est un langage de programmation polyvalent et facile à apprendre, privilégié pour sa clarté et sa prise en charge de multiples paradigmes de programmation. Son utilisation des espaces favorise une programmation claire, et il offre une vaste bibliothèque standard. Les interpréteurs Python sont disponibles sur de nombreux systèmes d'exploitation[46].

III.4 Description des bibliothèques utilisées

III.4.1 Pandas

Pandas est une bibliothèque open source sous licence qui fournit des structures de données et des outils d'analyse de données haute performance et faciles à utiliser pour le langage de programmation Python [47].

III.4.2 NumPy

NumPy est une bibliothèque dédiée au calcul scientifique en Python, offrant des objets de tableau multidimensionnel et diverses routines pour effectuer rapidement une gamme d'opérations sur ces tableaux. Ces opérations comprennent des fonctions mathématiques, de manipulation de formes, d'algèbre linéaire, de statistiques, de simulation aléatoire, et bien d'autres encore[48].

III.4.3 Matplotlib

Matplotlib est une bibliothèque de traçage complète pour créer des visualisations statiques, animées et interactives en Python. Elle a beaucoup d'extensions comme l'extension de mathématiques numériques NumPy[49].

III.4.4 OpenCV

OpenCV a été lancé chez Intel en 1999 par Gary Bradsky.La première version est sortie en 2000, il a été plus tard soutenu par Willow Garage puis Itseez. OpenCV-Python est une bibliothèque de liaisons Python destinée à résoudre les problèmes de vision par ordinateur en temps réel[50].

III.4.5 TensorFlow

TensorFlow est une plateforme Open Source de bout en bout dédiée aux machines learning, crée et maintenue par l'équipe Google Brain au sein de la machine de Google, une organisation de recherche en intelligence artificielle. TensorFlow est publié sous la licence open source Apache 2.0 [51].

III.4.6 Keras

Keras est une API d'apprentissage profond écrite en Python, fonctionnant sur la plateforme d'apprentissage automatique TensorFlow. Le but de cette bibliothèque est de permettre la conception rapide de réseaux de neurones[52].

III.5 Architecture du système proposée

Dans cette section nous présentons l'architecture du système (Figure III.2). En tant que pilier essentiel de ce projet, cette architecture définit la structure, les composants et les interactions nécessaires pour atteindre les objectifs de notre projet de manière efficace et fiable. Dans cette introduction, nous allons présenter les principes directeurs de cette architecture.

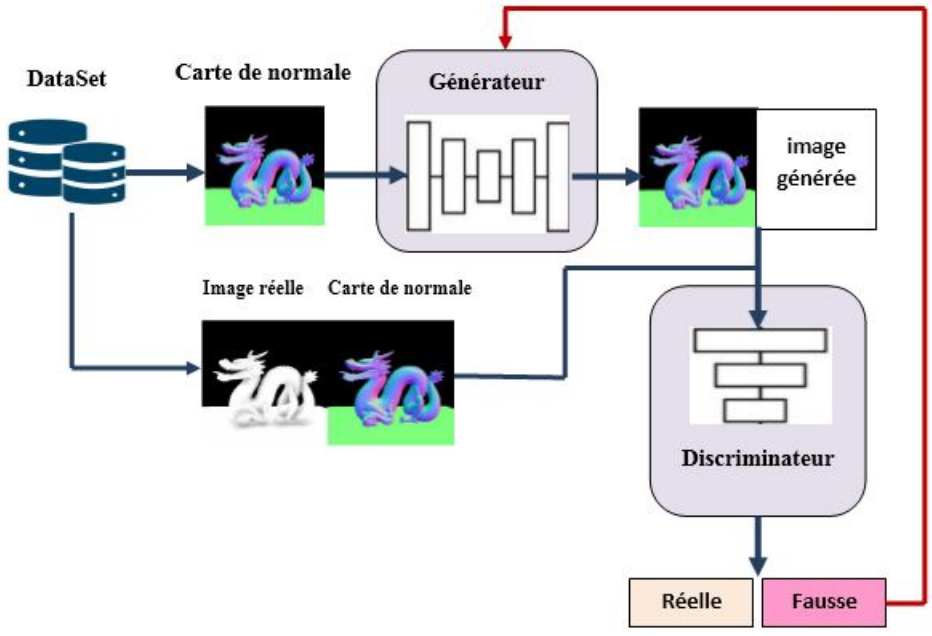

FIGURE III.2 – Architecture du système

III.5.1 Description

Le système proposé est basé sur une architecture GAN conditionnelle, où la génération de l'image de sortie dépend d'une image d'entrée donnée. Il comprend deux réseaux : un générateur et un discriminateur. L'utilisation de la conditionnalité dans Pix2Pix signifie que le générateur est conditionné à l'image d'entrée, ce qui lui permet de produire des résultats plus précis et significatifs.

Le processus implique de constituer un grand jeu de données d'entraînement composé de paires d'images : une image source (rendu 3D d'une scène sans calcul d'occlusion ambiante) et une image cible (rendu 3D de la même scène avec calcul d'occlusion ambiante). Ensuite, le modèle Pix2Pix est entraîné sur ces paires d'images source-cible afin d'apprendre la fonction de transfert permettant d'ajouter l'ombrage d'occlusion ambiante.

En inférence, lorsqu'une nouvelle image sans occlusion ambiante est donnée à l'encodeur de Pix2Pix, le décodeur produit une nouvelle image avec l'effet d'occlusion ambiante estimé par le modèle.

Le Générateur : est basé sur une architecture U-NET

L'entrée :

- Importer une image.
- Et générer une image similaire a l'image réelle.

La sortie :

- Prend une image presque similaire a l'image réelle.

Le Discriminateur : est basé sur une architecture PatchGAN

L'entrée (1) :

L'image normal + l'image générer par le générateur.

L'entrée (2) :

L'image réelle + l'image normal.

Le discriminateur tente de distinguer entre les vraies paires (image d'entrée et image réelle correspondante) et les fausses paires (image d'entrée et l'image générée par le générateur). Il s'agit également d'un réseau de neurones convolutifs, souvent entraîné simultanément avec le générateur dans un cadre d'apprentissage adversarial pour améliorer la qualité des images générées.

III.6 L'ensemble de données utlilisées

Il s'agit d'une base de données qui contient 10 modèles, chacun comprenant 1000 images, soit un total de 10 000 images. Les images sont classées dans 10 dossiers différents :

Dossier 1 : Dragon ,1000 images.

Dossier 2 : Ange ,1000 images.

Dossier 3 : Lucy0 ,1000 images.

Dossier 4 : LD-horseRtime,1000 images.

Dossier 5 : Chaval,1000 images.

Dossier 6 : Chaise ,1000 images.

Dossier 7 : Arbre,1000 images.

Dossier 8 : Bunny-in-scène,1000 images.

Dossier 9 : Buddha-in-scène,1000 images.

Dossier10 : Camel2,1000 images.

Les images du jeu de données sont générées en utilisant la méthode SSAOBMISRTR [53]. En total on à 10 000 images. Nous l'avons réparti en 8000 images (80%) pour l'entraînement, 1000 images (10%) pour la validation ou l'évaluation du modèle, et 1000 images (10%) pour les tests.

III.7 Algorithme pix2pix

1.Importer les bibliothèques nécessaires.

2.Début

3.def load (image_file) : (fonction charge une paire d'images à partir d'un fichier).

4.Dataset : Importer une paire d'images.

5.Visualisation : visualise les images chargées 'x' et 'y'.

6.Fonction normalize : [def normalize(input_image, real_image)].

7.Fonction resize : [def resize(input_image, real_image)].

8.Fonction_random_jitter : [def random_jitter(input_image, real_image)].

9.Chargement d'images pour l'entraînement :[def load_train_images (image_path)].

10.Chargement d'images pour test :[def load_test_image(image_path)].

11.Implémentation d'un générateur U-NET : [def generator()].

12.Fonction de perte de générateur : [loss_function = MeanSquaredError()].

13.Définit le discriminateur patchGAN : [def discriminator()].

14.Fonction 'save_images' :[def save_images(model, test_input, target, epoch)].

15.Entraînement le modèle : [def train_step(input_image, target, epoch)].

16.Évaluer le modèle : [def fit(train_ds, epochs, test_ds)].

17.Fin

1.Importer les bibliothèques nécessaires

```
import tensorflow as tf
import os
import numpy as np
import matplotlib.pyplot as plt
import time

from keras.models import Model, Sequential
from keras.layers import Dense, Conv2D, Flatten, BatchNormalization, LeakyReLU
from keras.layers import Conv2DTranspose, Dropout, ReLU, Input, Concatenate, ZeroPadding2D
from keras.optimizers import Adam
from keras.utils import plot_model
from keras.callbacks import ModelCheckpoint
from skimage.metrics import structural_similarity as ssim
from keras.losses import MeanSquaredError
from tensorflow.keras.layers import Input, Conv2DTranspose, Concatenate, UpSampling2D
```

2. Fonction charge une paire d'images

```
def load(image_file):
    image = tf.io.read_file(image_file)
    image = tf.image.decode_jpeg(image, channels=3)
    w = tf.shape(image)[1]
    w = w // 2
    real_image = image[:, :w, :]
    input_image = image[:, w:, :]

    input_image = tf.cast(input_image, tf.float32)
    real_image = tf.cast(real_image, tf.float32)
    return input_image, real_image
```

Cette fonction charge une paire d'images, les divise en deux parties égales et les convertit en tenseurs flottants avant de les retourner. **3.Fonction load dataset**

Charger une paire d'images à partir d'un fichier spécifique et inspecter les dimensions des tenseurs résultants, alors la création de dataset pour générer une carte des normales et algo-

rithme SSAO[53].

4.Visualisation les images

```
# visualize
fig, axes = plt.subplots(1,2, figsize = (16,5))
axes[0].imshow(x/255.0)
axes[1].imshow(y/255.0)
```

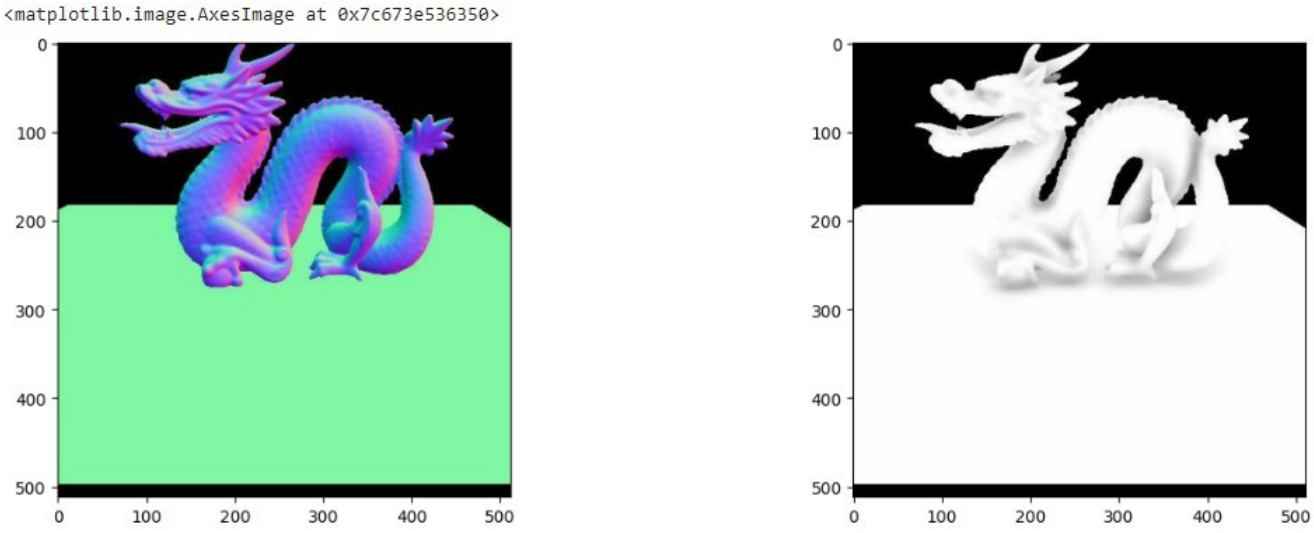

FIGURE III.3 – visualisation du modèle dragon

Dans la Figure III.3 ce code visualise les images chargées ‘x‘ et ‘y‘ à l’aide de matplotlib, utilise matplotlib pour afficher côte à côte les images d’entrée (‘x‘) et les images de sortie réelle (‘y‘) après les avoir normalisées pour les visualiser correctement.

5. Fonction normalize La fonction appelée ”normalize” qui normalise les valeurs des pixels

```
def normalize(input_image, real_image):
    input_image = (input_image / 127.5) - 1
    real_image = (real_image / 127.5) - 1
    return input_image, real_image
```

dans les images d'entrée et réelles. La normalisation est effectuée en soustrayant 1 et en divisant chaque valeur de pixel par 127.5, ce qui permet de ramener les valeurs dans une plage comprise entre -1 et 1.

6. Fonction resize

```
def resize(input_image, real_image):
    input_image = tf.image.resize(input_image, [IMAGE_SIZEX, IMAGE_SIZEY], method=tf.image.ResizeMethod.NEAREST_NEIGHBOR)
    real_image = tf.image.resize(real_image, [IMAGE_SIZEX, IMAGE_SIZEY], method=tf.image.ResizeMethod.NEAREST_NEIGHBOR)
    return input_image, real_image
```

La fonction nommée ”resize” qui redimensionne les images d'entrée et réelles à une taille spécifiée. Les images sont redimensionnées en utilisant la méthode du plus proche voisin, ce qui signifie que la valeur de chaque pixel dans l'image redimensionnée est déterminée en fonction de la valeur du pixel le plus proche dans l'image d'origine.

7. Fonction random_jitter

```
def random_jitter(input_image, real_image):
    #if tf.random.uniform(()) > 0.5:
    input_image = tf.image.flip_left_right(input_image)
    real_image = tf.image.flip_left_right(real_image)
    return input_image, real_image
```

```
x_jit, y_jit = random_jitter(x, y)
fig, axes = plt.subplots(1,2, figsize = (16,5))
axes[0].imshow(x_jit/255.0)
axes[1].imshow(y_jit/255.0)
```

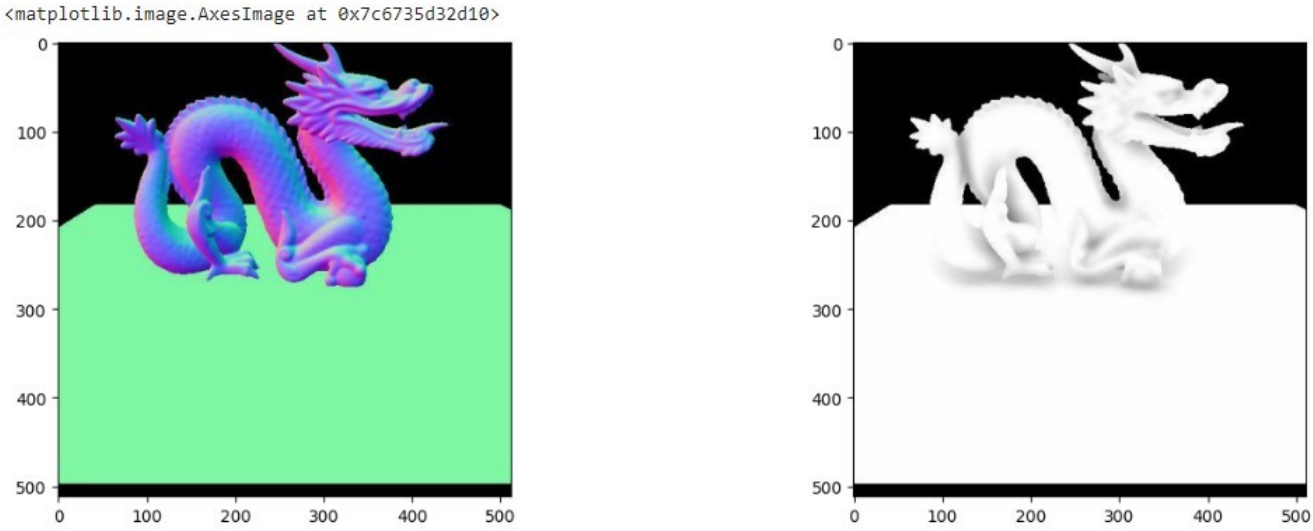

FIGURE III.4 – Visualisation de fonction random_jitter

Dans la Figure III.4 la fonction nommée "random_jitter" qui applique une augmentation de données aléatoire aux images d'entrée et réelles. Dans ce cas, l'augmentation consiste à retourner horizontalement les images avec une probabilité de 50 %. Cela signifie que chaque image a une chance égale d'être retournée horizontalement.

8. Fonction load_train_images

```
def load_train_images(image_path):
    input_image, real_image = load(image_path)
    input_image, real_image = resize(input_image, real_image)
    input_image, real_image = random_jitter(input_image, real_image)
    input_image, real_image = normalize(input_image, real_image)
    return input_image,real_image
```

La fonction nommée "load_train_images" qui charge, redimensionne, applique une augmentation de données aléatoire et normalise les images d'entraînement à partir d'un chemin d'accès spécifié. Tout d'abord, il charge les images à partir du chemin d'accès donné, puis les redimensionne à une taille spécifiée. Ensuite, il applique une augmentation de données aléatoire pour introduire de petites variations dans les images. Enfin, il normalise les valeurs des pixels dans les images pour les ramener à une plage spécifique.

9. Fonction load_test_images

```
def load_test_image(image_path):
    input_image, real_image = load(image_path)
    input_image, real_image = resize(input_image, real_image)
    input_image, real_image = normalize(input_image, real_image)
    return input_image, real_image
```

La fonction nommée ”load_test_image” qui charge, redimensionne et normalise une seule image de test à partir d'un chemin d'accès spécifié. Tout d'abord, il charge l'image à partir du chemin d'accès donné. Ensuite, il redimensionne l'image à une taille spécifiée. Enfin, il normalise les valeurs des pixels dans l'image pour les ramener à une plage spécifique.

10. Fonction load_test_images

```
# create input pipeline
#train_dataset = tf.data.Dataset.list_files(path + "/segmatationimage/DataSet/lymphoma/Pix2pix/Trainjpg/*.jpg")
train_dataset = tf.data.Dataset.list_files(path + "/dragon/*.jpg")
train_dataset = train_dataset.map(load_train_images)
train_dataset = train_dataset.shuffle(1).batch(BATCH_SIZE)
train_dataset
```

La création d'un pipeline d'entrée pour un modèle d'apprentissage automatique utilisant TensorFlow, en particulier pour l'entraînement d'un modèle de génération d'images Pix2Pix.

11. Chargement d'images pour test

```
test_dataset = tf.data.Dataset.list_files(path + "/dragon/*.jpg")
test_dataset = test_dataset.map(load_test_image)
test_dataset = test_dataset.batch(BATCH_SIZE)
test_dataset
```

La création d'un pipeline d'entrée pour les données de test d'un modèle d'apprentissage automatique utilisant TensorFlow.

12. Implémentation d'un générateur U-NET

```
# downsample block
def downsample(filters, size, batchnorm = True):
    init = tf.random_normal_initializer(0.,0.02)
    result = Sequential()
    result.add(Conv2D(filters, size, strides = 2, padding = "same", kernel_initializer = init, use_bias = False))
    if batchnorm == True:
        result.add(BatchNormalization())

    result.add(LeakyReLU())
    return result
down_model = downsample(3,4)
down_result = down_model(tf.expand_dims(x, axis = 0))
print(down_result.shape)
```

```
# upsample block
def upsample(filters, size, dropout = False):
    init = tf.random_normal_initializer(0, 0.02)
    result = Sequential()
    result.add(Conv2DTranspose(filters, size, strides = 2, padding = "same", kernel_initializer = init, use_bias = False))
    result.add(BatchNormalization())
    if dropout == True:
        result.add(Dropout(0.5))
    result.add(ReLU())
    return result
up_model = upsample(3,4)
up_result = up_model(down_result)
print(up_result.shape)
```

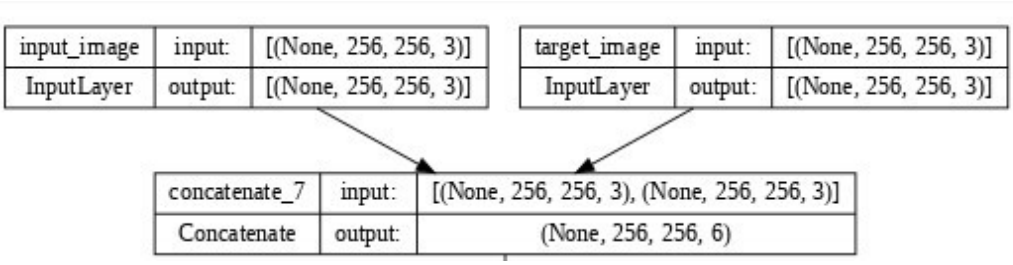

FIGURE III.5 – L'entrée des paires des images

```python
def generator():
    inputs = Input(shape = [IMAGE_SIZEX, IMAGE_SIZEY, 3])
    down_stack = [
        downsample(64, 4, batchnorm=False),
        downsample(128, 4),
        downsample(256, 4),
        downsample(512, 4),
        downsample(512, 4),
        downsample(512, 4),
        downsample(512, 4),
        downsample(512, 4)

    ]


    up_stack = [
        upsample(512, 4, dropout=True),
        upsample(512, 4, dropout=True),
        upsample(512, 4),
        upsample(512, 4),
        upsample(256, 4),
        upsample(128, 4),
        upsample(64, 4),
    ]
    init = tf.random_normal_initializer(0., 0.02)
    last = Conv2DTranspose(3, 4, strides = 2, padding = "same", kernel_initializer = init, activation ="tanh")
    x = inputs
    skips = []
    for down in down_stack:
        x = down(x)
```

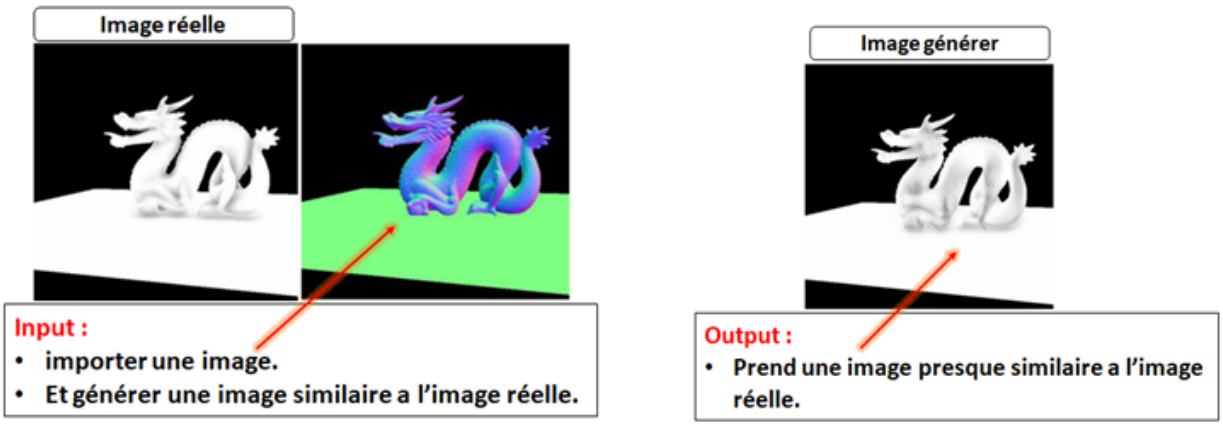

FIGURE III.6 – L'entrée et sortie de générateur

Générateur : est basé sur une architecture U-NET.

Down-sampling(64,4) : nombre des filtres et le size de filtre.

1) importer une image dans input.

2) appelle la fonction de down-sampling.

3) appliquer ensemble des filtres (conv 2d).

Up- sampling(64,4) : nombre des filtres et le size de filtre.

4) appelle la fonction up-sampling.

5) appliquer ensemble des filtres (conv 2d).

13. Fonction de perte de générateur

```
from keras.losses import BinaryCrossentropy
loss_function = BinaryCrossentropy(from_logits=True)
```

```
def generator_loss(disc_generated_output, gen_output, target):
    gan_loss = loss_function(tf.ones_like(disc_generated_output), disc_generated_output)
    l1_loss = tf.reduce_mean(tf.abs(target - gen_output))
    ssim_loss = 1.0 - tf.reduce_mean(tf.image.ssim(target, gen_output, 1.0))
    rmse_loss = tf.sqrt(loss_function(target, gen_output))
    total_gen_loss = gan_loss + (LAMBDA * l1_loss)
    return total_gen_loss, gan_loss, ssim_loss, rmse_loss
```

Définit une fonction de perte personnalisée pour un générateur dans un modèle de génération d'images

- La fonction **'generator_loss'**prend trois arguments :

'disc_generated_output' : Les sorties du discriminateur lorsqu'il est alimenté avec les images générées par le générateur.

'gen_output' : Les images générées par le générateur.

'target' : Les images de référence cibles.

- La fonction calcule plusieurs types de pertes :

'gan_loss' : La perte de type GAN (Generative Adversarial Network), calculée en comparant les sorties du discriminateur avec un tableau de 1s (car le générateur cherche à tromper le discriminateur).

'l1_loss' : La perte L1, calculée comme la moyenne de la valeur absolue de la différence entre les images générées et les images cibles.

'ssim_loss' : La perte SSIM (Structural Similarity Index Measure), calculée comme la différence entre 1 et la moyenne du SSIM entre les images générées et les images cibles.

'rmse_loss' : La perte RMSE (Root Mean Squared Error), calculée comme la racine carrée de la perte quadratique moyenne entre les images générées et les images cibles.

- La perte totale du générateur (**'total_gen_loss'**) est la somme pondérée de la perte GAN

et de la perte L1, où le poids de la perte L1 est déterminé par un coefficient **'LAMBDA'**.

14. Définit le discriminateur patchGAN

```
def discriminator():
    init = tf.random_normal_initializer(0., 0.02)

    inp = Input(shape = [IMAGE_SIZEX, IMAGE_SIZEY, 3], name = "input_image")
    tar = Input(shape = [IMAGE_SIZEX, IMAGE_SIZEY, 3], name = "target_image")
    x = Concatenate()([inp, tar])
    down1 = downsample(64,4,False)(x)
    down2 = downsample(128, 4)(down1)
    down3 = downsample(256, 4)(down2)

    zero_pad1 = ZeroPadding2D()(down3)
    conv = Conv2D(256, 4, strides = 1, kernel_initializer = init, use_bias = False)(zero_pad1)
    leaky_relu = LeakyReLU()(conv)
    zero_pad2 = ZeroPadding2D()(leaky_relu)
    last = Conv2D(1, 4, strides = 1, kernel_initializer=init)(zero_pad2)
    return Model(inputs = [inp, tar], outputs = last)
```

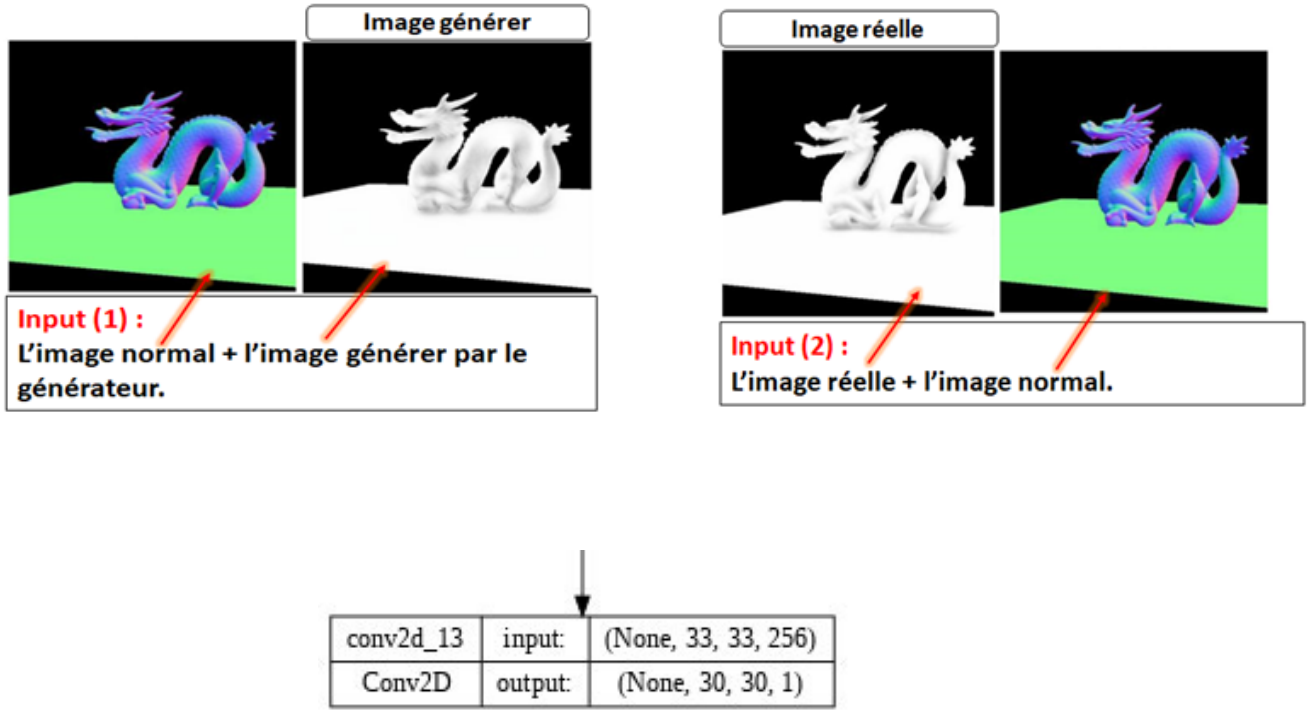

FIGURE III.7 – l'image générée

```
def discriminator_loss(disc_real_output, disc_generated_output):
    real_loss = loss_function(tf.ones_like(disc_real_output), disc_real_output)
    generated_loss = loss_function(tf.zeros_like(disc_generated_output), disc_generated_output)
    total_disc_loss = real_loss + generated_loss
    return total_disc_loss
```

Définit une architecture de modèle de discriminateur pour un modèle de génération d'images, ainsi que les fonctions de perte associées et les optimiseurs pour l'entraînement :

- Définition du modèle de discriminateur (**'discriminator'**).

- Fonction de perte du discriminateur (**'discriminator_loss'**).

*** Optimiseurs pour le générateur et le discriminateur :**

```
generator_optimizer = Adam(lr= 1e-4, beta_1=0.4)
discriminator_optimizer = Adam(lr = 1e-5, beta_1=0.5)
```

- Deux optimiseurs sont définis : un pour le générateur (**'generator_optimizer'**) et un pour le discriminateur (**'discriminator_optimizer'**).

- Ces optimiseurs sont spécifiés avec leurs taux d'apprentissage (**'lr'**) et leur hyperparamètre **'beta_1'**

15. Fonction sauvegarder les images

```
def save_images(model, test_input, target, epoch):
    prediction = model(test_input, training= True)
    plt.figure(figsize = (15,15))
    display_list= [test_input[0], target[0], prediction[0]]
    title = ["Input Image", "Ground Truth", "Predicton Image"]
    for i in range(3):
        plt.subplot(1, 3, i+1)
        plt.title(title[i])
        plt.imshow(display_list[i] * 0.5 + 0.5)
        plt.axis("off")
    plt.savefig(f"output/epoch_{epoch}.jpg")
    plt.close()
```

La fonction nommée "save_images" qui prend en entrée un modèle, une image d'entrée de test, une image cible et un numéro d'époque. Cette fonction utilise le modèle pour générer une prédiction basée sur l'image d'entrée de test. Ensuite, elle crée une figure avec trois sous-graphiques : l'image d'entrée de test, l'image cible (la vérité terrain) et l'image prédite.

16. Entraînement du modèle

```python
@tf.function
def train_step(input_image, target, epoch):
    with tf.GradientTape() as gen_tape, tf.GradientTape() as disc_tape:
        gen_output = gen(input_image, training = True)

        disc_real_output = disc([input_image, target], training = True)
        disc_generated_output = disc([input_image, gen_output], training = True)
        gen_total_loss, gen_gan_loss, gen_ssim_loss, gen_rmse_loss  = generator_loss(disc_generated_output, gen_output, target)
        disc_loss = discriminator_loss(disc_real_output, disc_generated_output)
        generator_gradients = gen_tape.gradient(gen_total_loss, gen.trainable_variables)
        discriminator_gradients = disc_tape.gradient(disc_loss, disc.trainable_variables)
        generator_optimizer.apply_gradients(zip(generator_gradients, gen.trainable_variables))
        discriminator_optimizer.apply_gradients(zip(discriminator_gradients, disc.trainable_variables))
        return gen_total_loss, disc_loss, gen_ssim_loss, gen_rmse_loss
```

Cette étape d'entraînement est utilisée pour une itération d'entraînement complète sur un lot de données. Elle calcule les pertes et met à jour les poids du générateur et du discriminateur en fonction de ces pertes, contribuant ainsi à l'entraînement du modèle de génération d'images.

17. Évaluation du modèle

```python
# Fit the model with the ModelCheckpoint callback
def fit(train_ds, epochs, test_ds):
    for epoch in range(epochs):
        start = time.time()
        for input_, target in test_ds.take(1):
            save_images(gen, input_, target, epoch)

        # Train
        print(f"Epoch {epoch}")
        for n, (input_, target) in train_ds.enumerate():
            gen_loss, disc_loss, gen_ssim, gen_rmse = train_step(input_, target, epoch)

        print("Generator loss {:.2f} Discriminator loss {:.2f} SSIM loss {:.2f} RMSE loss {:.2f}".format(gen_loss, disc_loss, gen_ssim,gen_rmse))
        print("Time take for epoch {} is {} sec\n".format(epoch + 1, time.time() - start))
```

Cette fonction assure l'entraînement du modèle sur un certain nombre d'époques, en exécutant une étape d'entraînement pour chaque batch dans le dataset d'entraînement à chaque époque. Après chaque époque, le modèle est évalué sur le dataset de test pour surveiller sa performance.

III.8 Discussion et résultats

Dans cette partie, nous présentons les résultats de notre algorithme sur différentes scènes. Nous testons également la validité de la technique proposée et nous discutons enfin des points forts et faibles en utilisant des métriques perceptuelles.

III.8.1 Concept de base

III.8.1.1 Outils et matériels utilisés

Pour évaluer la technique proposée, nous avons utilisé une machine équipée d'un processeur Intel® Core™ i3-3110M CPU @ 2.40GHz, fonctionnant sous le système d'exploitation Microsoft Windows 10 64 bits. Les images du dataset ont été créées à l'aide d'OpenGL 4.5. Nous avons utilisé le service hébergé de notebooks Google Colab et avons employé le GPU T4 pour l'entraînement des 10 modèles.

III.8.1.2 Les paramètres du Pix2Pix

- La taille des images : la taille originale des images est de (256 x 256).

- Epochs : Les époques représentent le nombre de répétitions de l'entraînement. Dans notre programme, le nombre d'époques n'est pas déterminé à l'avance et dépend de la fonction d'arrêt ; cependant, nous utilisons typiquement 100 époques.

- Batch size : Ce paramètre de l'algorithme d'apprentissage représente le nombre d'exemples de formation utilisés pour estimer le gradient d'erreur. Nous utilisons une taille de batch de 1.

- Lambda : Il s'agit d'un facteur de régularisation utilisé pour améliorer l'image générée par le générateur.

- L'optimiseur 'Adam' : Il s'agit de l'un des optimiseurs les plus utilisés en raison de sa rapidité. Il converge rapidement en corrigeant la latence du taux d'apprentissage et le contraste élevé.

- Loss générateur Il existe deux types de pertes pour le générateur :

- **L1 Loss :** Calcule la distance entre l'image produite par le générateur et l'image réelle. Nous devons diminuer sa valeur pour obtenir une image similaire à l'image réelle.
- **L :** Compare les pixels entre l'image produite par le générateur et l'image réelle. La

fonction de perte du générateur est la somme de L1 Loss et L.Loss

Générateur loss = L1 Loss + L

-Loss du Discriminateur : Mesure l'efficacité du discriminateur à différencier les images réelles des images générées. Une faible valeur de perte indique une efficacité accrue dans cette distinction.

-Loss SSIM(Structural Similarity Index) : Il s'agit d'une mesure de qualité où des valeurs plus élevées indiquent une meilleure similarité. La fonction de perte peut devoir être ajustée en conséquence.

- Loss RMSE (Root Mean Square Error) : C'est une mesure importante pour évaluer les performances d'un modèle de prédiction. Des erreurs élevées indiquent une performance inférieure du modèle.

III.8.2 Tests et évaluation

Dans cette section, nous allons présenter et évaluer les résultats de la technique Pix2pix proposée. L'objectif est de démontrer l'efficacité de notre approche en utilisant divers paramètres et métriques de performance. Nous allons d'abord décrire les conditions expérimentales, y compris les configurations matérielles et logicielles utilisées. Ensuite, nous détaillerons les résultats obtenus en appliquant notre technique sur différents ensembles de données. Enfin, nous discuterons de l'évaluation des performances en utilisant des mesures telles que le SSIM, le RMSE. Ces analyses nous permettront de valider la robustesse et l'efficacité de notre méthode.

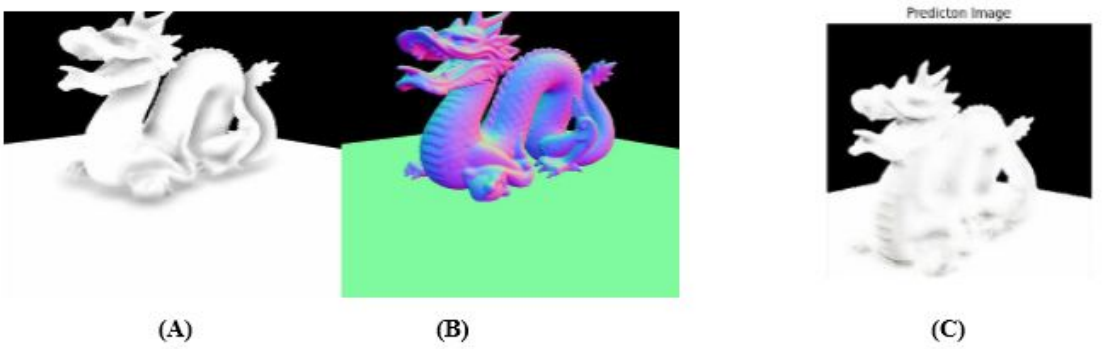

FIGURE III.8 – : SSAO + carte des normale[53]

Dans la figure III.8, l'image (A) représente le SSAO généré selon la méthode SSAOB-

MISRTR[53] et l'image (B) représente la carte des normales générée par la même méthode. En entrée, le générateur reçoit l'image (B) qui est la carte des normales, et génère une nouvelle image pour obtenir un résultat similaire à l'image SSAO (A). En sortie, nous obtenons une nouvelle image (C).

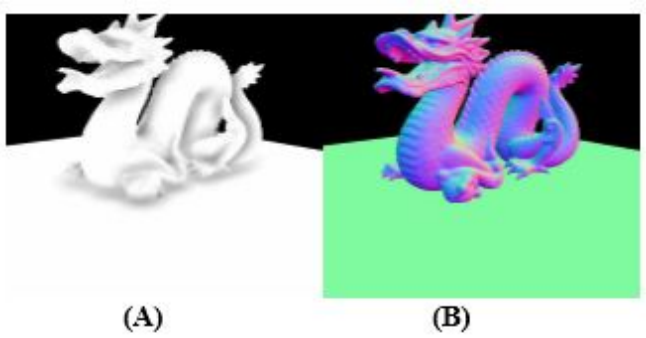

FIGURE III.9 – SSAO + carte des normale[53]

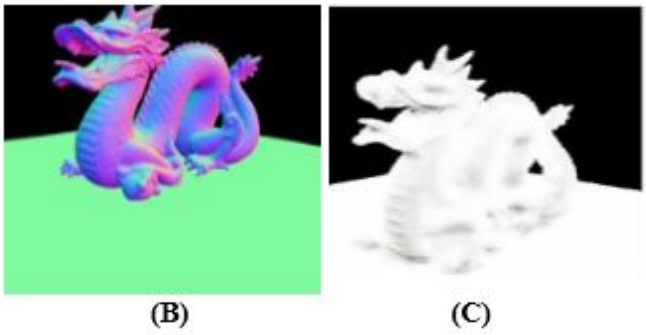

FIGURE III.10 – carte des normale +image générer[53]

Dans la figure III.9 l'image (A) représente le SSAO (images réelles) et l'image (B) représente la carte des normales. Dans la figure III.10 , l'image (B) représente la carte des normales et l'image (C) représente l'image générée par le générateur. En entrée, le discriminateur reçoit deux ensembles de données : celles de la figure III.9 et celles de la figure III.10. Il concatène ces ensembles et essaie de distinguer entre les paires réelles (image d'entrée (B) et image réelle correspondante (A)) et les paires fausses (image d'entrée (B) et image générée (C) par le générateur). Le discriminateur effectue cette distinction en classifiant les images, en divisant chaque image en patches, puis en comparant chaque patch de l'image générée à l'image réelle.

III.8.3 Résultats et comparaison

Epoch	SSIM loss	RMSE loss
9	0,98	0,02
11	0,98	0,04
13	0,98	0,02
14	0,98	0,02
15	0,99	0,03
25	0,98	0,04
26	0,98	0,02
43	0,98	0,03
45	0,99	0,03
55	0,98	0,03
Moyenne	0,98	0,04

TABLE III.1 – Représente la moyenne de SSIM loss et RMSE loss(dragon)

Table III.1 illustre l'évaluation de la technique Pix2pix proposée pour le modèle "dragon" après 100 époques. Nous avons choisi un minimum de 10 époques en fonction de la perte SSIM (une mesure de qualité où des valeurs plus élevées indiquent une meilleure similarité) et de la perte RMSE (une mesure importante pour évaluer les performances, où des valeurs plus faibles indiquent un modèle plus performant). Dans ce modèle, la valeur moyenne de la perte SSIM est de 0,98, ce qui doit être proche de 1, et la valeur moyenne de la perte RMSE est de 0,04, qui doit être proche de 0. Il est essentiel de respecter ces deux critères (SSIM et RMSE) pour obtenir la meilleure simulation possible.

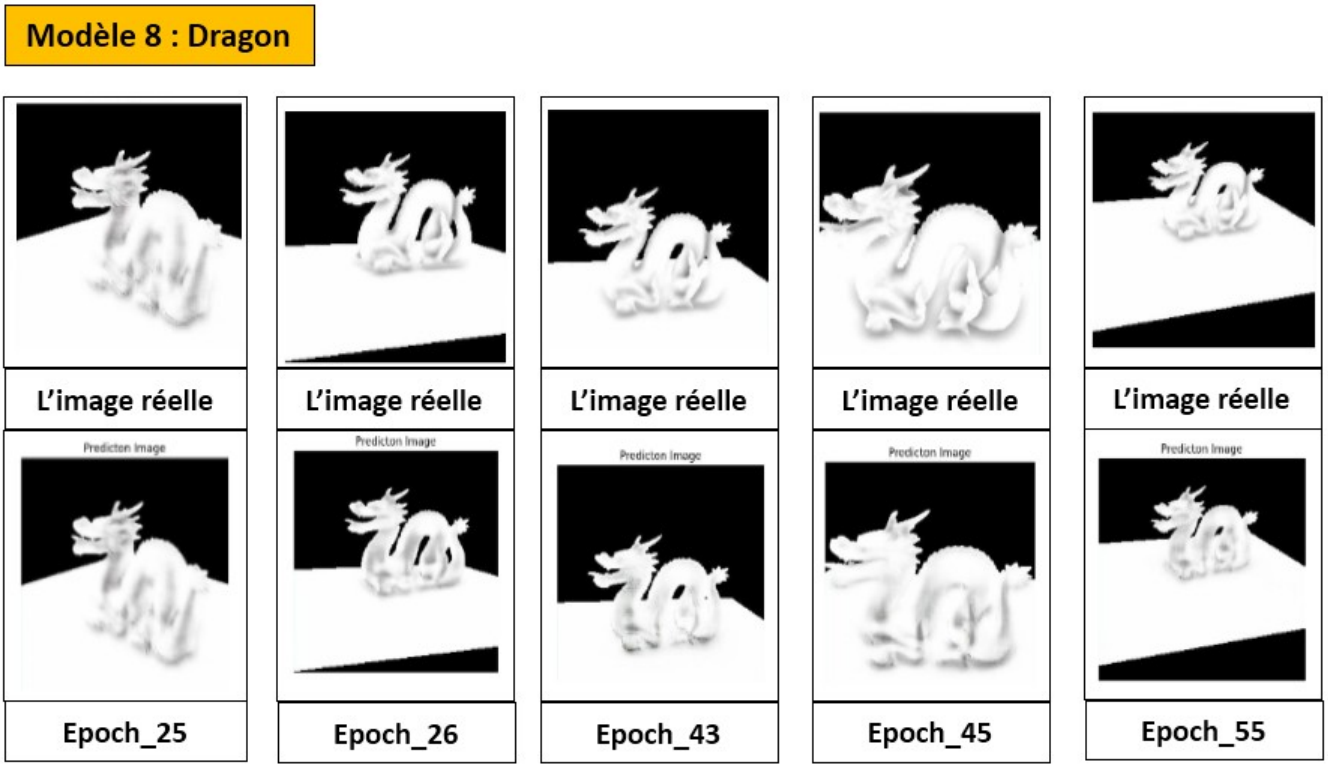

FIGURE III.11 – Comparaison entre SSAO(l'image réelle) et les images cibles

Lors de l'entraînement sur 100 époques selon le modèle "dragon", notre méthode de simulation d'occlusion ambiante utilisant la technique Pix2pix a obtenu d'excellents résultats en très peu de temps, comparée à la méthode SSAOBMISRTR[53]. Cette dernière repose sur le calcul de la carte des normales, de la carte de profondeur, et le calcul du nombre d'intersections avec la géométrie, ce qui est coûteux en termes de ressources. Pour remédier à cela, nous avons adopté une méthode de simulation d'occlusion ambiante basée sur le deep learning (Pix2pix), qui s'est avérée être une approche efficace et peu coûteuse. Ci-dessous, nous présentons les images représentant les résultats de l'entraînement avec cette méthode, ainsi que leur comparaison avec la méthode SSAO, également appelée images réelles avec occlusion ambiante(Figure III.11).

Les mêmes étapes sont appliquée pour les autres modèles, voici les résultats obtenus

Modèle 1 : Ange

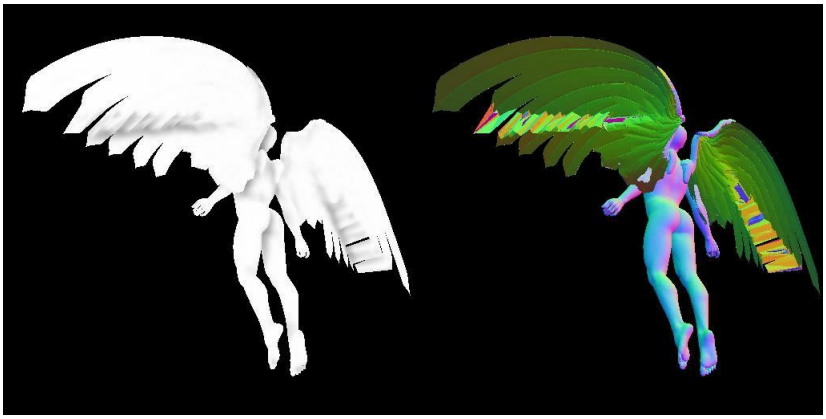

FIGURE III.12 – SSAO et carte des normale[53]

La Figure III.12 représente le SSAO et la carte des normales pour le modèle "ange" après un entraînement de 100 époques avec la technique Pix2pix. Le tableau suivant illustre les 10 meilleures époques et calcule la moyenne des pertes SSIM et RMSE (Table III.2) :

Epoch	SSIM loss	RMSE loss
5	0,96	0,04
7	0,96	0,05
9	0,99	0,04
10	0,99	0,04
17	0,99	0,02
18	0,99	0,03
19	0,96	0,05
20	0,96	0,04
21	0,99	0,04
22	0,98	0,02
Moyenne	0,98	0,03

TABLE III.2 – Représente la moyenne de SSIM loss et RMSE loss(ange)

Nous remarquons que la valeur moyenne de la perte SSIM est de 0,98 ce qui est proche de 1, et que la perte RMSE est de 0,03 ce qui est proche de 0. Grâce à cette technique, nous avons constaté qu'elle donne de bons résultats pour ce modèle (ange). La Figure III.13 présente une série de modèles qui ont atteint les objectifs souhaités après l'entraînement, en comparaison

avec les images réelles :

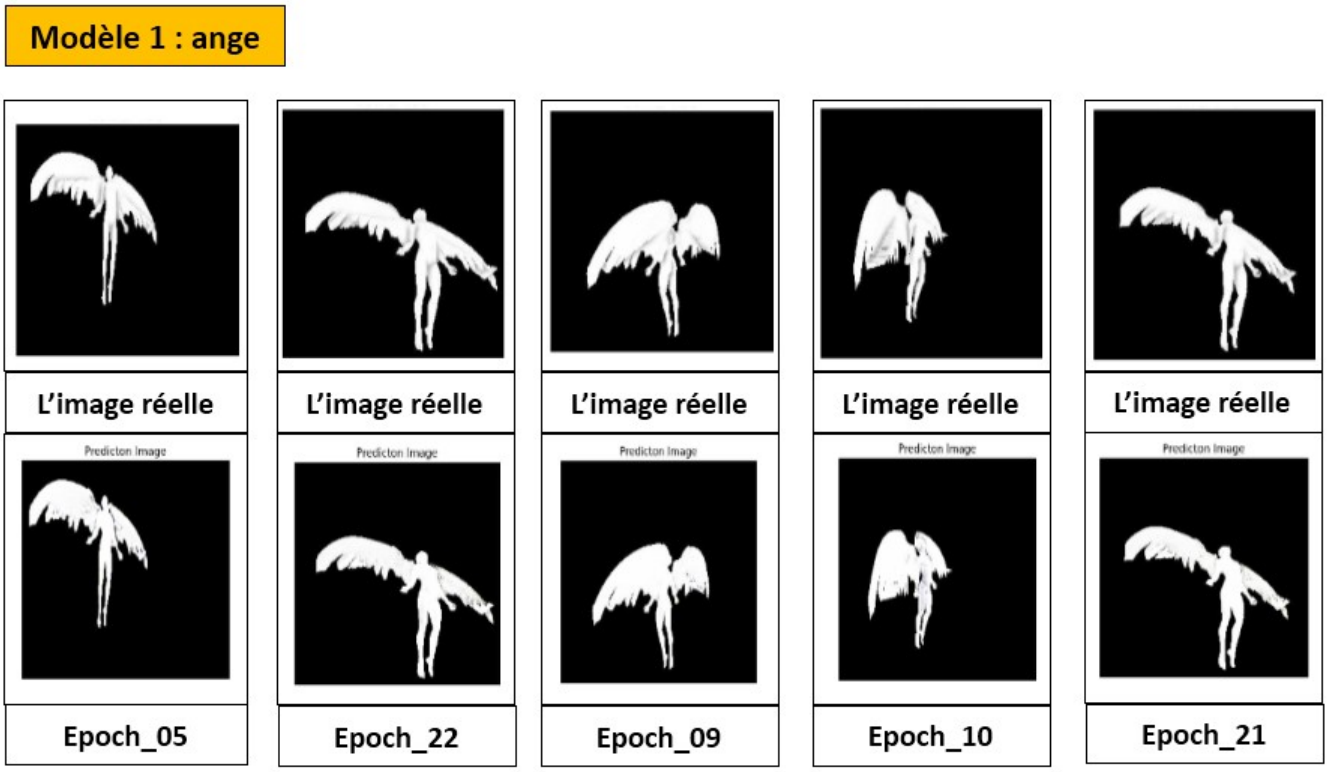

FIGURE III.13 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 2 : Arbre

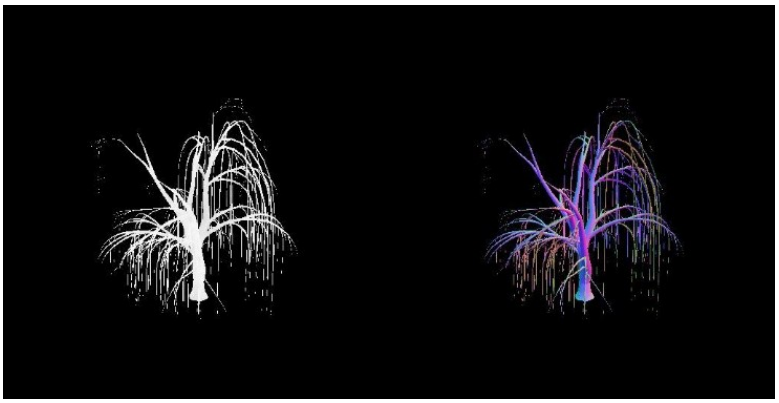

FIGURE III.14 – SSAO et carte des normale[53]

La Figure III.14 représente le SSAO et la carte des normales pour le modèle "arbre" après un entraînement de 100 époques avec la technique Pix2pix. Table III.3 illustre les 10 meilleures époques et calcule la moyenne des pertes SSIM et RMSE :

Epoch	SSIM loss	RMSE loss
33	0,98	0,03
9	0,98	0,03
11	0,98	0,02
26	0,98	0,03
27	0,98	0,03
36	0,97	0,02
37	0,98	0,02
38	0,97	0,02
39	0,96	0,04
40	0,97	0,03
Moyenne	0,98	0,03

TABLE III.3 – Représente la moyenne de SSIM loss et RMSE loss (Arbre)

Avec une moyenne de perte SSIM de 0,98 (proche de 1) et une perte RMSE de 0,03 (proche de 0), nous avons constaté de bons résultats pour ce modèle (arbre) grâce à cette technique. Ci-dessous, nous présentons une série de modèles ayant atteint les objectifs désirés lors de l'entraînement, comparés aux images réelles (Figure III.15) :

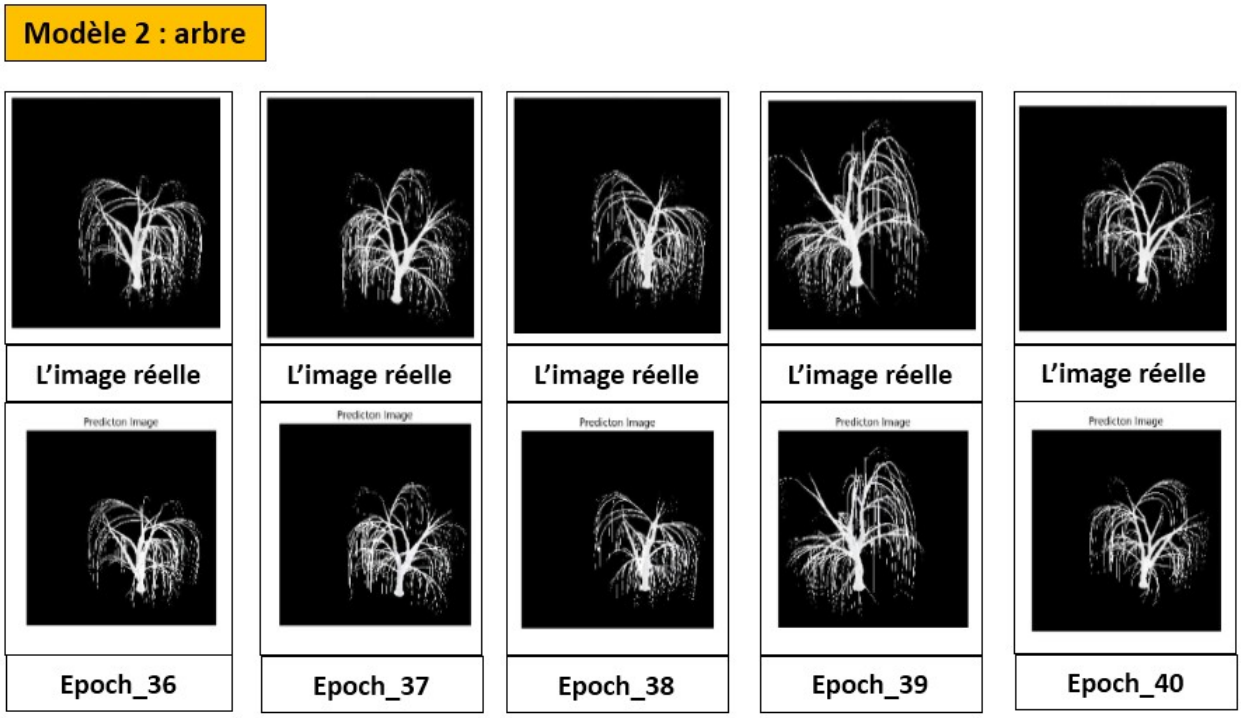

FIGURE III.15 – Comparaison entre SSAO(l'image réelle) et les images cibles

Modèle 3 : buddha_in_scene

FIGURE III.16 – SSAO et carte des normale[53]

La Figure III.16 affiche les résultats du SSAO et de la carte des normales pour le modèle "buddha_in_scene" après un entraînement de 100 époques avec la méthode Pix2pix. Le tableau ci-dessous(Table III.4), met en évidence les 10 meilleures époques et calcule la moyenne des pertes SSIM et RMSE :

Epoch	SSIM loss	RMSE loss
19	0,97	0,04
21	0,98	0,03
22	0,98	0,03
60	0,96	0,05
78	0,97	0,04
85	0,97	0,04
88	0,98	0,03
93	0,98	0,03
97	0,98	0,04
98	0,97	0,04
Moyenne	0,97	0,04

TABLE III.4 – Représente la moyenne de SSIM loss et RMSE loss (buddha_in_scene)

Les métriques obtenues montrent que le modèle a atteint d'excellentes performances. L'indice SSIM, qui mesure la similarité structurelle entre les images générées et les images réelles, est très proche de 1 avec une valeur moyenne de 0,97 .Cela indique une forte ressemblance. De plus, l'erreur quadratique moyenne (RMSE) est très faible à 0,04 ce qui signifie que les images générées sont presque identiques aux images réelles (Figure III.17) :

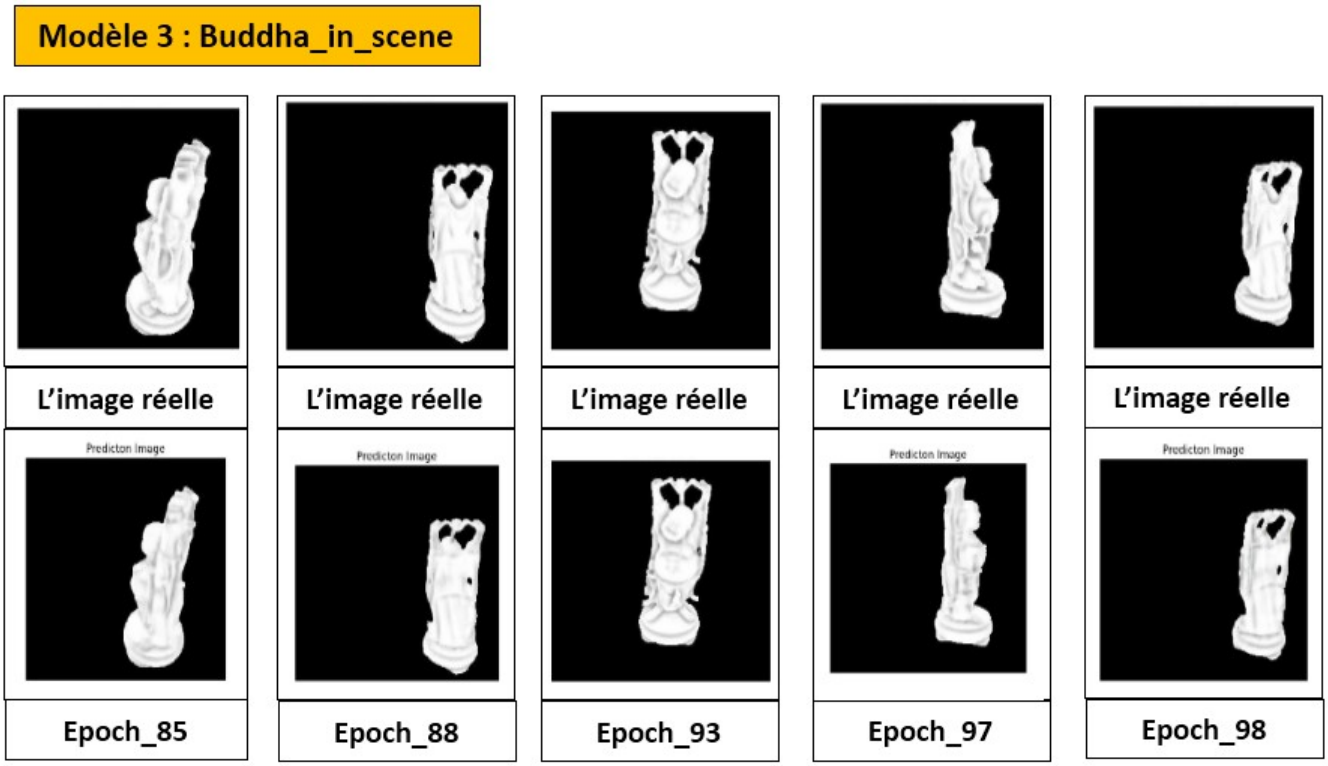

FIGURE III.17 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 4 : bunny_in_scene

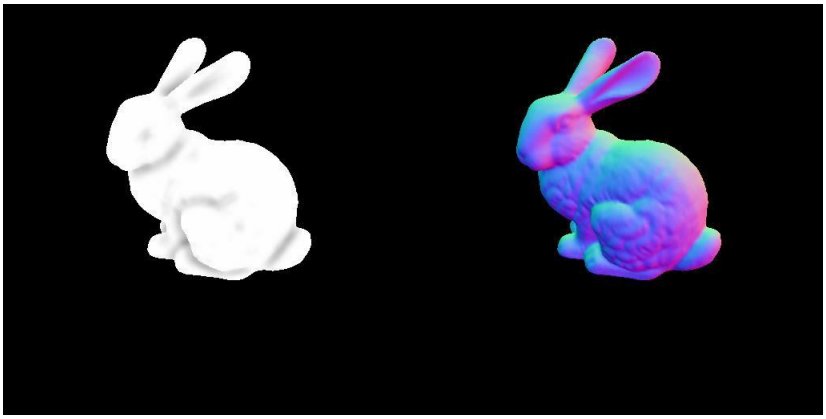

FIGURE III.18 – SSAO et carte des normales[53]

Figure III.18 montre le SSAO et la carte des normales pour le modèle "bunny_in_scene" après un entraînement de 100 époques avec la technique Pix2pix. Le tableau suivant (Table III.5), met en avant les 10 meilleures époques et calcule la moyenne des pertes SSIM et RMSE :

Epoch	SSIM loss	RMSE loss
8	0,98	0,03
9	0,98	0,04
10	0,99	0,03
12	0,98	0,03
14	0,98	0,03
25	0,98	0,03
46	0,98	0,04
74	0,98	0,03
75	0,98	0,04
88	0,97	0,05
Moyenne	0,98	0,03

TABLE III.5 – Représente la moyenne de SSIM loss et RMSE loss (bunny_in_scene)

Nous observons que la valeur moyenne de la perte SSIM = 0,98 est proche de 1 et que la perte RMSE = 0,03 est proche de 0. En utilisant cette méthode, nous avons constaté qu'elle a produit de bons résultats pour ce modèle (bunny_in_scene). Voici plusieurs modèles qui ont atteint les objectifs souhaités après l'entraînement, en comparaison avec les images réelles (Figure III.19) :

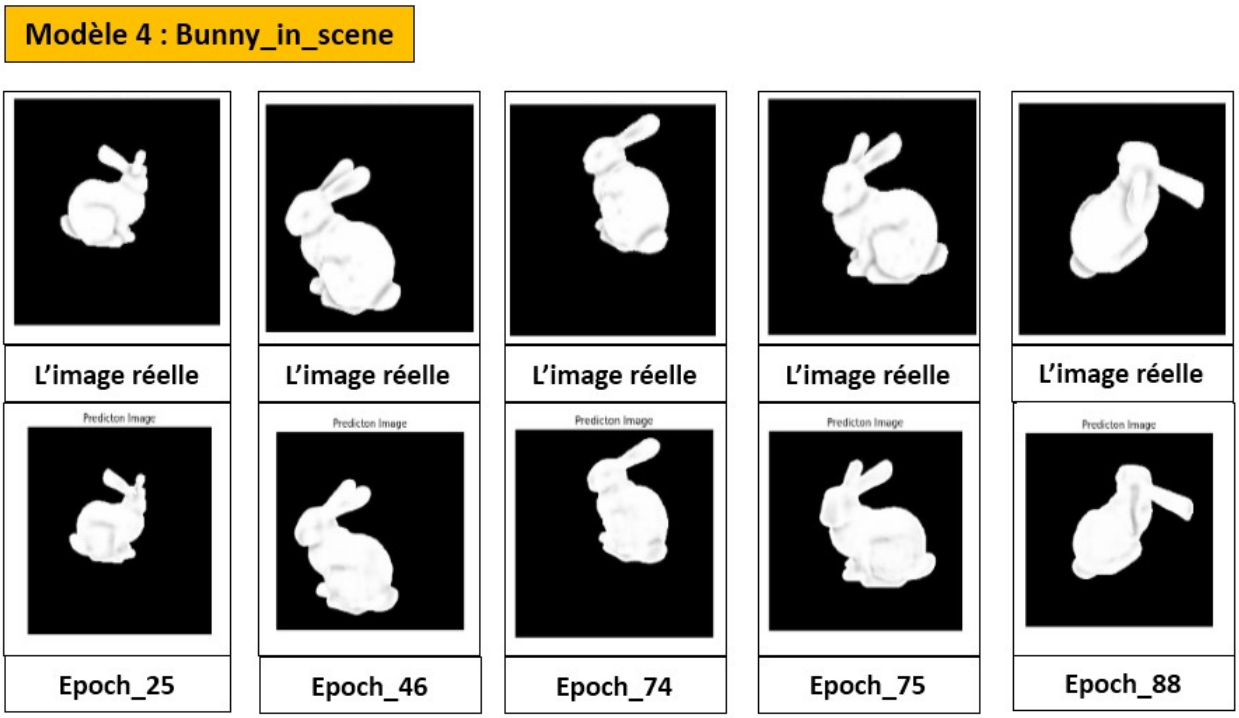

FIGURE III.19 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 5 : Camel2

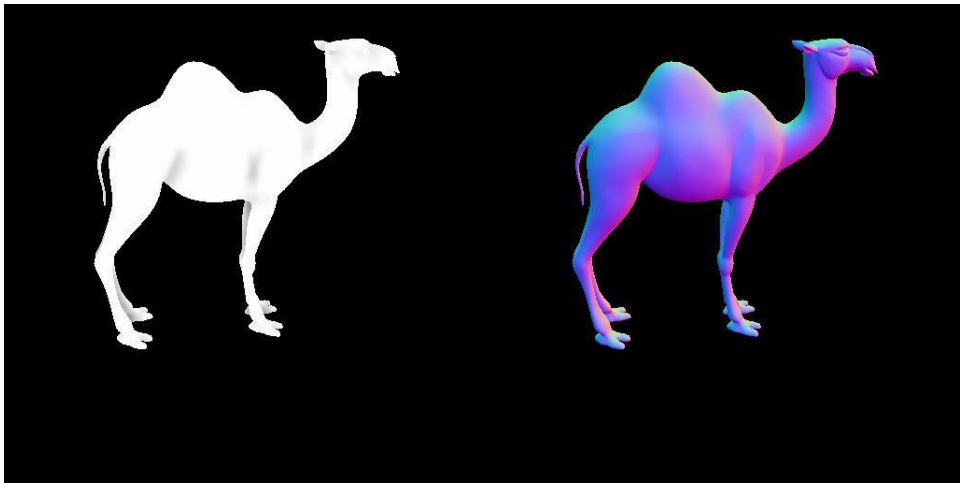

FIGURE III.20 – SSAO et carte des normale[53]

Figure III.20 met en lumière les résultats obtenus après l'application de l'occultation ambiante sur l'échantillon (SSAO) et la génération de la carte des normales pour le modèle "camel2", suite à un entraînement de 100 époques en utilisant la technique pix2pix. Table III.6 ci-dessous expose les performances des 10 meilleures époques en termes de qualité et de fidélité

des résultats.

Epoch	SSIM loss	RMSE loss
12	0,99	0,04
14	0,98	0,04
22	0,98	0,03
24	0,98	0,03
25	0,98	0,03
27	0,98	0,04
29	0,98	0,03
39	0,98	0,04
52	0,98	0,05
55	0,98	0,05
Moyenne	0,98	0,04

TABLE III.6 – Représente la moyenne de SSIM loss et RMSE loss (Camel2)

Dans ce modèle, nous observons que la valeur moyenne de la perte SSIM est de 0,98 ce qui est proche de 1, et que la perte RMSE est de 0,04 ce qui est presque proche de 0. Grâce à cette technique, nous avons constaté que le modèle (camel2) a produit de bons résultats (Figure III.21). Voici une série de modèles qui ont atteint les objectifs souhaités après l'entraînement, en comparaison avec les images réelles :

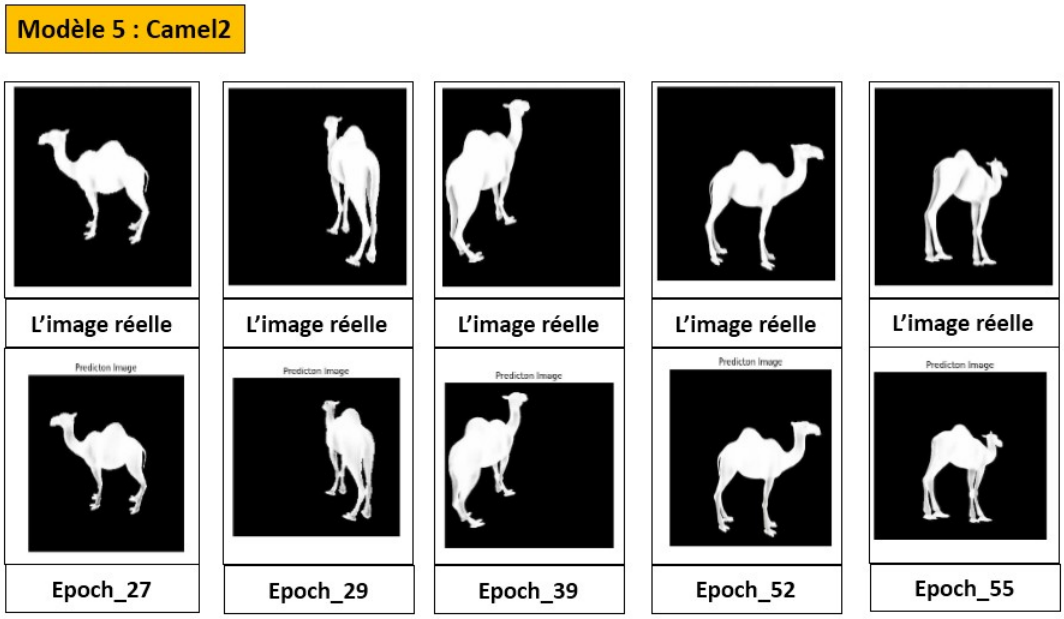

FIGURE III.21 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 6 : Chaise

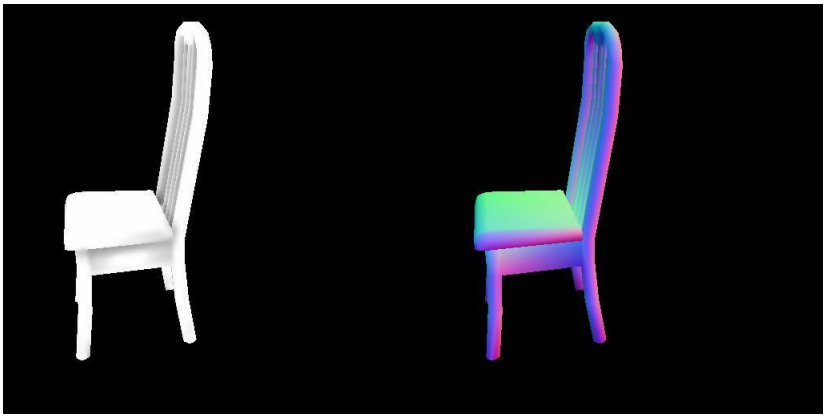

FIGURE III.22 – SSAO et carte des normale[53]

Figure III.22 présente les résultats obtenus pour la SSAO et la génération de la carte des normales pour le modèle de chaise après un entraînement de 100 époques avec la technique pix2pix. De plus, Table III.7 ci-dessous met en évidence les 10 meilleures époques, en calculant la moyenne des pertes SSIM et RMSE pour évaluer la qualité des résultats.

Epoch	SSIM loss	RMSE loss
2	0,98	0,03
3	0,98	0,03
7	0,99	0,04
8	0,98	0,04
9	0,98	0,04
10	0,98	0,03
11	0,99	0,03
21	0,98	0,05
23	0,98	0,04
30	0,98	0,04
Moyenne	0,98	0,04

TABLE III.7 – Représente la moyenne de SSIM loss et RMSE loss (Chaise)

Dans cette étude, nous constatons que la perte moyenne SSIM se situe autour de 0,98 presque atteignant 1, tandis que la perte RMSE est d'environ 0,04 s'approchant de 0. En exploitant cette méthodologie, nous avons observé des performances prometteuses du modèle (chaise).

Plusieurs itérations de modèles ont été entraînées avec succès, atteignant les objectifs définis et offrant des résultats comparables aux images réelles, comme illustré dans la Figure III.23.

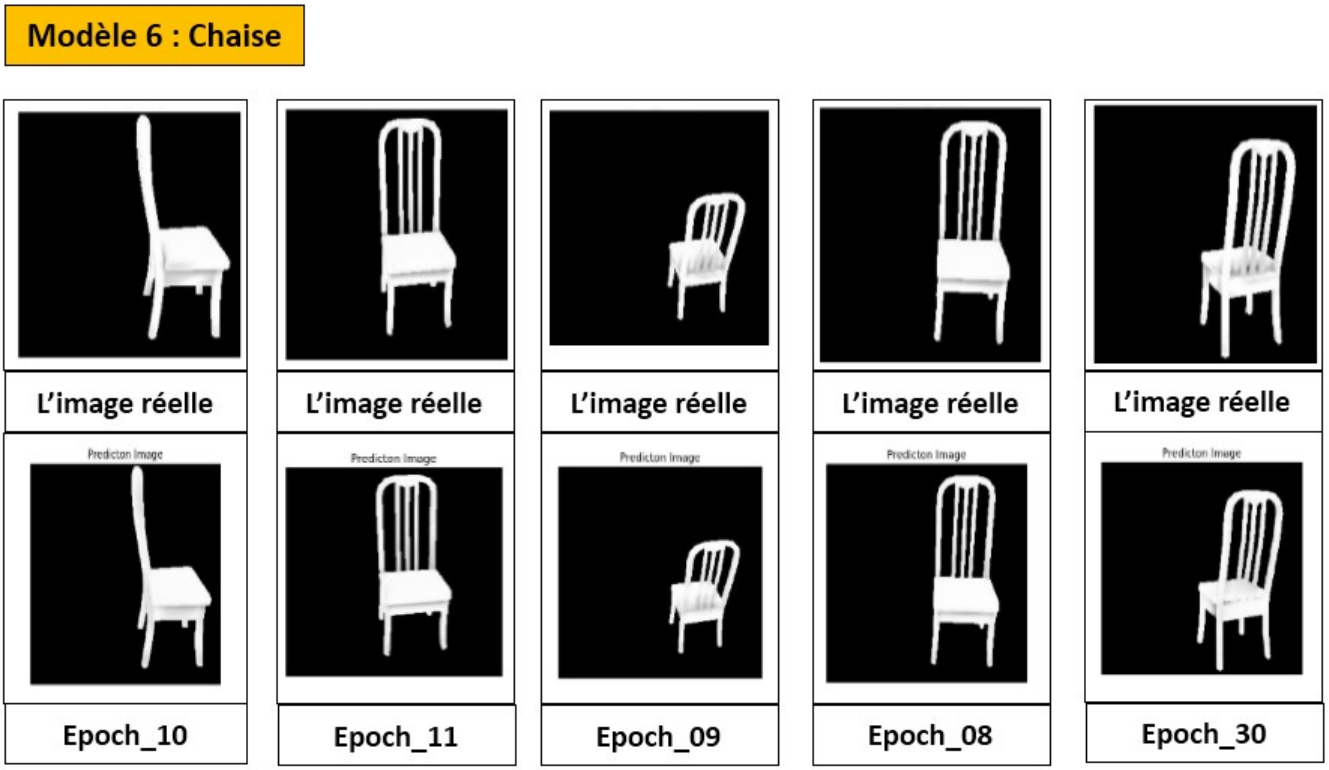

FIGURE III.23 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 7 : Chaval

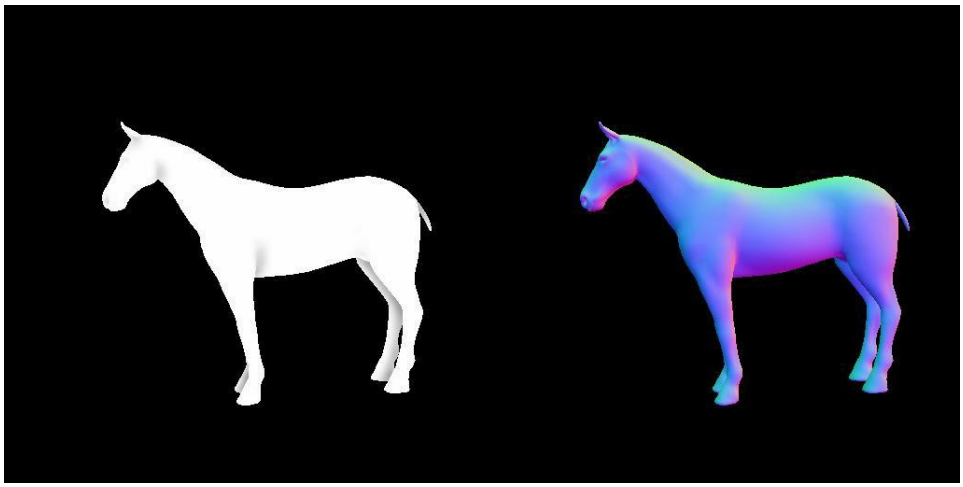

FIGURE III.24 – SSAO et carte des normale[53]

Figure III.24 expose les résultats obtenus suite à l'application de la technique pix2pix sur le modèle "cheval" après un entraînement de 100 époques. Elle offre un aperçu des résultats de la technique SSAO et de la génération de la carte des normales. En outre, Table III.8 présente les 10 meilleures époques, accompagnées du calcul de la moyenne des pertes SSIM et RMSE pour évaluer la qualité des résultats.

Epoch	SSIM loss	RMSE loss
5	0,97	0,09
6	0,96	0,09
7	0,97	0,07
9	0,98	0,07
19	0,98	0,07
20	0,96	0,08
29	0,98	0,08
31	0,97	0,07
33	0,97	0,06
34	0,99	0,04
Moyenne	0,97	0,07

TABLE III.8 – Représente la moyenne de SSIM loss et RMSE loss (Chaval)

Dans ce cas spécifique, la technique pix2pix n'a pas produit des résultats satisfaisants, comme en témoignent les moyennes des deux valeurs. Nous constatons que la perte moyenne SSIM est de 0,97 et la perte RMSE est de 0,07 pour ce modèle "cheval", même après plusieurs cycles d'entraînement, comme illustré dans la Figure III.25.

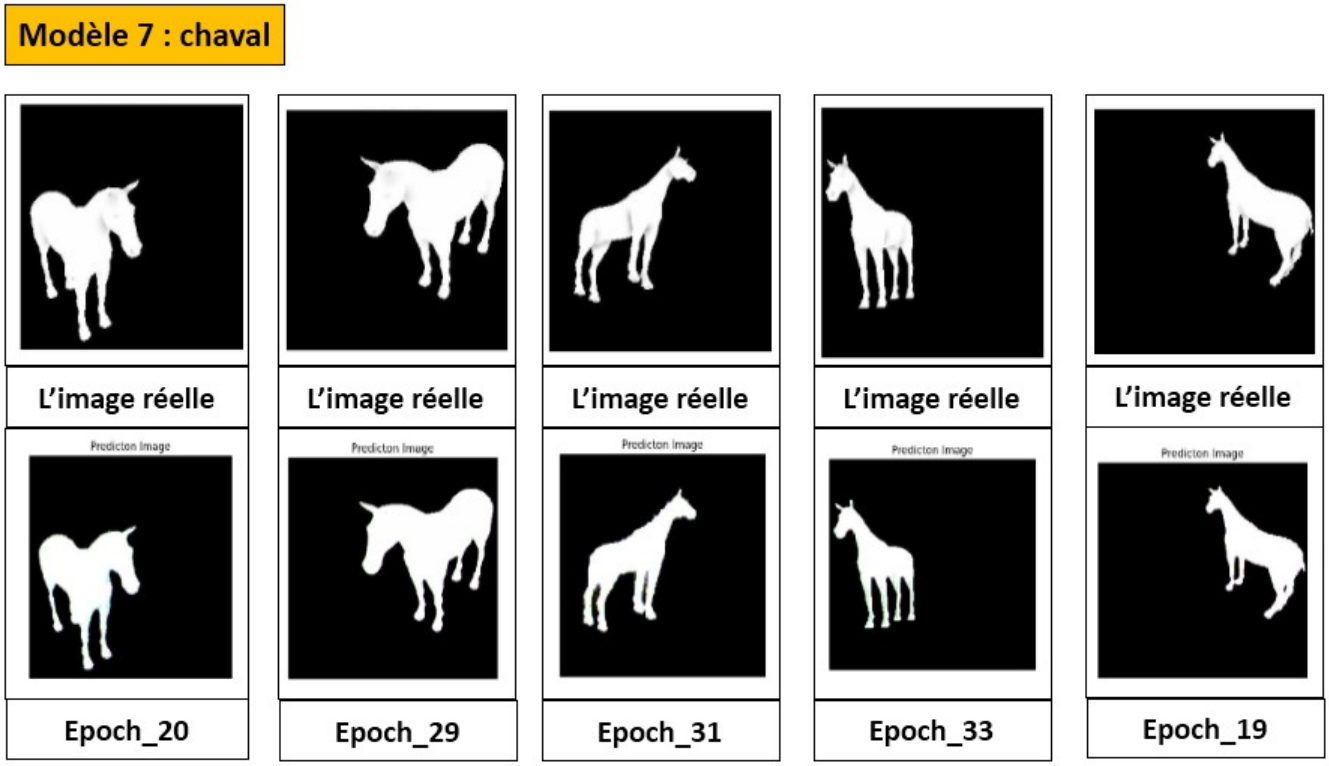

FIGURE III.25 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 9 : LD_HorseRtime

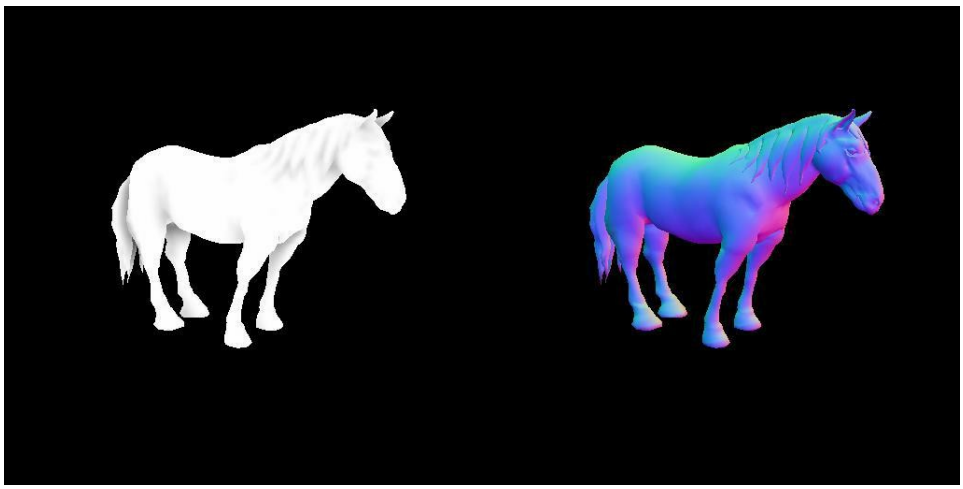

FIGURE III.26 – SSAO et carte des normale[53]

Figure III.26 illustre la représentation d'occlusion ambiante (SSAO) et de la carte des normales pour le modèle "LD_HorseRtime" après un entraînement de 100 époques avec la méthode pix2pix. Le tableau correspondant (Table III.9), récapitule les 10 meilleures époques en fonction de leurs performances respectives.

Epoch	SSIM loss	RMSE loss
12	0,98	0,03
39	0,99	0,03
62	0,98	0,03
63	0,98	0,02
66	0,98	0,03
67	0,99	0,02
80	0,99	0,03
82	0,98	0,03
83	0,99	0,02
90	0,98	0,03
Moyenne	0,98	0,03

TABLE III.9 – Représente la moyenne de SSIM loss et RMSE loss (LD_HorseRtime)

Nous notons que la valeur moyenne de la perte SSIM est de 0,98 se rapprochant de 1, et que la perte RMSE est de 0,03 se situant à proximité de 0. Cette approche a ainsi généré des résultats satisfaisants pour le modèle "LD_HorseRtime". Les objectifs souhaités ont été atteints après l'entraînement, comme (Figure III.27) :

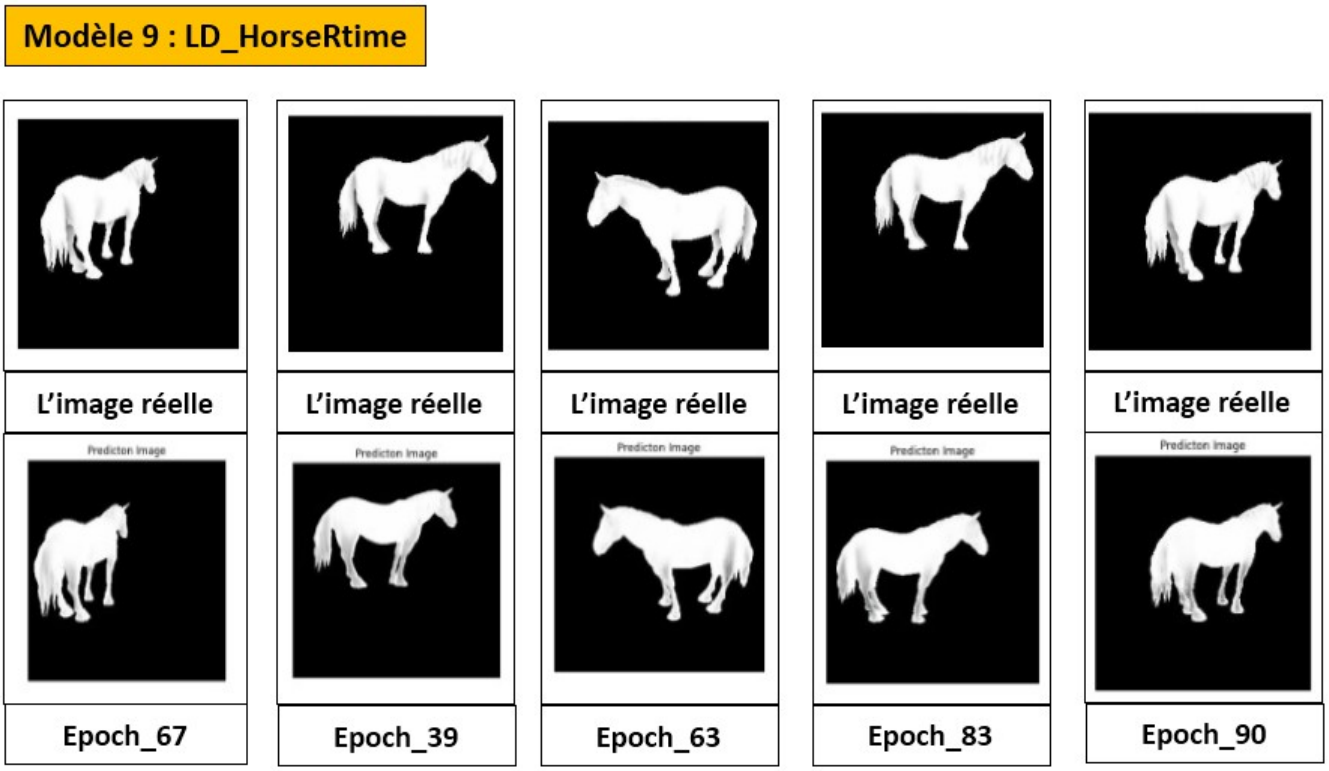

FIGURE III.27 – Comparaison entre SSAO (l'image réelle) et les images cibles

Modèle 10 : Lucy0

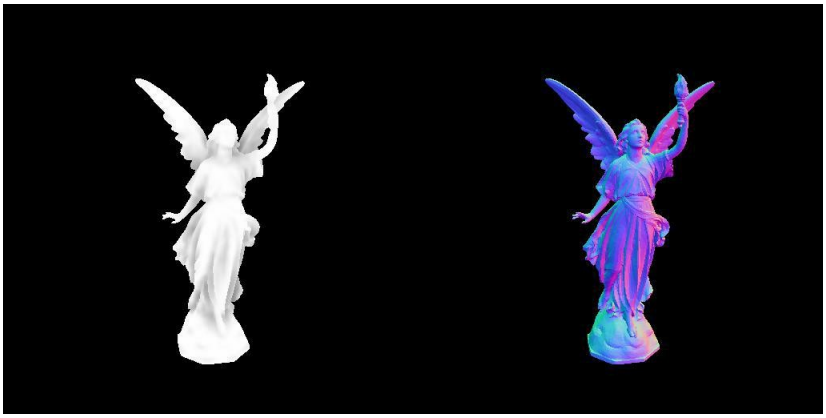

FIGURE III.28 – SSAO et carte des normale[53]

La Figure III.28 présente l'occlusion ambiante (SSAO) et la carte des normales pour le modèle "Lucy0", après un entraînement de 100 époques avec la technique pix2pix. Le tableau associé(Table III.10), met en évidence les 10 meilleures époques et calcule la moyenne des pertes SSIM et RMSE.

Epoch	SSIM loss	RMSE loss
5	0,98	0,03
6	0,98	0,03
7	0,98	0,04
8	0,98	0,03
9	0,98	0,03
53	0,98	0,03
54	0,98	0,03
93	0,98	0,04
96	0,96	0,04
98	0,98	0,03
Moyenne	0,98	0,03

TABLE III.10 – Représente la moyenne de SSIM loss et RMSE loss (Lucy0)

Dans ce modèle, nous notons que la valeur moyenne de la perte SSIM, à 0,98 est proche de 1, tandis que la perte RMSE, à 0,03 est quasiment proche de 0. Cette approche a ainsi produit des résultats satisfaisants pour le modèle "Lucy0". Ci-dessous, une série de modèles ayant

atteint les objectifs fixés après l'entraînement, comparés aux images réelles dans la Figure III.29.

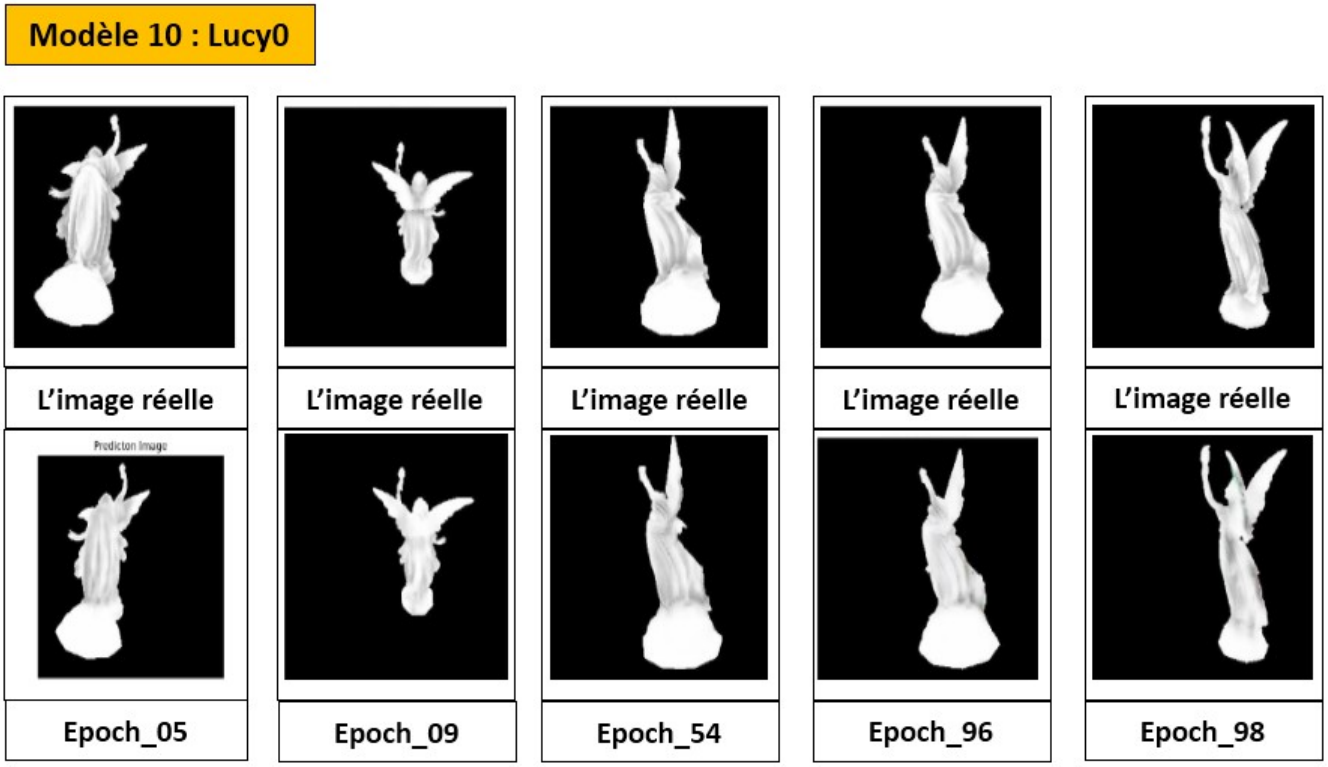

FIGURE III.29 – Comparaison entre SSAO (l'image réelle) et les images cibles

Notre étude montre que l'application de la technique pix2pix pour simuler l'occlusion ambiante a généré des résultats satisfaisants dans la plupart des cas. Cependant, des défis persistent, comme en témoigne le cas du modèle de cheval. Pour aller plus loin, des avenues de recherche intéressantes pourraient inclure l'exploration de variantes de l'architecture pix2pix, l'intégration de données supplémentaires pour enrichir l'apprentissage, ou encore l'utilisation de techniques de régularisation plus avancées pour mieux gérer les cas difficiles.

De plus, il serait bénéfique d'étendre cette étude à un plus large éventail de modèles pour évaluer la robustesse de la technique dans divers contextes. En outre, une analyse plus approfondie des cas où la technique a échoué pourrait fournir des informations précieuses pour identifier les lacunes et guider le développement de solutions plus efficaces.

En fin de compte, bien que des améliorations soient encore nécessaires, les résultats obtenus suggèrent que la technique pix2pix offre un potentiel prometteur pour la simulation d'occlusion ambiante dans divers domaines d'application, de la réalité virtuelle à la modélisation.

III.9 Conclusion

La simulation d'occlusion ambiante par Pix2Pix semble être une approche prometteuse. Cependant, son efficacité réelle dépendra des données d'entrée, de la qualité de l'entraînement du modèle et de sa capacité à générer des résultats réalistes et utiles pour la tâche spécifique à laquelle il est appliqué. Une analyse approfondie des résultats et des ajustements itératifs du modèle peuvent être nécessaires pour parvenir à des performances optimales.

Dans ce chapitre, nous avons exploré la simulation d'occlusion ambiante en utilisant des modèles de type Pix2Pix, qui semble présenter plusieurs résultats intéressants, comme en témoignent les différentes valeurs de métriques générées par les différentes époques. Cependant, pour tirer des conclusions plus précises sur l'efficacité et la performance de cette méthode, il est important d'analyser les résultats obtenus en profondeur.

Conclusion Générale

La simulation d'occlusion ambiante par pix2pix représente une avancée significative dans la modélisation des effets visuels, tant dans les environnements virtuels que réels. Cette étude nous permis de mettre en lumière les capacités et les limitations de cette approche, ainsi que son potentiel pour une variété d'applications.

L'utilisation de pix2pix pour simuler l'occlusion ambiante offre plusieurs avantages. Elle permet tout d'abord une modélisation précise et flexible des effets d'ombre et d'obscurité, éléments essentiels pour de nombreuses applications telles que la réalité virtuelle, la simulation de scènes 3D et la robotique. De plus, le modèle pix2pix est capable de générer des images photo réalistes qui restituent fidèlement les détails et les nuances des environnements simulés.

Cependant, des défis demeurent. L'entraînement d'un modèle pix2pix requiert souvent des ensembles de données volumineux et diversifiés, ainsi qu'une architecture de réseau soigneusement conçue pour obtenir des résultats optimaux. De plus, la qualité des images générées peut varier en fonction de la complexité de l'environnement et de la variabilité des conditions d'éclairage.

Malgré ces défis, notre étude met en évidence le potentiel prometteur de la simulation d'occlusion ambiante par pix2pix. En exploitant les avancées dans le domaine de l'apprentissage automatique et de la vision par ordinateur, nous sommes en mesure de créer des simulations plus réalistes et immersives, ouvrant ainsi des nouvelles perspectives pour une multitude d'applications.

Bibliographie

[1] https://tt-hardware.com/pc/occlusion-ambiante-quest-ce-que-cest/ (accédé le 21 mars 2024)

[2] Louis Bavoil, Miguel Sainz, ≪ Screen Space Ambient Occlusion ≫, septembre 2008.

[3] Louis Bavoil, Miguel Sainz, ≪ Image-Space Horizon-Based Ambient Occlusion ≫, SIG- GRAPH 2008, Talk Program, août 2008.

[4] Alexei Panteleev, ≪ VXAO : Occlusion ambiante du voxel ≫, 2016.

[5] NVIDIA, ≪ Pause-café RTX : Occlusion ambiante par Ray Traced (4 min 17 s) ≫, 13 juillet 2018.

[6] [Bavoil et Sainz, 2009] Bavoil, L. et Sainz, M. (2009). Occlusion ambiante multicouche a double résolution dans l'espace d'écran. Dans SIGGRAPH '09 : SIGGRAPH 2009 : Talks, pages 1-1, New York, NY, États-Unis. ACM.

[7] Zhang, Y. (2010). Application de machine apprentissage. In-TehOlajnica 19/2, 32000 Vukovar, Croatie. ISBN978-953-307-035-3.

[8] Mahesh, B. (2020). Machine apprentissage algorithmes-un revoir. International Journal of Science and Research (IJSR).[Internet] , 9 : 381–386.

[9] Évêque, C. M. et Nasrabadi, N. M. (2006). Reconnaissance de formes et apprentissage automatique , volume 4. Springer.

[10] https://www.data-transitionnumerique.com/machine-learning-python/ (accédé Le 01 Avril 2024)

[11] Radford, A., Metz, L. et Chintala, S. (2015). Sans surveillance représentation appren-tissage avec profond convolutif génératif réseaux contradictoires. arXiv préimpression arXiv :1511.06434.

[12] brightcape.co/apprentissage-supervise-vs-non-supervise/
(accédé Le 12 Avril 2024)

[13] Sutton, RS et Barto, AG (2018). Apprentissage par renforcement : une introduction . Presse du MIT.

[14] S. T. K RISHNA ET H. K. K ALLURI , Profond apprentissage et transfert apprentissage approches pour la classification des images , International Journal of recent Technology and Engineering (IJRTE), 7 (2019), pp. 427-432.

[15] SB M AIND , P. W ANKAR , ET AL ., Document de recherche sur les bases du réseau neuronal artificiel , International Journal on recent and innovation Trends in Computing and Communication, 2 (2014), pp. 96-100.

[16] Hashim Qamar, ≪ Apprentissage Perceptron ≫. Ingénieur en sécurité de l'informa- tion chez Rewterz . (2020). URL : http//www.linkedin.com/pulse/perceptron-learning- hashim-qamar

[17] https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53,
(accédé Le 20 Mars 2024)

[18] J. HUANG, Enhancement and colorization of infrared and other medical images,theseMaster, Department of Electrical and Computer Engineering, Lehigh University, 2010.

[19] Saadi Khaled Iben El Walid et Bendahi Khaled, Segmentation D'Image Médicale Via Non superviser Réseau de neurones convolutif , thèse MASTER ACADEMIQUE, Domaine : Electronique, Option :

Electronique des Systèmes, Faculté Technologie, Département de Electronique, UNIVERSITE MOHAMED BOUDIAF - M'SILA , 2021/2022.

[20] https ://www.hugomichel.io/post/gan/
(accédé le 27 février 2024)

[21] Kramer, ≪ Nonlinear principal component analysis using autoassociative neural networks≫, AIChE Journal, vol. 37, no 2, 1991, p. 233–243.

[22] Hinton et Salakhutdinov, ≪ Reducing the Dimensionality of Data with Neural Networks≫, Science, vol. 313, no 5786, 28 juillet 2006, p. 504–507.

[23] fr.wikipedia.org/wiki/Auto-encodeur-variationnel
(accédé Le 12 Avril 2024).

[24] C. J ANIESCH , P. Z SCHECH , ET K. H EINRICH , Apprentissage automatique etapprentissage profond , Electronic Markets, 31 (2021), pp.

[25] Vint, D., Anderson, M., Yang, Y., Ilioudis, C., Di Caterina, G. et Clémente, C. (2021). Automatique cible reconnaissance pour faible résolution images SAR pénétrant le feuillage à l'aide de CNN et de Gans. Télédétection , 13 : 596.

[26] Y.-J. CAO , LL. JIA , Y.-X. CHEN , N. L DANS , C. YANG , B. Z ACCROCHER, Z. L UI , X.-X. L Je , ET H.-H. D AI , Progrès récents des réseaux contradictoires génératifs en vision par ordinateur , IEEE Access, 7 (2018), pp. 14985-15006.

[27] J. B ROWNLEE , Une douce introduction aux réseaux contradictoires génératifs (gans), récupéré le 17 juin (2019), p. 2019.

[28] typeset.io/questions/what-are-the-advantages-and-disadvantages-of-gan-nr6vo6u6oi
(accédé Le 14 Avril 2024).

[29] Laurence Moroney. IA et apprentissage automatique pour les codeurs. O'Reilly Media, Inc., Octobre. 2020. ISBN : 9781492078197.

[30] H. A LQAHTANI , M. K AVAKLI -T HORNE , ET G. KUMAR , Applications des réseaux adverses génératifs (gans) : une revue mise à jour , Archives of Computational Methods in Engineering, 28 (2021), pp. 525– 552.

[31] S. REED , Z. Un KATA , X. OUI , L. L OGESWARAN , B. SCHIELE , ET H. L EE , Généraux-synthèse contradictoire de texte à image , dans Conférence internationale sur l'apprentissage automatique, PMLR, 2016, pp.

[32] https://pyimagesearch.com/2022/07/27/image-translation-with-pix2pix/
(accédé le 18 mars 2024)

[33] www.researchgate.net/figure/The-architecture-of-Pix2Pix-model-used-in-the-paper-G-is-the-generator-D-is-the-fig1-370889606
(accédé le 25 mars 2024)

[34] ai.plainenglish.io/understanding-pix2pix-gan-e21c2bedd213
(accédé le 19 mars 2024)

[35] J.T. Kajiya, The rendering equation, in : Proceedings of the 13th annual confer- ence on Computer graphics and interactive techniques, 1986, pp. 143–150.

[36] M. Mittring, Finding Next Gen : Cryengine 2, in : ACM SIGGRAPH 2007 courses, 2007,pp. 97–121.

[37] Y. LeCun, et al., Generalization and network design strategies, Connection. Per- spect.19 (143–155) (1989) 18.

[38] H. Zhang, I. Goodfellow, D. Metaxas, A. Odena, Self-attention generative adver- sarial networks, in : International conference on machine learning, PMLR, 2019, pp. 7354–7363.

[39] https ://machinelearningmastery.com/a-gentle-introduction-to-pix2pix-generative-adversarialnetwork/,

Consulter le : 03/05/2024.

[40] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image trans-lation with conditional adversarial networks. In Proceedings of the IEEE conference on computer vision andpattern recognition, pages 1125–1134, 2017.

[41] G.RESEARCH,

https://research.google.com/colaboratory/faq.html : :text=Colabortory

(accéder Le 03 Mai 2024).

[42] TUTORIALSPOINT,

https://www.tutorialspoint.com/googlecolab/whatisgooglecolab.htm ::text

(accéder Le 03 Mai 2024).

[43] MEDIUM,

https://medium.com/deep-learning-turkey/google-colab-freegpu-tutorial-e113627b9f5d

(accẽdé Le 03 Mai 2024).

[44] G.VANROSSUM et F.L.DRAKE, An introduction to Python , Livre Bristol ,2003.

[45] G.VANROSSUMetF.L.DRAKEJR, Python Reference Manual. Python tutorial , Livre , 1995.

[46] PYTHON,
https ://www.python.org./
(accédé Le 03 Mai 2024).
[47] PANDA,
https ://pandas.pydata.org/docs/
(accédé Le 03 Mai 2024).
[48] NUMPY,
https ://numpy.org/doc/stable/
(accédé Le 03 Mai 2024).
[49] MATPLOTLIB,
https ://matplotlib.org/stable/index.html
(accédé Le 03 Mai 2024).
[50] OPENCV,
https://docs.opencv.org/3.4/d0/de3/tutorialpyintro.html
(accédé Le 03 Mai 2024).
[51] TENSORFLOW,
https ://www.tensorflow.org/ ?hl=fr,
(accédé Le 03 Mai 2024).
[52] KERAS,
https ://keras.io/about/
(accédé Le 03 Mai 2024).
[53] Zerari, A.E.M., Babahenini, M.C. Screen Space Ambient Occlusion Based Multiple Importance Sampling for Real-Time Rendering. 3D Res 9, 1 (2018).
https://doi.org/10.1007/s13319-017-0152-9.

[54] F. Gao, Y. Yang, J. Wang, J. Sun, E. Yang et H. Zhou, A deep convolutional generative adversarial networks (DCGANs)-based semi-supervised method for object recognition in synthetic aperture radar (SAR) images," Remote Sensing, t. 10, no 6, p. 846, 2018.

[55] B. Huang, W. Chen, X. Wu, C.-L. Lin et P. N. Suganthan, "High-quality face image generated with conditional boundary equilibrium generative adversarial networks," Pattern Recognition Letters, t. 111, p. 72-79, 2018.

[56] J. Adler et S. Lunz, "Banach wasserstein gan," Advances in Neural Information Processing Systems, t. 31, 2018.

[57] Sen Pei, Richard Yi Da Xu, Shiming Xiang, Gaofeng Meng 5 Aug 2021.

[58] Junbo Zhao, Michael Mathieu, Yann LeCun [Submitted on 11 Sep 2016 (v1), last revised 6 Mar 2017 (this version, v4)]

https://doi.org/10.48550/arXiv.1609.03126

[59] D. Berthelot, T. Schumm et L. Metz, "Began : Boundary equilibrium generative adversarial networks," arXiv preprint arXiv :1703.10717, 2017.

[60] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever et P. Abbeel, "Infogan : Interpretable representation learning by information maximizing generative adversarial nets," Advances in neural information processing systems, t. 29, 2016.

[61] J. Nie, Y. Xiao, L. Huang et F. Lv, "Time-frequency analysis and target recognition of HRRP based on CN-LSGAN, STFT, and CNN," Complexity, t. 2021, 2021.

[62] Naveen Kodali, Jacob Abernethy, James Hays, Zsolt Kira [Submitted on 19 May 2017 (v1), last revised 10 Dec 2017 (this version, v5)].

https://doi.org/10.48550/arXiv.1705.07215

[63] SNGAN, by Preferred Networks, Inc., Ritsumeikan University, and National Institute of Informatics 2018 ICLR, Over 4200 Citations (Sik-Ho Tsang @ Medium).

[64] Conference: 2017 IEEE International Conference on Computer Vision (ICCV) October 2017 , DOI:10.1109/ICCV.2017.244.

[65] Tony F Chan and Jianhong Jackie Shen. Image processing and analysis: variational, PDE, wavelet, and stochastic methods, volume 94. Siam, 2005.

[66] 12(14):3038 DOI:10.3390/electronics12143038 Electronics 12(14):3038 , July 2023.

[67] Albahar, B., Lu, J., Yang, J., Shu, Z., Shechtman, E., Huang, J.B.: Pose with style: detail-preserving pose-guided image synthesis with conditional StyleGAN. ACM Trans. Graph. (TOG) **40**(6), 1–11 (2021).

[68] Werhahn, M., Xie, Y., Chu, M., Thuerey, N.: A multi-pass GAN for fluid flow super- resolution. Proc. ACM Comput. Graph. Interact. Tech. 2(2), 1–21 (2019).

[69] Holden, D., Saito, J., Komura, T.: Neural network ambient occlusion. In: SIGGRAPH ASIA 2016 Technical Briefs, pp. 1–4 (2016).

[70] Nalbach, O., Arabadzhiyska, E., Mehta, D., Seidel, H.P., Ritschel, T.: Deep shading: convolutional neural networks for screen space shading. In: Computer Graphics Forum, vol. 36, no. 4, pp. 65–78 (2017).

[71] Erra, U., Capece, N.F., Agatiello, R., Peytavie, A., Bosch, C.: Ambient occlusion baking via a feed-forward neural network. In: Eurographics (Short Papers), pp. 13–16 (2017).

[72] Inoue, N., Ito, D., Hold-Geoffroy, Y., Mai, L., Price, B., Yamasaki, T.: RGB2AO: ambient occlusion generation from RGB images. In: Computer Graphics Forum, vol. 39, no. 2, pp. 451–462 (2020).

[73] Abbas F., Malah M., Babahenini M.C., Attentional conditional generative adversarial network for ambient occlusion approximation, in: Bennour A., Ensari T., Kessentini Y., Eom S. (Eds.), Intelligent Systems and Pattern Recognition, Springer International Publishing, Cham, 2022, pp. 349–361.