

République algérienne démocratique et populaire
Ministère de l'enseignement supérieur et de la recherche
scientifique
Université du 8 mai 1945-Guelma-
Faculté des mathématiques, de l'informatique et des sciences
de la matière
Département d'informatique



Mémoire de fin d'étude Master

Filière : Informatique

Option : Système Informatique.

Thème

Analyse et Manipulation de l'espace latent d'un
GAN pour la génération d'image

Présenté par : Djemmam Ayet El Nour

<u>Membres de jury</u>	<u>Qualité</u>
GUERROUI Nadia	Président
BORDJIBA Yamina	Encadrante
FAROU Brahim	Examineur

Juin 2023

Remerciement

Tout d'abord, je tiens à remercier DIEU

De m'avoir donné la force et le courage de mener à bien ce modeste travail.

Je tiens à remercier mon encadrante Madame BORDJIBA yamina, pour sa patience, et surtout pour sa confiance, ses remarques et ses conseils, et sa bienveillance.

Je voudrais également remercier les membres du jury pour avoir accepté d'évaluer ce travail et pour toutes leurs remarques et critiques,

Je remercie toutes les personnes qui ont participé de près ou de loin à la réalisation de ce travail.

Merci à vous tous

Dédicaces

Je dédie cet ouvrage A mes parents qui m'ont soutenu et encouragé durant ces années d'études. Qu'ils trouvent ici le témoignage de ma profonde reconnaissance.

A mes frères, mes cousins et Ceux qui ont partagé avec moi tous les moments d'émotion lors de la réalisation de ce travail. Ils m'ont chaleureusement supporté et encouragé tout au long de mon parcours. A ma famille, mes proches et à ceux qui me donnent de l'amour et de la vivacité.

A tous mes amis « Aya, Anfel, Lina, Djouhaina, Rana, Belkis, Amira... » qui m'ont toujours encouragé, et à qui je souhaite plus de succès.

A tous ceux que j'aime

Résumé

Dans le domaine de la synthèse et de l'analyse d'images, les réseaux adversaires génératifs (Generative Adversarial Networks - GAN) ont connu d'importantes avancées. En particulier, le modèle StyleGAN est considéré comme l'un des modèles les plus performants pour la génération d'images de visages. Ses résultats sont d'une fidélité et d'une qualité visuelles remarquables.

L'édition d'une image réelle nécessite la conversion de l'image d'entrée dans les variables latentes de StyleGAN. Cependant, il est encore difficile de trouver des variables latentes qui permettent une manipulation significative. Dans ce projet de fin d'étude, l'objectif visé est de mettre en œuvre une application capable de contrôler la génération des images en manipulant l'espace latent.

Le système proposé commence par extraire les attributs du visage de l'image d'entrée à l'aide d'un CNN via la technique d'apprentissage par transfert, cette étape est cruciale dans notre système. Les attributs détectés seront ensuite présentés à l'utilisateur, afin qu'il puisse modifier ceux qu'il souhaite changer. Ces attributs modifiés seront ensuite concaténés avec un vecteur généré aléatoirement, pour former le vecteur latent d'entrée du générateur StyleGAN3. Ce générateur a été entraîné sur une sélection aléatoire à partir de la base de données CelebA.

Le système d'extraction des attributs a été entraîné sur 10% de la base de données CelebA environ 20k images, et a obtenu un taux de 86%, et le générateur de stylegan3 a été entraîné sur une partie sélectionnée de jeu de données CelebA, bien que ces résultats restent insuffisantes et soient encourageants, il est important de souligner que l'augmentation de la durée de l'entraînement permettra une nette amélioration de la qualité des Images générés ainsi la méthode d'entraînement de ce Gan.

Mots-clés : Extraction des attributs, Manipulation de l'espace latent, Contrôle de génération d'images, Apprentissage par transfert, Stylegan3.

Abstract

In the field of image synthesis and analysis, Generative Adversarial Networks (GANs) have made significant advances. In particular, the StyleGAN model is considered one of the most powerful models for generating images of faces. Its results are of remarkable visual fidelity and quality.

Editing a real image requires conversion of the input image into StyleGAN's latent variables. However, it is still difficult to find latent variables that allow meaningful manipulation. In this end-of-study project, the aim is to implement an application capable of controlling image generation by manipulating latent space.

The proposed system begins by extracting face attributes from the input image using a CNN via the transfer learning technique, a crucial step in our system. The detected attributes are then presented to the user, so that he can modify those he wishes to change. These modified attributes are then concatenated with a randomly generated vector, to form the latent input vector of the StyleGAN3 generator. This generator was trained on a random selection from the CelebA database.

The attribute extraction system was trained on 10% of the CelebA database around 20k images, and obtained a rate of 86%, and the stylegan3 generator was trained on a selected part of the CelebA dataset. Although these results remain insufficient and are encouraging, it is important to emphasize that increasing the training duration will allow a clear improvement in the quality of the Images generated as well as the training method of this Gan.

Keywords: Attribute extraction, Latent space manipulation, Image generation control, Transfer learning, Stylegan3.

المخلص

في مجال توليف الصور وتحليلها، حققت شبكات الخصومة التوليدية (GANs) تقدماً كبيراً. على وجه الخصوص، يعتبر نموذج StyleGAN أحد أقوى النماذج لتوليد صور للوجوه. نتائجه ذات إخلاص بصري وجودة ملحوظة.

يتطلب تحرير صورة حقيقية تحويل صورة الإدخال إلى متغيرات StyleGAN الكامنة. ومع ذلك، لا يزال من الصعب العثور على متغيرات كامنة تسمح بمعالجة ذات مغزى. في مشروع نهاية الدراسة هذا، الهدف هو تنفيذ تطبيق قادر على التحكم في توليد الصور عن طريق التلاعب بالفضاء الكامن.

يبدأ النظام المقترح باستخراج سمات الوجه من صورة الإدخال باستخدام CNN عبر تقنية التعلم النقلي، وهي خطوة حاسمة في نظامنا. ثم يتم تقديم السمات المكتشفة إلى المستخدم، حتى يتمكن من تعديل تلك التي يرغب في تغييرها. ثم يتم اختلاق هذه السمات المعدلة بمتجه مولد عشوائياً، لتشكيل متجه الإدخال الكامن للمولد StyleGAN3. تم تدريب هذا المولد على اختيار عشوائي من قاعدة بيانات CelebA.

تم تدريب نظام استخراج السمات على 10% من قاعدة بيانات CelebA حول 20 ألف صورة، وحصل على معدل 86%، وتم تدريب مولد stylegan3 على جزء مختار من مجموعة بيانات CelebA. على الرغم من أن هذه النتائج لا تزال غير كافية لكنها مشجعة، فمن المهم التأكيد على أن زيادة مدة التدريب ستسمح بتحسين واضح في جودة الصور التي تم إنشاؤها وكذلك طريقة التدريب لهذا Gan.

الكلمات الرئيسية: استخراج السمات، التلاعب بالفضاء الكامن، التحكم في توليد الصور، نقل التعلم، Stylegan3.

Table des Matières

RESUME	1
LISTE DES FIGURES	8
LISTE DES TABLEAUX.....	10
INTRODUCTION GENERALE	11
1 CHAPITRE 1	13
RESEAUX ANTAGONISTES GENERATIFS « GAN ».....	13
1.1 INTRODUCTION.....	13
1.2 APPRENTISSAGE AUTOMATIQUE	13
1.2.1 L'apprentissage supervisé :	14
1.2.2 Apprentissage non supervisé	15
1.2.3 Apprentissage semi supervisé.....	15
1.3 APPRENTISSAGE PROFOND.....	16
1.4 LES RESEAUX DE NEURONES.....	17
1.5 ARCHITECTURE D'UN RESEAU DE NEURONE	18
1.6 LES DIFFERENTS TYPES DE RESEAUX DE NEURONES	18
1.7 LES RESEAUX DE NEURONES CONVOLUTIFS	21
1.7.1 L'architecture d'un CNN.....	21
1.8 DU RÉSEAU DE NEURONES ARTIFICIELS (ANN) À L'APPRENTISSAGE PROFOND	25
1.9 LES MODÈLES GÉNÉRATIFS	25
1.10 LES RESEAUX ANTAGONISTES GENERATIFS (GAN)	26
1.10.1 Principe des GAN	27
1.11 LES DIFFERENTES ARCHITECTURES DES GANS	27
1.11.1 Le Conditionnel GAN (cGAN).....	27
1.11.2 Progressive GAN (PROGAN) :	28
1.12 LES AUTO-ENCODEURS (AE)	30
1.12.1 Fonctionnement et Architecture des auto-encodeurs.....	30
1.12.2 Exemples d'auto-encodeurs	31

1.13	ÉVALUATION DES PERFORMANCES :	31
1.13.1	Matrice de confusion :	31
1.13.2	L'exactitude (Accuracy) :	32
1.13.3	La précision (Precision) :	33
1.13.4	Le rappel (Recall) :	33
1.13.5	L'aire sous la courbe ROC (AUC-ROC) :	33
1.14	CONCLUSION	34
2	CHAPITRE 2	35
	STYLEGAN	35
2.1	INTRODUCTION	35
2.2	ORIGINE DU STYLEGAN	35
2.3	ARCHITECTURE DE STYLEGAN	36
2.4	VERSIONS ULTERIEURES DE STYLEGAN	39
2.4.1	StyleGan2	39
2.4.2	StyleGAN2-ADA	40
2.4.3	StyleGan3	41
2.5	DEFINITION DE L'ESPACE LATENT	42
2.6	IMPORTANCE D'ESPACE LATENT DANS L'APPRENTISSAGE PROFOND	44
2.7	EXEMPLES D'ESPACES LATENTS	45
2.7.1	Espace des caractéristiques de l'image	45
2.7.2	L'espace latent de Style Gan :	45
2.7.3	L'espace latent de l'auto-encodeur Variational (VAE)	47
2.7.4	L'espace latent de Cycle GAN :	47
2.7.5	L'espace latent de GPT-2 (Generative Pre-trained Transformer 2) :	47
2.8	LES BASES DE DONNEES	48
2.8.1	CelebA	48
2.8.2	Flickr-Faces FFHQ	49
2.8.3	Des visages étiquetés dans la nature (LFW)	49
2.8.4	MegaFace	49
2.9	TRAVAUX CONNEXES	50
2.9.1	Article 1	50
2.9.2	Article 2	50
2.9.3	Article 3	51

2.9.4	Article 4	52
2.9.5	Article 5	53
2.9.6	Article 6	54
2.9.7	Article 7	54
2.9.8	Article 8	55
2.10	CONCLUSION	56
3	CHAPITRE 3	57
	CONCEPTION	57
3.1	INTRODUCTION	57
3.2	OBJECTIFS	57
3.2.1	Extraction des attributs de visage de l'image d'entrée	58
3.2.2	Génération et construction du nouveau vecteur latent	58
3.2.3	La génération d'images	58
3.3	ARCHITECTURE GENERALE DU SYSTEME PROPOSE	59
3.4	PRE-TRAITEMENT DE L'IMAGE D'ENTREE	60
3.5	L'EXTRACTION DES ATTRIBUTS PAR NOTRE MODELE CNN	60
3.6	GENERATION ET CONSTRUCTION DU NOUVEL VECTEUR LATENT	63
3.7	GENERATION D'IMAGES	64
3.8	LE CHOIX DES MODELES UTILISES	65
3.9	CONCLUSION	68
4	CHAPITRE 4	70
	IMPLEMENTATION	70
4.1	INTRODUCTION	70
4.2	ENVIRONNEMENT DE DEVELOPPEMENT	70
4.2.1	Google Colab	70
4.2.2	Jupyter	72
4.2.3	Anaconda	72
4.3	LANGAGES DE PROGRAMMATION ET BIBLIOTHEQUES UTILISEES	72
4.3.1	Python	72
4.4	LES BIBLIOTHEQUES UTILISEES PENDANT LA PROGRAMMATION	73
4.5	APPRENTISSAGE ET TEST	74
4.5.1	Base de données utilisée	74

4.5.2	Apprentissage et test du MobileNetV2.....	75
4.5.3	Apprentissage et test du générateur de StyleGAN3	76
4.6	ÉVALUATION ET RESULTATS	77
4.6.1	Évaluation et Résultats du système d'extraction des attributs.....	78
4.6.2	Évaluation et résultats du système de génération	82
4.7	CONCLUSION.....	85
	CONCLUSION GENERALE.....	86
	RÉFÉRENCES.....	88

Liste des Figures

FIGURE 1-1 - L'INTELLIGENCE ARTIFICIEL ET L'APPRENTISSAGE AUTOMATIQUE	14
FIGURE 1-2- SCHEMA D'APPRENTISSAGE SUPERVISE	15
FIGURE 1-3- EXEMPLE D'APPRENTISSAGE PROFOND	17
FIGURE 1-4 ARCHITECTURE D'UN NEURONE FORMEL	17
FIGURE 1-5 ARCHITECTURE D'UN RESEAU DE NEURONE.....	18
FIGURE 1-6- DIFFERENTES ARCHITECTURES DES RESEAUX DE NEURONES ARTIFICIELS	20
FIGURE 1-7- ARCHITECTURE CLASSIQUE D'UN CNN.....	21
FIGURE 1-8- REPRESENTATION D'UNE COUCHE CONVOLUTIVE.....	22
FIGURE 1-9- EXEMPLE D'UN MAX POOLING	22
FIGURE 1-10-MODELE DE RESEAU NEURONAL PAR DROPOUT.....	23
FIGURE 1-11-FONCTION DE RELU	24
FIGURE 1-12- TYPES DE FONCTION D'ACTIVATION	24
FIGURE 1-13-EXEMPLE D'UNE ARCHITECTURE D'UN CNN	25
FIGURE 1-14- ARCHITECTURE D'UN GAN	27
FIGURE 2-1- COMPARAISON ENTRE LES GAN TRADITIONNELS ET STYLEGAN	37
FIGURE 2-2-MODELE DE GENERATION DU STYLEGAN.....	38
FIGURE 2-3- EXEMPLE D'IMAGES DE VISAGES GENERES PAR STYLEGAN.....	39
FIGURE 2-4-LE CHANGEMENT EFFECTUE DANS L'ARCHITECTURE ORIGINALE DE STYLEGAN... 40	
FIGURE 2-5-ARCHITECTURE DE STYLEGAN2-ADA	41
FIGURE 2-6-ARCHITECTURE DE STYLEGAN3.....	42
FIGURE 2-7 – EXEMPLE D'ESPACE LATENT REGROUPE DES ANIMAUX.	43
FIGURE 2-8-REPRESENTATION DE L'ESPACE LATENT DANS LE RESEAU DE NEURONE CONVOLUTIF	44
FIGURE 2-9- LA RELATION ENTRE L'APPRENTISSAGE PROFONDE ET L'ESPACE LATENT.....	44
FIGURE 2-10- ARCHITECTURE DU CNN	45
FIGURE 2-11- MAPPING NETWORK DE STYLEGAN.	46
FIGURE 2-12-EXEMPLE D'IMAGE DE CELEBA	48
FIGURE 2-13-EXEMPLE D'IMAGES DE FFHQ	49
FIGURE 3-1-OBJECTIF DE L'APPLICATION	57
FIGURE 3-2- ARCHITECTURE GENERALE DU SYSTEME PROPOSE.....	59
FIGURE 3-3- SCHEMA GENERALE D'EXTRACTION DES ATTRIBUTS	61

FIGURE 3-4-LES FREQUENCES RELATIVES DES ATTRIBUTS DANS LA BASE DE DONNEES CELEBA.	62
FIGURE 3-5- GENERATION ET CONSTRUCTION DU NOUVEAU VECTEUR LATENT	64
FIGURE 3-6- ARCHITECTURE DU STYLEGAN3	65
FIGURE 3-7- LA PARTIE AJOUTEE A L'ARCHITECTURE DU MOBILENETV2	67
FIGURE 4-1-INTERFACE DE GOOGLE COLAB	71
FIGURE 4-2- RESULTATS DE GENERATION AVEC LE GENERATEUR STYLEGAN3 PRE-ENTRAINE	83
FIGURE 4-3- RESULTATS DE GENERATION AVEC LE GENERATEUR DE STYLEGAN3 ENTRAINE .	84
FIGURE 4-4- ENSEMBLE D'IMAGES GENEREES PAR STYLEGAN3 ENTRAINE	85

Liste des Tableaux

TABLEAU 1- MATRICE DE CONFUSION	32
TABLEAU 2- L'ARCHITECTURE GENERALE DE MODELE MOBILENETV2	66
TABLEAU 3- PARAMETRES D'APPRENTISSAGE DES EXPERIMENTATIONS REALISEES.....	76
TABLEAU 4- ÉVALUATION DE SYSTEME D'EXTRACTION DES ATTRIBUTS	78

Introduction générale

L'édition d'images et la manipulation de l'espace latent sont des domaines d'intérêt en plein essor dans le domaine de l'intelligence artificielle. L'apprentissage profond est actuellement l'approche la plus répandue dans le domaine de l'IA, utilisant des réseaux neuronaux artificiels qui imitent le fonctionnement du cerveau humain. Parmi ces modèles, les réseaux adversaires génératifs sont l'un des plus performants dans la génération d'images, permettant d'éditer et de contrôler les images générées pour répondre à des besoins spécifiques.

L'espace latent du GAN joue un rôle crucial dans la génération d'images et est généralement représenté par un vecteur de bruit multidimensionnel. Il s'agit d'une représentation abstraite qui peut être manipulée pour obtenir des changements visuels dans les images générées. Les opérations sur l'espace latent permettent aux utilisateurs de contrôler des aspects tels que la couleur, la texture, la forme et d'autres propriétés visuelles des images générées.

Ce mémoire se concentre sur cette problématique du contrôle de l'édition des attributs des images générées en agissant sur l'espace latent d'un modèle génératif profond appelé StyleGAN3. L'objectif principale est de concevoir et mettre en œuvre une application qui permette ce type de contrôle sur des images de visage. Nous combinons un réseau neuronal convolutionnel CNN pour extraire les attributs d'une image d'entrée, puis nous utilisons ces attributs pour modifier l'espace latent du modèle et générer des images similaires qui correspondent aux attributs spécifiés. Les modèles utilisés sont entraînés sur la base de la base de données CelebA.

Notre mémoire est structuré en quatre chapitres qui sont présentés brièvement comme suit :

Chapitre 1 : Réseau adversaire génératif (GAN)

Dans ce chapitre, nous examinons divers concepts liés à l'apprentissage en profondeur et aux réseaux génératifs contradictoires, avec un accent particulier sur les types de modèles les plus étudiés dans la littérature.

Chapitre 2 : StyleGAN

Dans ce chapitre, nous avons examiné de près les modèles StyleGAN, qui sont des architectures de modèles génératifs profonds largement étudiées et utilisées dans la littérature. Nous avons abordé les principaux types de modèles StyleGAN qui ont été largement étudiés. Cela inclut StyleGAN original, StyleGAN2 et StyleGAN2-ADA (Adaptive Discriminator Augmentation), StyleGAN3, et nous avons souligné les caractéristiques clés de ces modèles.

Chapitre 3 : Conception

Dans ce chapitre, nous offrons une analyse détaillée de la conception de notre projet, en présentant en détail l'architecture et la structure globale de notre application. Nous mettons l'accent sur les décisions prises tout au long du processus de développement. De plus, nous décrivons les différentes fonctionnalités des composants clés de notre application et expliquons comment ils interagissent entre eux pour atteindre les objectifs fixés.

Chapitre 3 : Implémentation

Ce chapitre offre un aperçu détaillé de la mise en œuvre de notre projet. Tout d'abord, il décrit l'environnement de développement utilisé, ainsi que les langages et bibliothèques qui ont été employés. Ensuite, il détaille les étapes de mise en œuvre de chaque composant du système proposé. Les tests et les validations effectués pour garantir le bon fonctionnement de l'application sont également abordés. Enfin, nous analysons les résultats obtenus et évaluons les performances du système par rapport aux objectifs initialement fixés. Nous avons également présenté et interprété les différents résultats obtenus dans cette section.

Pour conclure cet article, nous proposons une synthèse globale des résultats obtenus et les principaux apports de notre étude. De plus, nous formulons des recommandations et suggestions pour les travaux futurs.

1 Chapitre 1

Réseaux antagonistes génératifs « GAN »

1.1 Introduction

L'apprentissage profond ou « Deep learning » est une branche de l'intelligence artificielle dérivée de l'apprentissage automatique ou « machine learning ». Contrairement à la programmation qui consiste à exécuter des règles préétablies, ce type d'apprentissage permet aux machines d'apprendre de manière autonome. En fait, les machines sont capables de s'entraîner sur des données, de reconnaître des modèles et de prendre des décisions basées sur ces données sans intervention humaine.

L'apprentissage profond a été utilisé dans diverses applications, notamment la reconnaissance d'images et de la parole, le traitement du langage naturel et la conduite autonome. Il a permis d'atteindre des performances de pointe dans de nombreuses tâches et a été à l'origine de nombreuses avancées récentes dans le domaine de l'intelligence artificielle. Cependant, l'apprentissage profond nécessite également de grandes quantités de données et de ressources informatiques, ce qui en fait une technologie gourmande en ressources.

L'élément de base de l'apprentissage profond est le réseau neuronal qui se compose de neurones artificiels, ce dernier prend des valeurs d'entrée, applique des poids à ces entrées et produit une sortie. Dans un réseau neuronal profond, plusieurs couches de neurones sont empilées les unes sur les autres, chaque couche transformant l'entrée de la couche précédente jusqu'à ce qu'une sortie finale soit produite.

Dans ce chapitre nous présentons un modèle profond génératif connu sous le nom GAN.

1.2 Apprentissage automatique

L'apprentissage automatique ou Machine Learning (ML) est un sous-ensemble de l'intelligence artificielle qui se concentre sur le développement d'algorithmes et de modèles statistiques capables de faire des prédictions ou de classer des données sur la base des informations apprises à partir des données d'apprentissage. C-à-dire d'effectuer des tâches sans être explicitement programmés.

Le ML est de plus en plus populaire en raison de ses nombreux avantages. La disponibilité accrue de données et la capacité de calcul croissante ont permis de développer des logiciels "intelligents" capables de résoudre des problèmes complexes et variés. Grâce à l'apprentissage machine, les ordinateurs sont capables d'apprendre en permanence et d'améliorer leur performance, parfois même au-delà des capacités humaines. [1]

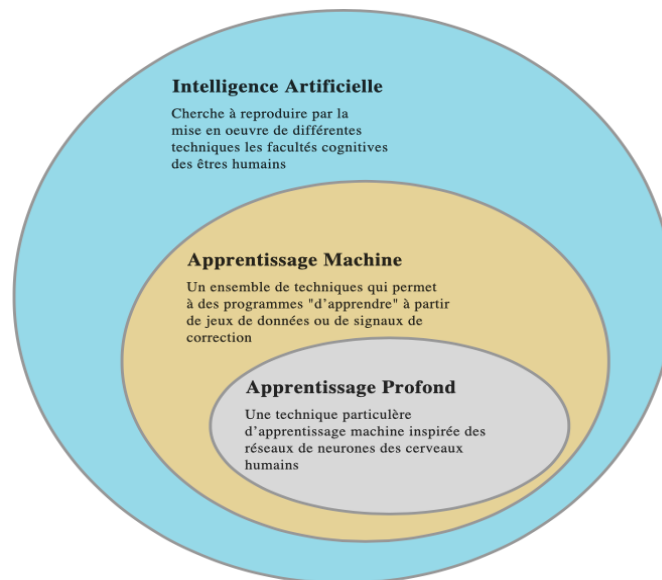


Figure 1-1 - L'intelligence artificiel et l'apprentissage automatique [2]

Il existe trois principaux types d'apprentissage automatique :

1.2.1 L'apprentissage supervisé :

L'apprentissage supervisé est une méthode d'apprentissage automatique où un modèle est entraîné sur un ensemble de données étiquetées. Le processus d'apprentissage automatique implique la création d'un algorithme capable d'apprendre à prédire une fonction. Pour y parvenir, l'algorithme est entraîné à partir d'exemples annotés comprenant des variables d'entrée et leurs variables de sortie correspondantes. Ce processus d'entraînement est répété jusqu'à l'obtention d'une performance satisfaisante, et Grâce à lui, le modèle est en mesure d'apprendre à partir des données et de suivre des règles précises pour prédire avec précision la valeur de sortie en fonction de la valeur d'entrée fournie.[3]

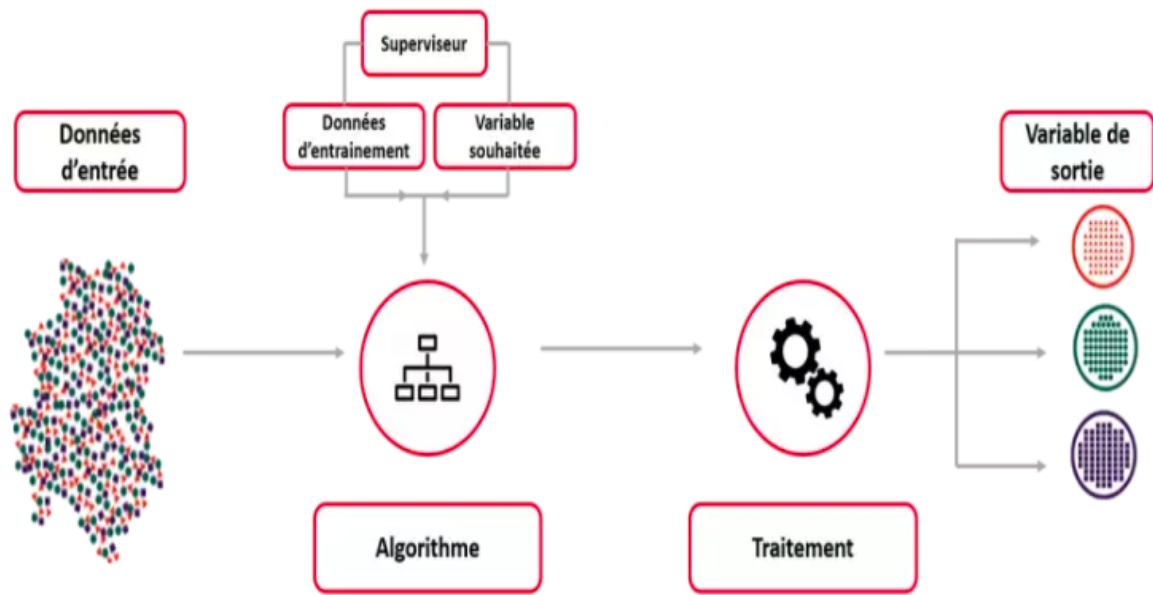


Figure 1-2- Schéma d'apprentissage supervisé [3]

1.2.2 Apprentissage non supervisé

L'apprentissage non supervisé est une méthode d'apprentissage automatique où l'algorithme apprend à partir de données non étiquetées sans la nécessité d'une supervision humaine. Il consiste à apprendre les poids d'un réseau sans utiliser de données annotées. Cette approche permet d'utiliser de vastes quantités de données non annotées, qui sont facilement accessibles en grande quantité. Pour y parvenir, une tâche prétexte est utilisée pour extraire des informations de haut niveau utiles à la classification, sans nécessiter d'annotation préalable. Cette méthode est souvent utilisée pour découvrir des motifs cachés ou des structures dans les données, ou pour réduire la dimensionnalité des données en trouvant les caractéristiques les plus pertinentes. Contrairement à l'apprentissage supervisé, où l'algorithme est entraîné à prédire des sorties en fonction de données étiquetées, l'apprentissage non supervisé n'a pas de sortie étiquetée prédéfinie à prédire. [4].

1.2.3 Apprentissage semi supervisé

L'apprentissage semi supervisé est une méthode d'apprentissage automatique qui s'intéresse à la prise de décision dans laquelle un agent apprend à prendre des décisions en interagissant avec un environnement dynamique. Il implique la capacité d'un algorithme à apprendre de ses erreurs dans le but d'atteindre un objectif. Cela implique aussi que l'algorithme explore plusieurs approches différentes afin de trouver une solution optimale pour atteindre son objectif. Les trois éléments principaux dans ce processus c'est : l'agent, l'environnement et la politique tels qu'Après avoir pris en compte l'état courant de l'environnement (représenté par

"s" pour "state"), l'agent utilise une politique d'action appelée " π " pour sélectionner une action à effectuer. L'objectif de l'apprentissage par renforcement est d'améliorer la politique d'action de l'agent en utilisant un système de récompenses positives et négatives. Ce type d'apprentissage est souvent utilisé dans des situations où les données d'apprentissage ne sont pas disponibles ou sont coûteuses à obtenir, comme dans le domaine des jeux, de la robotique et des systèmes de recommandation.

Il existe également d'autres sous-types d'apprentissage automatique, notamment l'apprentissage par transfert et l'apprentissage multitâche, mais les trois principales catégories sont l'apprentissage supervisé, non supervisé et par renforcement.

1.3 Apprentissage profond

L'apprentissage profond plus connu sous le nom « Deep Learning » regroupe un ensemble de techniques d'apprentissage automatique basées sur des approches mathématiques. Ces techniques sont utilisées pour modéliser des données et améliorer la compréhension de celles-ci. Le machine Learning a été développé dans les années 1950 par Alan Turing. Le Deep Learning est une forme avancée d'apprentissage automatique, qui utilise un réseau de neurones artificiels interconnectés, similaires au cerveau humain, pour traiter et mémoriser des informations comme les êtres humains [5]. Le principe de base de l'apprentissage profond est d'entraîner un réseau de neurones à partir d'un ensemble de données d'apprentissage afin qu'il puisse effectuer des tâches spécifiques, telles que la reconnaissance d'images, la classification de textes, la prédiction de séries temporelles, etc. et Pour résoudre des problèmes de la meilleure façon possible. Ce processus est effectué à l'aide d'un algorithme d'optimisation qui cherche à minimiser une fonction de perte ou de coût définie pour la tâche en question. Une fois que le réseau de neurones a été entraîné, il peut être utilisé pour effectuer des prédictions sur de nouvelles données en général plus ou moins similaires à celles utilisées pour l'entraînement (figure 2.2)., il offre des moyens d'apprendre les représentations de données de manière supervisée et non Supervisée à l'aide de la hiérarchie des couches, qui permet un traitement multiple, et par rapport au méthodes conventionnelles de ML

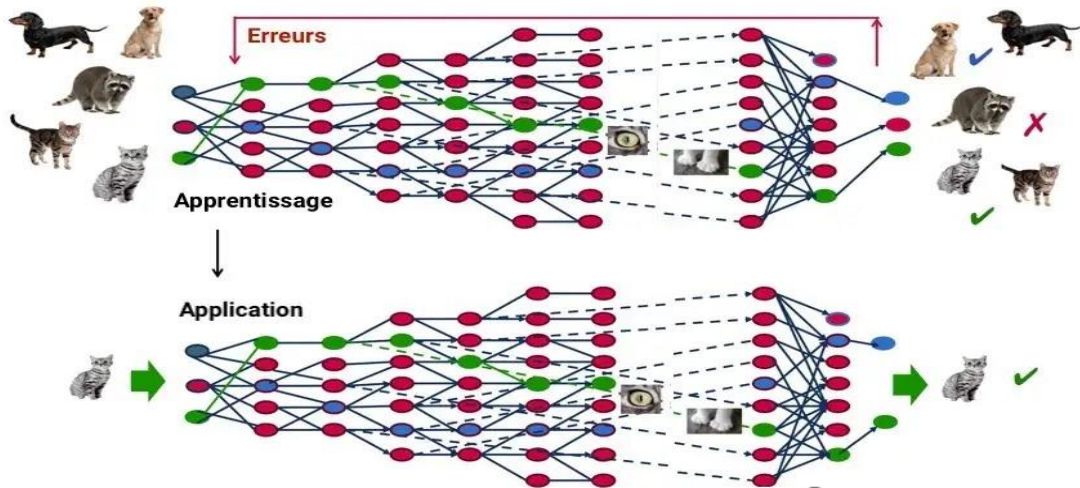


Figure 1-3- Exemple d'apprentissage profond [5]

1.4 Les réseaux de neurones

Un réseau neuronal artificiel (ou réseau neuronal profond) s'inspire de la structure du cerveau humain (imitent la façon dont les neurones biologiques s'envoient mutuellement des signaux). Alors les Réseaux de neurones artificiels (ANN) sont des modèles d'apprentissage automatique composés de deux ou plusieurs couches qui sont constituées d'une couche d'entrée, d'une ou plusieurs couches cachées et d'une couche de sortie neurones, ces derniers interagissent avec un flux de données d'apprentissage pour apprendre à effectuer des tâches. Les réseaux de neurones artificiels sont appliqués à de nombreux domaines : la reconnaissance d'images et la vision par ordinateur, la reconnaissance vocale et, plus généralement, le traitement du langage naturel (NLP).

La figure suivante représente la relation entre des données d'un espace X et un espace de sortie Y dans neurone formel [6]:

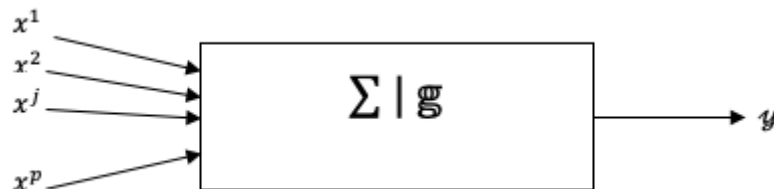


Figure 1-4 Architecture d'un neurone formel [7]

Chaque neurone artificiel se connecte à un autre neurone et possède un poids et un seuil associés. Si la sortie d'un neurone dépasse le seuil défini, le neurone est activé et transmet des

données à la couche suivante du réseau. En revanche, si la sortie est inférieure au seuil, aucune donnée n'est transmise à la couche suivante du réseau [8]

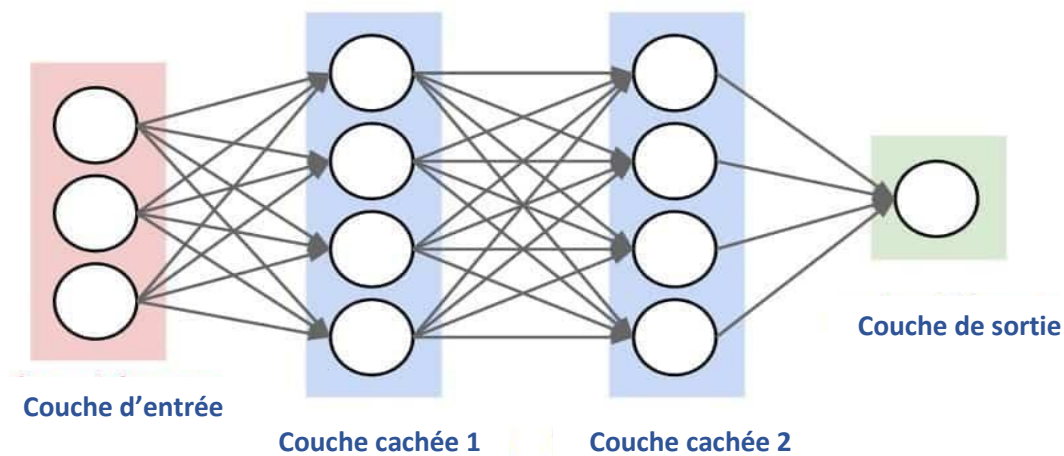


Figure 1-5 Architecture d'un réseau de neurone [9]

1.5 Architecture d'un réseau de neurone

L'architecture d'un réseau de neurones fait référence à la manière dont les neurones sont organisés et connectés les uns aux autres au sein du réseau. En d'autres termes, il s'agit de la structure et de l'ordre dans lequel les neurones sont arrangés dans le réseau. En général, l'architecture d'un réseau de neurones dépend de la tâche d'apprentissage à accomplir [10].

Un réseau de neurone est structuré en 3 couches qui sont :

- **La Couche d'entrée (Input) :** le contenu de cette couche fluctue en fonction des exigences et des spécifications du réseau de neurones en question. Cela permet au réseau de neurones de s'adapter efficacement aux différents scénarios et de fournir des résultats précis et fiables.
- **La/les couche(s) cachée(s) :** la couche cachée se situe entre la couche d'entrée et la couche de sortie et chaque neurone de cette couche est connecté à tous les neurones de la couche précédente en entrée et à tous les neurones de la couche suivante en sortie.
- **La Couche de sortie (output) :** Le nombre de neurones dans cette couche est variable et correspond au contexte du réseau.

1.6 Les différents types de réseaux de neurones

Il existe différentes sortes de réseaux neuronaux. En règle générale, les réseaux de neurones sont classés en fonction du nombre d'épaisseurs qui séparent l'entrée de données de la sortie du

résultat, chacun conçu pour résoudre des tâches spécifiques en fonction de leur architecture et de leur fonctionnement. Voici quelques exemples de réseaux de neurones couramment utilisés :

1. Réseaux de neurones multicouches (MLP) : connus sous le nom de réseaux de neurones à propagation avant.
2. Réseaux de neurones convolutifs (CNN) : Ces réseaux sont plus utilisés pour le traitement des images et de la vision par ordinateur. Elles contiennent des couches de convolution qui filtrent les intrants pour en extraire les caractéristiques significatives, suivies de couches de regroupement qui réduisent la taille de la représentation. [1]
3. Réseaux de neurones récurrents (RNN) : Ces réseaux ont pour but de traiter des données séquentielles.
4. Réseaux de neurones générateurs adverses (GAN) : Ces réseaux sont communément utilisés pour produire des images, des vidéos ou des sons réalistes à partir de données brutes. [12]
5. Réseaux de neurones auto-encodeurs (AE) : Ces réseaux sont utilisés pour la compression et la reconstruction de données. Ils sont constitués de deux parties : un encodeur qui transforme les données en une représentation compressée, et un décodeur qui transforme la représentation compressée en une reconstruction de l'entrée originale. [13]
6. Réseaux de neurones récurrents à mémoire à court terme (LSTM) : Ces réseaux sont une forme de RNN qui ont la capacité de conserver des informations sur une longue période de temps. [14]

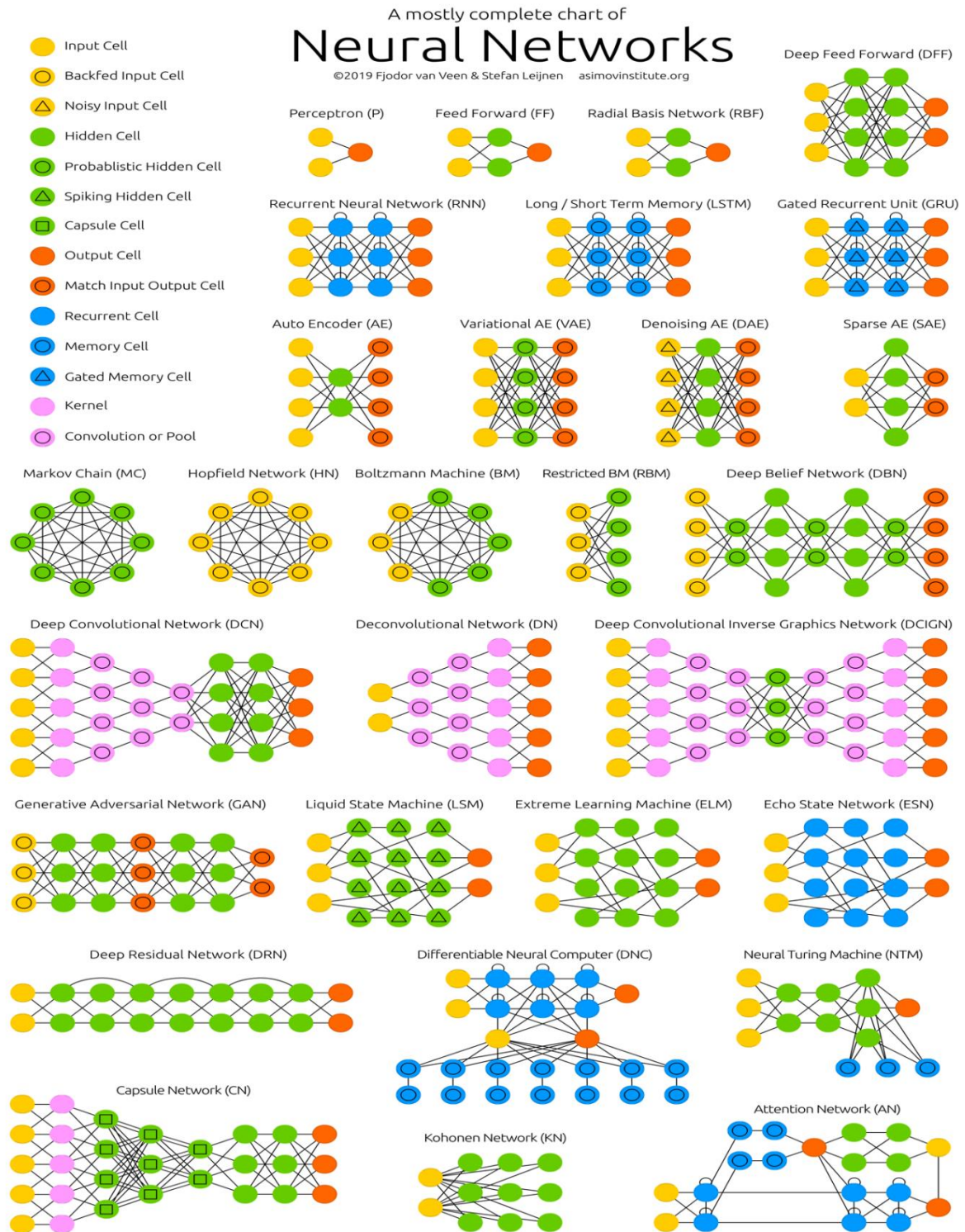


Figure 1-6- Différentes architectures des Réseaux de neurones artificiels [15]

La figure 6 présente la majorité des architectures populaires des réseaux de neurones.

1.7 Les réseaux de neurones convolutifs

Les réseaux de neurones convolutifs, également appelés CNN (Convolutional Neural Networks) ou (ConvNet) constituent une classe de réseaux de neurones profonds qui ont été élaborés à l'origine pour la reconnaissance d'images. Ce sont des Modèles de programmation puissants qui permettent de résoudre des problèmes de vision informatique tels que : la classification d'images, la détection d'objets, la segmentation sémantique, la reconnaissance faciale, etc.

1.7.1 L'architecture d'un CNN

On trouve deux parties principales :

- **Une partie convolutive** : Son but ultime est d'extraire les caractéristiques propres à chaque image en les compressant pour réduire leur taille initiale. Il génère à la fin une carte caractéristique.
- **Une partie classification** : La carte caractéristique obtenue en sortie de la partie convolutive est fournie en entrée dans une deuxième partie, constituée de couches entièrement connectées appelées perceptron multicouche (MLP pour Multi Layer Perceptron). Le rôle de cette partie est de combiner les caractéristiques du code CNN afin de classer l'image.

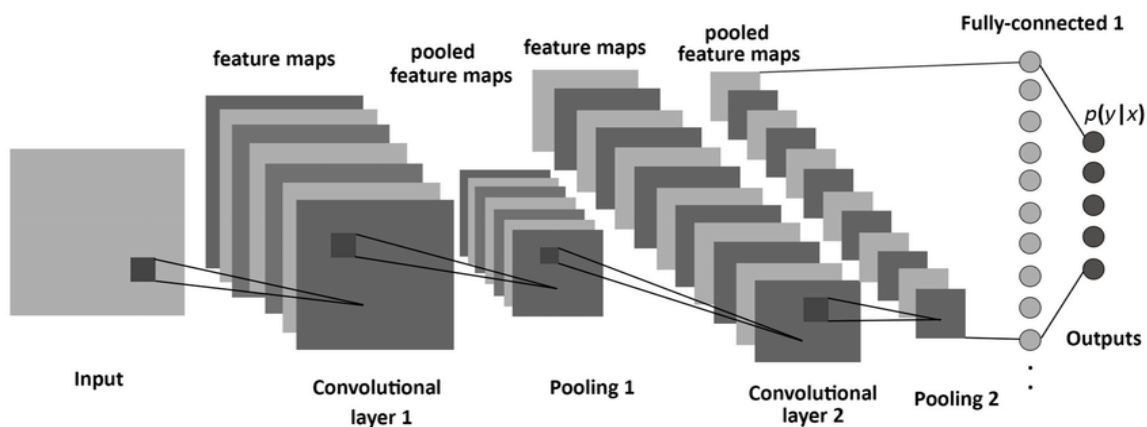


Figure 1-7- Architecture classique d'un CNN [16]

Dans ce qui suit nous entrerons dans plus de détails sur les deux parties précédentes, un CNN est généralement composé de plusieurs couches, chacune avec sa propre fonctionnalité. Voici les principales couches d'un CNN :

- **Les couches de convolution** : c'est la première couche d'un CNN qui joue un rôle principal pour son fonctionnement, cette couche utilise un ensemble de filtres pour

extraire les caractéristique (features) de l'image d'entrée (input image). Chaque filtre est une petite matrice de poids (figure 8) qui parcourt l'image d'entrée, en effectuant un produit scalaire entre les entrées et ces filtres et au même temps une carte d'activation est produite et elles sont modulées par une fonction d'activation, généralement une unité linéaire rectifiée (communément appelée Relu). La taille et le nombre de ces filtres sont définis à priori.

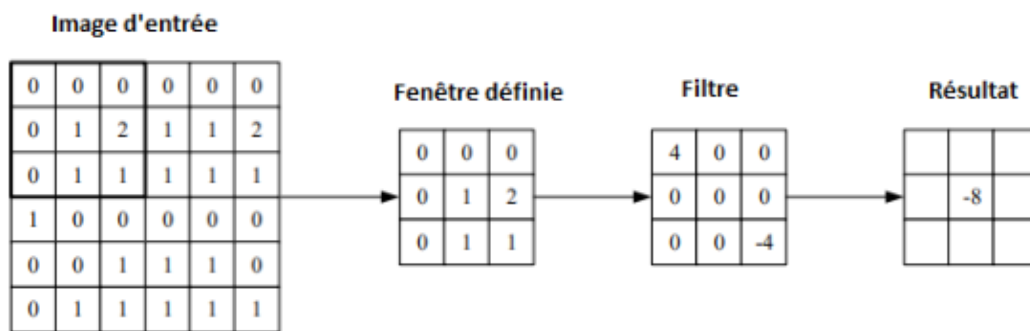


Figure 1-8- Représentation d'une couche convolutive[17]

- **Les couches de regroupements (pooling) :** cette couche a pour objectif de diminuer la taille de la carte de caractéristique produite par la couche de convolution. Elles fonctionnent indépendamment des différentes profondeurs du réseau et ne requièrent pas de poids à entraîner. Le regroupement maximal est la technique la plus utilisée pour diminuer la taille d'une image. Elle consiste à réduire une fenêtre donnée à une valeur unique en sélectionnant la plus grande valeur parmi les éléments de la fenêtre.

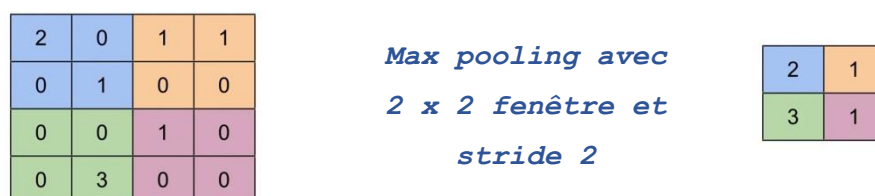


Figure 1-9- Exemple d'un max pooling [18]

- **Les couches entièrement connectées (fully connected layers) :** c'est la dernière couche du CNN, Elle utilise les caractéristiques extraites des couches précédentes pour effectuer une classification. Les neurones de ces couches sont tous reliés aux neurones des cartes d'activation précédentes.[17]

Afin d'améliorer les performances d'un CNN, il existe d'autres types de couches qui peuvent être utilisées. Parmi celles-ci, on peut citer :

- **Le dropout :**

Le Dropout est l'un des types de couches utilisés pour éviter le sur apprentissage « l'overfitting » du réseau. Cette technique a été introduite par Srivastava et al. Dans leur article "Dropout : A Simple Way to Prevent Overfitting in Neural Networks" (2014) [19] . Le principe de cette technique est de désactiver aléatoirement un certain pourcentage de neurones lors de l'entraînement du réseau. Cela permet d'éviter les dépendances excessives entre les neurones et force le réseau à apprendre des caractéristiques plus robustes et plus générales. En désactivant les neurones de façon aléatoire, le stalling agit comme une forme de régularisation et réduit le risque de surentraînement en rendant le réseau plus résistant aux petits changements dans les données d'entraînement. De nombreuses études ont montré que l'utilisation de la temporisation peut permettre d'améliorer les performances des réseaux neuronaux, en particulier dans les scénarios où les données d'apprentissage sont limitées.

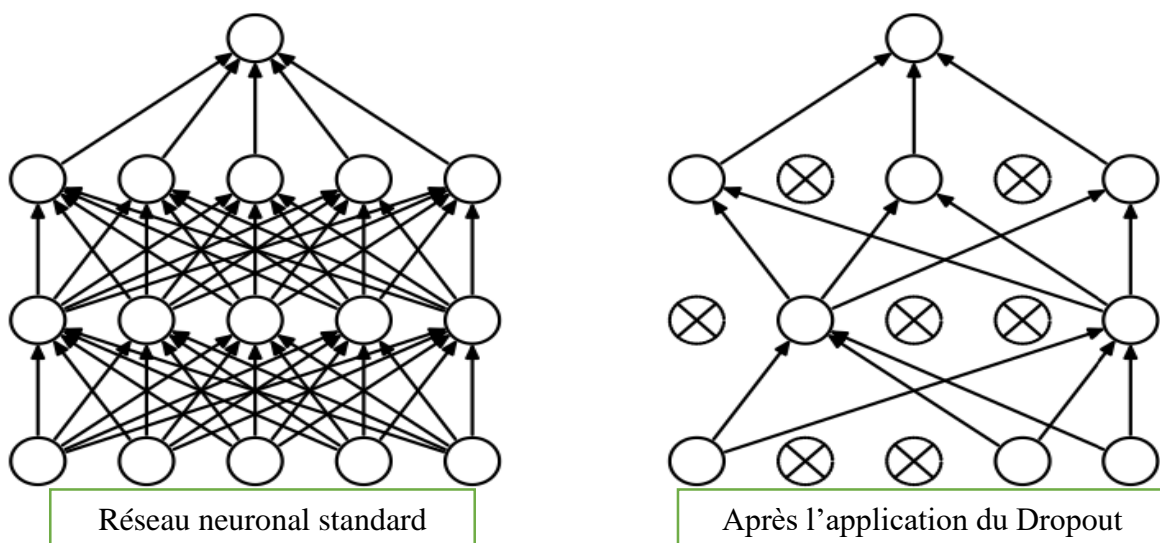


Figure 1-10-Modèle de réseau neuronal par Dropout[19]

- **La couche de ReLU (Rectified Linear Unit) :** Cette couche de correction utilise une fonction d'activation définie dans la figure suivante

$$\text{fonction_ReLU}(x) = \max(x, 0)$$

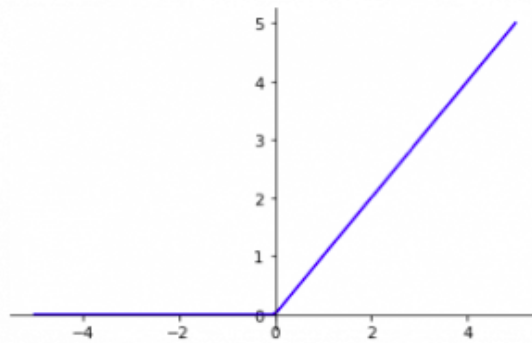


Figure 1-11-Fonction de ReLU [20]

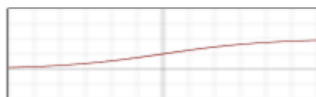
Où x est la sortie du neurone. L'avantage de cette fonction est qu'elle introduit des non-linéarités dans le réseau, ce qui permet de modéliser des relations complexes entre les données d'entrée et les activations neuronales. Il existe différents types de ces fonctions qu'on peut citer quelques-uns les plus connus (figure 12) :

Fonction "Marche/Heaviside"



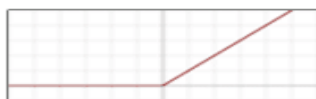
$$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ 1 & \text{si } x \geq 0 \end{cases}$$

Fonction Sigmoide



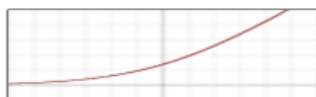
$$f(x) = \frac{1}{1 + e^{-x}}$$

Fonction "Unité de rectification linéaire" (ReLU)



$$f(x) = \begin{cases} 0 & \text{si } x < 0 \\ x & \text{si } x \geq 0 \end{cases}$$

Fonction "Unité de rectification linéaire douce" (SoftPlus)



$$f(x) = \ln(1 + e^x)$$

Figure 1-12- Types de fonction d'activation

Il existe d'autres types d'architecture de réseaux de neurones artificiels comme l'exemple illustré dans la figure 10, consiste à empiler deux couches convolutives avant chaque couche de mise en commun

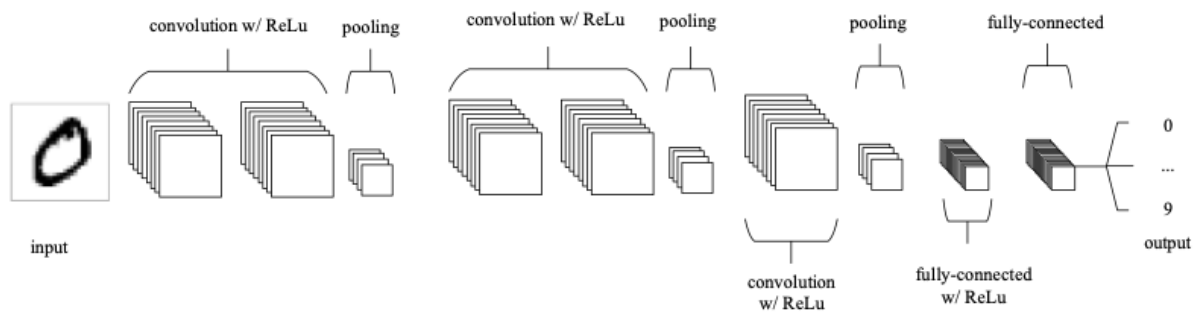


Figure 1-13-Exemple d'une architecture d'un CNN[17]

1.8 Du Réseaux de neurones artificiels (ANN) à l'apprentissage profond

Depuis des décennies, les réseaux de neurones artificiels ont été utilisés pour résoudre des problèmes de classification, de prédiction, de classification, d'optimisation, de reconnaissance et d'autres problèmes. Dans divers domaines tels que la reconnaissance de caractères manuscrits, la détection de fraude et la reconnaissance vocale. L'utilisation des ANN à des limites importantes. En effet, pour effectuer des tâches complexes, ils ont souvent besoin d'un grand nombre de paramètres, ce qui peut entraîner un surapprentissage et une faible généralisation à de nouveaux exemples. En outre, ils ne sont pas très adaptés pour traiter des données complexes telles que des images, des vidéos et des séquences de texte. C'est ici que l'apprentissage intervient, une branche de l'apprentissage automatique qui utilise des réseaux de neurones profonds pour apprendre des représentations de haut niveau des données [21].

Contrairement aux ANN classiques, les réseaux de neurones profonds contiennent plusieurs couches cachées, ce qui leur permet de modéliser des relations complexes entre les données. L'apprentissage profond a connu une croissance exponentielle ces dernières années grâce à l'augmentation de la puissance de calcul et à la disponibilité de grands ensembles de données. Il est maintenant utilisé dans de nombreux domaines, tels que la reconnaissance d'images, la reconnaissance de la parole, la traduction automatique et la conduite autonome, et il a permis de réaliser des progrès significatifs. L'apprentissage profond a ainsi révolutionné l'utilisation des réseaux de neurones et a ouvert la voie à de nombreuses applications innovantes dans différents secteurs.

1.9 Les modèles génératifs

Un modèle génératif est une classe de modèles d'apprentissage automatique qui a la capacité de générer de nouvelles données telles que des images, du texte ou du son à partir d'un ensemble

de données d'apprentissage. Contrairement aux modèles prédictifs, qui sont utilisés pour faire des prédictions sur des données existantes, les modèles génératifs peuvent créer de nouvelles données qui n'existent pas encore. Il existe plusieurs types de modèles génératifs, chacun ayant des caractéristiques et des avantages différents. Les modèles génératifs probabilistes utilisent des probabilités pour générer de nouvelles données. Ils apprennent à estimer la distribution de probabilité des données d'entraînement et utilisent cette distribution pour générer de nouvelles données. Les réseaux de neurones génératifs adversaires (GAN) sont basés sur une compétition entre deux réseaux de neurones, un générateur et un discriminateur. Le générateur crée de nouvelles données à partir d'un bruit aléatoire, tandis que le discriminateur tente de distinguer les données générées des données réelles. Les deux réseaux s'entraînent simultanément, ce qui permet au générateur de créer des données plus réalistes au fil du temps.

Les modèles de Markov cachés (HMM) sont basés sur des processus stochastiques qui utilisent des états cachés pour représenter les caractéristiques d'un ensemble de données. Les HMM peuvent être utilisés pour générer des séquences de données, telles que du texte ou de la parole [22].

Les modèles génératifs ont la capacité de créer des données similaires aux données d'apprentissage qu'ils ont reçues, ce qui en fait un outil très puissant pour la synthèse de données, car ils sont utilisés dans de nombreux domaines, tels que la génération automatique de texte, la synthèse d'images, la modélisation de la voix et la musique générative. Ils sont également utilisés pour la simulation et la prédiction dans les domaines scientifiques, tels que la météorologie et la physique [23].

1.10 Les réseaux antagonistes génératifs (GAN)

Les GAN ont été introduits pour la première fois par Ian Goodfellow [12] et ses collègues en 2014 dans l'article "Generative Adversarial Networks" publié dans la conférence NIPS. Depuis leur mise en place, les GAN ont été utilisés dans de nombreuses applications, tels que la génération d'images, la synthèse de voix, la traduction de langues et bien plus encore. Le GAN ou Réseau Antagoniste Génératif (Generative Adversarial Network en anglais) est une méthode de l'apprentissage automatique permettront de créer une cartographie qui transforme une distribution simple et connue, telle qu'une distribution gaussienne, en une distribution plus complexe et spécifique à un domaine de modèles donné. Le principe de base de cette méthode repose sur la confrontation de deux réseaux au sein d'un cadre de travail spécifique pour générer

des données qui ressemblent à des données réelles., Ces deux réseaux sont appelés générateur et discriminateur.

1.10.1 Principe des GAN

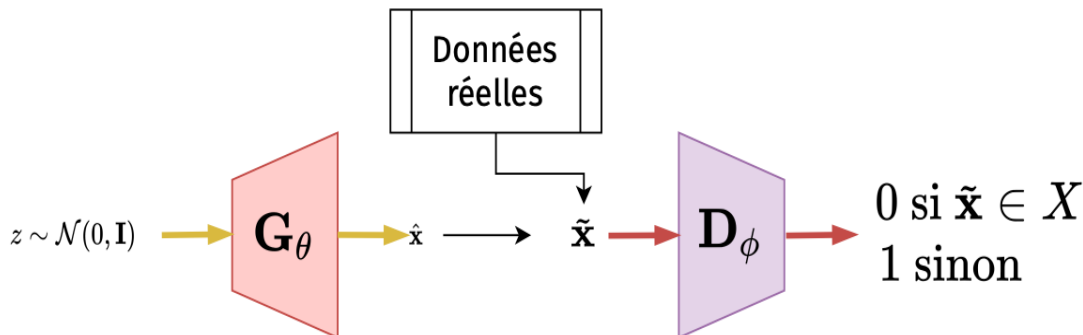


Figure 1-14- Architecture d'un Gan [12]

Cette technique met en compétition deux réseaux de neurones artificiels, le premier est le générateur G qui va génère des images synthétiques pour tromper le discriminateur D qui est le deuxième réseau de neurone dans cette architecture (Figure 11) et qui est chargé de différencier les images réelles des images générées. Le générateur G utilise une distribution connue $z \sim P_z$ pour produire une distribution de probabilité P_g correspondant aux échantillons générés $G(z)$. L'objectif de GAN est de permettre au générateur d'apprendre comment produire des échantillons qui sont près de la distribution de données réelles. Cette optimisation est réalisée en utilisant une fonction de perte conjointe pour le discriminateur et le générateur. [27]

$$\min_G \max_D \mathbb{E}_{x \sim p_r} \log[D(x)] + \mathbb{E}_{z \sim p_z} \log[1 - D(G(z))]. \quad (1)$$

1.11 Les différentes architectures des GANs

De nombreuses architectures de GAN ont été développées. Voici une description des architectures les plus importantes :

1.11.1 Le Conditionnel GAN (cGAN)

Un réseau contradictoire génératif conditionnel ou cGAN [24] ,est un réseau qui utilise un générateur (G) et un discriminateur (D) pour générer des images conditionnelles. Contrairement aux GAN classiques, il peut conditionner la génération d'images à partir d'étiquettes ou de vecteurs de conditionnement Y en plus du bruit de base Z. En plus du vecteur de bruit Z, pour générer une image $G(z,y)$, et le discriminateur D prend l'image réelle x et un

vecteur conditionnel y en entrée, et génère une image $G(z, y)$ et le vecteur conditionnel y , et évaluez leur authenticité en renvoyant une probabilité comprise entre 0 et 1 (vrai et faux), comme la montre dans la figure 15 :

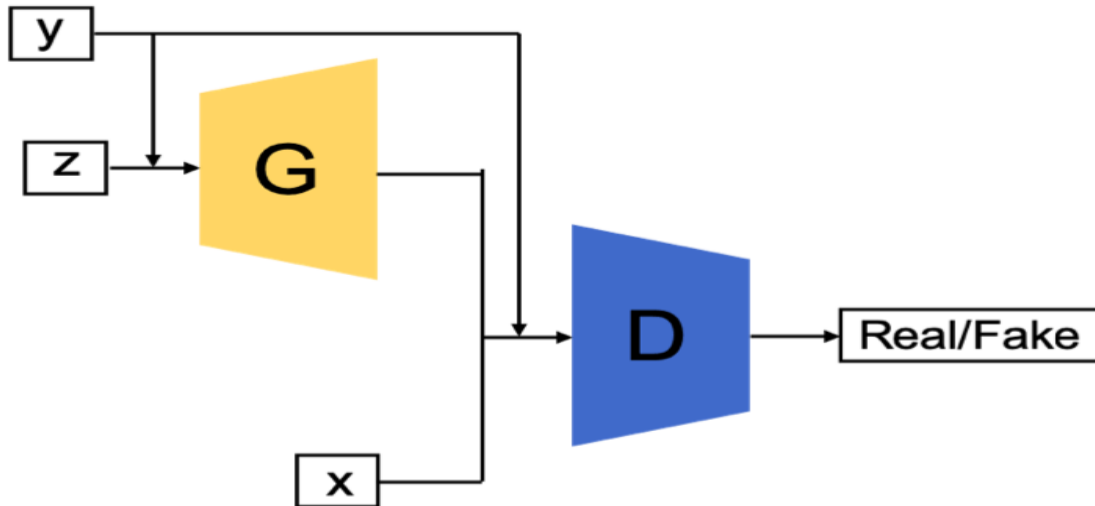


Figure 1-15-Architecture du cGAN [24]

La fonction objective d'un jeu minimax du cGAN est donnée par :

$$\min_G \max_D V(D, G) = \mathbb{E}_{x \sim p(x)} [\log D(x|y)] + \mathbb{E}_{z \sim p_z(z)} [\log(1 - D(G(z|y)))] \quad (2)$$

Où G est le générateur, D est le discriminateur, x est un échantillon de données réelles, z est un vecteur de bruit aléatoire, y est l'information conditionnelle (par exemple, une étiquette de classe), $p_{data}(x)$ est la distribution de données réelles, et $G(z|y)$ et $D(x|y)$ représentent les sorties du générateur et du discriminateur respectivement, conditionnées par y . L'objectif du générateur G est de minimiser $V(D, G)$ en générant des échantillons qui peuvent tromper le discriminateur D , tandis que l'objectif du discriminateur D est de maximiser $V(D, G)$ en distinguant correctement entre les échantillons réels et les échantillons générés par G .

Les cGAN (Conditional Generative Adversarial Networks) sont des modèles de réseaux de neurones profonds très puissants et utilisables dans nombreux domaines du machine learning tels que : La traduction d'image à image, la création d'images à partir de texte, la génération de vidéo, la génération de visages.

1.11.2 Progressive GAN (PROGAN) :

Le Progressive GAN [25], est une architecture de GAN proposée par Tero Karras, Timo Aila, Samuli Laine et Jaakko Lehtinen de NVIDIA Research en 2017. Cette architecture permet de générer des images haute résolution et de qualité supérieure en utilisant une méthode

progressive d'entraînement du générateur. Le principe fondamental du Progressive GAN consiste à faire évoluer de manière progressive le générateur et le discriminateur en partant d'une faible résolution et en ajoutant progressivement de nouvelles couches pour modéliser des détails de plus en plus fins au fur et à mesure de l'apprentissage. Cette méthode accélère l'apprentissage et stabilise considérablement le processus, permettant ainsi de générer des images d'une qualité exceptionnelle. Cette méthode progressive permet d'éviter les problèmes de stabilité et de convergence qui peuvent survenir lors de l'entraînement de GAN sur des images haute résolution, tout en permettant de générer des images de qualité supérieure avec des détails plus fins et plus précis.

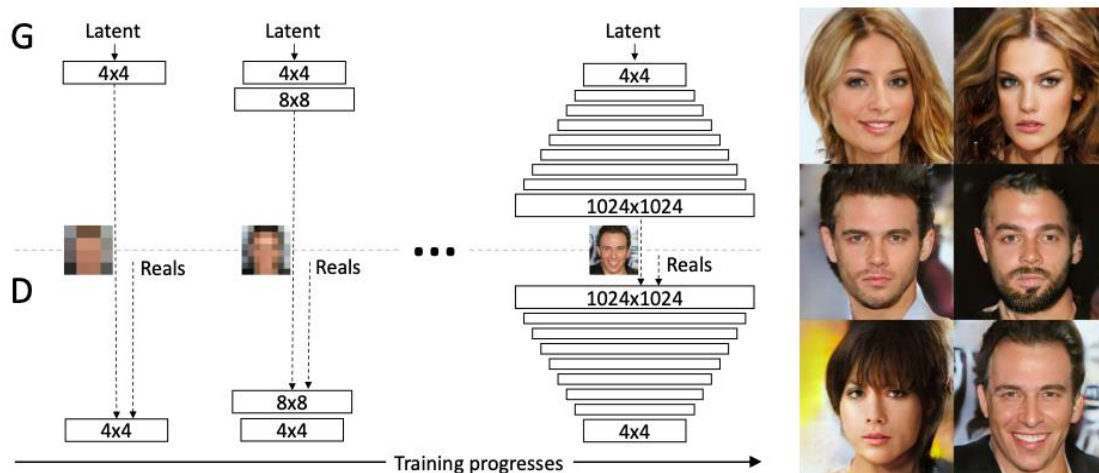


Figure 1-16- Architecture de ProGAN[25]

La figure 16 illustre comment le Progressive GAN utilise un modèle générateur et un modèle discriminant avec la même structure que les GAN traditionnels, mais commence avec des images de très petite taille, comme 4x4 pixels.

Progressive GAN a été employé avec succès pour produire des images photoréalistes à haute résolution dans une variété de domaines,

Dans ce qui suit (le prochain chapitre) on s'intéresse beaucoup plus par le type d'architecture StyleGAN parce qu'il est une addition à l'architecture GAN qui introduit des changements très importants dans le modèle du générateur.

1.12 Les auto-encodeurs (AE)

Un auto-encodeur [26] (ou « Auto-encoder » en anglais) est un type de réseau de neurones profonds qui est conçu pour apprendre à représenter une donnée d'entrée en utilisant une quantité réduite de données. En d'autres termes, il cherche à compresser la donnée en un format plus compact. Les auto-encodeurs sont couramment utilisés en apprentissage automatique pour des tâches telles que la compression de données, l'apprentissage de représentations et la détection de motifs.

1.12.1 Fonctionnement et Architecture des auto-encodeurs

Les auto-encodeurs sont des réseaux de neurones dont le nombre de neurones dans la couche d'entrée est identique à celui de la couche de sortie. Ils ont une architecture particulière appelée « architecture bottleneck », où les couches cachées sont plus petites que les couches d'entrée. Cette architecture peut être décomposée en trois parties distinctes comme le montre la figure 17 :

- **Encodeur** : Il est chargé de transformer l'entrée en une représentation dans un espace latent de dimension réduite. Cette opération permet de compresser l'entrée en une représentation plus économique.
- **Goulot d'étranglement (ou *bottleneck*)** : Cette partie du réseau stocke la représentation encodée de l'entrée, qui sera ensuite transmise au décodeur pour effectuer la reconstruction de l'entrée initiale. [27]
- **Décodeur** : Son rôle est de reconstruire une sortie plus proche à l'entrée à l'aide de la représentation latente de l'entrée

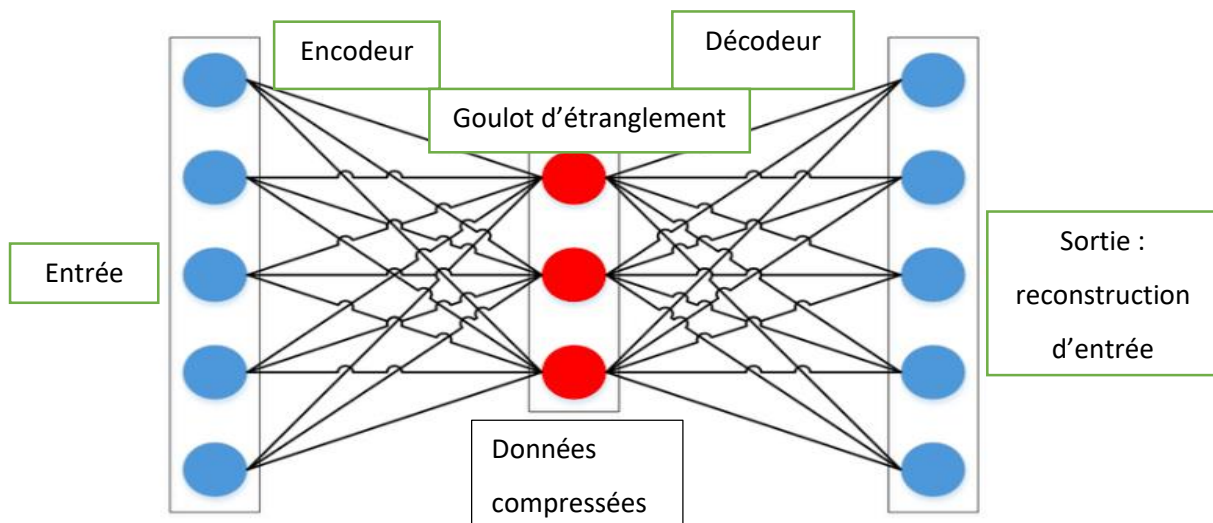


Figure 1-17- Architecture d'un auto-Encodeur [27]

L'objectif principal de l'auto-encodeur est d'obtenir une sortie similaire à l'entrée. Bien que l'architecture du décodeur puisse différer de celle de l'encodeur, dans la plupart des cas, elle est conçue pour être l'image miroir de l'encodeur. Il est important que la dimensionnalité de l'entrée et de la sortie soit la même. Les couches convolutives sont souvent utilisées pour l'encodeur et les couches déconvolutives pour le décodeur en pratique. [27]

1.12.2 Exemples d'auto-encodeurs

Il existe plusieurs exemples des auto-encodeurs, on peut citer :

- Auto-encodeurs variationnels (VAEs) [26]
- Les auto-encodeurs de débruitage
- Auto-encodeurs profonds
- Auto-encodeurs adversariaux (AAE)

1.13 Évaluation des performances :

Il existe plusieurs mesures de performances pour évaluer la qualité d'un modèle. Les mesures les plus couramment utilisées sont :

1.13.1 Matrice de confusion :

La matrice de confusion, également appelée tableau de contingence, est un outil couramment utilisé en apprentissage automatique pour évaluer les performances. Elle permet de mesurer la fréquence à laquelle les prédictions d'un modèle correspondent ou ne correspondent pas à la réalité en comparant les prédictions avec les étiquettes réelles. [28]

Pour construire une matrice de confusion, il est nécessaire de disposer d'un ensemble de données d'évaluation et de validation qui contient les prédictions du modèle ainsi que les valeurs réelles correspondantes.

Chaque colonne du tableau contient une classe prédite par le modèle et les lignes des classes réelles, comme le montre le tableau 1. Les quatre principales terminologies utilisées pour définir la matrice de confusion sont : TP, TN, FP FN

- **TP (true positive)** : la prédiction est positive, et la valeur réelle aussi est effectivement positive. Exemple : Une personne malade et prévu malade.
- **TN (true negative)** : la prédiction et la valeur réelle sont négatives.
- **FP (false positive)** : la prédiction est positive par contre la valeur réelle est négative. Exemple : Une personne saine et prévu malade.
- **FN (false negative)** : la prédiction est négative et la valeur réelle est négative aussi.

	Positive	Négative	
Positive	Vrai positive (TP)	Faux Négative (FN) Erreur de type 2	Sensitivité $\frac{TP}{(TP + FN)}$
Négative	Faux positive (FP) Erreur de type 1	Vrai Négative (TN)	Spécificité $\frac{TN}{(TN + FP)}$
	Précision $\frac{TP}{(TP + FP)}$	Valeur prédictive négative $\frac{TN}{(TN + FN)}$	Accuracy $\frac{TP + TN}{(TP + FP + FN)}$

Tableau 1- Matrice de confusion[28]

A partir de cette matrice on peut obtenir des informations pertinentes en utilisant plusieurs indicateurs appelés métriques.

Ces métriques permettent de mesurer les performances d'un modèle voici les plus couramment utilisées :

1.13.2 L'exactitude (Accuracy) :

C'est la mesure la plus simple, Elle représente le pourcentage de prédictions correctes faites par le modèle sur l'ensemble des prédictions, calculé comme suit :

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{N} \quad (3)$$

1.13.3 La précision (Precision) :

Mesure la proportion de vrais positifs parmi les éléments prédits comme positifs par le modèle. Elle est particulièrement utile dans les cas où les faux positifs sont coûteux, calculé comme suit :

$$\text{Précision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \quad (4)$$

1.13.4 Le rappel (Recall) :

Mesure la proportion de vrais positifs parmi les éléments qui étaient effectivement positifs. Cette mesure est particulièrement utile dans les cas où les faux négatifs sont coûteux, calculé comme suit :

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \quad (5)$$

1.13.5 L'aire sous la courbe ROC (AUC-ROC) :

Cette mesure permet d'évaluer la capacité du modèle à distinguer les classes en mesurant la surface sous la courbe ROC (Receiver Operating Characteristic). Cette courbe représente deux paramètres [29] :

- ✓ Taux de vrais positifs (TPR) : est l'équivalent du rappel.
- ✓ Taux de faux positifs (TFP) : est défini comme suit :

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \quad (6)$$

- ✓ La figure suivante montre une courbe ROC typique :

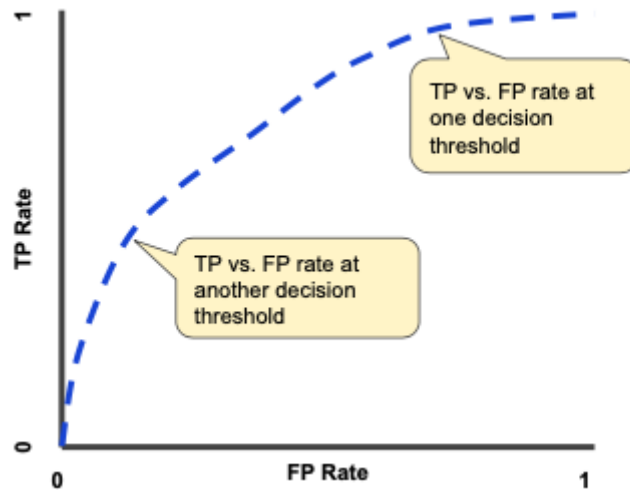


Figure 1-18- Taux de VP par rapport aux FP [29]

1.14 Conclusion

Les techniques d'apprentissage profond ont connu une croissance exponentielle ces dernières années, entraînant des progrès dans le domaine de l'intelligence artificielle (IA). Ces techniques se sont avérées particulièrement efficaces pour traiter des problèmes complexes tels que la reconnaissance d'images, la traduction automatique et la génération de texte. Certaines de ces techniques les plus couramment utilisées comprennent les réseaux de neurones convolutifs (CNN) pour la vision par ordinateur, les réseaux de neurones récurrents (RNN) pour le traitement du langage naturel, et les réseaux antagonistes génératifs (GAN) qui offrent des possibilités fascinantes pour générer des données réelles, augmenter les données, transformer les styles, améliorer les performances des modèles et comprendre les données. Leurs usages ne cessent d'évoluer et de se diversifier, et ils sont devenus des outils précieux dans de nombreux domaines.

2 Chapitre 2

StyleGAN

2.1 Introduction

Tout d'abord GAN est une architecture qui consiste en deux parties principales : un générateur et un discriminateur. Le générateur prend un bruit aléatoire en entrée et génère une image, tandis que le discriminateur évalue si l'image générée est réelle ou fausse. Le but de l'apprentissage est d'optimiser le générateur de sorte qu'il puisse tromper le discriminateur en lui faisant croire que l'image générée est réelle. StyleGAN est une architecture particulière de GAN qui utilise une approche modulaire pour générer des images avec une qualité et une résolution encore plus élevée.

StyleGAN est une architecture de réseau de neurones convolutifs utilisée pour générer des images de haute qualité. Elle a été développée par une équipe de chercheurs de NVIDIA en 2018, L'objectif est de générer des images d'un réalisme inégalé en utilisant des techniques d'apprentissage profond. [30]

2.2 Origine du StyleGAN

En décembre 2018, les chercheurs de Nvidia " Tero Karras, Samuli Laine et Timo Aila " ont publié un logiciel de pré-peinture accompagnée présentant StyleGan[31], un réseau adversarial génératif (GAN) permettant de produire des portraits souvent très convaincants de faux visages humains. Ce dernier était capable de fonctionner sur les processeurs GPU core de Nvidia. Peu après, en février 2019, Phillip Wang, ingénieur chez Uber, a utilisé ce modèle pour créer un site web qui génère un nouveau visage à chaque fois que la page est rechargée, donnant l'impression de vrais visages humains. La capacité de StyleGAN à produire des visages humains réalistes était considérable, étant donné que les humains ont évolué pour reconnaître et comprendre le domaine complexe des visages humains.

StyleGAN a été comparé aux "deepfakes" et suscite des inquiétudes quant à son utilisation à des fins malveillantes. Cependant, il est important de souligner que l'objectif principal de StyleGAN était de représenter une percée dans la génération d'images réalistes. Son utilisation pour créer des visages synthétiques est largement considérée comme une vitrine de ses capacités technologiques, plutôt qu'un outil conçu pour causer des dommages délibérés.

2.3 Architecture de StyleGan

L'objectif de StyleGAN est de générer des images haute résolution d'un réalisme inégalé en utilisant des techniques d'apprentissage profond. Pour y parvenir, l'architecture utilise une approche modulaire, en séparant la génération des images en deux parties principales : la génération des attributs de style et la génération des détails.

La génération des attributs de style se fait à l'aide d'un générateur de bruit aléatoire qui est ensuite transformé en un vecteur de style. Ce vecteur de style est ensuite utilisé pour contrôler les attributs de l'image, tels que la couleur, la forme, la texture, etc.

La génération des détails se fait à l'aide d'un réseau de neurones convolutifs qui utilise le vecteur de style pour générer les détails de l'image. Ces détails sont ensuite fusionnés avec les attributs de style pour produire l'image finale. [30]

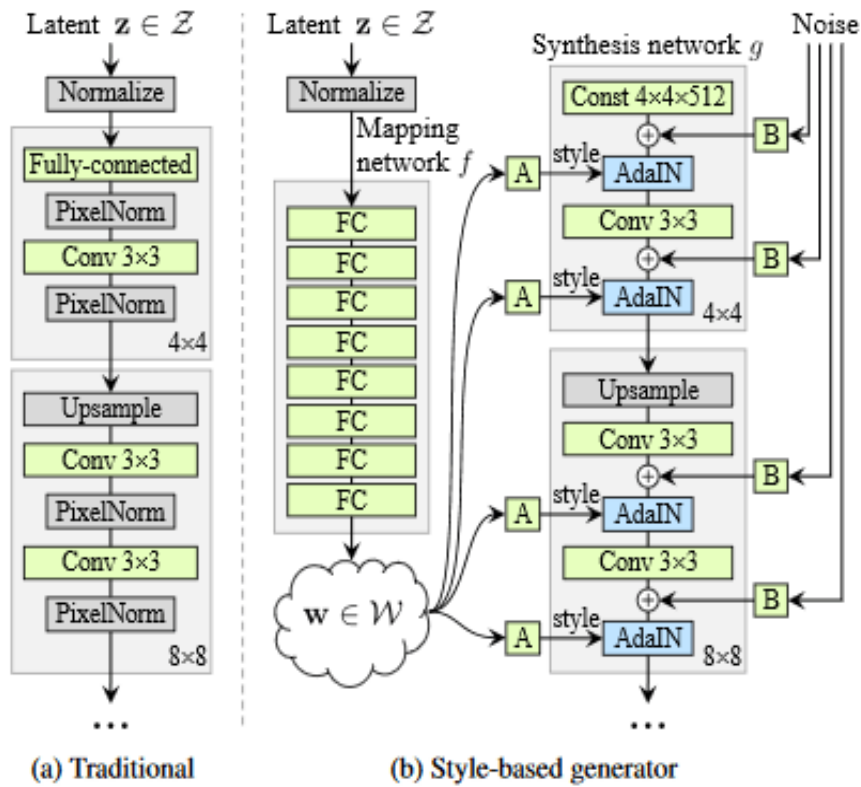


Figure 2-1- Comparaison entre les GAN traditionnels et StyleGAN [30]

Dans cette architecture, les vecteurs latent Z dans le réseau traditionnel passent par une phase de normalisation ensuite ils passent directement dans le bloc de traitement par contre dans le réseau StyleGAN le générateur est basé sur une cascade de réseaux de neurones, chacun d'eux contrôlant des aspects différents de l'image générée.

La première partie de l'architecture de StyleGAN appelée "The Mapping Network" (Réseau de cartographie) est un réseau de neurones constitué de huit couches entièrement connectées, et qui va transformer un vecteur de bruit de dimension basse appelé vecteur latent Z en un vecteur de style de dimension plus élevée et plus représentatif contient plus de facteurs latents indépendants W . Ce dernier introduit dans le module de transformation affine (A) afin de l'utiliser pour alimenter le réseau de Synthèse G. [31] [32]

La transformation affine A et la Normalisation adaptative des instances (AdaIN) : Adaptive Instance Normalization (AdaIN) est une technique qui est dérivé de l'algorithme de normalisation par lots et est une extension de la normalisation d'instance, définie comme suit [39] :

$$AdaIN(x, y) = \sigma(y) \left(\frac{x - \mu(x)}{\sigma(x)} \right) + \mu(y) \quad (7)$$

La

deuxième

partie de l'architecture de StyleGAN est Synthesis Network G, qui est un réseau de neurones convolutionnel profond qui utilise la méta-information stylistique pour générer des images en haute résolution. Il commence par un tenseur constant à faible résolution $4 \times 4 \times 512$ et augmente la résolution par bloc de calcul à un rythme standard de 4×4 à 1024×1024 (9 résolutions au total) [32].

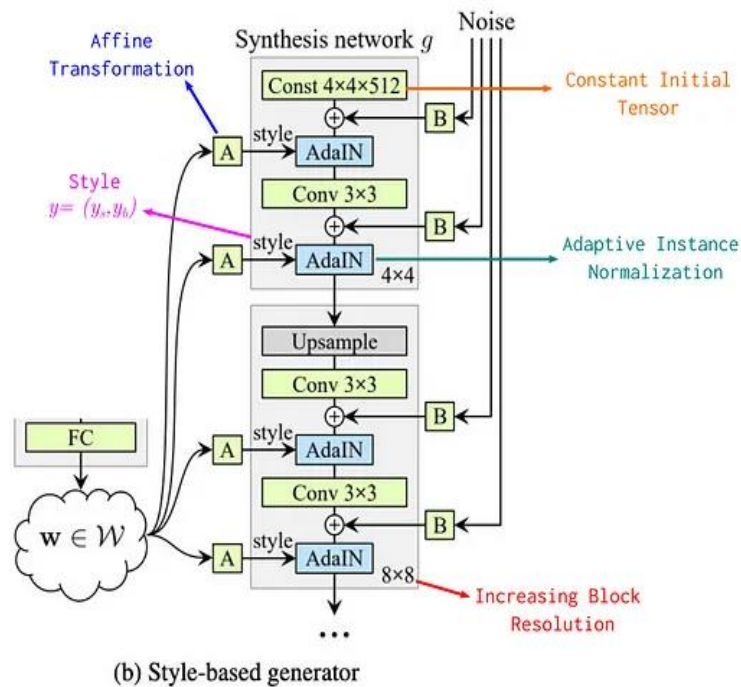


Figure 2-2-Modèle de génération du StyleGAN [30]

La sortie W est introduite dans différentes couches du réseau de synthèse et résolu par la transformation affine (A) et la normalisation adaptative de l'instance (AdaIN). Grâce à la transformation par couche dans StyleGAN, il est possible d'intégrer des informations de style aux données spatiales à chaque niveau du réseau, ce qui n'est pas le cas avec un code latent conventionnel qui ne s'applique qu'à la première couche. De cette manière, le rôle du style dans la génération de l'image est considérablement renforcé, ce qui permet un contrôle fin sur les attributs de l'image tels que l'orientation, la couleur et la texture, et d'obtenir des images de haute qualité, plus variées et uniques comme le montre dans la figure 21. [30,32]

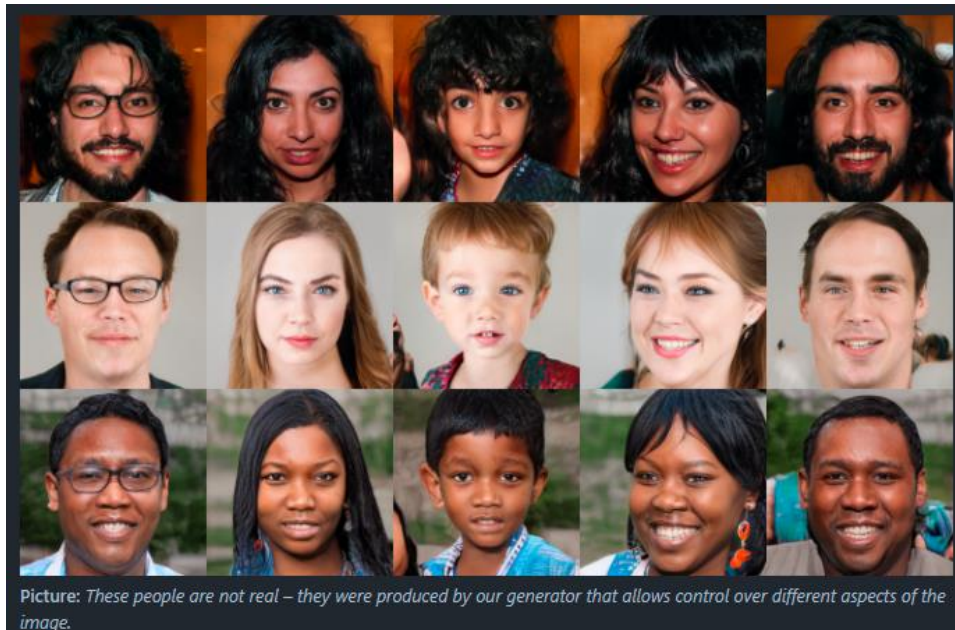


Figure 2-3- Exemple d'images de visages générés par StyleGAN [30]

2.4 Versions ultérieures de StyleGAN

Les chercheurs de Nvidia n'ont pas arrêté d'améliorer le réseau neuronal génératif StyleGAN, tels qu'il existe plusieurs versions améliorées de StyleGAN comme : StyleGAN2 et StyleGAN2-ADA, StyleGAN3.

2.4.1 StyleGan2

StyleGAN2 [33] est une version améliorée de StyleGAN, publiée par les mêmes auteurs en 2019, par rapport à son prédécesseur, il utilise plusieurs techniques avancées pour améliorer les performances de génération d'images, la qualité et la diversité des images générées, et des styles plus diversifiés. L'une des principales améliorations apportées à StyleGAN2 est l'utilisation d'architectures de réseaux de neurones cartographiés plus profonds et plus complexes. Ces réseaux de neurones "cartographiques" permettent une meilleure séparation des attributs de l'image, c'est-à-dire un contrôle plus fin et plus précis de la génération des images, ainsi qu'une meilleure séparation des caractéristiques de bas niveau des caractéristiques de haut niveau [33]

De plus, StyleGAN2 utilise une technique appelée augmentation dynamique des données illustrée dans la figure 22 pour augmenter la quantité de données d'entraînement en modifiant légèrement les images existantes à chaque itération d'entraînement. Cette technique permet d'entraîner des réseaux de neurones sur de plus grandes quantités de données.

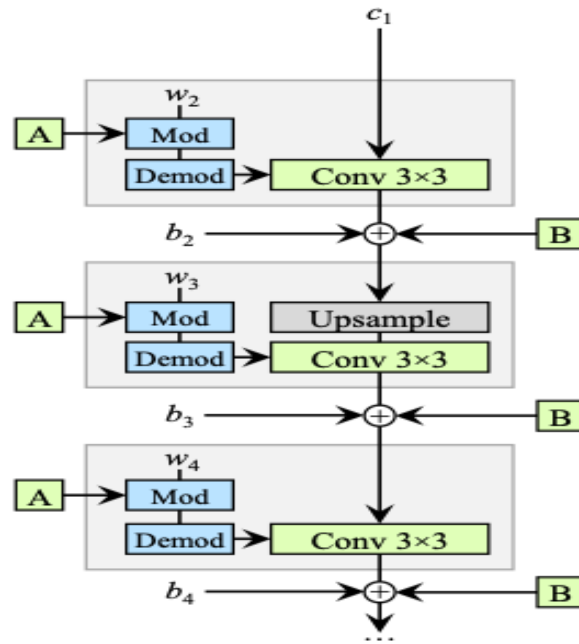


Figure 2-4-Le changement effectué dans l'architecture originale de StyleGAN[33]

Enfin, StyleGAN2 contrôle mieux les variations aléatoires en utilisant des vecteurs de bruit de contrôle supplémentaires qui peuvent contrôler la variation de certaines caractéristiques de l'image, telles que l'éclairage ou le flou. Cela améliore la diversité des images générées tout en maintenant une qualité élevée. Il a été utilisé pour créer des images de paysages, d'objets et de visages, et pour la composition vidéo.

2.4.2 StyleGAN2-ADA

StyleGAN2-ADA [34] est une version améliorée de StyleGAN2, développée par les mêmes auteurs (T. Karras, S. Laine et T. Aila) et publiée en 2020. L'objectif de StyleGAN2-ADA était de rendre la génération d'images réalistes plus accessible, même avec des données limitées, en utilisant un mécanisme adaptatif pour la normalisation au niveau de l'instance, appelé "Adaptive Instance Normalization" (AdaIN), qui peut mieux contrôler la variabilité de la génération d'images. Ce mécanisme permet de modifier la moyenne et l'écart type de chaque couche en sortie du générateur en fonction du vecteur de style fourni en entrée comme montré dans la figure 23. Cette technique permet d'obtenir des résultats nettement meilleurs pour les ensembles de données comportant moins de 30 000 images d'entraînement et prend également en charge la précision mixte, ce qui peut se traduire par un entraînement environ 1,6 fois plus rapide et des résultats environ 1,3 fois plus performants. En plus de cette amélioration, StyleGAN2-ADA intègre également des méthodes d'optimisation plus avancées pour améliorer

la stabilité de l'apprentissage et de la génération d'images, ainsi que des techniques d'augmentation de données pour améliorer la qualité et la diversité des images générées. [34]

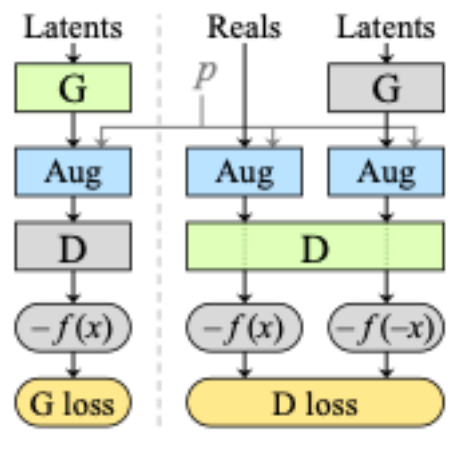


Figure 2-5-Architecture de StyleGan2-ADA[34]

2.4.3 StyleGan3

StyleGan3 [35] est un développement récent dans le domaine des réseaux adversaires génératifs (GANs), c'est la dernière version de la famille des réseaux de neurones génératifs StyleGAN, développée par NVIDIA en 2021.

StyleGan3 [35] est une amélioration de StyleGAN2-ADA, il vise à résoudre un problème rencontré dans StyleGAN2 appelé "texture collante", qui se manifeste lors des transitions de morphing d'une image à une autre. En clair, lorsque l'on utilise StyleGAN2 pour animer une transition entre deux visages par exemple, les textures peuvent sembler figées ou incohérentes. StyleGAN3 cherche à corriger ce problème en rendant les transitions plus naturelles et fluides, pour ce faire il propose des fonctionnalités supplémentaires montrés dans la figure 26 telles que :

- Une architecture de réseau de neurones plus avancée : L'architecture du générateur StyleGAN3 et les configurations d'apprentissage (StyleGAN3-T, StyleGAN3-R) utilisent une conception sans alias qui permet d'améliorer la qualité de l'image et la stabilité de l'apprentissage [35].
- Le modèle StyleGAN3-R présente des rotations de haute précision, ce qui permet une synthèse d'image plus précise à des angles spécifiques [24].

- Des modèles pré-entraînés pour plusieurs jeux de données, y compris FFHQ, AFHQv2, et MetFaces. Malgré leur nature convolutive hiérarchique, le processus de synthèse de StyleGAN3 est très efficace et produit des images de haute qualité [25]
- Un nouvel algorithme d'optimisation appelé « pathlength regularization » : permet de générer des images plus cohérentes et plus naturelles.
- StyleGAN3 permet la visualisation interactive et la manipulation des images générées.
- Une amélioration de la stabilité de l'apprentissage du réseau.[35]

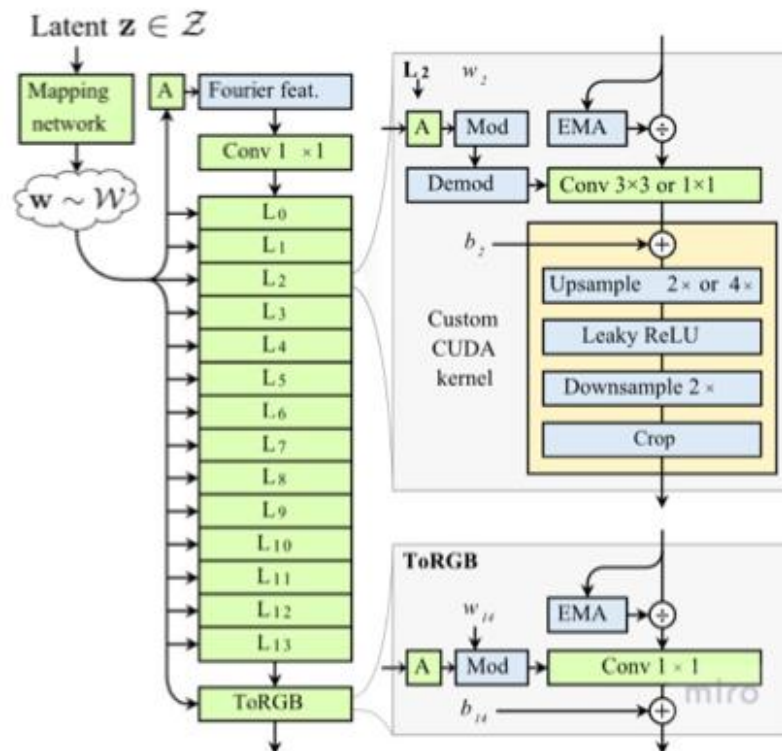


Figure 2-6-Architecture de StyleGAN3[36]

2.5 Définition de l'Espace latent

Un espace latent également connu sous le nom d'espace de caractéristiques latentes ou d'espace d'intégration, en générale est un espace abstrait multidimensionnel qui permet de représenter de manière significative les événements observés dans le monde extérieur (figure2-7). Ces derniers sont regroupés dans l'espace latent en fonction de leur similarité.

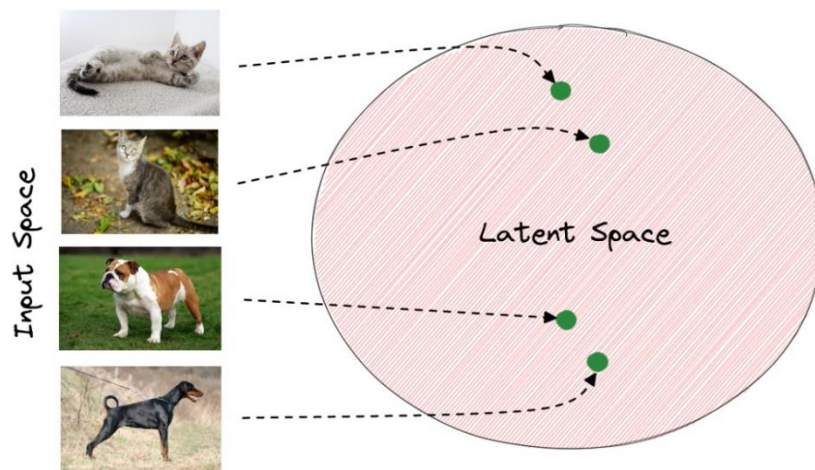


Figure 2-7 – Exemple d’espace latent regroupe des animaux [33]

De manière similaire, En Informatique l’espace latent cherche à fournir à un ordinateur une représentation comprimée du monde, en lui offrant une compréhension spatiale de ce dernier.

L'espace latent est un concept qui est souvent utilisé en intelligence artificielle, notamment dans le domaine de la génération d’images, Il fait référence à un espace abstrait et multidimensionnel qui contient des valeurs de caractéristiques que nous ne pouvons pas interpréter directement, mais qui encode une représentation interne significative d'un ensemble d'observations. La motivation pour apprendre un espace latent (ensemble de sujets cachés/représentations internes) sur des données observées (ensemble d'événements) est que de grandes différences dans les espaces/événements observés peuvent être dues à de petits changements dans l'espace latent (pour le même sujet). Par conséquent, l'entraînement de l'espace latent aidera le modèle à mieux comprendre les données observées que les données observées elles-mêmes, qui constituent un très grand espace d'entraînement.

Dans le cas de la génération d’images à l’aide de modèles génératifs tels que les GAN ou les auto-encodeurs, l'espace latent est utilisé pour encoder des informations sur les caractéristiques visuelles de l’image, telles que la couleur, la texture, le motif, etc. Il est représenté comme l'espace vectoriel de nombres aléatoires d'entrée utilisé pour générer l’image.

En ajustant les vecteurs de l'espace latent, il est possible d'explorer différentes régions de l'espace de représentation et de générer des images qui présentent des changements plus ou moins subtils. [33]

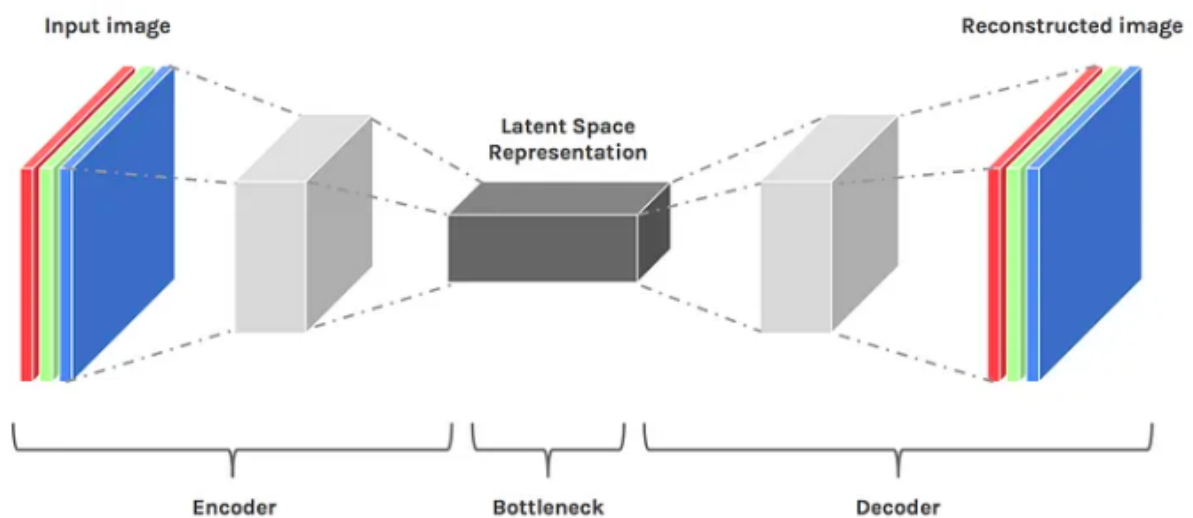


Figure 2-8- Représentation de l'espace latent dans le réseau de neurone convolutif [37]

2.6 Importance d'espace latent dans l'apprentissage profond

L'objectif principal de l'apprentissage profond est de prendre des données brutes, telles que les valeurs de pixels d'une image, et de les transformer en une représentation interne ou un vecteur de caractéristiques adapté. À partir de cette représentation, le sous-système d'apprentissage, souvent un classificateur, peut détecter ou classer des modèles dans les données d'entrée. Il est intéressant de noter que l'apprentissage profond et l'espace latent sont des concepts étroitement liés.

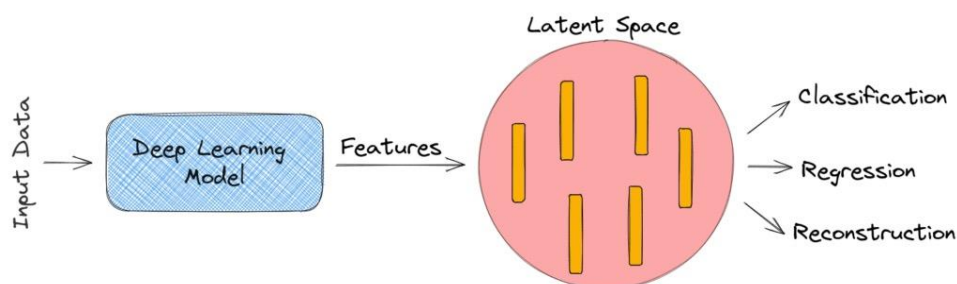


Figure 2-9- La relation entre l'apprentissage profonde et l'espace latent [38]

En fait, les représentations internes générées par l'apprentissage profond constituent souvent les représentations utilisées dans l'espace latent. Cela dit, l'apprentissage en profondeur peut être utilisé pour transformer les données en une représentation adaptée à un espace latent, qui peut ensuite être utilisée pour la génération d'images et d'autres tâches. [38].

2.7 Exemples d'espaces latents

Les espaces latents sont généralement uniques à chaque modèle d'apprentissage profond. Il existe plusieurs exemples d'apprentissage profond où la présence d'un espace latent est essentielle pour comprendre la complexité de la tâche et obtenir des performances de pointe. Dans ce suit nous décrivons quelques espaces latents utilisés dans différents modules d'apprentissage profond

2.7.1 Espace des caractéristiques de l'image

L'espace latent permet au modèle de réaliser sa tâche (comme la classification) en utilisant des caractéristiques discriminantes à faible dimension plutôt que les pixels bruts de haute dimension. La figure suivante montre l'architecture générale d'un CNN :

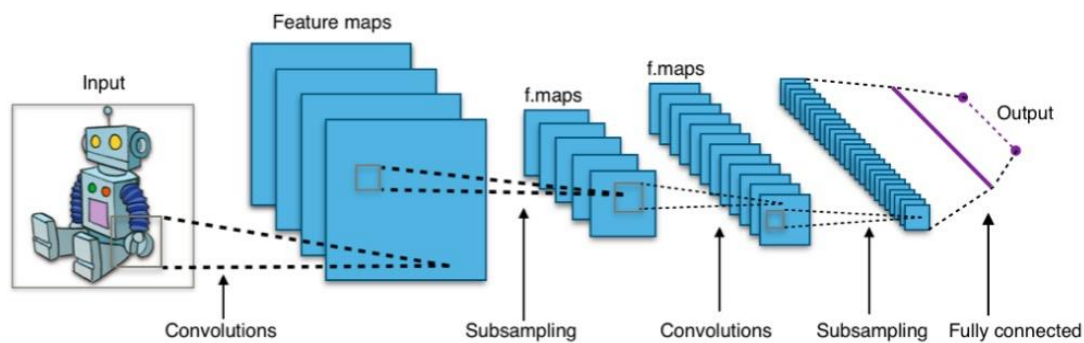


Figure 2-10- Architecture du CNN [16]

Après l'entraînement, la dernière couche du modèle capture les modèles importants d'entrée requis pour la tâche de classification d'images. Dans l'espace latent, les images représentant le même objet ont des représentations très similaires. En général, la distance entre les vecteurs dans l'espace latent correspond à la similarité sémantique de l'image originale.

2.7.2 L'espace latent de Style Gan :

StyleGAN [30] dispose de deux espaces latents sont : L'espace Z qui est l'espace latent initial, et W qui est l'espace latent intermédiaire, il est défini comme un espace de vecteurs de nombres aléatoires, qui est utilisé comme entrée pour le générateur de Style GAN afin de synthétiser des images réalistes et haute résolution. Le vecteur latent initial Z est un vecteur de dimension 512 qui est échantillonné à partir d'une distribution normale standard. Ce qui signifie que chaque élément du vecteur est choisi aléatoirement avec une probabilité égale. Cela permet de générer des images différentes à chaque fois que le générateur est utilisé. Les vecteurs intermédiaires W sont également de dimension 512 sont obtenus en passant le vecteur latent initial Z à travers une série de couches entièrement connectées. Comme nous allons mentionner précédemment

Chaque couche transforme le vecteur en un nouveau vecteur de même dimension en appliquant une transformation linéaire suivie d'une fonction d'activation non linéaire (ReLU). Les couches sont conçues pour capturer des caractéristiques à différentes échelles spatiales dans l'image générée. Ils sont utilisés pour contrôler les différentes échelles spatiales dans l'image générée [30]

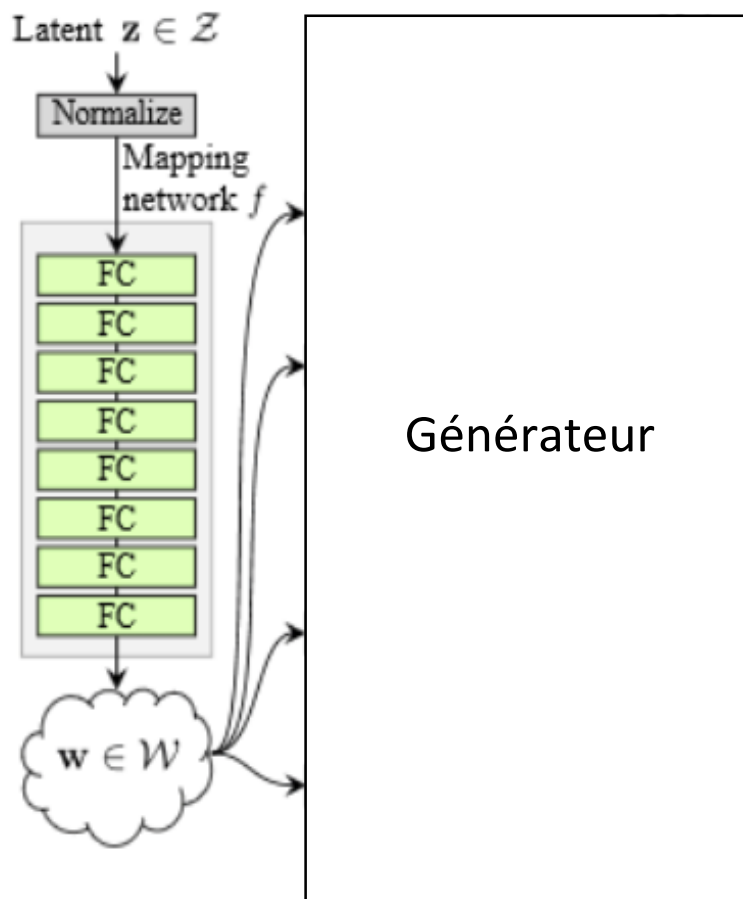


Figure 2-11- Mapping network de StyleGAN [3]

Ce qui distingue l'espace latent de Style GAN des autres espaces latents utilisés dans les modèles de génération d'images, c'est sa capacité à permettre des opérations sémantiques sur les images générées [39]. En d'autres termes, cela signifie que des modifications apportées aux vecteurs de l'espace latent peuvent avoir un effet cohérent et significatif sur les images générées. Par exemple, en ajustant les valeurs de certains éléments du vecteur latent, il est possible de modifier des attributs tels que l'âge, la couleur des cheveux, l'expression faciale, et bien d'autres encore. Cette propriété de l'espace latent de StyleGAN est particulièrement utile pour la synthèse de visages réalistes à des fins de jeux vidéo, de films, de publicités, et d'autres applications similaires.

L'espace latent de StyleGAN est un outil puissant qui permet de contrôler les caractéristiques des images générées avec une grande précision.

2.7.3 L'espace latent de l'auto-encodeur Variational (VAE)

Un Variational Autoencoder (VAE) est un modèle basé sur des réseaux neuronaux qui permet d'apprendre une représentation compressée des données qui se situe entre l'encodeur et le décodeur, appelée Espace latent. Cette représentation capture les caractéristiques les plus importantes des données afin de réduire la quantité d'informations échangées entre ces deux composants en éliminant le bruit et en ne conservant que les informations pertinentes et importantes. Et permet ainsi de réduire leur complexité tout en conservant leur information essentielle. [40]

2.7.4 L'espace latent de Cycle GAN :

Le CycleGAN est un type de réseau de neurones génératif qui permet de transformer des images d'un domaine en images d'un autre domaine sans avoir besoin de paires d'images correspondantes pour l'apprentissage. L'espace latent dans le modèle Cycle GAN correspond aux caractéristiques latentes ou aux représentations abstraites des images dans chaque domaine. C'est un espace de réduction de dimensionnalité où les images sont encodées avant transformation puis décodées pour reconstruire l'image dans un autre domaine. L'espace latent dans le modèle CycleGAN joue un rôle crucial dans la capacité du modèle à apprendre les correspondances entre deux domaines d'image. En codant des images dans cet espace latent, le modèle peut extraire des caractéristiques et des représentations communes entre les images dans les deux domaines, conduisant à une traduction précise et de haute qualité. [41]

2.7.5 L'espace latent de GPT-2 (Generative Pre-trained Transformer 2) :

GPT-2 (Generative Pre-trained Transformer 2) est un modèle de langage naturel basé sur une architecture de réseau neuronal connue sous le nom de Transformer, Plus précisément GPT-2 utilise une architecture de transformateur génératif pré-entraîné qui met en œuvre un réseau neuronal profond et utilise des mécanismes d'attention pour générer du texte. L'espace latent de GPT-2 est défini par les poids des couches du réseau neuronal du modèle, qui produit des vecteurs de taille 768 en sortie de la dernière couche du modèle. Ces vecteurs peuvent être considérés comme des représentations sémantiques des textes d'entrée, dans le sens où ils capturent des informations sur le sens et le contenu des phrases. L'espace latent de GPT-2 peut donc être utilisé pour diverses tâches de traitement de texte, telles que la classification de textes, la génération de textes, la traduction de textes, etc. [42]

2.8 Les bases de données

Afin de former et d'évaluer des modèles génératifs profonds, des bases de données vastes et diverses doivent être utilisées. Voici quelques-unes des plus couramment utilisées dans ce domaine :

2.8.1 CelebA

CelebFaces Attributes Dataset (CelebA) [43] est un ensemble de données d'attributs faciaux à grande échelle contenant plus de 200 000 images de célébrités, chacune annotée avec 40 attributs. Les images de cet ensemble de données couvrent une grande variété de poses et d'arrière-plans encombrés. CelebA a les caractéristiques d'une grande diversité, d'une grande quantité et d'annotations riches, y compris :

- **10,177** number of **identities**,
- **202,599** number of **face images**, and
- **5 landmark locations**, **40 binary attributes** annotations per image.



Figure 2-12-Exemple d'image de celebA

2.8.2 Flickr-Faces FFHQ

Flickr-Faces-HQ (FFHQ) est une grande base de données qui contient 70 000 images de haute qualité avec une résolution de 1024x2 comme dans la (Fig. 27). Cet ensemble de données contient plus de variations que CelebA-HQ [28] en termes d'âge, de race et de fond d'image, et a également une meilleure couverture pour les accessoires tels que les lunettes, les lunettes de soleil, les chapeaux, etc. Les images sont extraites de Flickr.[35]



Figure 2-13-Exemple d'images de FFhq[30]

2.8.3 Des visages étiquetés dans la nature (LFW)

Cet ensemble de données a été créé par, Huang et al. Qui ont utilisé des photographies collectées dans le cadre du Berkeley Field Faces Project. Les photos ont été prises dans des environnements non contrôlés, avec différents réglages, expressions et flashes. En, Huang et al. ont nettoyé manuellement les données, conçu un nouveau protocole et publié un ensemble de données appelé Labeled Faces in the Wild. La base de données comprend 13 233 images faciales de 5 749 personnes. À des fins de comparaison, l'ensemble de données a été divisé en 10 sous-ensembles distincts de paires d'images pour la validation croisée. Chaque sous-ensemble contient 300 paires positives (deux images de la même personne) et 300 paires négatives (deux images de personnes différentes). Si la base de données n'est utilisée que pour les tests, toutes les paires sont utilisées pour obtenir des résultats de performance. Depuis lors, de nombreuses études ont été menées pour améliorer les performances de cette base de données.[44]

2.8.4 MegaFace

MegaFace [45] est une base de données volumineuse de visages destinée principalement à l'évaluation de systèmes de reconnaissance faciale à grande échelle. Elle est créée pour éprouver la capacité des systèmes de reconnaissance faciale à identifier et à vérifier les visages dans des conditions réelles.

Les bases de données mentionnées ci-dessus se concentrent spécifiquement sur les visages. Elles permettent d'entraîner des modèles génératifs profonds pour la détection et l'analyse des

visages, ce qui améliore leur capacité à générer des visages réalistes et à effectuer des tâches complexes, telles que la détection des expressions faciales, l'estimation de l'âge ou l'identification des attributs faciaux.

2.9 Travaux connexes

2.9.1 Article 1

EmbedGAN: A Method to Embed Images in GAN Latent Space est un article de recherche publié par (**Zhijia Chen, Weixin Huang et Ziniu Luo**) . 2019

L'article [49], propose une méthode appelée EmbedGAN pour incorporer des images dans l'espace latent d'un réseau génératif antagoniste (GAN). En utilisant un réseau de neurones appelé "Embedded". Cette méthode permet de projeter une image donnée dans l'espace latent du GAN en optimisant les poids du GAN pour qu'il produise une sortie proche de l'image d'entrée, Plus précisément, EmbedGAN utilise une combinaison de deux réseaux GAN, l'un pour la génération et l'autre pour la reconnaissance d'images, ainsi qu'une fonction de perte pour évaluer la distance entre l'image d'entrée et l'image générée par le GAN. La méthode proposée consiste à entraîner le réseau Embedded qui prend en entrée une image et produit un vecteur latent qui représente cette image dans l'espace latent du GAN. Ce dernier est ensuite utilisé comme entrée pour le générateur du GAN afin de générer une nouvelle image similaire à l'image d'origine. Cette méthode est basée sur la minimisation d'une fonction de coût qui mesure la différence entre les images générées par le générateur à partir des vecteurs latents produits par le Embedded et les images d'origine. Cette fonction de coût est optimisée en utilisant la descente de gradient stochastique.

Les auteurs ont réalisé des expériences pour évaluer la qualité des images générées par cette méthode et ont comparé les résultats avec d'autres méthodes existantes. Les résultats montrent que la méthode proposée est efficace pour incorporer des images dans l'espace latent d'un GAN et générer des images de haute qualité.

L'utilisation d'EmbedGAN peut être utile pour plusieurs applications, telles que la génération d'images à partir d'une description textuelle ou la modification d'images existantes en ajustant leur vecteur latent dans l'espace du GAN.

2.9.2 Article 2

"KE-GAN: Knowledge Embedded Generative Adversarial Networks for Semi-Supervised Scene Parsing" réalisé par (**Qi, M., Wang, Y., Qin, J., & Li, A.** 2018)

C'est un article[47], propose une méthode pour la segmentation de scènes semi-supervisée en utilisant un réseau de neurones génératif antagoniste (GAN) avec des connaissances incorporées. Plus précisément, Les auteurs ont proposé d'utiliser un GAN pour générer des images de scènes à partir d'un espace latent. Les images générées sont ensuite utilisées pour améliorer la segmentation de scènes semi-supervisée en fournissant des exemples supplémentaires pour l'apprentissage. La méthode proposée utilise également des connaissances préalables sur les caractéristiques des scènes pour guider le processus de génération d'images. Ces connaissances sont incorporées dans le GAN sous forme de vecteurs d'attributs qui sont utilisés comme entrée supplémentaire pour le générateur.

La méthode KE-GAN a été testée sur différents jeux de données pour évaluer la qualité de la segmentation de scènes obtenue avec cette méthode et compare les résultats avec d'autres méthodes existantes. Les résultats montrent que la méthode proposée est efficace pour améliorer la segmentation de scènes semi-supervisée en utilisant des images générées par un GAN avec des connaissances incorporées.

2.9.3 Article 3

“Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space?” réalisé par (**Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, et Peter Wonka. (2019, October)**).

L'article [48], présente une méthode efficace appelée Image2StyleGAN qui permet d'intégrer des images dans l'espace latent de StyleGAN. Un réseau de neurones génératif qui est capable de générer des images réalistes de haute qualité, Cette méthode permet d'obtenir des résultats similaires à la génération d'images à partir de vecteurs de bruit aléatoire, mais avec une plus grande flexibilité et un contrôle accru sur le résultat fin et qui permet aussi d'effectuer des opérations de modification sémantique d'images telles que le morphing, le transfert de style et le transfert d'expression sur des photographies existantes. La méthode utilisée comporte plusieurs étapes :

Tout d'abord les auteurs ont utilisé le réseau de neurones StyleGAN entraîné sur l'ensemble de données FFHQ comme exemple pour montrer les résultats de leur algorithme. Ensuite ils ont mentionné qu'Il y a deux approches pour insérer des images dans l'espace latent de StyleGAN. La première approche consiste à apprendre un encodeur qui mappe une image donnée dans l'espace latent. La seconde approche consiste à sélectionner un code latent initial aléatoire et à l'optimiser en utilisant la descente de gradient. Dans cet article, la seconde approche qui a été

choisie car elle est plus générale et stable. Plus précisément, l'image n'est pas projetée directement dans l'espace latent de StyleGAN. Au lieu de cela, l'algorithme d'optimisation utilisé dans cet article optimise un code latent initial aléatoire pour qu'il corresponde à l'image d'entrée. Cet algorithme utilise une fonction de perte pour mesurer la différence entre l'image d'entrée et l'image générée à partir du code latent optimisé. L'algorithme utilise ensuite la descente de gradient stochastique pour minimiser cette fonction de perte et optimiser le code latent initial. Une fois que le code latent a été optimisé, il peut être utilisé pour générer des images similaires à l'image d'entrée ou pour effectuer des opérations de modification sémantique sur cette image en modifiant le code latent. L'algorithme d'optimisation est exécuté pendant un certain nombre d'étapes (5000 étapes dans cet article), après quoi le code latent optimisé est obtenu. Ce code peut ensuite être utilisé pour générer des images similaires à l'image d'entrée ou pour effectuer des opérations de modification sémantique sur cette image.

La méthode utilisée dans cet article est basée sur la descente de gradient stochastique et vise à minimiser une fonction de perte qui mesure la différence entre l'image d'entrée et l'image générée par le code latent optimisé. Cette méthode permet d'insérer des images dans l'espace latent de StyleGAN et d'effectuer des opérations de modification sémantique sur ces images. En utilisant cette approche, des images peuvent être générées à partir de diverses sources, y compris de vraies photographies. Cela permet la création d'images personnalisées réalistes et un niveau de contrôle accru sur la génération d'images aléatoires à partir de vecteurs de bruit. La méthode a également des applications potentielles dans la génération d'images pour la réalité virtuelle et la conception assistée par ordinateur.

Un nouvel article a été publié par les mêmes auteurs intitulé "Image2StyleGAN++ : How to Edit the Embedded Images ?" considéré comme une extension de l'article précédent "Image2StyleGAN : How to Embed Images Into the StyleGAN Latent Space ?".

2.9.4 Article 4

"Image2StyleGAN++: How to Edit the Embedded Images?" réalisé par (Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, et Peter Wonka. 2020).

Cet article [49], propose une méthode pour éditer les images incorporées dans l'espace latent de StyleGAN. Plus précisément, l'article propose une méthode pour contrôler la modification sémantique des images intégrées en utilisant des vecteurs de style et des vecteurs d'attributs. Les vecteurs de style sont utilisés pour contrôler les aspects globaux de l'image, tels que la pose

et l'orientation, tandis que les vecteurs d'attributs sont utilisés pour contrôler les aspects locaux, tels que la couleur des cheveux ou la forme du nez.

La méthode proposée consiste à utiliser un réseau de neurones supplémentaire appelé Attribute Encoder Network (AEN) pour extraire les vecteurs d'attributs à partir d'une image donnée. Ces vecteurs d'attributs sont ensuite combinés avec un vecteur de style aléatoire pour produire une image modifiée.

L'article présente également une méthode pour contrôler la modification sémantique en utilisant des masques binaires qui indiquent quelles parties de l'image doivent être modifiées. Les masques binaires peuvent être créés manuellement ou automatiquement à partir d'une image source et d'une image cible. Il présente aussi des expériences pour évaluer la qualité de la méthode proposée et compare les résultats avec d'autres méthodes existantes. Les résultats montrent que la méthode proposée est efficace pour produire des images réalistes avec une modification sémantique précise.

2.9.5 Article 5

“Improved StyleGAN Embedding: where are the Good Latents?” réalisé par **(Peihao Zhu, Rameen Abdal, Yipeng Qin, John Femiani, et Peter Wonka 2021)**.

Cet article [50] est la troisième version révisée de l'article "Image2StyleGAN : How to Embed Images Into the StyleGAN Latent Space ?" publié par les mêmes auteurs en octobre 2021". Cette version mise à jour explore le problème de la recherche de vecteurs latents de haute qualité dans les modèles StyleGAN en utilisant une combinaison de techniques d'optimisation et de mesure de similarité. Pour ce faire, les auteurs de cet article utilisent une mesure de la qualité de l'image appelée "perte perceptuelle", qui calcule la différence entre les caractéristiques visuelles des images générées et celles des images de référence. Ils appliquent également une méthode d'optimisation itérative pour trouver des vecteurs latents qui minimisent cette perte perceptuelle, et présente en détail l'architecture du réseau utilisé pour l'optimisation des vecteurs latents, ainsi que les paramètres spécifiques utilisés dans l'expérimentation.

Les auteurs proposent aussi une méthode améliorée de navigation dans l'espace latent pour les modèles StyleGAN, permettant aux utilisateurs de trouver et de manipuler facilement les attributs visuels souhaités dans les images générées. Ils introduisent une nouvelle technique appelée "incorporations augmentées" qui améliore l'interprétabilité et le contrôle des images

générées par l'apprentissage de cartes spatiales latentes. L'article présente les détails des techniques d'ensemble augmenté, y compris l'architecture de réseau et le processus d'apprentissage. Il aborde également les paramètres d'évaluation utilisés pour mesurer la qualité des images générées et compare les résultats avec d'autres méthodes existantes. Dans l'ensemble.

Cet article se concentre sur la difficulté de trouver de bons vecteurs latents dans les modèles StyleGAN et propose une nouvelle approche pour améliorer la contrôlabilité et la qualité visuelle des images générées.

2.9.6 Article 6

"Latent Walking Techniques for Conditioning GAN-Generated Music," réaliser par (**L. Eisenbeiser**)

L'article [51] "Latent Walking Techniques for Conditioning GAN-Generated Music" de L. Eisenbeiser présente une méthode pour générer de la musique à l'aide d'un réseau GAN conditionnel en utilisant des techniques de "Latent Walking". Plus précisément, l'article propose une méthode pour naviguer dans l'espace latent du réseau GAN conditionnel pour générer des variations progressives de la musique générée.

La méthode utilise un réseau GAN conditionnel qui est entraîné à générer de la musique en fonction d'une condition donnée, telle que le genre musical ou le tempo. En utilisant les techniques de "Latent Walking", il est possible de créer des variations progressives dans la musique générée en naviguant dans l'espace latent du réseau GAN conditionnel. Les auteurs proposent également une méthode pour contrôler les variations en utilisant des vecteurs de contrôle qui sont ajoutés au vecteur latent. Enfin, l'article évalue la qualité de la musique générée en utilisant la méthode proposée en comparant les résultats avec ceux obtenus avec d'autres méthodes existantes telles que WaveNet et MIDI-VAE. Les résultats montrent que la méthode proposée est efficace pour produire de la musique réaliste avec une variation contrôlable, et elle a été évaluée expérimentalement et a montré qu'elle était efficace pour produire des résultats de haute qualité.

2.9.7 Article 7

" StyleSpace Analysis: Disentangled Controls for StyleGAN Image Generation " Réaliser par (**Zongze Wu Hebrew University, Dani Lischinski Hebrew University, Eli Shechtman Adobe Research**)

Dans cet article [52], Les auteurs se concentrent sur l'exploration de l'espace de style latent de StyleGAN2, pour la génération d'images l'enchevêtrement et comment trouver ces contrôles démêlés. Pour ce faire les auteurs proposent une nouvelle méthode appelée "StyleSpace Analysis" qui permet d'identifier les contrôles démêlés dans l'espace latent de StyleGAN2. Cette approche permet de dissocier les différents styles et attributs présents dans les images générées par StyleGAN, comme la couleur de la peau, la forme du visage ou la texture des cheveux. Elle permet ainsi aux utilisateurs de modifier séparément chaque facteur de variation, en obtenant des résultats cohérents et réalistes. Ils utilisent des modèles pré-entraînés sur plusieurs ensembles de données différents pour explorer l'espace latent du style et identifier ces contrôles. Les auteurs présentent plusieurs résultats qui démontrent comment ces contrôles peuvent être utilisés pour manipuler les attributs visuels d'images réelles. Ces résultats ouvrent la voie à des manipulations d'images sémantiquement significatives.

Cette méthode est considérée comme une contribution significative à la communauté de recherche en apprentissage profond, et très efficace pour la génération et la modification d'images à l'aide de GAN.

2.9.8 Article 8

"Designing an Encoder for StyleGAN Image Manipulation" rédigé par (Omer Tov, Yuval Alaluf, Yotam Nitzan, Or Patashnik et Daniel Cohen-Or, tous affiliés à l'Université de Tel-Aviv.)

Dans cet article [53], de recherche, les auteurs proposent une méthode pour manipuler les images générées par StyleGAN en utilisant un encodeur personnalisé. Les auteurs commencent par présenter StyleGAN.

Ils présentent une d'architecture d'encodeur qui est capable de transformer des images en un espace latent approprié pour StyleGAN. Cette architecture est utilisée pour manipuler les images générées par StyleGAN en modifiant le code latent. Cette méthode permet de modifier les caractéristiques des images, telles que la couleur des cheveux, la forme du visage ou l'expression faciale, sans perdre la cohérence globale de l'image.

Ils ont testé leur architecture sur plusieurs ensembles de données et ils ont comparé à d'autres méthodes de manipulation des images StyleGAN. Selon les auteurs leur méthode est plus performante en termes de qualité visuelle et de mesures quantitatives.

2.10 Conclusion

Les modèles StyleGAN sont des modèles profonds génératifs qui permettent de créer des images de haute qualité à partir d'un espace latent. Ce dernier est un vecteur de nombres aléatoires qui représente la variabilité des caractéristiques de l'image. L'importance de l'espace latent dans la génération d'images réside dans sa capacité à explorer et à manipuler plus facilement les caractéristiques de l'image. En ajustant les valeurs de l'espace latent, il est possible de modifier des aspects tels que la couleur, la forme, la texture et la complexité de l'image générée. En outre, les modèles StyleGAN offrent une certaine flexibilité en permettant aux utilisateurs de contrôler séparément le style et le contenu de l'image. Cette capacité permet de générer des images expressives et réalistes qui peuvent être utilisées dans diverses applications telles que les jeux vidéo, les films, l'art numérique et la mode.

3 Chapitre 3

Conception

3.1 Introduction

L'édition d'images peut être effectuée à différents niveaux de complexité et d'abstraction. Les opérations courantes consistent simplement à appliquer des filtres à une image, par exemple en augmentant son contraste ou en la convertissant en niveaux de gris. Ces opérations ne sont pas très complexes, et ne nécessitent pas une connaissance de la scène ou de l'objet représenté par l'image. En revanche, si l'objectif est de modifier les attributs d'un visage par exemple (comme ajouter un sourire, changer la couleur des cheveux ou encore le sexe), cette édition devient une modification plus compliquée et plus difficile à réaliser. Dans de tels cas, l'obtention de résultats réalistes nécessite généralement des solutions qui automatisent ces opérations importantes s'appuient sur des modèles génératifs.

Dans ce contexte, nous proposons dans ce travail, une méthode pour l'édition d'images basée sur l'apprentissage profond. La méthode proposée repose sur l'utilisation d'un réseau de neurone convolutif pour l'extraction des attributs d'abord, ensuite la génération et la construction d'un vecteur latent, qui sera enfin injecter aux module de synthèse d'image du styleGAN3[35].

3.2 Objectifs

Notre application a été conçue pour atteindre l'objectif suivant : à partir d'une image d'entrée donnée et de certains attributs cibles spécifiques, générer une image similaire correspondant à ces attributs cibles, à l'aide du modèle génératif profond StyleGAN3.

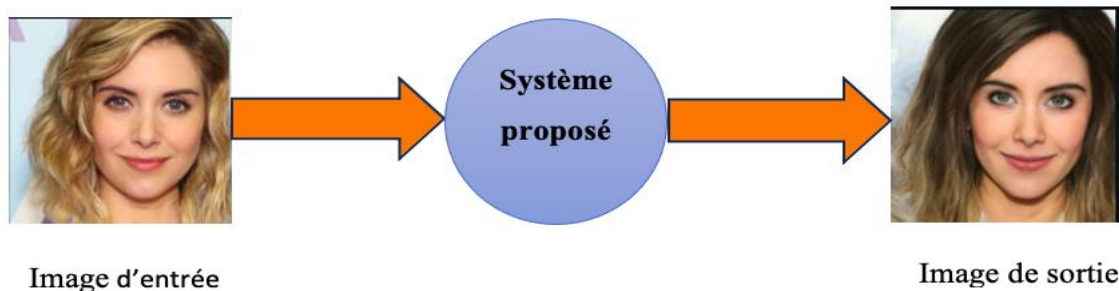


Figure 3-1-Objectif de l'application

Afin d'accomplir cela, nous avons proposé d'utiliser une approche en trois étapes principales sont :

3.2.1 Extraction des attributs de visage de l'image d'entrée

Pour extraire les caractéristiques significatives des visages, nous avons utilisé le réseau neuronal convolutionnel (CNN) MobileNetV2 [54]. Il s'agit d'un modèle pré-entraîné que nous formons sur la base de données CelebA pour extraire 40 valeurs d'attributs spécifiques sous la forme d'un vecteur Y .

Cette étape constitue la première partie de l'architecture de notre système, qui nous aidera à accomplir la phase suivante. Par la suite, de tous les attributs extraits, nous choisissons six attributs que l'utilisateur est en mesure de modifier à sa guise. Ainsi, notre système permet à l'utilisateur de personnaliser de manière sélective les attributs de l'image générée, c'est-à-dire de contrôler la génération de l'image cible.

3.2.2 Génération et construction du nouveau vecteur latent

La plupart des modèles de génération d'images utilisent le vecteur aléatoire Z comme entrée de leur générateur. Et comme nous l'avons discuté dans le chapitre 2 de ce mémoire, plusieurs recherches affirment que la manipulation de ce vecteur est plus efficace pour produire des images de meilleure qualité.

Nous proposons d'agir sur ce vecteur pour contrôler le générateur de StyleGAN3. La construction du nouveau vecteur latent est donc une phase intermédiaire cruciale dans cette architecture. Elle consiste à combiner le résultat obtenu dans l'étape précédente, qui est le vecteur d'attributs modifié Y' avec un vecteur Z générée aléatoirement pour construire le nouveau vecteur latent Z' , le vecteur résultant sera injecté dans le générateur StyleGAN3, cette combinaison est effectuée par une simple concaténation.

3.2.3 La génération d'images

Pour la génération d'images, nous considérons le modèle StyleGAN3 comme un réseau de génération de visages pré-entraîné. Le nouveau vecteur latent Z' construit dans l'étape précédente est utilisé comme entrée pour le générateur de StyleGAN3, que nous entraînons sur la base de données CelebA. Ainsi, durant cette dernière étape du processus, la manipulation préalable de l'espace latent permet de contrôler la génération des images.

3.3 Architecture générale du système proposé

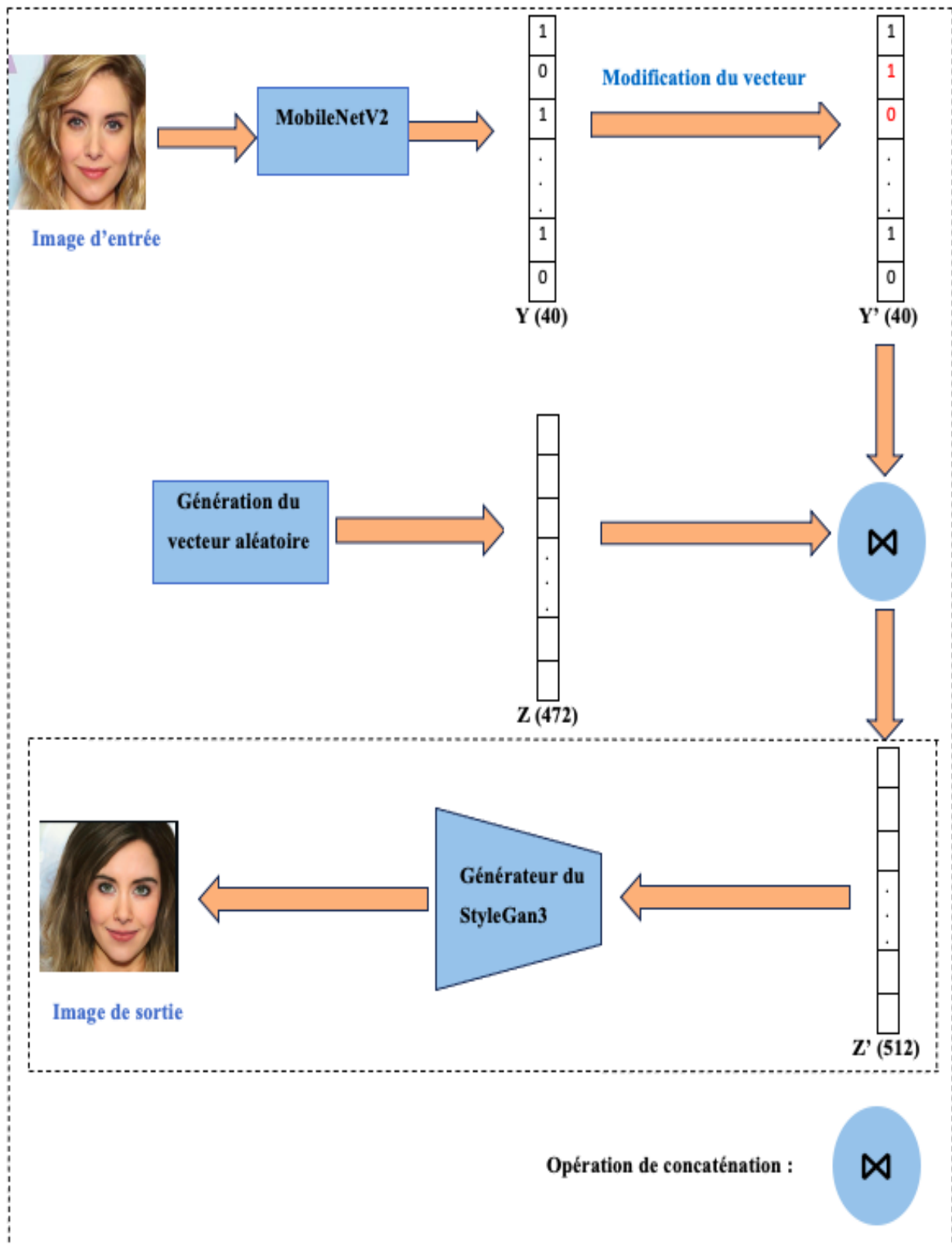


Figure 3-2- Architecture générale du système proposé

La figure 34 présente le schéma global de notre système proposé et décrit principalement les trois étapes mentionnées précédemment, notons que :

- Y est le vecteur d'attributs extrait
- Y' est le vecteur d'attributs modifié par l'utilisateur
- Z est le vecteur latent généré aléatoirement
- Z' est le nouveau vecteur latent construit

Dans ce qui suit, nous allons fournir une description plus détaillée de ces étapes :

3.4 Pré-traitement de l'image d'entrée

Pour assurer le bon fonctionnement du modèle MobileNetV2, qui est conçu pour traiter des images de 128x128 pixels avec 3 canaux (RGB), nous effectuons une étape de prétraitement et de redimensionnement des images. Pour ce faire, nous utilisons tout d'abord la fonction "get_frontal_face_detector" de la bibliothèque dlib, qui est basée sur les caractéristiques de Haar et peut être utilisée pour détecter des visages dans une image. Nous redimensionnons et normalisons ensuite chaque visage détecté pour qu'il corresponde aux spécifications requises par le modèle. Cette étape de prétraitement garantit que les images sont prêtes à être utilisées par ce modèle et offre des performances optimales lors de l'extraction des attributs de l'image d'entrée.

3.5 L'extraction des attributs par notre modèle CNN

Nous avons utilisé le réseau de neurone convolutif (CNN), MobileNetV2 [54], qui utilise le modèle pré-entraîné sur de grands ensembles de données « ImageNet », souvent utilisés pour des tâches de classification d'images. Nous l'avons entraîné sur la base de données CelebA pour extraire les caractéristiques significatives des visages, à savoir 40 valeurs d'attributs spécifiques sous forme d'un vecteur Y . La figure 35 illustre cette phase du système, l'extraction des attributs.

Avant de commencer l'extraction des attributs des images de la base de données CelebA par le modèle pré-entraîné MobileNetV2, nous avons tout d'abord analysé les données de cette base, où la figure 36 montre la distribution des attributs.

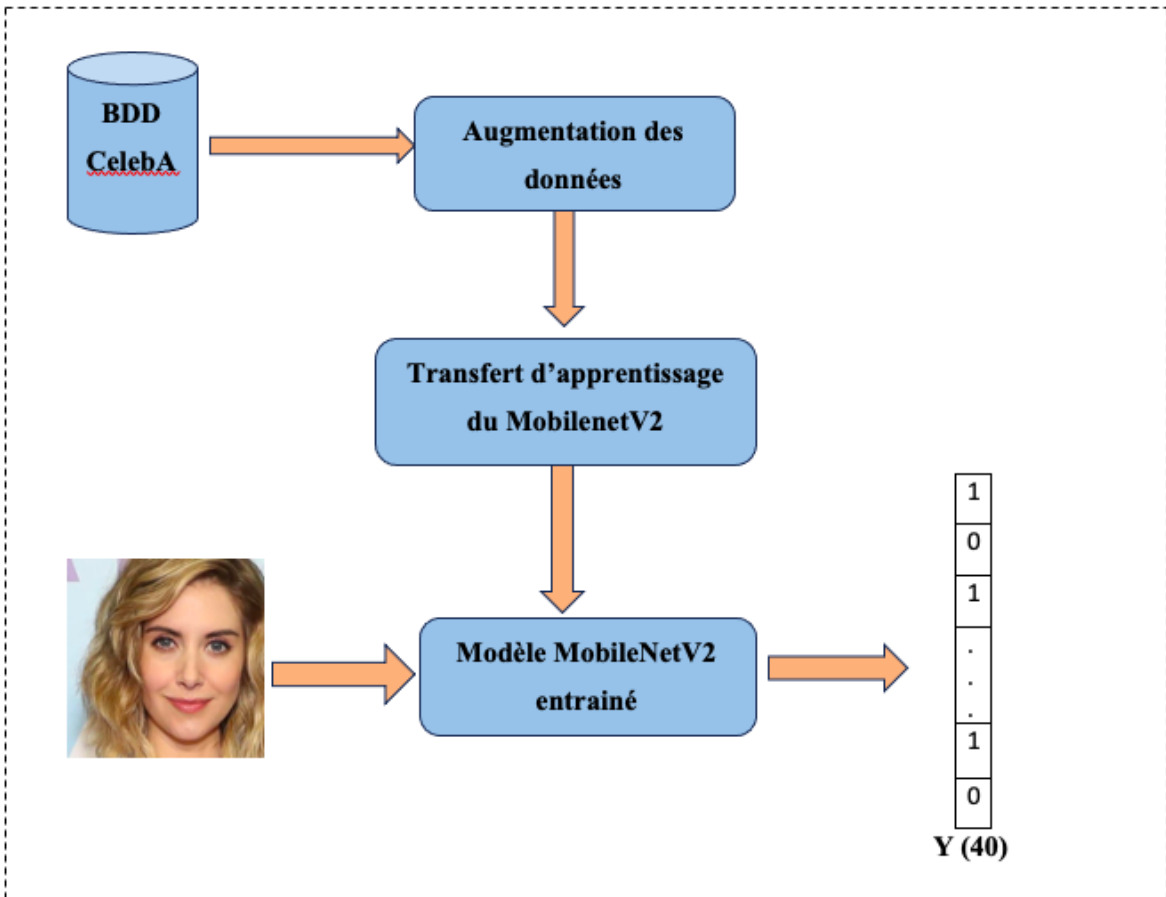


Figure 3-3- Schéma générale d'extraction des attributs

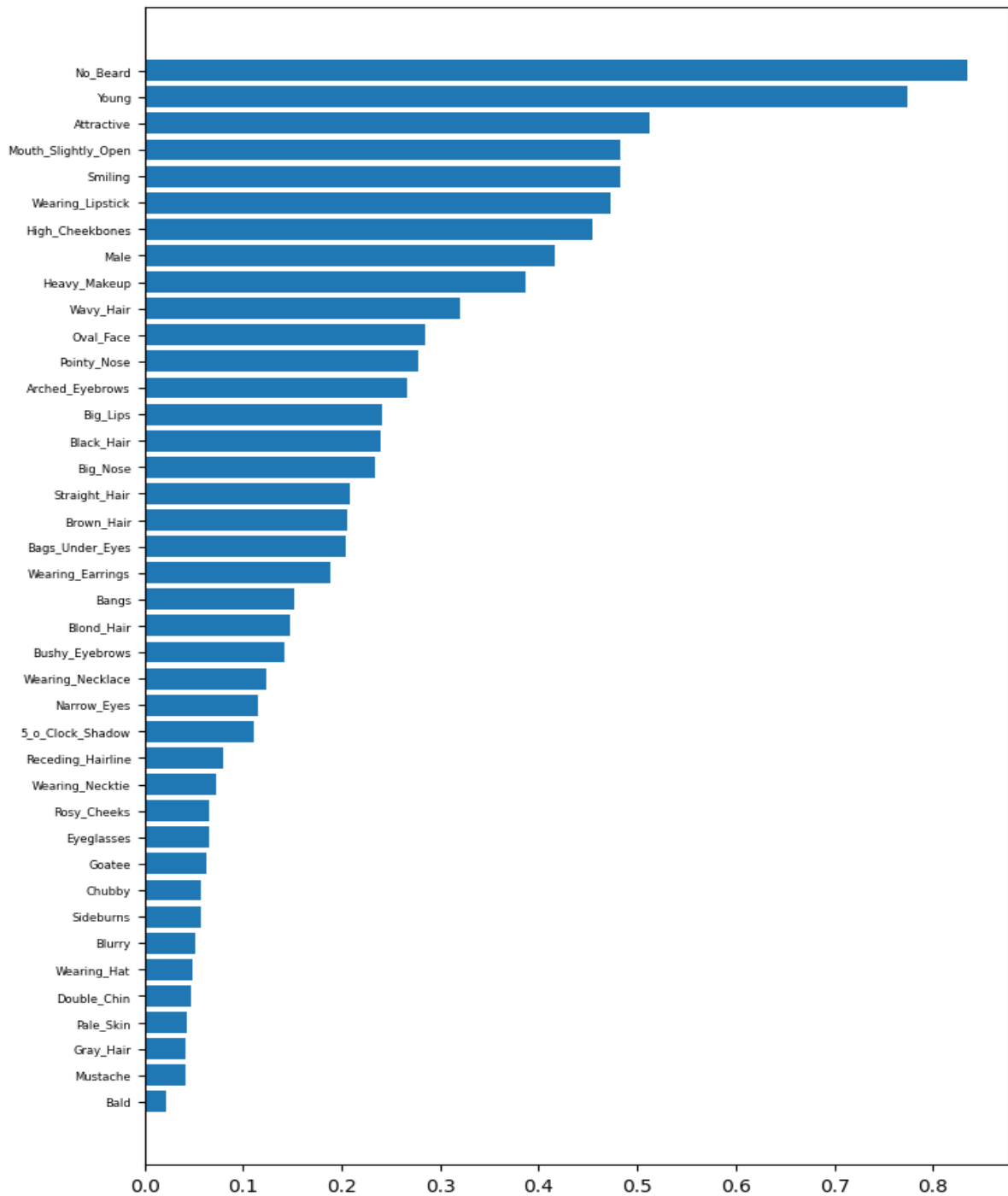


Figure 3-4-Les fréquences relatives des attributs dans la base de données celebA.

Le problème observé ici est le déséquilibre des données. Les attributs faciaux ne sont pas uniformément répartis, mais présentent plutôt une grande variabilité. Certains attributs rares apparaissent moins de 10 %, tandis que certains attributs très courants apparaissent plus de 80 %. Cette distribution déséquilibrée peut causer un problème lors de l'entraînement du modèle, connu en apprentissage profond, à savoir "le surajustement" ou "Overfitting", qui se produit lorsque le modèle surajuste les données. Dans ce cas, il peut apprendre à prédire

systématiquement les valeurs de 1 pour tous les attributs communs et de -1 pour les attributs rares.

Pour résoudre ce problème, nous avons adopté une approche recommandée utilisant des techniques d'augmentation des données. Cette méthode consiste à créer de nouvelles images en appliquant des transformations aléatoires aux données existantes. Elle permet donc d'augmenter la taille des données tout en équilibrant la distribution des attributs.

Ensuite, pour entraîner le modèle MobileNetV2 à extraire les attributs du visage d'entrée, nous avons utilisé la technique d'apprentissage par transfert (Transfer Learning). Cette technique consiste à exploiter les connaissances acquises par un modèle pré-entraîné sur une tâche spécifique pour résoudre une autre tâche similaire. Au lieu de construire un modèle à partir de zéro et de l'entraîner sur un ensemble de données particulier, l'apprentissage par transfert tire parti des caractéristiques apprises par le modèle préexistant.

Une fois le modèle entraîné, le réseau de neurones convolutifs MobileNetV2 est prêt à prendre en entrée une image faciale et à renvoyer en sortie son vecteur d'attributs associé, noté Y , qui se compose de 40 valeurs.

3.6 Génération et construction du nouvel vecteur latent

L'objectif de ce travail est de contrôler la génération d'images en manipulant l'espace latent. Nous pensons que cette manipulation serait efficace pour produire des images de meilleure qualité. Elle consiste à concaténer le résultat obtenu à l'étape précédente avec un vecteur généré aléatoirement pour construire le nouveau vecteur latent. Le vecteur résultant est utilisé comme entrée du générateur StyleGAN3.

Autrement dit, la construction du nouveau vecteur est basée sur l'idée d'utiliser le résultat de prédiction du module d'extraction des attributs, qui est modifié en un autre vecteur d'attributs Y' en fonction des attributs choisis par l'utilisateur (attributs cibles). De plus, un vecteur aléatoire Z de taille 472 est généré. Enfin, ces deux vecteurs sont concaténés pour construire le nouveau vecteur latent Z' . De cette manière, nous pouvons combiner l'information des attributs avec le vecteur latent pour guider le processus de génération et obtenir des images qui intègrent les attributs sélectionnés par l'utilisateur. Comme le montre la figure 37.

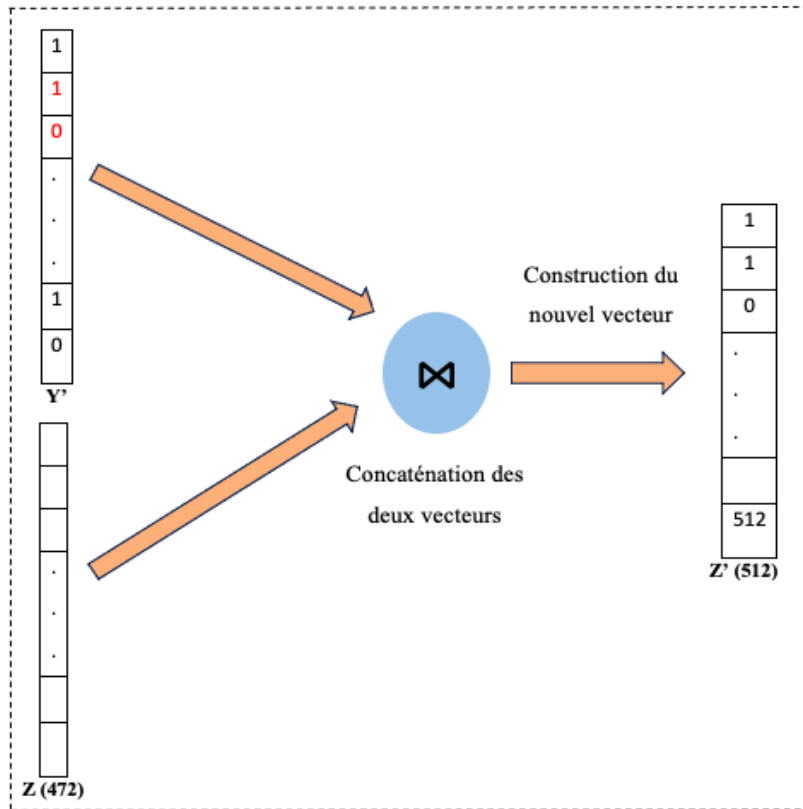


Figure 3-5- Génération et construction du nouveau vecteur latent

3.7 Génération d'images

Pour cette phase, nous utilisons le modèle StyleGAN3, qui utilise une architecture multi-résolution pour générer des détails fins à différentes échelles spatiales. Ceci est réalisé en utilisant plusieurs blocs générateurs et discriminateurs, chacun opérant à une résolution spécifique, pour créer de nouvelles images. Il prend en entrée le nouveau vecteur latent aléatoire Z et génère des images en utilisant des couches convolutives, des blocs résiduels, des mécanismes d'attention et des blocs de normalisation (Figure 38).

Avant de pouvoir générer avec StyleGAN3 des images correspondant à notre besoin, nous avons d'abord opté pour l'utilisation de son générateur pré-entraîné sur la base de données ffhq, auquel nous injectons le nouveau vecteur latent Z' construit comme entrée au générateur.

Ensuite, nous avons procédé à une étape d'apprentissage de ce générateur sur une partie de la base de données CelebA choisie aléatoirement. Cette étape d'apprentissage permet d'entraîner le générateur afin qu'il puisse générer des images réalistes d'une qualité, correspondant attributs que l'on souhaite générer.

Une fois le GAN correctement entraîné, nous avons utilisé son réseau de synthèse (le générateur) pour la génération.

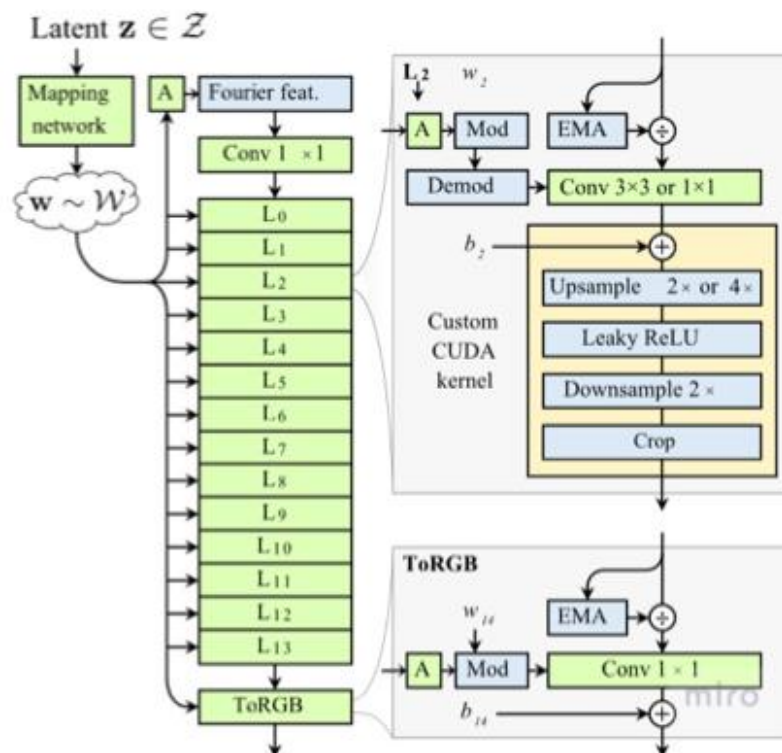


Figure 3-6- Architecture du styleGAN3 [36]

3.8 Le choix des modèles utilisés

MobileNetV2 :

Dans le but d'obtenir de bons résultats dans la partie de l'extraction des attributs de l'image d'entrée, nous avons choisi d'utiliser la technique de transfert d'apprentissage, comme mentionné précédemment, par le modèle du CNN MobileNetV2 [54] en raison de plusieurs caractéristiques de ce modèle telles que :

- ✓ **Légèreté et efficacité** : MobileNetV2 est conçu pour être à la fois léger et efficace en termes de calculs, de mémoire et de ressources nécessaires. Pour atteindre cet objectif, il utilise des opérations de convolution profonde spéciales appelées convolutions séparables en profondeur. Ces convolutions permettent de réduire considérablement le nombre de paramètres et les opérations nécessaires par rapport à des réseaux plus complexes et plus profonds. En réduisant le nombre de calculs et de ressources nécessaires, MobileNetV2 peut être utilisé dans diverses tâches telles que la

classification, la détection d'objets, la segmentation sémantique, avec une efficacité accrue. Dans la figure 32 nous vous montrons l'architecture générale de ce modèle.

- ✓ Pré-entraînement sur ImageNet : MobileNetV2 est généralement pré-entraîné sur de grands ensembles de données d'images, y compris l'ensemble de données populaire ImageNet. Cela signifie qu'il a acquis une connaissance approfondie de l'image et appris à extraire des caractéristiques générales utiles pour la plupart des tâches de vision par ordinateur.

Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	-

Tableau 2- L'architecture générale de modèle MobileNetV2 [54]

Pour adapter cette architecture à nos besoins qui est l'extraction des 40 attributs de l'image d'entrée on a supprimé la dernière couche du modèle MobileNetV2 qui permet de retourner 1000 probabilités de classes, et nous avons ajouté deux blocs de couches supplémentaires permettant d'extraire au final un vecteur d'attributs composé de 40 valeurs. Ces couches sont décrites dans la figure 39.

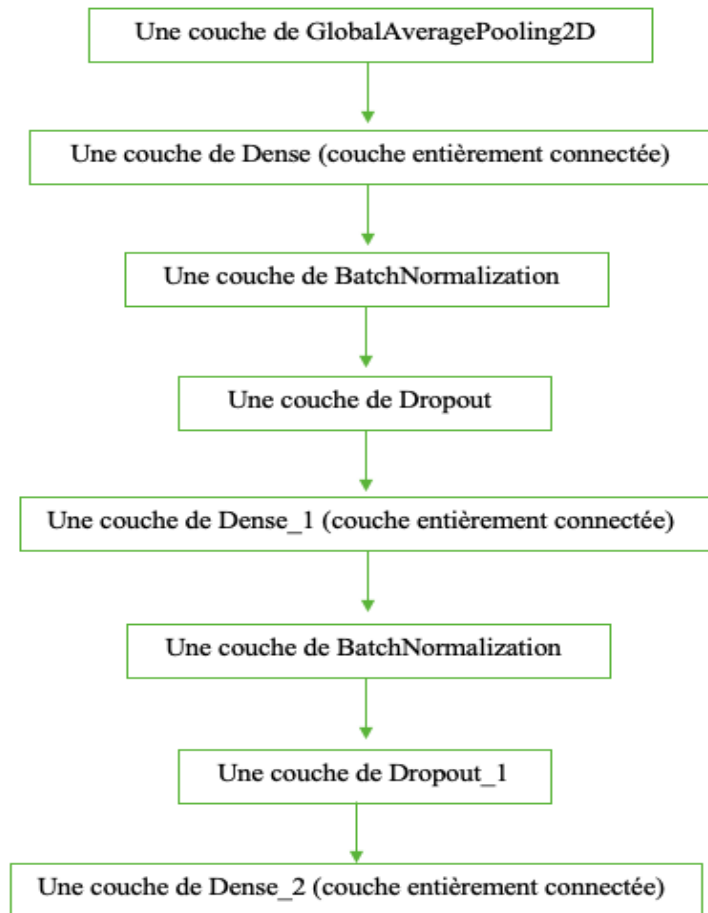


Figure 3-7- la partie ajoutée à l'architecture du MobileNetV2

Description des couches utilisés :

- Une couche de GlobalAveragePooling2D : Cette couche aide à réduire les paramètres du modèle et fournit un résumé global des caractéristiques.
- Une couche de Dense (couche entièrement connectée) : Cette couche est responsable de l'apprentissage des modèles complexes et des relations entre les caractéristiques extraites
- Une couche de BatchNormalization : Cette couche normalise les valeurs en soustrayant la moyenne et en divisant par l'écart type
- Une couche de Dropout : Cette couche permet de supprimer 30% des anneaux de la couche précédente afin de minimiser et éviter le surapprentissage (Overfitting)
- Une dernière couche de Dense : pour la prédiction des 40 valeurs d'attributs (classification multi-labels), avec la fonction d'activation « sigmoid », cette dernière

convient aux tâches de classification binaire, où chaque attribut est traité comme un problème de classification binaire distinct.

L'ajout de ces couches permet d'améliorer les performances du Modèle (augmenter le taux de précision du modèle).

StyleGAN3 :

Tout d'abord nous avons commencé d'utiliser le StyleGAN2-ADA en tant que générateur dans notre projet, mais nous avons été confrontés à des difficultés parce que la version de TensorFlow1, utilisée dans l'implémentation de StyleGAN-ADA, n'était plus disponible après la mise à jour de GoogleColab en mai. Pour cela, nous avons opté pour l'utilisation de StyleGAN3[35] comme modèle de génération profond dans notre architecture pour diverses raisons :

- ✓ **Qualité de résultat :** StyleGAN3 est réputé pour générer des images de bonne qualité, avec une résolution élevée et un réalisme accru. Il utilise une architecture de réseau de neurone avancée et des techniques d'entraînement spécifiques pour générer des images réalistes et précises.
- ✓ **Contrôle précis :** StyleGAN3 permet un contrôle fin des attributs et des caractéristiques des images générées. Il nous permet de manipuler l'espace latent de manière détaillée, ce qui signifie que nous pouvons adapter et modifier spécifiquement les attributs cibles que nous voulons obtenir dans nos images générées.
- ✓ **Adaptabilité :** StyleGAN3 est adaptable à différentes tâches de génération d'images. Il est capable de produire des images réelles dans une variété de domaines, des visages humains aux paysages, en passant par les animaux et plus encore. Cette flexibilité nous permet de mettre notre architecture au service d'une grande variété d'applications et de domaines.
- ✓ **Entraînement à partir de zéro :** contrairement à StyleGAN2, qui utilise un pré-entraînement progressif, StyleGAN3 est conçu pour être formé à partir de zéro, ce qui lui permet d'apprendre des fonctionnalités spécifiques d'ensembles de données spécifiques et de générer des images plus réalistes et diversifiées.

3.9 Conclusion

Pour atteindre notre objectif, nous avons conçu un système en trois étapes principales. Tout d'abord, nous avons utilisé la technique d'apprentissage par transfert pour le réseau neuronal

convolutionnel (CNN) MobileNetV2 afin d'extraire les attributs d'une image d'entrée. Ensuite, nous avons conçu un système qui combine ces attributs extraits avec une partie d'un vecteur latent généré aléatoirement à l'aide d'une concaténation. Enfin, le nouveau vecteur latent a été intégré en tant qu'entrée pour le générateur entraîné de StyleGan3 afin de générer une nouvelle image correspondant aux attributs cibles. Cette approche nous permet de créer des images personnalisées en manipulant l'espace latent de l'image.

4 Chapitre 4

Implémentation

4.1 Introduction

Dans ce chapitre, nous présentons les étapes de mise en œuvre du système de génération d'images contrôlé par la manipulation de l'espace latent. Notre approche consiste à entraîner un modèle de réseau de neurone convolutif MobilenetV2 par la technique de transfert d'apprentissage afin d'extraire les attributs de l'image d'entrée, et un deuxième modèle génératif profond StyleGan3 pour générer des images similaires correspondant au attributs cibles.

Pour cela, nous utilisons le système proposé est décrit dans le chapitre précédent. Ensuite, nous décrivons les différents paramètres utilisés dans la mise en œuvre et l'apprentissage du système proposé. Enfin, nous présentons un aperçu des résultats obtenus.

4.2 Environnement de développement

Notre application a été développée en exploitant la puissance de calcul considérable offerte par la plateforme populaire de Google Colab, reconnue pour son efficacité dans les tâches d'apprentissage automatique.

4.2.1 Google Colab



Google Colaboratory, également connu sous le nom de « Colab », est un service cloud de Google Research. Il s'agit d'un environnement de développement intégré (IDE) qui permet à tout utilisateur d'écrire du code source Python dans son éditeur et de l'exécuter directement depuis le navigateur. Colab est spécialement conçu pour les tâches d'apprentissage automatique, l'analyse de données, les projets éducatifs, etc. Il prend en charge les bibliothèques populaires telles que TensorFlow, PyTorch, Keras, etc. et permet d'accéder sans frais à des ressources informatiques, dont des GPU ce qui en fait un outil idéal pour la formation et la recherche en apprentissage automatique.[55]

- **GPU et TPU**

Les utilisateurs gratuits de Colab bénéficient d'un accès gratuit aux runtimes GPU et TPU pendant 12 heures maximum. Son exécution GPU est équipée d'un processeur Intel Xeon à 2,20 GHz, de 13 Go de RAM, d'un accélérateur Tesla K80 et de 12 Go de VRAM GDDR5. Le runtime TPU se compose d'un processeur Intel Xeon à 2,30 GHz, de 13 Go de RAM et d'un Cloud TPU .Avec Colab Pro ou Pro+, vous pouvez déployer davantage de processeurs, de TPU et de GPU pendant plus de 12 heures.[56]

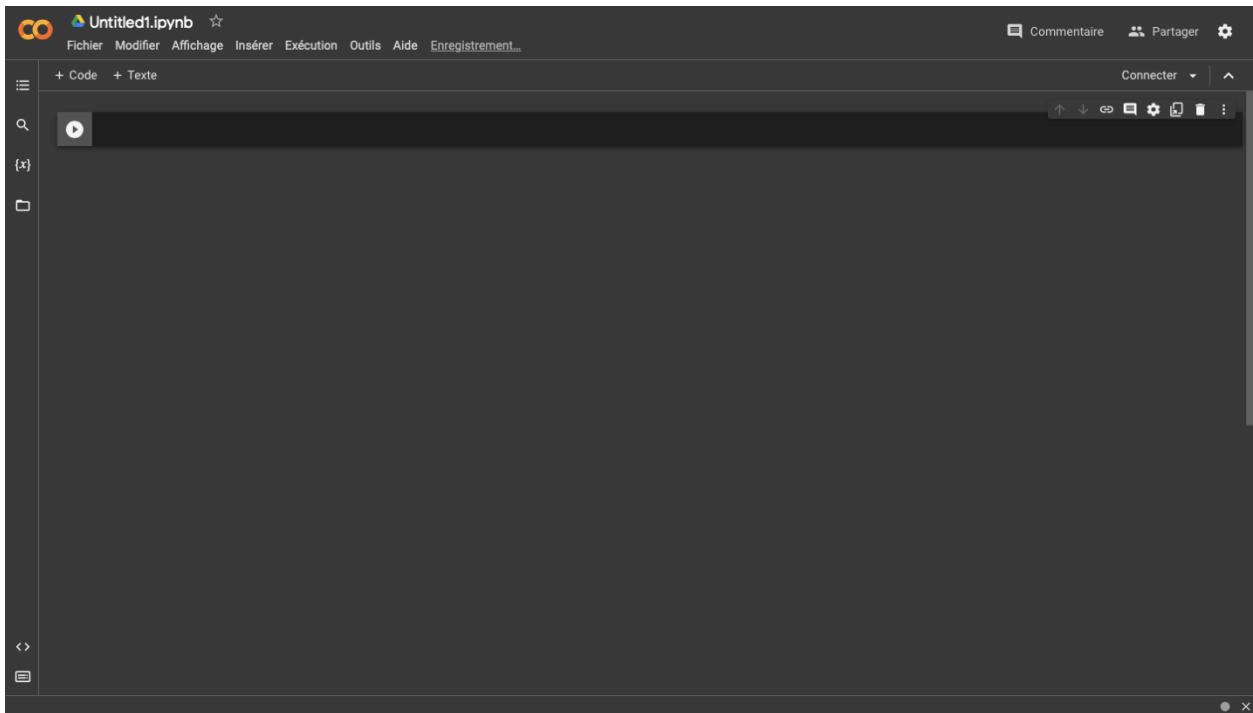


Figure 4-1-Inerface de Google Colab

- **Bibliothèques préinstallées de Colab**

Google Colab fournit plusieurs bibliothèques préinstallées, vous pouvez donc importer celles dont vous avez besoin à partir de vos extraits de code. Ces bibliothèques incluent NumPy, Pandas, matplotlib, PyTorch, TensorFlow, Keras et bien d'autres bibliothèques ML.[56]

4.2.2 Jupyter



Jupyter Notebook est un outil open source gratuit pour écrire du code informatique et le partager à des fins de collaboration, il est compatible avec près de 40 langages de programmation différents tels que : Julia, Python...

Jupyter est une application web basée client qui offre une expérience interactive où les utilisateurs peuvent travailler avec ces éléments de manière intégrée, directement dans leur navigateur web. Grâce à l'interface des notebooks Jupyter le code est exécuté dans des cellules individuelles, afficher les résultats, visualiser les graphiques et interagir avec les données.[57]

4.2.3 Anaconda



Anaconda est une plateforme logicielle gratuite et open source qui est spécialement conçue pour la programmation dans les environnements Python et R. Elle est largement utilisée dans des domaines tels que la science des données, l'intelligence artificielle et l'apprentissage automatique. En tant que distribution Python scientifique, Anaconda comprend un ensemble complet de packages essentiels pour l'analyse de données.[58]

4.3 Langages de programmation et bibliothèques utilisées

4.3.1 Python



Python est le langage de programmation open source le plus utilisé par les informaticiens. Ce langage s'est propulsé au sommet de la gestion d'infrastructure, de l'analyse de données ou du développement de logiciels. En fait, parmi les nombreuses fonctionnalités de Python, les

développeurs peuvent se concentrer sur ce qu'ils font, plutôt que sur la façon dont ils le font. Il libère les développeurs des contraintes formelles qui tourmentaient leur temps avec les langages plus anciens. Par conséquent, développer du code en Python est plus rapide que dans d'autres langages.[59]

4.4 Les bibliothèques utilisées pendant la programmation

- **Keras**

Keras est une bibliothèque open-source écrite en Python (sous licence MIT), basée en grande partie sur les travaux du développeur Google François Chollet, et faisant partie du projet ONEIROS (Open Neuroelectronic Intelligent Robot Operating System). La première version du logiciel multiplateforme est sortie le 28 mars 2015. Le but de cette bibliothèque est de permettre la construction rapide de réseaux de neurones. Dans ce cas, Keras ne fonctionne pas comme un framework à part entière, mais comme une interface de programmation d'application (API) pour accéder et programmer différents frameworks d'apprentissage automatique. Les frameworks pris en charge par Keras incluent Theano, Microsoft Cognitive Toolkit (anciennement connu sous le nom de CNTK) et TensorFlow.[60]

- **Tensorflow**

TensorFlow est une bibliothèque open source de Machine Learning développée par Google. Qui fournit une boîte à outils permettant de résoudre des problèmes mathématiques complexes de manière efficace Elle offre une plateforme complète pour le développement et l'exécution d'applications de Machine Learning et de Deep Learning. [61]

- **Numpy**

Le terme NumPy est l'abréviation « Numerical **Python** », c'est une bibliothèque Python populaire principalement utilisée pour les calculs mathématiques et scientifiques. Elle comprend de nombreuses fonctionnalités et outils qui peuvent s'avérer utiles pour les projets de Data Science. Se familiariser avec NumPy est une étape essentielle dans tout projet de formation en Data Science. Découvrez tout ce que vous devez savoir pour maîtriser Numpy [62]

- **Matplotlib**

Matplotlib est une bibliothèque Python open source, initialement développée par le neurobiologiste John Hunter en 2002. TensorFlow est une bibliothèque largement utilisée par les personnes travaillant avec Python ou NumPy. Elle trouve son utilité dans divers contextes, tels que les serveurs d'application web, les shells et les scripts Python. En utilisant les APIs de matplotlib, les développeurs ont également la possibilité d'intégrer des graphiques à des applications avec interface graphique.[63]

- **OS**

Le module os de Python offre des fonctionnalités pour effectuer des opérations courantes liées au système d'exploitation. Il est conçu pour être indépendant du système d'exploitation de la machine, ce qui signifie qu'il peut être utilisé sur n'importe quel système d'exploitation.[64]

- **Tkinter**

Tkinter est une bibliothèque open source et portable pour la création d'interfaces utilisateur graphiques (GUI) dans les scripts Python. Elle repose sur Tk, qui est une bibliothèque graphique utilisée par Tcl/Tk et Perl, et qui est elle-même implémentée en C. Ainsi, on peut dire que Tkinter est une implémentation multicouche, où Tkinter utilise Tk pour fournir ses fonctionnalités.[65]

4.5 Apprentissage et test

4.5.1 Base de données utilisée

Comme nous avons mentionné précédemment, nous avons utilisé l'ensemble de données CelebA qui contient 202 599 images de visages en couleur et 40 vecteurs binaires d'attributs. Pour notre étude, nous utilisons des versions alignées et recadrées des images dont les dimensions ont été réduites à 64×64 pixels. Les données sont divisées en trois partitions officielles : une partition d'entraînement de 182 000 images et une partition de test de 20 000 images. et une partition du test contient le reste d'images. Parmi ces images nous avons utilisé 8975 images comme mentionné précédemment aussi.

4.5.2 Apprentissage et test du MobileNetV2

Pour entraîner notre modèle de CNN MobileNetV2, nous avons utilisé l'architecture finale conçue dans le chapitre précédent. En raison des ressources limitées de Google Colab, telles que le manque de GPU et la RAM d'exécution pleine, nous avons dû restreindre notre ensemble de données celebA à seulement 10% (environ 20 000 images) et extraire le visage de chaque image pour l'entraînement.

Afin de remédier au problème de déséquilibre des attributs dans les images, nous avons appliqué sur ces données une techniques d'augmentation de données.

Lors de l'apprentissage, nous avons exploré différentes possibilités en modifiant plusieurs paramètres du réseau, notamment :

- Le nombre d'épochs : Nous avons testé différentes valeurs pour déterminer le nombre d'épochs optimal et nécessaire pour l'apprentissage.
- Le taille du lot (batch size) : Nous avons expérimenté différentes tailles de lot pour déterminer celle qui offre un bon compromis entre l'efficacité de l'apprentissage et l'utilisation des ressources.
- Le nombre d'itérations : Nous avons ajusté le nombre d'itérations par epoch, pour optimiser la convergence du modèle.
- Le type d'optimiseur et le taux d'apprentissage : Nous avons testé différents optimiseurs et taux d'apprentissage pour trouver la configuration qui permet d'obtenir de bonnes performances d'apprentissage.

En effectuant ces ajustements, nous avons cherché à maximiser les performances de notre modèle dans les limites imposées par les ressources de calcul disponibles. Donc, nous avons mesuré les performances du modèle en termes de précision (Accuracy) et de perte (Loss), puis enregistré les résultats dans le tableau ci-dessous.

Le numéro d'expérimentation	Nombre d'epochs	Le Batch size	Le type d'optimiseur	Le taux d'apprentissage
1	14	56	Adam	0.001
2	30	56	Adam	0.001
3	100	32	AdaDelta	1.0
4	10	56	Adam	0.0001

Tableau 3- Paramètres d'apprentissage des expérimentations réalisées

4.5.3 Apprentissage et test du générateur de StyleGAN3

Initialement, nous avons prévu d'utiliser le générateur conditionnel de StyleGAN2, aussi connu sous le nom de StyleGAN2-ADA, car il est considéré comme le meilleur choix pour injecter du conditionnement d'attributs dans le générateur, mais comme nous avons rencontré le problème de Tensorflow mentionné plus haut, nous avons opté pour StyleGAN3 pour atteindre notre objectif final.

Cependant, il convient de noter qu'avant de procéder à l'entraînement personnalisé de ce modèle, nous l'avons d'abord testé en tant que modèle pré-entraîné afin d'évaluer ces résultats. Pour ce faire, nous avons utilisé le vecteur d'attributs prédit lors de l'étape d'extraction des attributs et modifié par l'utilisateur selon son choix. Nous avons généré un vecteur latent aléatoire de taille 472 et combiné ce dernier avec le vecteur d'attributs finale pour former un nouveau vecteur latent. Cette combinaison est nécessaire pour que la taille du vecteur latent soit compatible avec l'entrée du générateur StyleGAN3.

Le StyleGAN3 pré-entraîné utilisé est entraîné sur la base de données FFHQ. Les résultats obtenus par cette étape ne correspondent pas à nos attentes initiales, bien que le générateur ait pu générer des images de visages de haute qualité mais ne satisfaisaient pas notre objectif, qui est une image similaire avec des attributs choisies. Pour cela, nous nous sommes tournés vers l'idée d'entraîner le modèle sur une partie aléatoire de notre base de données CelebA en suivant les mêmes étapes précédentes. Le processus d'apprentissage se déroule de la manière suivante :

1. Prétraitement des données : nous avons utilisés 8975 images de la base de données choisie aléatoirement, ensuite nous avons redimensionné la taille de ces images au 64*64 qui représente une des dimensions accepté par le modèle StyleGAN (il nécessite des dimensions d'images en puissance de 2), et enfin compressé ces images au format zip.
2. Lancement d'apprentissage avec les paramètres suivant :

```
--outdir=/content/out/results \  
--cfg=stylegan3-r \  
--data=/content/mydataset.zip \  
--gpus=1 \  
--batch=16 \  
--batch-gpu=8 \  
--gamma=6.6 \  
--mirror=1 \ --kimg=200 \ --snap=8 \  
--metrics=fid50k_full, kid50k_full
```

3. Relancement d'apprentissage plusieurs fois par le paramètre résumé :

```
resume_from = '/content/drive/MyDrive/network-snapshot-000200.pkl'  
--outdir=/content/drive/MyDrive/out/results \  
--cfg=stylegan3-r \  
--data=/content/mydataset.zip \  
--gpus=1 \  
--batch=32 \  
--batch-gpu=8 \ --gamma=6.6 \ --mirror=1 \ --kimg=500 \ --snap=20  
--resume=$resume_fr
```

4.6 Évaluation et Résultats

4.6.1 Évaluation et Résultats du système d'extraction des attributs

Tableau 4- Évaluation de système d'extraction des attributs

Tests	Taux du loss et du accuracy	Les courbes de précisions et de perte associées
Test : 1	Évaluation Loss : 0.3003 Évaluation Accuracy : 0.8618	
Test : 2	Évaluation Loss : 0.3005 Évaluation Accuracy : 0.8625	

<p>Test : 3</p>	<p>Evaluation Loss: 0.5633</p> <p>Evaluation Accuracy: 0.7180</p>	
<p>Test : 4</p>	<p>Évaluation Loss : 0.0930</p> <p>Évaluation Accuracy : 0.8674</p>	

Dans le tableau précédent nous avons présenté les différentes expérimentations et les graphes d'évaluations associés avec ces valeurs de résultats. Après la réalisation de plusieurs tests et l'évaluation de notre système d'extraction des attributs, nous avons choisi de travailler avec le modèle d'expérimentation 1 qui a obtenu une précision de 86% avec un nombre d'époques optimale, ensuite nous l'avons testé sur quelques images, voici quelques exemples de résultat d'exécution de modèle :





Image d'entrée



```

1/1 [=====] - 8s 8s/ste
5_o_Clock_Shadow : -1 ( 0.021301258 )
Arched_Eyebrows : -1 ( 0.424729 )
Attractive : 1 ( 0.91477513 )
Bags_Under_Eyes : -1 ( 0.048426524 )
Bald : -1 ( 0.0026720809 )
Bangs : -1 ( 0.08643035 )
Big_Lips : -1 ( 0.30928847 )
Big_Nose : -1 ( 0.052644145 )
Black_Hair : -1 ( 0.09105623 )
Blond_Hair : -1 ( 0.21556982 )
Blurry : -1 ( 0.032258913 )
Brown_Hair : -1 ( 0.32728386 )
Bushy_Eyebrows : -1 ( 0.076071896 )
Chubby : -1 ( 0.0043296125 )
Double_Chin : -1 ( 0.0031803257 )
Eyeglasses : -1 ( 0.0032052367 )
Goatee : -1 ( 0.0071705035 )
Gray_Hair : -1 ( 0.0021639748 )
Heavy_Makeup : 1 ( 0.8394606 )
High_Cheekbones : -1 ( 0.40764982 )
Male : -1 ( 0.019374155 )
Mouth_Slightly_Open : 1 ( 0.57632303 )
Mustache : -1 ( 0.006412674 )
Narrow_Eyes : -1 ( 0.07211687 )
No_Beard : 1 ( 0.9835755 )
Oval_Face : -1 ( 0.25922585 )
Pale_Skin : -1 ( 0.083590865 )
Pointy_Nose : 1 ( 0.5977922 )
Receding_Hairline : -1 ( 0.011753232 )
Rosy_Cheeks : -1 ( 0.1122039 )
Sideburns : -1 ( 0.009557752 )
Smiling : 1 ( 0.50807333 )
Straight_Hair : -1 ( 0.2299633 )
Wavy_Hair : 1 ( 0.54338944 )
Wearing_Earrings : -1 ( 0.15150093 )
Wearing_Eat : -1 ( 0.03148094 )
Wearing_Lipstick : 1 ( 0.93100107 )
Wearing_Mecklace : -1 ( 0.15947536 )
Wearing_Mecktie : -1 ( 0.0054585706 )
Young : 1 ( 0.97501016 )

```

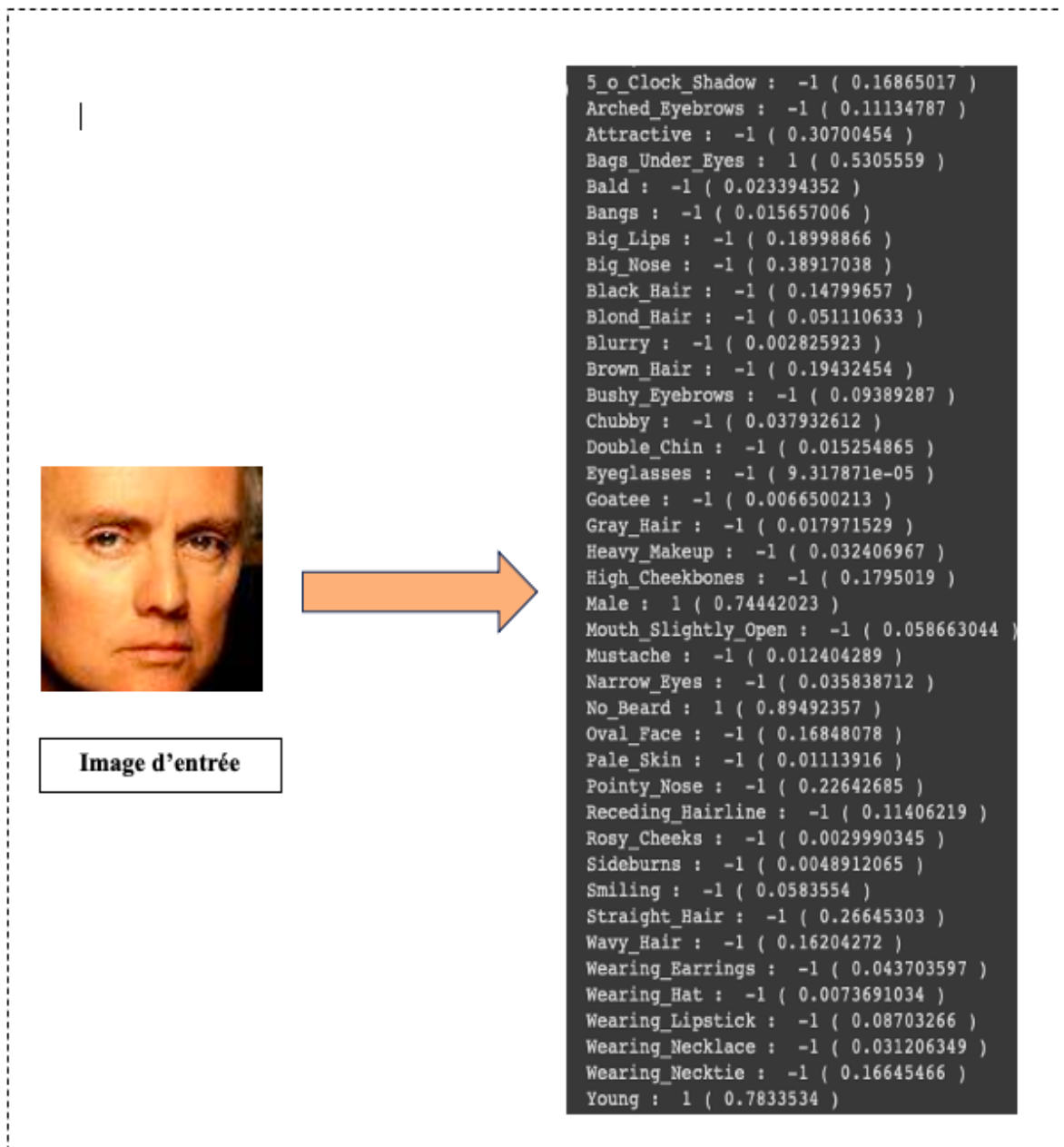
Il est noté que le modèle ici retourne des valeurs de prédictions réelles, pour cela nous avons effectué une simple modification pour que le modèle retourne une valeur de 1 si la valeur prédite est supérieure à 0.5 sinon il retourne -1



Image d'entrée



```
5_o_Clock_Shadow : -1 ( 0.007435365 )
Arched_Eyebrows : 1 ( 0.52195096 )
Attractive : 1 ( 0.820961 )
Bags_Under_Eyes : -1 ( 0.00431521 )
Bald : -1 ( 0.00014230424 )
Bangs : -1 ( 0.19185226 )
Big_Lips : -1 ( 0.29461852 )
Big_Nose : -1 ( 0.017705547 )
Black_Hair : -1 ( 0.029875873 )
Blond_Hair : 1 ( 0.6656831 )
Blurry : -1 ( 0.00022516136 )
Brown_Hair : -1 ( 0.12947549 )
Bushy_Eyebrows : -1 ( 0.03744571 )
Chubby : -1 ( 0.00039224065 )
Double_Chin : -1 ( 0.00042402535 )
Eyeglasses : -1 ( 7.03469e-06 )
Goatee : -1 ( 6.840049e-05 )
Gray_Hair : -1 ( 0.00029995272 )
Heavy_Makeup : 1 ( 0.75926286 )
High_Cheekbones : -1 ( 0.20266017 )
Male : -1 ( 0.024441348 )
Mouth_Slightly_Open : -1 ( 0.03653227 )
Mustache : -1 ( 8.606848e-06 )
Narrow_Eyes : -1 ( 0.021599445 )
No_Beard : 1 ( 0.9980959 )
Oval_Face : -1 ( 0.2866192 )
Pale_Skin : -1 ( 0.019368239 )
Pointy_Nose : 1 ( 0.72459316 )
Receding_Hairline : -1 ( 0.006144073 )
Rosy_Cheeks : -1 ( 0.16145678 )
Sideburns : -1 ( 6.5431435e-05 )
Smiling : -1 ( 0.07738902 )
Straight_Hair : -1 ( 0.15201712 )
Wavy_Hair : 1 ( 0.5238 )
Wearing_Earrings : -1 ( 0.22809644 )
Wearing_Hat : -1 ( 0.008434175 )
Wearing_Lipstick : 1 ( 0.9182257 )
Wearing_Necklace : -1 ( 0.16088243 )
Wearing_Necktie : -1 ( 0.001245994 )
Young : 1 ( 0.94932157 )
```



4.6.2 Évaluation et résultats du système de génération

- Le générateur StyleGAN3 pré-entraîné

Nous avons effectué plusieurs tests de génération en utilisant les images précédentes dont nous avons déjà extrait les vecteurs d'attributs. Ces vecteurs ont été convertis en vecteurs binaires, où la valeur est définie à 1 si l'attribut est présent et à 0 si l'attribut est absent (en comparant avec 0,5), afin de pouvoir les concaténer. Dans la génération d'image, nous avons choisie de généré des images avec l'attributs 'Bags_Under_Eyes', afin de tester les résultats et les observer.

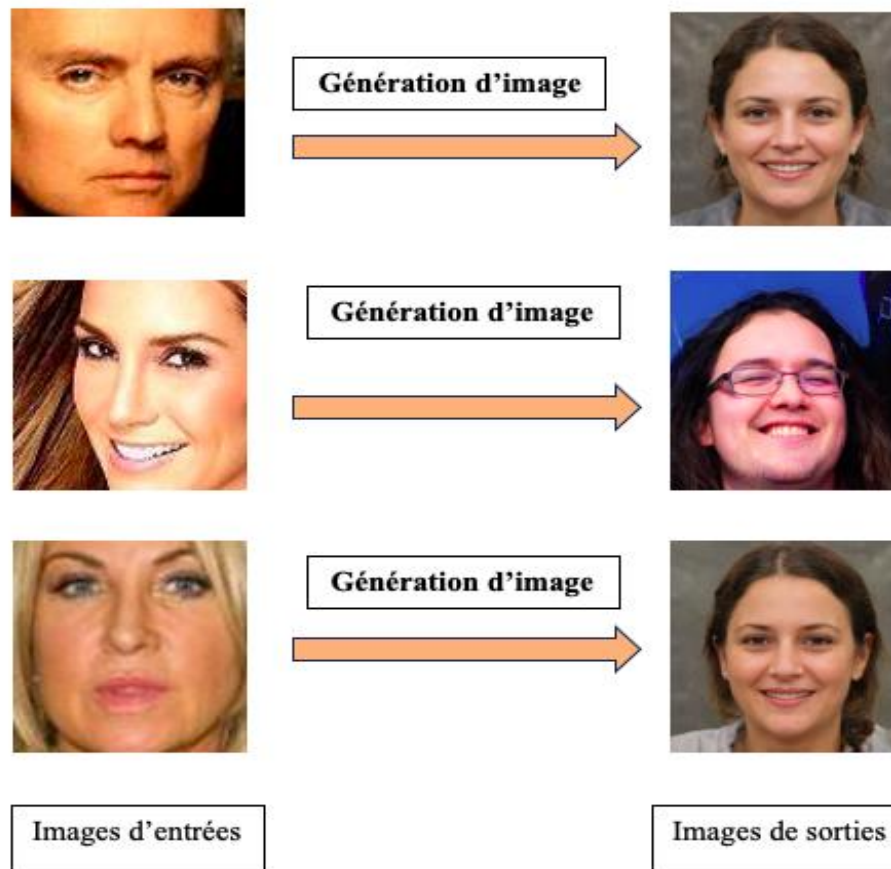


Figure 4-2- Résultats de génération avec le générateur StyleGAN3 pré-entraîné

Après avoir effectué plusieurs tests sur le générateur pré-entraîné StyleGAN3 (figure 41), nous avons constaté que les images générées étaient de très bonne qualité, mais elles ne répondaient pas à notre objectif principal, qui était de générer des images similaires avec des attributs cibles. Dans la plupart des tests réalisés, les visages générés étaient différents, ce qui a motivé notre décision de procéder à l'entraînement de ce modèle.

- **Le générateur StyleGAN3 entraîné par nos soins**

Cette fois, nous avons choisi de générer des images similaires avec l'attribut 'Black_hair' :

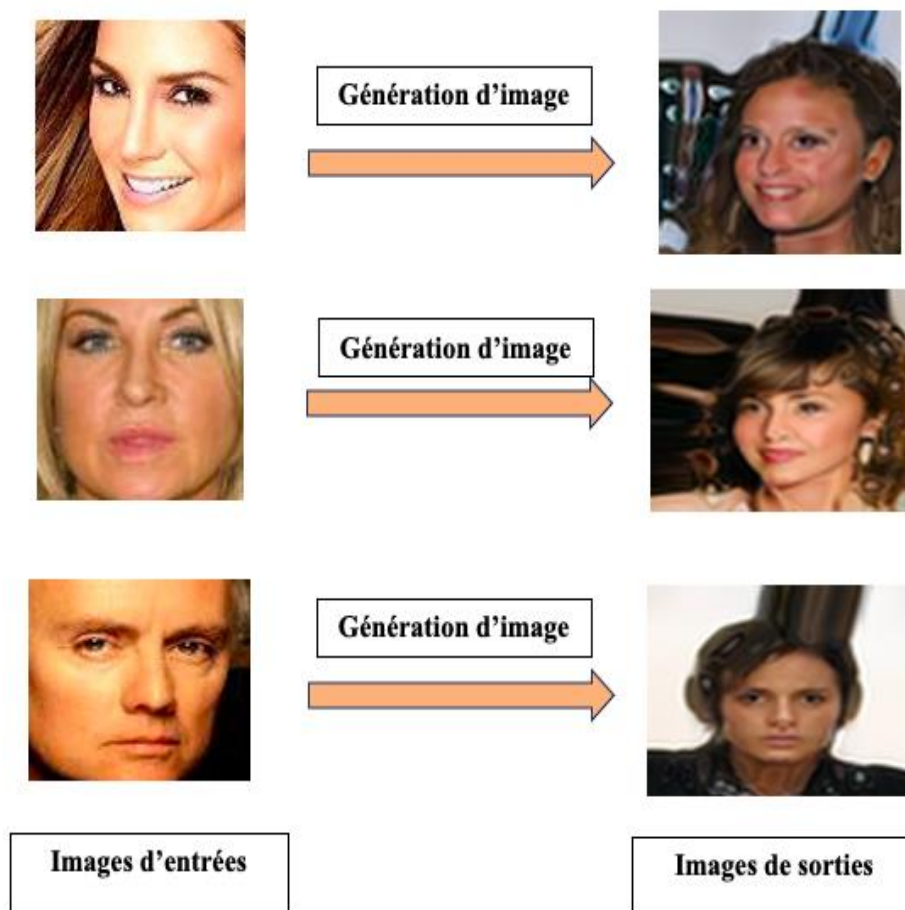


Figure 4-3- Résultats de génération avec le générateur de StyleGAN3 entraîné

Après avoir entraîné notre modèle à plusieurs reprises sur différents comptes Google Colab, nous avons obtenu les résultats présentés dans la figure 42, Nous constatons que les images générées sont de moins bonne qualité que celles générées par le modèle pré-entraîné StyleGAN3. Cela est principalement dû aux exigences d'apprentissage, nécessitant plusieurs cartes graphiques (GPU) et plusieurs jours d'entraînement. Nous avons également constaté que les images générées n'étaient pas complètes et n'atteignaient pas entièrement notre objectif, qui était de générer une image similaire avec l'attribut choisi.



Figure 4-4- Ensemble d'images générées par StyleGAN3 entraîné

4.7 Conclusion

Dans ce chapitre, nous présentons les résultats de diverses expériences menées sur l'extraction d'attributs de l'image d'entrée et la génération d'images faciales. Les résultats obtenus, bien qu'insuffisants et incomplets par rapport à nos objectifs, ont le potentiel d'améliorer les performances de notre système. Pour y parvenir, nous envisageons d'utiliser un autre modèle de génération conditionnelle tel que StyleGAN2-ADA, ainsi qu'un ensemble de données plus vaste afin de mieux entraîner le générateur de ce modèle. De plus, pour obtenir des images de haute qualité, il est nécessaire d'utiliser des machines et des outils plus puissants, tels que Colab Pro, qui ne sont pas encore disponibles en Algérie.

Conclusion Générale

L'édition d'images à l'aide des modèles génératifs profonds est une application intéressante de l'intelligence artificielle. Ces modèles, tels que les générateurs (GAN) et les auto-encodeurs variationnels (VAE), sont capables de générer des images réalistes et d'effectuer des opérations spécifiques sur ces images, tels que transformations d'images, contrôler les attributs d'images générés.

Dans ce mémoire, nous avons concentré nos efforts sur le problème du contrôle des images générées en modifiant l'espace latent d'un modèle génératif profond. Notre attention s'est particulièrement portée sur l'opération de génération d'une image similaire avec des attributs cibles choisis à partir d'une image d'entrée.

Le système que nous proposons se déroule en trois phases principales : l'extraction des attributs de l'image d'entrée à l'aide d'un modèle de réseau neuronal convolutionnel, puis la construction et la génération d'un nouveau vecteur latent en concaténant le vecteur d'attributs extrait précédemment et modifié, par un vecteur latent généré aléatoirement. Cette étape est cruciale dans notre système pour pouvoir passer à l'étape finale, qui est la génération d'images à partir du nouveau vecteur latent construit à l'aide du modèle génératif profond StyleGAN3. Nous avons utilisé ces deux cas de générateurs (module de synthèse), l'un pré-entraîné sur la base de données ffhq, et l'autre que nous avons entraîné nous-mêmes. L'entraînement de notre système d'extraction d'attributs a été effectué sur la base de données celebA et nous avons obtenu une précision de 86%. Le système de génération d'images faciales a également été entraîné sur le même ensemble de données.

Au cours de la réalisation de cette application, nous avons fait face à plusieurs difficultés. Tout d'abord, nous avons commencé la programmation en utilisant le générateur StyleGAN2-ADA. Cependant, en raison de problèmes de mises à jour de Google Colab, le système a été interrompu en mai. Cela nous a obligés à utiliser une autre méthode de génération, à savoir la dernière version du générateur StyleGAN, appelée StyleGAN3.

En ce qui concerne le processus d'entraînement pour la génération de visages, nous avons rencontré deux problèmes majeurs qui ont ralenti la phase d'implémentation. Tout d'abord, nous avons dû faire face à des problèmes de compatibilité des versions des bibliothèques utilisées.

De plus, nous avons rencontré des difficultés liées au manque de GPU disponibles pour assurer une exécution rapide, ainsi qu'à la limitation de la mémoire RAM sur Google Colab.

Ces obstacles ont représenté des défis significatifs tout au long du processus de conception et d'implémentation de notre application. Les résultats de notre système s'affichent progressivement et restent incomplets et pourrait être encore améliorés et suffisants dans des projets futurs. Dans cette optique, nous proposons les points suivants pour les travaux futurs :

- Utiliser un système d'extraction des attributs plus performant, car la performance du système de génération en dépend largement.
- Refaire l'apprentissage du système de génération en utilisant plusieurs GPU, ainsi que plusieurs bases de données, pour le généraliser.
- Utiliser une autre méthode d'apprentissage de ce type de GAN
- Utiliser un type de générateur profond conditionnel pour pouvoir intégrer le vecteur d'attributs cible comme condition de génération.

Références

- [1] Zhou Z-H. Machine Learning. Springer Nature; 2021.
- [2] Qu'est-ce que l'apprentissage automatique ? - PraedictIA n.d. <https://praedictia.com/page/lapprentissage-machine/quest-ce.html> (accessed May 11, 2023).
- [3] Qu'est-ce que l'apprentissage supervisé ? | Linedata n.d. <https://fr.linedata.com/quest-ce-que-lapprentissage-supervise> (accessed May 11, 2023).
- [4] Lorre G. Apprentissage non-supervisé de représentations pour l'analyse de séquences vidéos n.d.
- [5] Deluzarche C. Définition | Deep Learning - Apprentissage profond | Futura Tech. Futura n.d. <https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/> (accessed May 11, 2023).
- [6] Dreyfus G. LES RÉSEAUX DE NEURONES n.d.
- [7] st-m-app-rn.pdf n.d.
- [8] ARTIFICIAL NEURAL NETWORKS. n.d.
- [9] Réseau de neurones artificiels : qu'est-ce que c'est et à quoi ça sert ? n.d. <https://www.lebigdata.fr/reseau-de-neurones-artificiels-definition> (accessed May 12, 2023).
- [10] Touzet C. LES RESEAUX DE NEURONES ARTIFICIELS, INTRODUCTION AU CONNEXIONNISME n.d.
- [11] Convolutional Neural Networks from the ground up | by Alejandro Escontrela | Towards Data Science n.d. <https://towardsdatascience.com/convolutional-neural-networks-from-the-ground-up-c67bb41454e1> (accessed May 12, 2023).
- [12] Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. Generative Adversarial Nets. Advances in Neural Information Processing Systems, vol. 27, Curran Associates, Inc.; 2014.
- [13] Auto-encodeur. Data Analytics Post n.d. <https://dataanalyticspost.com/Lexique/auto-encodeur/> (accessed May 12, 2023).
- [14] Long short-term memory - Wikipedia n.d. https://en.wikipedia.org/wiki/Long_short-term_memory (accessed May 12, 2023).
- [15] The Neural Network Zoo - The Asimov Institute n.d. <https://www.asimovinstitute.org/neural->

network-zoo/ (accessed May 12, 2023).

[16] Blanc-Durand P. Réseaux de neurones convolutifs en médecine nucléaire : Applications à la segmentation automatique des tumeurs gliales et à la correction d'atténuation en TEP/IRM. 2018. <https://doi.org/10.13140/RG.2.2.23231.97441>.

[17] O'Shea K, Nash R. An Introduction to Convolutional Neural Networks 2015.

[18] Kim S. A Beginner's Guide to Convolutional Neural Networks (CNNs). Medium 2019. <https://towardsdatascience.com/a-beginners-guide-to-convolutional-neural-networks-cnns-14649dbddce8> (accessed May 12, 2023).

[19] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research* 2014;15:1929–58.

[20] Nair V, Hinton GE. Rectified Linear Units Improve Restricted Boltzmann Machines n.d.

[21] Que sont les réseaux de neurones ? | IBM n.d. <https://www.ibm.com/fr-fr/topics/neural-networks> (accessed May 12, 2023).

[22] rédac T. Generative Model ou Modèle Génératif : Tout comprendre. *Formation Data Science | DataScientest.com* 2021. <https://datascientest.com/generative-model-ou-modele-generatif> (accessed May 12, 2023).

[23] Cuofano G. Modèles génératifs en bref. *FourWeekMBA* 2023. <https://fourweekmba.com/fr/mod%C3%A8les-g%C3%A9n%C3%A9ratifs-en-bref/> (accessed May 12, 2023).

[24] Zhang H, Sindagi V, Patel VM. Image De-raining Using a Conditional Generative Adversarial Network 2019.

[25] Karras T, Aila T, Laine S, Lehtinen J. Progressive Growing of GANs for Improved Quality, Stability, and Variation 2018.

[26] Kingma DP, Welling M. Auto-Encoding Variational Bayes 2022.

[27] Auto-encodeurs en Deep Learning : tout savoir | Blent.ai n.d. <https://blent.ai/blog/a/auto-encodeurs-deep-learning> (accessed May 15, 2023).

[28] Matrice de confusion : qu'est-ce que c'est et comment l'utiliser ? n.d. <https://datascientest.com/matrice-de-confusion> (accessed May 16, 2023).

[29] Classification: courbe ROC et AUC | Machine Learning | Google Developers n.d. <https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=fr>

(accessed May 16, 2023).

[30] Karras T, Laine S, Aila T. A Style-Based Generator Architecture for Generative Adversarial Networks 2019.

[31] StyleGAN - A Style-Based Generator Architecture for Generative Adversarial Networks | Engineering Education (EngEd) Program | Section n.d. <https://www.section.io/engineering-education/stylegan-a-style-based-generator-architecture-for-gans/> (accessed May 16, 2023).

[32] [Review] Image2StyleGAN — Embedding An Image Into StyleGAN. | by Oscar Guarnizo | Medium n.d. <https://oscar-guarnizo.medium.com/review-image2stylegan-embedding-an-image-into-stylegan-c7989e345271> (accessed May 16, 2023).

[33] Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T. Analyzing and Improving the Image Quality of StyleGAN. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA: IEEE; 2020, p. 8107–16. <https://doi.org/10.1109/CVPR42600.2020.00813>.

[34] Karras T, Aittala M, Hellsten J, Laine S, Lehtinen J, Aila T. Training Generative Adversarial Networks with Limited Data 2020.

[35] Karras T, Aittala M, Laine S, Härkönen E, Hellsten J, Lehtinen J, et al. Alias-Free Generative Adversarial Networks 2021.

[36] Melnik A, Miasayedenkau M, Makarovets D, Pirshtuk D, Akbulut E, Holzmann D, et al. Face Generation and Editing with StyleGAN: A Survey 2023.

[37] Tiu E. Understanding Latent Space in Machine Learning. Medium 2020. <https://towardsdatascience.com/understanding-latent-space-in-machine-learning-de5a7c687d8d> (accessed May 25, 2023).

[38] Latent Space in Deep Learning | Baeldung on Computer Science n.d. <https://www.baeldung.com/cs/dl-latent-space> (accessed May 17, 2023).

[39] Park T, Liu M-Y, Wang T-C, Zhu J-Y. Semantic Image Synthesis with Spatially-Adaptive Normalization 2019.

[40] Vonintsoa. Qu'est-ce qu'un auto-encodeur ? Le guide complet. INTELLIGENCE-ARTIFICIELLECOM 2022. <https://intelligence-artificielle.com/auto-encodeur-guide-complet/> (accessed May 21, 2023).

[41] Zhu J-Y, Park T, Isola P, Efros AA. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks 2020.

- [42] OpenAI GPT2 n.d. https://huggingface.co/docs/transformers/model_doc/gpt2 (accessed May 22, 2023).
- [43] CelebA Dataset n.d. <https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html> (accessed June 4, 2023).
- [44] Zheng T, Deng W. Cross-Pose LFW: A Database for Studying Cross-Pose Face Recognition in Unconstrained Environments n.d.
- [45] Miller D, Brossard E, Seitz S, Kemelmacher-Shlizerman I. MegaFace: A Million Faces for Recognition at Scale 2015.
- [46] Chen Z, Huang W, Luo Z. embedGAN: A Method to Embed Images in GAN Latent Space. In: Yuan PF, Yao J, Yan C, Wang X, Leach N, editors. Proceedings of the 2020 DigitalFUTURES, Singapore: Springer Singapore; 2021, p. 208–16. https://doi.org/10.1007/978-981-33-4400-6_20.
- [47] Qi M, Wang Y, Qin J, Li A. KE-GAN: Knowledge Embedded Generative Adversarial Networks for Semi-Supervised Scene Parsing. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA: IEEE; 2019, p. 5232–41. <https://doi.org/10.1109/CVPR.2019.00538>.
- [48] Abdal R, Qin Y, Wonka P. Image2StyleGAN: How to Embed Images Into the StyleGAN Latent Space? 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea (South): IEEE; 2019, p. 4431–40. <https://doi.org/10.1109/ICCV.2019.00453>.
- [49] Abdal R, Qin Y, Wonka P. Image2StyleGAN++: How to Edit the Embedded Images? 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA: IEEE; 2020, p. 8293–302. <https://doi.org/10.1109/CVPR42600.2020.00832>.
- [50] Zhu P, Abdal R, Qin Y, Femiani J, Wonka P. Improved StyleGAN Embedding: Where are the Good Latents? 2021.
- [51] Eisenbeiser L. Latent Walking Techniques for Conditioning GAN-Generated Music. 2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON), New York, NY, USA: IEEE; 2020, p. 0548–53. <https://doi.org/10.1109/UEMCON51285.2020.9298149>.
- [52] Wu_StyleSpace_Analysis_Disentangled_Controls_for_StyleGAN_Image_Generation_CVPR_2021_paper.pdf n.d.
- [53] Tov O, Alaluf Y, Nitzan Y, Patashnik O, Cohen-Or D. Designing an encoder for StyleGAN image manipulation. ACM Trans Graph 2021;40:1–14. <https://doi.org/10.1145/3450626.3459838>.

- [54] Sandler M, Howard A, Zhu M, Zhmoginov A, Chen L-C. MobileNetV2: Inverted Residuals and Linear Bottlenecks 2019.
- [55] Isaac. Google Collab ou Google Colaboratory : qu'est-ce que c'est. Hardware libre 2021. <https://www.hwlibre.com/fr/colaboratoire-google/> (accessed May 29, 2023).
- [56] Das T. Google Colab : tout ce que vous devez savoir. Geekflare 2022. <https://geekflare.com/fr/google-colab/> (accessed May 29, 2023).
- [57] L B. Jupyter Notebook : tout savoir sur le notebook préféré des Data Scientists n.d. <https://www.lebigdata.fr/jupyter-notebook> (accessed June 11, 2023).
- [58] Anaconda pour Python - Présentation et installation | Jedha n.d. <https://www.jedha.co/formation-python/anaconda-python> (accessed June 11, 2023).
- [59] Python : définition et utilisation de ce langage informatique n.d. <https://www.journaldunet.fr/web-tech/dictionnaire-du-webmastering/1445304-python-definition-et-utilisation-de-ce-langage-informatique/> (accessed May 29, 2023).
- [60] Keras : une bibliothèque open source pour la constitution de réseaux neuronaux. IONOS Digital Guide 2020. <https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/quest-ce-que-keras/> (accessed May 29, 2023).
- [61] L B. TensorFlow : tout savoir sur la bibliothèque Machine Learning open source n.d. <https://www.lebigdata.fr/tensorflow-definition-tout-savoir> (accessed May 29, 2023).
- [62] rédac T. NumPy : la bibliothèque Python la plus utilisée en Data Science. Formation Data Science | DataScientest.com 2021. <https://datascientest.com/numpy> (accessed May 29, 2023).
- [63] user. Matplotlib : Tout savoir sur la bibliothèque Python de Dataviz. Formation Data Science | DataScientest.com 2021. <https://datascientest.com/matplotlib-tout-savoir> (accessed May 30, 2023).
- [64] pythonforge. Le module os en Python - système d'exploitation. Pythonforge 2021. <https://pythonforge.com/module-os-systeme-dexploitation/> (accessed May 30, 2023).
- [65] La bibliothèque GUI Python Tkinter – Très Facile n.d. <https://www.tresfacile.net/la-bibliotheque-gui-python-tkinter/> (accessed May 30, 2023).