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

N° d'ordre :IVA16/IVA/M2/2023

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Image et vie artificielle**

Adversarial Image Generation using Optimisation

Par :

NEZAR TEMIM

Soutenu le19/06/2023 devant le jury composé de :

Babahenini Djihane	MCB	Président
Mokhtari Bilal	MCA	Rapporteur
Boucetta Mebarek	MAA	Examinateur

Année universitaire 2022-2023

Remerciements

Je tiens à exprimer ma profonde gratitude envers mon superviseur, le Dr Mokhtari Bilal, pour ses précieux conseils, son soutien indéfectible et son mentorat d'expert tout au long de ce projet. Sa dévotion à l'excellence et sa passion pour la recherche ont été essentielles pour façonner ma compréhension et me pousser à donner le meilleur de moi-même. Je suis vraiment reconnaissant pour sa patience, son encouragement et ses commentaires éclairés, qui ont considérablement amélioré la qualité de ce travail.
J'aimerais également exprimer ma sincère appréciation envers ma famille pour leur amour constant, leur encouragement et leur croyance inébranlable en mes capacités. Leur soutien a été une source de force et de motivation, m'inspirant à persévérer face aux défis et à poursuivre mes aspirations académiques. Je leur suis redevable pour leurs sacrifices et leur compréhension tout au long de mon parcours de recherche.
À mes amis, je suis immensément reconnaissant pour leur soutien constant, leur compréhension et leur encouragement. Leur présence a apporté joie et rires à ma vie, rendant le processus de recherche plus agréable et mémorable. Leur croyance inébranlable en mes capacités et leur volonté de donner un coup de main ont été inestimables.

NEZAR Temim

Résumé

Les systèmes de classification et de reconnaissance d'images ont connus un grand progrès grâce à l'utilisation des réseaux profond. Cependant, ces systèmes sont vulnérables aux attaques d'images contradictoires, ce qui peut entraîner des erreurs de classification. Nous avons proposé une approche novatrice utilisant des images contradictoires qui représentent des images contenant un bruit invisible à l'oeil humain, mais qui peuvent inciter les modèles d'apprentissage profond à faire des mauvaise prédictions. Ces images vont donc être utilisées pour améliorer les performances des modèles d'apprentissage automatique au lieu de les affaiblir. Elle sont donc considérées comme étant une opportunité plutôt qu'un défaut. La généaration de ce type d'images se fait en utilisant les réseaux antagonistes génératifs, et un algorithme d'optimisation permettant de sélectionner les meilleures images parmi toutes les images générées. Les images générées enrichissent ensuite la base de données d'entraînement, ce qui va avoir un impact positif sur la précision du modèle. Les résultats obtenus sur différentes bases de données montrent une augmentation significative de la précision et du taux d'apprentissage correct grâce à l'utilisation de ces images générées.

Abstract

Classification and image recognition systems have made significant progress thanks to the use of deep networks. However, these systems are vulnerable to adversarial attacks, which can lead to classification errors. We have proposed an innovative approach using adversarial images that contain imperceptible noise to the human eye but can mislead deep learning models into making incorrect predictions. These images are used to improve the performance of machine learning models rather than weaken them, presenting an opportunity rather than a flaw. The generation of such images is accomplished using generative adversarial networks, along with an optimization algorithm to select the best images from the generated set. These generated images then enrich the training database, resulting in a positive impact on model accuracy. Results obtained on different databases demonstrate a significant increase in precision and correct learning rate through the use of these generated images.

ملخص

حققت أنظمة تصنيف الصور والتعرف عليها تقدمًا كبيرًا بفضل استخدام الشبكات العميقة. ومع ذلك ، فإن هذه الأنظمة عرضة لهجمات الصور المتضاربة ، والتي يمكن أن تؤدي إلى سوء التصنيف. لقد اقترحنا نهجًا جديدًا باستخدام الصور العدائية التي تمثل صورًا تحتوي على ضوضاء غير مرئية للعين البشرية ، ولكنها يمكن أن تخدع نماذج التعلم العميق في ارتكاب أخطاء في التنبؤ. لذلك سيتم استخدام هذه الصور لتحسين أداء نماذج التعلم الآلي بدلاً من إضعافها. لذلك فهي تعتبر فرصة وليست عيبًا. يتم إنشاء هذا النوع من الصور باستخدام شبكات الخصومة التوليدية ، وخوارزمية تحسين تسمح بتحديد أفضل الصور من بين جميع الصور التي تم إنشاؤها. ثم تثري الصور التي تم إنشاؤها قاعدة بيانات التدريب ، مما سيكون له تأثير إيجابي على دقة النموذج. النتائج التي تم الحصول عليها في قواعد البيانات المختلفة تظهر زيادة كبيرة في الدقة ومعدل التعلم الصحيح بفضل استخدام هذه الصور التي تم إنشاؤها.

Table des matières

Table des figures

Introduction générale

Les systèmes de vision par ordinateur sont devenus de plus en plus performants, et cela grâce à des algorithmes sophistiqués notamment ceux utilisant l'apprentissage profond. Cependant, ces systèmes sont également connus pour leur vulnérabilité aux attaques, représentées sous forme de perturbations appliquées aux images qui peuvent conduire à une classification erronée. Ces images sont appelées images antagonistes (contradictoires). Elles sont visuellement similaires aux images originales, mais peuvent conduire à des mauvaises décisions établis par les réseaux profonds. En revanche, cette nuisance est peut-être exploitée comme étant une opportunité et non pas un défaut, et cela en essayant de générer des images antagonistes de telle sorte qu'elles seront utilisées pour améliorer la performance des modèles d'apprentissage automatique plutôt que de les affaiblir. Dans ce travail donc, nous proposons un système basé sur la génération d'images en utilisant l'une des techniques les plus répondues, en occurrence les réseaux antagonistes génératifs. De plus, nous exploitant l'optimisation pour obtenir les meilleures images antagonistes qui permettent à un réseau profond d'avoir une meilleure performance. Les images générées sont donc utilisées pour augmenter la base originale, et à force que le réseau apprend à partir des images antagonistes, il va forcément améliorer sa performance.

Ce manuscrit est organisé comme suit. Dans le premier chapitre, nous décrivons le domaine de la génération d'images, les techniques utilisées pour la génération des images, ainsi que les réseaux antagonistes génératifs. Dans le deuxième chapitre, nous présentons les différentes approches utilisées pour la génération d'images antagonistes en utilisant les techniques d'optimisation permettant d'avoir les meilleures images. Nous décrivons ensuite dans le dernier chapitre la conception du système proposé, y compris les méthodes d'optimisation utilisées, les contraintes appliquées pour garantir la réalisme des images générées, ainsi que les métriques d'évaluation utilisées pour mesurer la qualité des résultats. Nous présentons également les résultats obtenus lors de l'évaluation de ce système sur différentes bases de données.

Chapitre 1

Génération d'images et Réseaux Antagistes Génératifs

1.1 Introduction

La génération d'images est un domaine de l'intelligence artificielle qui vise à créer des images réalistes et de haute qualité à partir de données brutes. Elle peut être utilisée pour la génération d'images réalistes, l'amélioration et la restauration d'images, la génération des images artistiques, etc. La génération d'images peut ne pas être bénéfiques, bien loin de cela, ce qui est le cas pour la génération d'images trompeuses en utilisant les réseaux antagonistes génératifs (GAN). C'est l'un des outils les plus populaires utilisées pour créer générer des images incluant un bruit non visible à l'un humain, mais qui trompe les réseaux profonds, ce qui peut conduire à des graves conséquences. Dans ce premier chapitre, nous allons décrire le domaine de la génération d'images dans le contexte de l'utilisation des GANs, tout en illustrant les avantages et les défis de ce type de techniques. Nous aborderons également l'impact potentiel des GANs sur les industries créatives et les futures applications.

1.2 Génération d'images

1.2.1 Définition de la génération d'images

La génération d'images fait référence à la création d'images à partir de rien ou à partir d'images existantes, à l'aide d'algorithmes informatiques qui utilisent souvent des modèles d'apprentissage automatique tels que les réseaux de neurones convolutionnels (CNN) ou les réseaux de neurones génératifs (GAN). Les images générées peuvent par exemple créer des besoins créatifs et artistiques, ou pour d'autres besoins plus techniques et spécifiques comme l'évaluation des performances de modèles et d'algorithmes de machine learning. [1].
Les réseaux antagonistes génératifs, ou GAN, est une techniques très utilisée pour la génération des images, telle que la génération de visage par exemple. Les réseaux GANs sont composées de deux parties : un générateur et un discriminateur. Le générateur permet de créer une nouvelle image à partir d'une image existante, le plus souvent en injectant un bruit non visible à l'œil humain, en produisant une image de mauvaise qualité qui s'améliore au cours de l'apprentissage. Le discriminateur en revanche permet

de distinguer les images créer par le générateur et les images originales [2]. A la fin du processus d'apprentissage, le générateur va apprendre à créer des images vraisemblables aux images originales et de très bonne qualité, voir meilleurs, et qui seront difficile à distinguer par le discriminateur. En effet, la génération d'images peut être réalisée à l'aide d'autres techniques, telles que les auto-encodeurs variationnels (VAEs), les modèles autorégressifs, etc. Les VAEs implique l'encodage et le décodage d'images dans un espace latent, tandis que les modèles autorégressifs reposent sur la modélisation de la distribution de probabilité des pixels de l'image à travers une séquence de probabilités conditionnelles. Ces méthodes trouvent des applications pratiques dans l'infographie, l'art et le design, ainsi que la réalité virtuelle et réalité augmentée, etc [3].
Dans ce qui suit, nous allons introduire les différentes techniques existantes utilisées dans le domaine de la génération d'images.

1.2.2 Méthodes de la génération d'images

Le domaine de la génération d'images progresse rapidement, et il existe une variété de méthodes utilisées dans ce domaine. La figure 1.1 résume les différents types de techniques utilisés dans le domaine de la génération d'images.

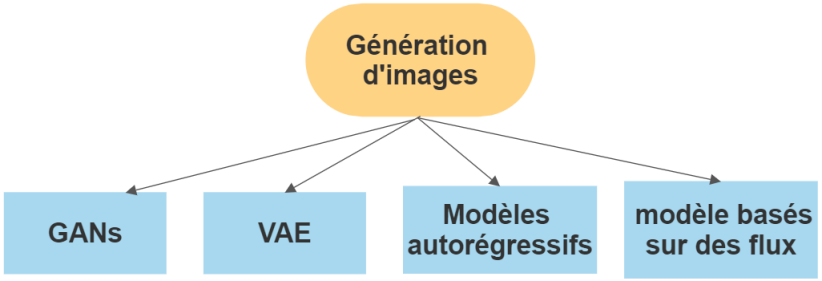

FIGURE 1.1 – Différents types des techniques utilisées dans la génération d'images

Dans ce qui suit, nous allons décrire chacune de ces techniques.

1.2.2.1 Réseaux Antagonistes Génératifs (GAN)

Les GAN font partie des techniques les plus utilisées et les plus efficaces pour la génération d'images. La figure 1.12 montre la architecture et le diagramme d'action de GAN.
Dans la littérature, il existe différents types de réseaux GANs. La figure 1.2 illustre ces différents types.

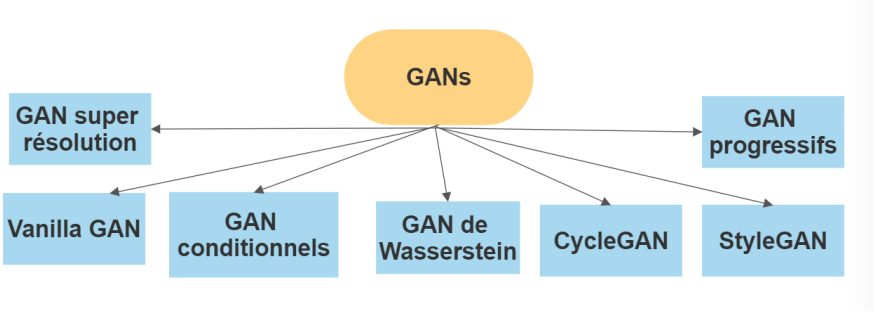

FIGURE 1.2 – Différents types des modèles de GANs

— **La super résolution GAN (SRGAN) :** Est un type de réseau antagoniste génératif (GAN) utilisé pour les tâches de super résolution d'image. La super-résolution fait référence au processus d'augmentation de la résolution d'une image.
— **Vanilla GAN :** C'est représentent l'architecture GAN de base, composée d'un générateur et d'un discriminateur [4]. Le générateur prend un bruit aléatoire en entrée et génère des images, tandis que le discriminateur décide si une image est authentique ou fausse.
— **Conditional GAN :** Les réseaux antagonistes génératifs conditionnels (cGANs) sont une extension des réseaux antagonistes génératifs (GANs) classiques. Ils utilisent une condition supplémentaire, telle qu'une étiquette ou une donnée d'entrée, pour guider la génération d'images. Tant le générateur que le discriminateur prennent en compte cette condition lors de leur fonctionnement [5]. Le générateur crée des images correspondant à la condition spécifiée, tandis que le discriminateur évalue la qualité des images générées et leur adéquation à la condition. et d'autres tâches où la génération d'images doit être contrôlée et personnalisée.
— **CycleGAN :** C'est un type de GAN qui implique deux générateurs et deux discriminateurs, qui sont utilisés pour superposer les images d'entrée aux images de sortie et vice versa. Il est couramment utilisé pour les tâches de traduction d'image en image [6].
— **WGAN :** C'est un type de GAN qui utilise la distance de Wasserstein comme mesure de distance entre les distributions générées et réelles. Il est plus stable que le GAN vanille, et peut générer des images de haute qualité [7].
— **StyleGAN :** C'est un type de GAN qui utilise un réseau de mappage pour mapper un vecteur de bruit aléatoire à un espace latent [8], qui est ensuite utilisé pour générer des images de haute qualité avec un style et une apparence contrôlables.
— **GAN progressifs :** Ce sont des réseaux antagonistes génératifs (GAN) qui génèrent des images de haute résolution en augmentant progressivement la résolution des images de sortie [9]. Le réseau du générateur démarre avec une faible résolution et ajoute des couches pour augmenter la résolution des images générées.

1.2.2.2 Autoencodeurs variationnels (VAE)

Les Autoencodeurs variationnels VAE sont des méthodes populaires de génération d'images. Elles impliquent d'encoder une image dans un espace latent, puis de la décoder à nouveau en une nouvelle image. Les réseaux d'encodeur et de décodeur sont formés simultanément pour minimiser l'erreur entre les images originales et reconstruites. La figure 1.3 montre l'architecture de travail de Autoencodeurs variationnels [3].

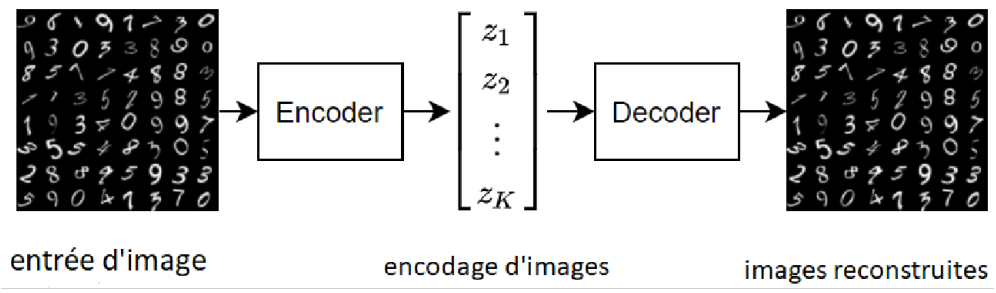

FIGURE 1.3 – Architecture de travail de autoencodeurs variationnels [3]

Comparaison des résultats d'entrée et de sortie après l'utilisation de VAE est illustrée la figure 1.4.

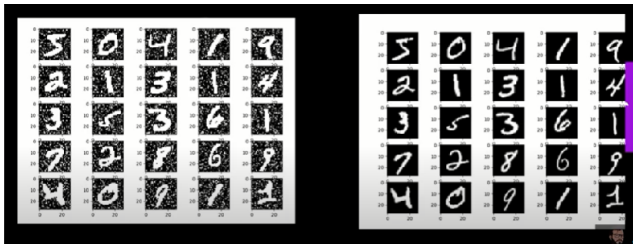

FIGURE 1.4 – La différence entre les résultats d'entrée et de sortie après utilisation de VAE

La figure 1.5 montre les différents types d'auto-encodeurs variationnels (VAE)

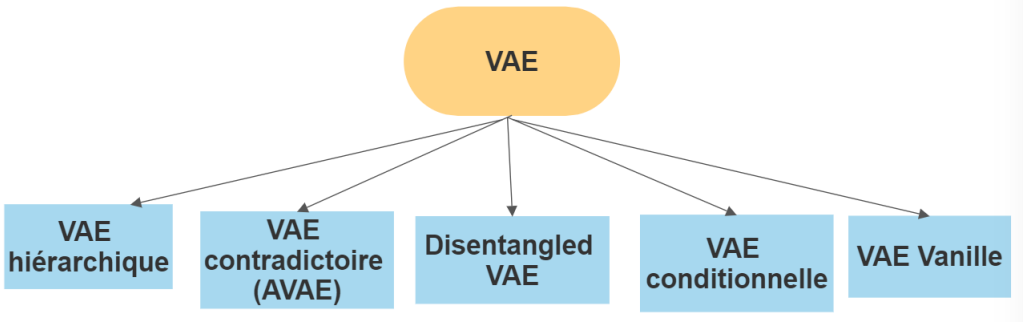

FIGURE 1.5 – Différents types des modèles des autoencodeurs variationnels (VAE)

— **VAE standard :** Il s'agit du type de VAEs le plus basique, qui utilise un réseau de neurones entièrement connecté pour l'encodeur et le décodeur [10].
— **VAE convolutif (CVAE) :** Il s'agit d'un VAE qui utilise des réseaux de neurones convolutifs pour l'encodeur et le décodeur [11]. Les CVAE sont couramment utilisés sur des données de type images.
— **VAE récurrent (RVAE) :** Il s'agit d'un VAE qui utilise des réseaux de neurones récurrents pour l'encodeur et le décodeur [12]. Les RVAE sont couramment utilisés pour les données séquentielles, telles que les données de séries chronologiques ou le traitement du langage naturel.
— **Adversarial autoencoder (AAE) :** Il s'agit d'un type de VAE qui utilise le mécanisme d'entraînement contradictoire des GAN pour améliorer la qualité des données générées [13].
— **VAE désenchevêtré :** Le VAE désenchevêtré est une amélioration du VAE classique qui cherche à obtenir des représentations de données plus claires et explicites. Il utilise des méthodes telles que des contraintes spécifiques, des régularisations et des architectures de réseau adaptées pour favoriser des dimensions latentes indépendantes et liées à des caractéristiques spécifiques des données [14]. L'objectif est de créer des représentations désenchevêtrées pour faciliter l'interprétation et l'utilisation des données dans divers domaines.
— **VAE Hiérarchique (HVAE) :** Il s'agit d'un VAE qui utilise une structure hiérarchique pour apprendre des représentations hiérarchiques des données [15]. Les HVAE sont couramment utilisés pour les données qui ont plusieurs niveaux d'abstraction, comme les images ou le langage.

1.2.2.3 Modèles autorégressifs

Les modèles autorégressifs sont des modèles de génération de données qui se basent sur les dépendances séquentielles. Ils génèrent les éléments d'une séquence un par un, en utilisant les éléments précédemment générés comme contexte. Ces modèles sont souvent utilisés dans le traitement du langage naturel et la génération de texte, et peuvent également être appliqués à d'autres types de données séquentielles. Ils captent les relations entre les éléments d'une séquence et permettent de générer des séquences cohérentes [16]. Cependant, ils peuvent être lents à générer les données et peuvent être sensibles aux erreurs accumulées. La figure 1.7 montre l'architecture de travail de Modèles autorégressifs.

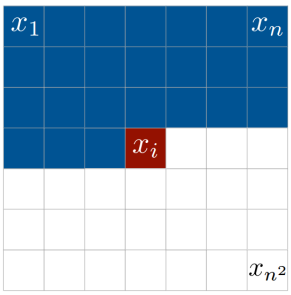

FIGURE 1.6 – L'architecture de travail de modèles autorégressifs [17]

La figure 1.7 montre les différents types des modèles autorégressifs.

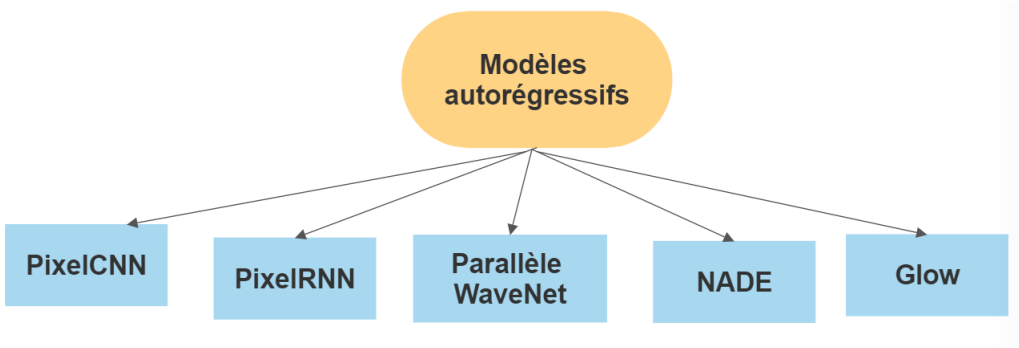

FIGURE 1.7 – Différents types des modèles des modèles autorégressifs

— **PixelCNN :** PixelCNN est un type de modèle autorégressif qui prédit les intensités de pixels d'une image en tenant compte des valeurs de tous les pixels générés précédemment [18]. PixelCNN utilise un réseau neuronal convolutif (CNN) pour modéliser la distribution conditionnelle de chaque pixel compte tenu des pixels précédents. La figure 1.8 illustre le fonctionnement de model PixelCNN.

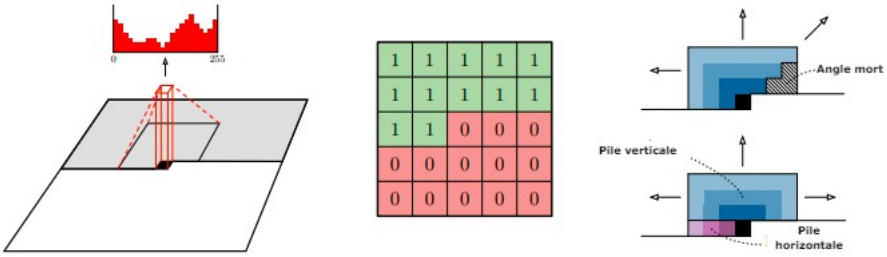

FIGURE 1.8 – Fonctionnement de model pixelCNN [19].

— **PixelRNN :** PixelRNN est un autre type de modèle autorégressif qui modélise la distribution des intensités de pixels à l'aide d'un réseau neuronal récurrent (RNN) [18]. PixelRNN prédit la valeur de pixel en prenant en compte les pixels précédents de manière séquentielle.
— **Parallel WaveNet :** Parallel WaveNet est un modèle autorégressif qui a été développé à l'origine pour la synthèse vocale, mais il peut également être utilisé pour la génération d'images [20]. Parallel WaveNet utilise un réseau neuronal convolutif dilaté pour modéliser la distribution conditionnelle des intensités des pixels. Contrairement à PixelCNN, Parallel WaveNet peut générer des images en parallèle, ce qui le rend plus efficace en termes de calcul.
— **NADE (Neural Autoregressive Density Estimator) :** NADE utilise un réseau de neurones à anticipation pour modéliser la distribution conditionnelle des intensités de pixels [21]. NADE est efficace en termes de calcul car il peut prédire la prochaine valeur de pixel en temps constant.
— **Glow :** Utilise un modèle génératif basé sur le flux pour générer des images. Glow transforme l'image d'origine en une séquence de caractéristiques de dimension inférieure et applique des transformations inversibles pour modéliser la distribution conditionnelle de chaque caractéristique compte tenu des caractéristiques précédentes [22].

1.2.2.4 Modèles basés sur des flux

Les modèles basés sur les flux sont des modèles génératifs utilisés principalement dans le domaine de la vision par ordinateur. Ils permettent de transformer une distribution de probabilité simple en une distribution cible représentant les données d'intérêt. Cette transformation est réalisée par un réseau de neurones appelé transformateur de flux, qui est construit de manière à être inversible. Pendant l'apprentissage, les modèles optimisent les paramètres du transformateur de flux pour minimiser la divergence entre la distribution cible et les données réelles [23]. La figure 1.9 montre l'architecture de travail les modèles basés sur des flux.

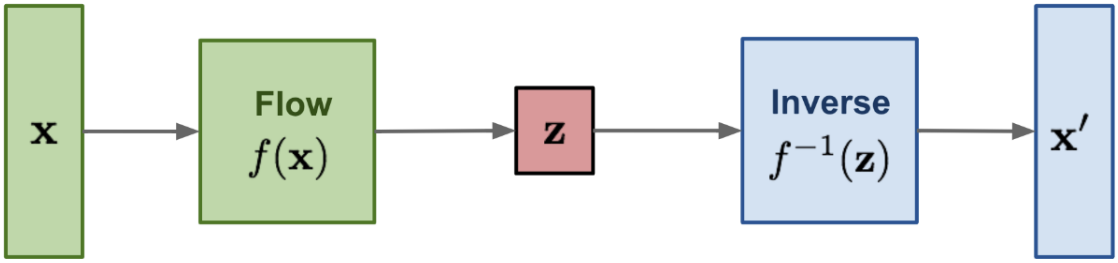

FIGURE 1.9 – Architecture de travail les modèles basés sur des flux [24]

La figure 1.10 montre Normalize the flow on an image sur les modèles basés sur des flux.

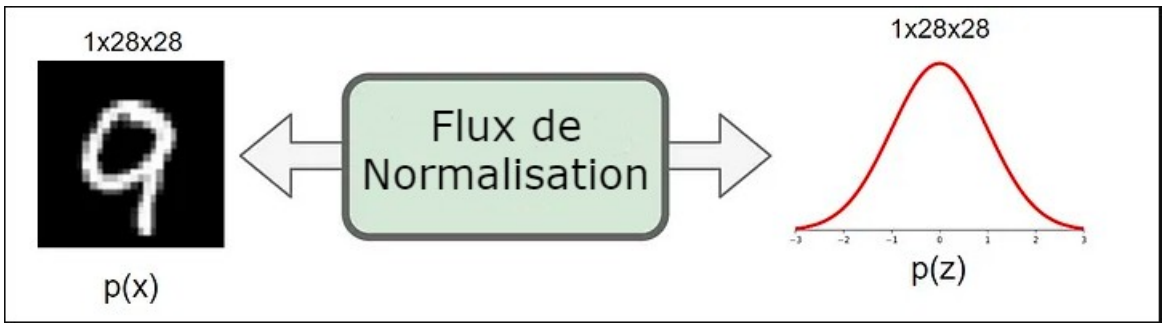

FIGURE 1.10 – Normalize the flow on an image.

La figure 1.11 montre les différents types des modèles basés sur des flux.

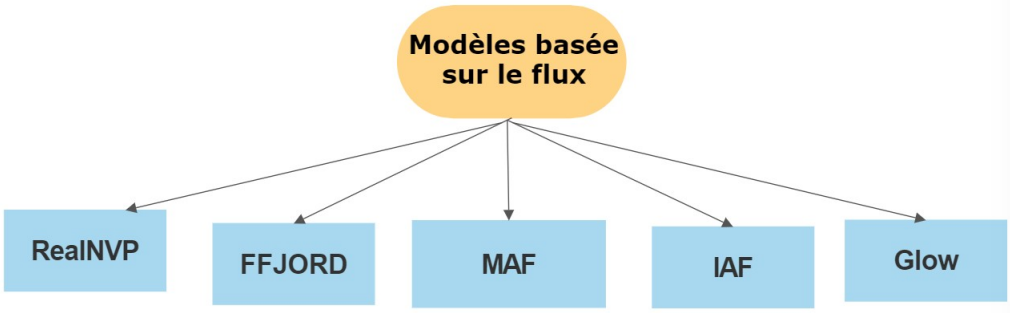

FIGURE 1.11 – Différents types des modèles basés sur des flux.

— **RealNVP (Real-valued Non-Volume Preserving) :** RealNVP est un modèle qui utilise un réseau neuronal inversible pour modéliser la fonction de densité de probabilité des données [25]. Il prend une séquence de transformations inversibles qui mappent la distribution simple à la distribution cible.
— **Glow :** Glow est un autre type de modèle basé sur le flux qui utilise une séquence de transformations inversibles pour modéliser la fonction de densité de probabilité des données. Glow utilise des couches de couplage affines pour modéliser la distribution complexe des données [26]. Il peut générer des images de haute qualité avec un grand nombre de pixels.

— **FFJORD (Free-form Jacobian with ODE-based Regularization for Density estimation) :** FFJORD utilise un solveur ODE (équation différentielle ordinaire) pour calculer les transformations avant et arrière [27]. FFJORD utilise une formulation en temps continu pour modéliser la densité des données.
— **MAF (Masked Autoregressive Flow) :** Modélise la fonction de densité de probabilité des données à l'aide d'un réseau neuronal autorégressif [28]. Il applique une séquence de transformations affines, dont chaque transformation étant appliquée à la sortie de la transformation précédente.
— **IAF (Inverse Autoregressive Flow) :** Utilise un réseau neuronal autorégressif inversible pour modéliser la fonction de densité de probabilité des données [29]. IAF applique une séquence de transformations inversibles qui mappe la distribution simple à la distribution cible.

1.3 Réseaux antagonistes génératifs GANs

1.3.1 Présentation des GANs

Les réseaux antagonistes génératifs (GAN) sont un type de modèle d'apprentissage en profondeur largement utilisé pour générer de nouvelles données (images) similaire à un ensemble données original. Ian Goodfellow et al. [30] qui ont introduit le concept des GANs en 2014, et ils sont depuis devenus un outil populaire dans les domaines de l'apprentissage automatique et de l'intelligence artificielle [31]. La figure 1.12 montre la architecture et le diagramme d'action de GAN.

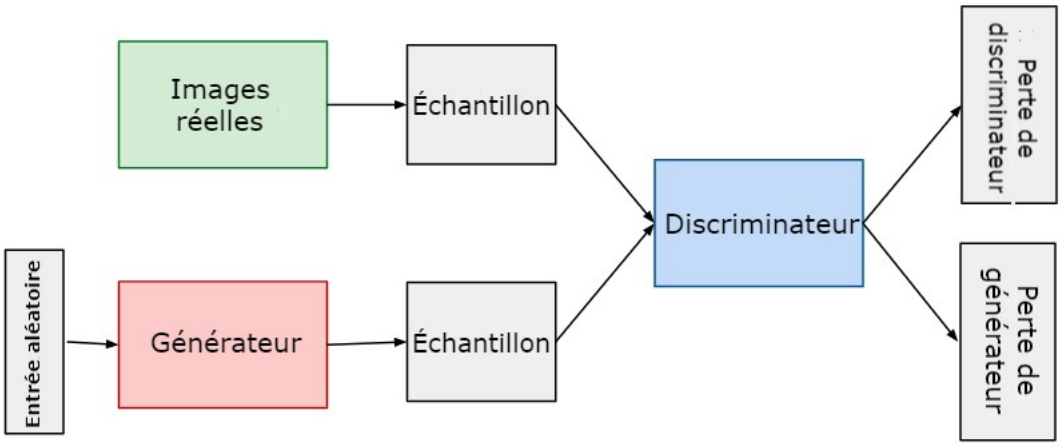

FIGURE 1.12 – Architecture de GAN [32]

Les GAN sont composés de deux réseaux de neurones : un générateur et un discriminateur. Le générateur prend en compte une image d'entrée sur laquelle un bruit aléatoire est appliquer afin de génèrer de nouvelles données qui ressemblent au fil du temps aux données d'apprentissage [33]. Le discriminateur, quand à lui, tente de faire la différence entre les données générées par le générateur et les données originales [30].
Pendant le processus de formation, le générateur et le discriminateur sont formés simultanément. Le générateur vise à tromper le discriminateur en générant des données que le discriminateur classe comme authentiques, tandis que le discriminateur s'efforce

à identifier avec précision les données générées comme étant fausses [31]. L'interaction entre les deux réseaux aboutit finalement à ce que le générateur apprenne à produire des données qui correspondent étroitement aux données d'entraînement, et qui sont difficilement différencier des données originale [33].
Cependant, Les GAN ont été initialement conçus pour tromper les réseaux profonds. Au fil de l'entraînement, les GAN ont démontré leur capacité à générer des échantillons de haute qualité et à tromper les réseaux profonds. [31].
Les GAN présentent plusieurs avantages, sont sa capacité à générer de nouvelles données diverses et représentatives qui capturent la distribution sous-jacente des données de formation [33]. Ils ont été appliqués à une variété de tâches, y compris la génération d'images et de vidéos, la génération de texte et même la découverte de médicaments.

1.3.2 Bref historique et évolution des GANs

Les réseaux antagonistes génératifs (GAN) ont été introduits pour la première fois en 2014 par Ian Goodfellow et al. [34]. L'idée originale derrière les GAN était de créer un modèle capable de générer de nouvelles données ressemblant étroitement à un ensemble données d'entraînement. L'architecture proposée dans l'article original consistait en un générateur et un discriminateur, qui ont été entraînés à l'aide d'un cadre de jeu minimax.

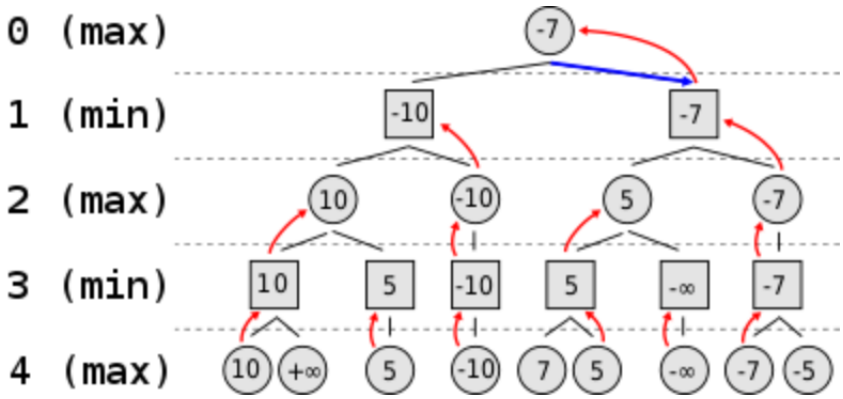

FIGURE 1.13 – Architecture de jeu minimax [35].

Depuis leur introduction, les GAN ont connu un développement important et diverses modifications et améliorations ont été proposées. L'une des premières modifications était les GAN conditionnels, qui ont été introduites la même année. Ils permettaient de générer des données conditionnées sur une classe ou un attribut particulier. Cette modification a rendu les GAN adaptés à des tâches telles que la traduction d'image à image.
En 2015, les DCGAN (Deep Convolutional GAN) [34] ont été proposés. Ils utilisaient des réseaux de neurones de convolutions pour générer des images plus réalistes et de plus haute résolution. En 2016, CycleGAN a été introduit [6], permettant de générer de nouvelles images à partir d'un domaine différent sans la nécessite de la mise en correspondance dans le domaine source.
Les progrès les plus récents dans les GAN incluent les GAN progressifs, qui peuvent générer des images haute résolution de manière progressive, ainsi que StyleGAN, qui permet un contrôle précis des attributs des images générées. Depuis, le développement des l'application utilisant des GAN a connu beaucoup de progrès dans différents domaines, tels que le traitement de texte, la musique, etc.

Dans l'ensemble, les GAN sont devenus un outil puissant pour générer des nouvelles données qui ressemblent aux données originales, ouvrant des possibilités pour de nombreuses applications passionnantes dans divers domaines.

1.3.3 Fonctionnement d'un GAN

Un GAN (Generative Adversarial Network) est créé en suivant plusieurs étapes. Tout d'abord, on définit la sortie finale souhaitée du GAN, puis on compile un jeu initial de données d'apprentissage basé sur ces paramètres [36]. Ensuite, ces données sont utilisées pour entraîner un générateur à produire des sorties similaires à celles souhaitées. Au début, les sorties du générateur peuvent être de mauvaise qualité, mais le générateur s'améliore progressivement en comparant ses sorties avec les sorties réelles souhaitées et en ajustant ses paramètres. Ce processus est répété jusqu'à ce que le générateur atteigne un niveau de précision satisfaisant dans la production des sorties souhaitées [31].
Ensuite, les images générées sont introduites dans le discriminateur avec les points de données réels du conception d'origine [36]. Le discriminateur filtre les informations et retourne une probabilité de 0 ou 1 pour représenter l'authenticite de chaque image (1 pour image vraie et 0 pour image fausse).
Enfin, ces valeurs sont vérifiées manuellement et répétées jusqu'à ce que le résultat souhaité soit obtenu. Deux réseaux tentent d'optimiser des fonctionnalités différentes et opposées dans un jeu à somme nulle.

1.3.4 Présentation de la structure du GAN

Un réseau antagoniste génératif (GAN) se compose de deux parties :

— le générateur apprend à produire des données réalistes en utilisant les sorties du discriminateur comme exemples négatifs. Le générateur cherche à tromper le discriminateur en générant des sorties indiscernables des vraies données. Il ajuste ses paramètres en fonction des commentaires du discriminateur pour améliorer la qualité de ses sorties. Ainsi, le générateur s'entraîne à générer des données fiables en se basant sur les retours du discriminateur, ce qui conduit à des sorties de plus en plus réalistes au fil de l'entraînement.

— Le discriminateur apprend à distinguer les fausses des vraies données. Le discriminateur punit le générateur pour avoir produit des résultats incompréhensibles.

Lorsque l'apprentissage démarre, le générateur produit une fausse donnée simple et claire, et l'avantage est qu'il apprend vite puisqu'il n'a aucun mal à dire qu'il s'agit d'une fausse donnée. La figure 1.14 montre la production de la première génération d'images par le GAN.

FIGURE 1.14 – Première génération de génération d'images produite par un GAN.

Fur et à mesure que l'entraînement progresse, la sortie du générateur peut tromper le discriminateur. La figure 1.15 montre la production de la deuxième génération d'images.

FIGURE 1.15 – Deuxième génération d'images produite par le GAN.

Enfin, si le générateur fonctionne bien, les résultats seront excellents, ce qui affecte négativement le discriminateur pour distinguer le vrai du faux. Il commence à classer les fausses données comme réelles et leur précision diminue. La figure 1.16 montre la dernière génération ou l'accès aux images finales par GAN.

FIGURE 1.16 – Capacité GAN à générer des images indiscernables de l'original.

1.3.5 Processus de formation du GAN pour la génération d'images

Le générateur est formé pour produire des images réalistes qui peuvent tromper le discriminateur [37], tandis que le discriminateur est formé pour faire la distinction entre les images réelles et les images générées. Ce processus est connu sous le nom de formation contradictoire, où les deux réseaux sont formés simultanément [34]. La figure 1.17 montre formation de GAN pour la génération d'images.
Le générateur et le discriminateur sont entraînés à l'aide de différentes fonctions de perte. Le générateur est formé pour minimiser la différence entre les images générées et les images réelles, tandis que le discriminateur est formé pour maximiser la différence entre les images réelles et fausses [37]. Pendant l'entraînement, le générateur génère de fausses images à partir d'un bruit aléatoire et le discriminateur essaie de différencier les images réelles des fausses [30]. Les fonctions de perte du générateur et du discriminateur sont mises à jour en fonction de leurs performances, et ce processus est répété jusqu'à ce que le générateur puisse produire des images de haute qualité pouvant tromper le discriminateur.
Après la formation, le GAN peut être évalué en générant de nouvelles images et en évaluant leur qualité. De plus, le GAN peut être affiné en ajustant les hyperparamètres [37], en modifiant l'architecture ou en utilisant différentes fonctions de perte pour améliorer la qualité des images générées [31]. Bien que la formation des GAN pour la génération d'images puisse être difficile et prendre du temps, il s'agit d'une technique puissante pour générer des images de haute qualité.

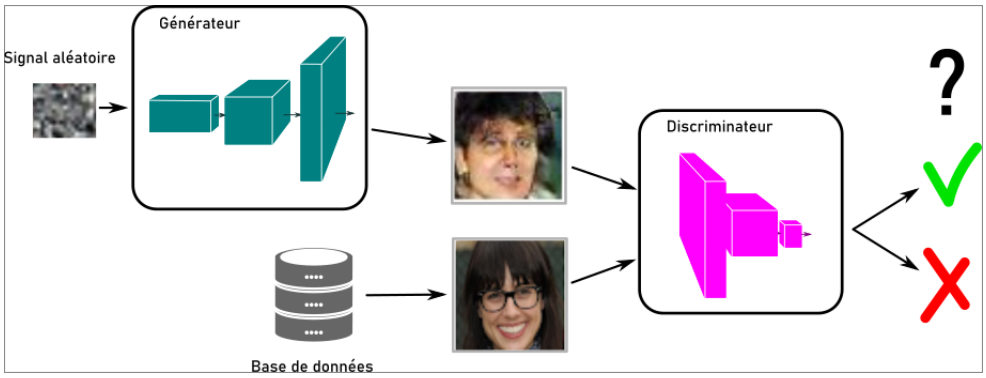

FIGURE 1.17 – Schéma d'entrainement d'un GAN.

1.4 La relation entre la génération d'images et les GAN

La génération d'images et les GAN (Generative Adversarial Networks) sont étroitement liés. La génération d'images est le processus de création d'images à partir de rien, souvent à l'aide de modèles informatiques [1]. Pour générer des images réalistes, les modèles de génération d'images doivent comprendre les caractéristiques visuelles et les motifs présents dans les données d'entraînement. C'est là que les GAN interviennent. Les GAN avec un générateur d'images sont donc des outils puissants pour la génération d'images réalistes dans divers domaines, allant de l'art génératif à la synthèse d'images pour des applications pratiques.

1.5 Applications des GAN pour la génération d'images

Les GAN (Generative Adversarial Networks) sont largement utilisés pour la génération d'images et ont plusieurs applications dans ce domaine. Voici quelques-unes des principales applications des GAN pour la génération d'images :

1.5.1 Transfert de style

Le transfert de style est une technique qui combine le style d'une image de référence avec le contenu d'une autre image. Elle utilise des réseaux de neurones convolutifs pour extraire les caractéristiques de style et de contenu des images sources et de référence. En ajustant ces caractéristiques [38], une nouvelle image est générée qui conserve le contenu de l'image source tout en adoptant le style de l'image de référence. Le transfert de style est utilisé pour créer des œuvres d'art stylisées et permet d'explorer de nouvelles formes d'expression artistique en fusionnant des aspects visuels uniques. La figure 1.18 montre des exemples d'application de Transfert de style entre deux images différentes.

FIGURE 1.18 – Application de transfert de style sur les images [39].

Plusieurs algorithmes et méthodes sont utilisés pour le transfert de style. L'algorithme de transfert de style neuronal de Gatys et al. [38] est l'une des méthodes les plus populaires, qui utilise une combinaison de fonctions de perte de contenu et de perte de style pour optimiser l'image générée. En revanche, les styleGAN peuvent apprendre à générer de nouvelles images en incorporant à la fois les caractéristiques de style et de contenu d'une image d'entrée. La figure 1.19 montre la structure du travail de transfert de style dans le model styleGAN.

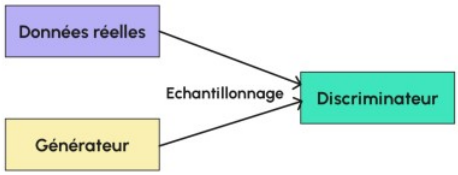

FIGURE 1.19 – Architecture des transfert de style avec styleGAN [40]

Le transfert de style a une large gamme d'applications dans l'infographie, l'art et le design [38]. Il peut être utilisé pour créer des images et des animations stylisées, des filtres et des effets artistiques, et synthétiser des images avec les styles souhaités. De plus, il est également utilisé dans les applications de réalité virtuelle et de réalité augmentée pour créer des environnements réalistes et immersifs. Le transfert de style est une technique puissante pour produire des images et des graphiques visuellement attrayants et esthétiques.

1.5.2 La super-résolution d'image

La super-résolution d'image est le processus de génération d'une image de haute résolution à partir d'une ou plusieurs images basse résolution [38]. L'objectif de la super-résolution est d'améliorer les détails et la netteté de l'image, la rendant plus attrayante visuellement et utile pour des applications telles que l'analyse d'images et la vision par ordinateur. La figure 1.20 montre le résultat de lapplication de la super-résolution d'image par le réseau antagoniste génératif pour obtenir une nouvelle image de meilleure qualité.

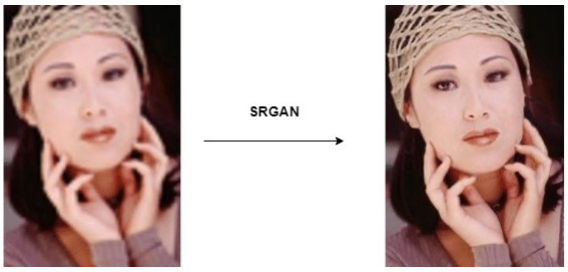

FIGURE 1.20 – Super-résolution d'image à l'aide d'un réseau antagoniste génératif [41].

Il existe deux approches principales de la super-résolution à savoir les méthodes basées sur l'interpolation et les méthodes basées sur l'apprentissage [42]. Les méthodes basées sur l'interpolation utilisent des techniques simples telles que l'interpolation bicubique pour augmenter la taille de l'image. Ces méthodes sont rapides et simples, mais elles ne produisent pas de résultats de haute qualité. Les méthodes basées sur l'apprentissage, quant à elles, utilisent des techniques d'apprentissage automatique telles que les réseaux de neurones profonds pour apprendre la relation entre les images basse résolution et haute résolution. Ces méthodes sont plus complexes et prennent plus de temps, mais elles peuvent produire des résultats de haute qualité.
La super-résolution a de nombreuses applications dans divers domaines, tels que l'imagerie médicale, la télé-détection et la surveillance. Il peut être utilisé pour améliorer la résolution des images satellites, améliorer la précision des diagnostics médicaux et améliorer les détails des images des caméras de sécurité [43].

1.5.3 L'imagerie d'inpainting

L'imagerie d'inpainting est une technique utilisée en vision par ordinateur pour réparer ou remplacer les parties manquantes ou endommagées d'une image en utilisant des informations provenant des zones environnantes [42]. Il existe plusieurs applications pratiques, telles que la restauration de photos anciennes ou endommagées, l'effacement d'objets ou de personnes indésirables d'une image ou la génération d'images composites visuellement attrayantes.
Il existe plusieurs méthodes pour l'imagerie d'inpainting, y compris des techniques traditionnelles telles que la synthèse de texture et l'inpainting basé sur des patchs [44], ainsi que des approches plus récentes basées sur l'apprentissage en profondeur telles que les réseaux antagonistes génératifs (GAN) et les réseaux de neurones convolutifs (CNN).
Les méthodes traditionnelles consistent à identifier des patchs ou des textures similaires dans les zones environnantes d'une image et à les utiliser pour remplir les zones manquantes ou endommagées [44]. Bien que ces méthodes puissent produire des résultats satisfaisants [45], elles peuvent avoir du mal avec des motifs plus complexes ou irréguliers.
La figure 1.21 montre des exemples d'application de L'imagerie d'inpainting par le GAN sur des images différentes.

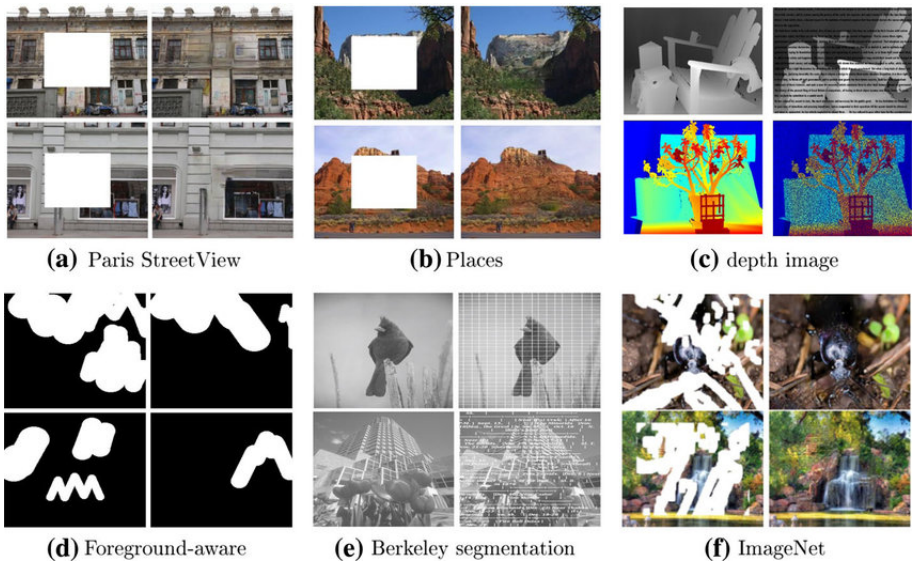

FIGURE 1.21 – L'imagerie d'inpainting à l'aide d'un GAN [46].

Les approches basées sur l'apprentissage en profondeur telles que les GAN et les CNN ont donné des résultats prometteurs pour l'imagerie d'inpainting. Ces méthodes apprennent à prédire les pixels manquants en fonction du contexte environnant et, bien qu'elles nécessitent beaucoup de données d'entraînement, elles peuvent produire des résultats plus réalistes et visuellement attrayants que les méthodes traditionnelles [42].

1.5.4 L'image de synthèse

L'image de synthèse est un domaine de l'informatique qui traite la génération de nouvelles images numériques à l'aide de techniques informatiques [47]. Ces techniques consistent à générer des images à partir de zéro ou en modifiant des images existantes. La figure 1.22 montre des exemples d'application d'images synthétiques générées par un GAN pour modifier les caractéristiques des personnes après un changement d'âge.

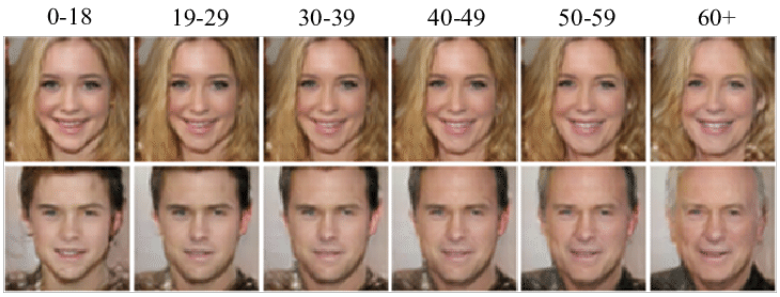

FIGURE 1.22 – Exemples d'images de synthèse générées par un GAN [46].

La synthèse d'images peut être réalisée à l'aide de différentes méthodes, telles que la génération procédurale qui utilise des algorithmes pour créer des images basées sur des règles mathématiques, ou la synthèse de texture qui implique la combinaison et la répétition d'éléments ou de textures d'images plus petits [48].
Une autre approche pour la synthèse d'images consiste à utiliser des algorithmes d'apprentissage profond, en particulier les réseaux antagonistes génératifs (GAN), qui ont démontré leur capacite à générer des images très réalistes [49].

Les applications de la synthèse d'images comprennent les graphismes informatiques, les jeux, la réalité virtuelle et les effets spéciaux dans les films [47]. Elle a également des applications pratiques dans des domaines tels que la médecine, où des images synthétiques peuvent être utilisées pour simuler des procédures médicales et la formation.

1.6 Défis et tendances

Les réseaux antagonistes génératifs (GAN) ont montré un succès remarquable dans la génération d'images de haute qualité qui sont visuellement similaires aux images du monde réel. Cependant, malgré les progrès considérables réalisés dans les GAN, plusieurs défis perssistent.

Voici les peincipales défis majeurs :

— **Mode Collapse :** Il fait référence à une situation où le GAN ne génère qu'un nombre limité d'échantillons qui ne sont pas assez diversifiés. Ce problème survient lorsque le générateur a tendance à produire des images similaires et que le discriminateur ne peut pas les différencier.
— **Manque de contrôle :** les GAN génèrent des images basées sur un vecteur de bruit aléatoire, ce qui rend difficile le contrôle de la sortie générée. Par conséquent, il est difficile de générer des images avec des attributs ou des caractéristiques spécifiques.
— **Métriques d'évaluation :** Il est difficile d'évaluer la performance des GAN en raison du manque de métriques d'évaluation objectives. Certaines métriques couramment utilisées incluent le score de début (IS) et la distance de début de Fréchet (FID), mais ces métriques ne sont pas toujours en corrélation avec la perception humaine de la qualité de l'image.
— **Biais des données :** les GAN apprennent à partir des données sur lesquelles ils sont formés, donc si les données de formation sont biaisées, les images générées peuvent également présenter un biais.
— **Ressources informatiques :** les GAN nécessitent de nombreuses ressources informatiques, en particulier pour la génération d'images de haute résolution. Le processus de formation peut prendre beaucoup de temps et nécessiter un matériel puissant, ce qui peut limiter l'accessibilité des GAN à un public plus large.

Les tendances de l'application des GAN pour la génération d'images peuvent inclure principalement les points suivants :

— **Contrôle amélioré :** Les chercheurs travaillent sur le développement de méthodes pour améliorer le contrôle des GAN sur les images générées. Ces méthodes incluent des GAN conditionnels, qui permettent au générateur de générer des images avec des attributs spécifiques, et des GAN à croissance progressive, qui génèrent des images haute résolution étape par étape.
— **Intégration des connaissances du domaine :** les GAN peuvent bénéficier de l'intégration des connaissances du domaine pour générer des images plus réalistes et plus précises. Ceci peut être réalisé en formant des GAN sur des ensembles de données spécifiques qui capturent les caractéristiques du domaine cible.
— **Modèles hybrides :** la combinaison de GAN avec d'autres modèles d'apprentissage en profondeur peut conduire à une génération d'images plus robuste et plus efficace.

Par exemple, les GAN peuvent être combinés avec des auto-encodeurs pour générer des images de haute qualité à partir d'entrées de faible qualité.

— **Traiter le biais :** résoudre le problème du biais des données est crucial dans l'application des GAN pour la génération d'images. Les chercheurs peuvent travailler au développement de méthodes pour atténuer le biais des données en augmentant les données de formation ou en utilisant des modèles génératifs moins sensibles au biais.

— **Meilleures métriques d'évaluation :** le développement de meilleures métriques d'évaluation en corrélation avec la perception humaine de la qualité de l'image peut améliorer l'évaluation des GAN. Ceci peut être réalisé en utilisant des évaluateurs humains pour évaluer la qualité des images générées.

1.7 Attaques contradictoires (adversarial attacks)

1.7.1 Description des attaques contradictoires

Les attaques contradictoires sont un type d'attaque appliquées sur les modèles d'apprentissage automatique, où une attaquant manipule délibérément les données d'entrée pour amener le modèle de CNN à faire des prédictions ou des décisions incorrectes. Ces attaques peuvent être menées en ajoutant de petits changements, souvent imperceptibles, aux données d'entrée, qui sont conçus pour tromper le modèle en classant mal l'entrée. Les attaques contradictoires peuvent être utilisées à diverses fins, telles que pour échapper aux systèmes de détection, pour voler des informations sensibles ou pour saboter les performances d'un modèle. La figure 1.23 montre Le processus de conversion d'une image originale en une image contradictoires.

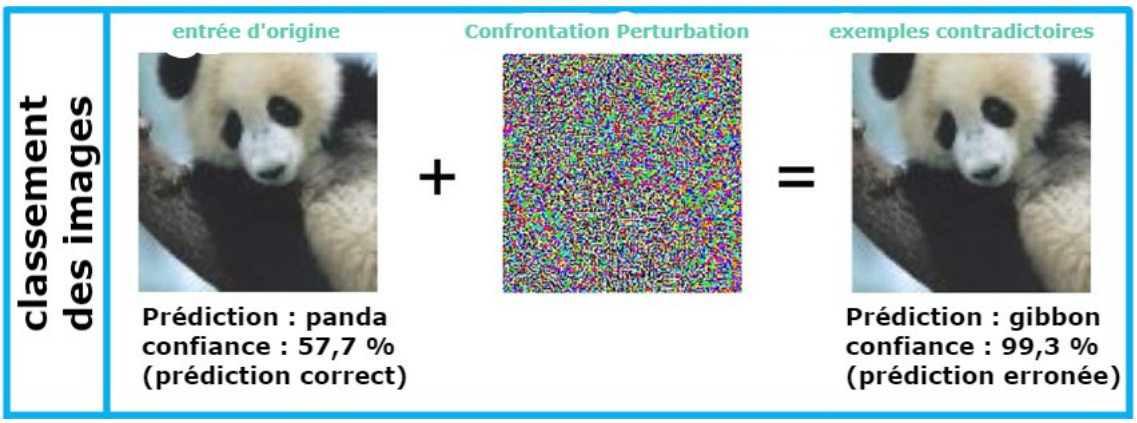

FIGURE 1.23 – Processus de conversion d'une image originale en une image contradictoires [50]

1.7.2 Utilisation des GANs pour la génération des images contradictoires

Les réseaux antagonistes génératifs (GANs) peuvent être utilisés pour générer des images antagonistes en entraînant le réseau générateur à produire des images spécifiquement conçues pour tromper le réseau discriminateur. En ce qui suit les étapes de base impliquées dans la génération d'images contradictoires à l'aide de GAN.

— **Former un GAN :** Tout d'abord, il faut former un GAN en lui fournissant un ensemble de données d'images.
— **Générer des exemples contradictoires :** une fois le GAN est formé, il faut générer des exemples contradictoires en manipulant l'entrée du réseau de générateurs. Le réseau de générateurs prend un vecteur de bruit aléatoire en entrée et produit une image. Pour générer un exemple contradictoire, une petite perturbation est ajutée au vecteur de bruit, qui est conçue pour amener le réseau discriminateur à mal classer l'image générée.
— **Évaluer les exemples contradictoires :** l'efficacité des exemples contradictoires est évaluée en le testant sur le réseau discriminateur. Si le réseau discriminateur classe par erreur les exemples contradictoires comme réels, la génération est alors considéréé comme réussie.

1.7.3 Différents types d'attaques contradictoires

Il existe plusieurs types d'attaques contradictoires qui peuvent être utilisées pour manipuler des images. En ce qui suit quelques types courants d'attaques contradictoires.
— **Fast Gradient Sign Method (FGSM) :** Cette attaque consiste à calculer le gradient de la fonction de perte par rapport à l'image d'entrée, puis à perturber légèrement l'image d'entrée dans la direction du gradient, généralement en fonction du signe du gradient. Cette attaque est relativement simple et rapide, mais peut être efficace contre de nombreux classificateurs d'images [51].
— **La méthode itérative de base (BIM) :** C'est une variante itérative de la méthode de signe de gradient rapide (FGSM) pour générer des exemples contradictoires. Le BIM est une attaque en boîte blanche, ce qui signifie qu'il nécessite une connaissance de l'architecture et des paramètres du modèle pour générer des exemples contradictoires [52].
— **Projected Gradient Descent (PGD) :** Cette attaque est similaire à FGSM, mais au lieu d'une seule perturbation, elle implique de prendre plusieurs étapes dans la direction du gradient, chaque étape étant contrainte à un petit écart maximum par rapport à l'image d'origine. Cette attaque est plus puissante que FGSM et peut être plus difficile à défendre [53].

1.7.4 Contre-attaque du BIM contre les attaques FGSM et PGD

Le BIM (Basic Iterative Method) est une méthode utilisée en réponse au FGSM (Fast Gradient Sign Method) et au PGD (Projected Gradient Descent) pour contrer les attaques adverses. Alors que le FGSM perturbe rapidement les données d'entrée en utilisant le gradient de la fonction de perte, et que le PGD effectue plusieurs étapes itératives pour trouver la perturbation optimale, le BIM adopte une approche itérative visant à minimiser la classification erronée par le modèle. Cette méthode renforce la robustesse du modèle contre les attaques FGSM et PGD [54]. L'utilisation du BIM rend les exemples d'entrée plus difficiles à attaquer et limite l'efficacité des attaques adverses.

FIGURE 1.24 – Résultats d'image attaquer par perturbation du (BIM) [55]

1.8 Le lien entre le bruit et la génération d'images

L'ajout de bruit peut être une technique bénéfique pour améliorer la qualité des images générées par les réseaux neuronaux. En introduisant du bruit dans les données d'entrée, le réseau peut apprendre à générer des images plus résistantes aux perturbations. Cela est dû au fait que le réseau doit apprendre à produire des images cohérentes à la fois avec des données propres et des données bruitées [56].

Le bruit aléatoire : L'ajout de bruit aléatoire à une image générée permet de lui conférer un aspect plus réaliste. Par exemple, lors de la génération de textures, un bruit aléatoire peut être utilisé pour simuler les variations et les irrégularités présentes dans le monde réel.

Le bruit latent : Dans les modèles de génération d'images tels que les réseaux génératifs adverses (GAN), un bruit latent est généralement introduit à l'entrée du générateur. Ce bruit latent est ensuite transformé en une image générée par le générateur. En ajustant le bruit latent, il est possible d'explorer différentes variations et de contrôler le résultat de la génération.

Le bruit de perturbation : Pendant le processus d'entraînement des modèles de génération d'images, le bruit peut être utilisé comme une perturbation ajoutée aux données d'entraînement. Cela permet d'introduire de la diversité dans les exemples d'entraînement et d'aider le modèle à généraliser de manière plus efficace, même en présence de données nouvelles ou bruitées.

1.9 Conclusion

Les réseaux antagonistes génératifs (GAN) sont une méthode prometteuse pour générer des images réalistes et de haute qualité. Ils ont été utilisés dans de nombreux domaines dont l'art, la mode, la publicité, la médecine, la recherche scientifique, etc. Les GAN fonctionnent à l'aide de deux réseaux de neurones : un générateur et un discriminateur. Le générateur crée des images à partir de bruit aléatoire, tandis que le discriminateur évalue la qualité de ces images et apprend à les distinguer des vraies. L'un des avantages des GAN est qu'ils peuvent apprendre à générer des images dans le cas où les données de formation sont rares ou coûteuses à obtenir. Cependant, l'utilisation des GAN peut également conduire à des résultats erronés, notamment en ce qui concerne la génération d'images trompeuses. Les attaques contradictoires sont des attaques où un attaquant manipule les données d'entrée pour induire des prédictions incorrectes. Elles peuvent être

appliquées aux GAN en modifiant les données d'entrée pour générer des images trompeuses. Ces attaques peuvent avoir des conséquences préoccupantes, telles que tromper les systèmes de reconnaissance d'image ou perturber les performances du modèle GAN. Cela inclut l'utilisation de techniques de génération de données contradictoires et de détection d'attaques pour renforcer la robustesse du modèle GAN. L'objectif de ce projet est de créer des images trompeuses qui peuvent amplifier la base de données. Dans le chapitre suivant, nous expliquerons comment utiliser l'optimisation pour extraire les meilleures images générées et nous discuterons des différentes méthodes d'optimisation afin de déterminer celle que nous utiliserons dans notre travail.

Chapitre 2

Utilisation de l'optimisation pour la génération d'images

2.1 Introduction

L'optimisation des images est largement utilisée dans le domaine de l'apprentissage automatique pour extraire les meilleures images pouvant répondre aux exigences nécessaires dans notre travail. Notre objectif principal est de produire des images réalistes en combinant des modèles de génération et des outils d'optimisation avancés, afin d'optimisé les images contradictoires générer. L'objectif est de rendre ces images parfaitement similaires à l'image d'origine, ce qui représente un défi pour le réseau qui les reconnaît. Cette approche entraîne une amélioration de la qualité des résultats d'image et de la fiabilité de modèle.
Ce chapitre donne un aperçu de l'utilisation de l'optimisation pour la génération d'images, en présentant son histoire, ses principes, ses algorithmes et ses développements récents. Il décrit également différentes méthodes d'optimisation, compare leur efficacité et met en évidence les techniques utilisées. Les défis rencontrés et les efforts de recherche déployés pour surmonter ces obstacles ont également été discutés. En conclusion, ce chapitre propose une discussion sur les orientations futures possibles de l'utilisation de l'optimisation dans le domaine de la génération d'images.

2.2 Génération des images en utilisant l'optimisation

L'optimisation pour la génération d'images consiste à améliorer l'efficacité et la qualité de la production d'images à l'aide d'algorithmes informatiques. Pour améliorer la qualité de l'image et la vitesse de traitement, de nombreux paramètres sont modifiés au cours de ce processus.
L'utilisation de réseaux antagonistes génératifs est une autre méthode pour améliorer la production d'images (GAN) [31]. Les GAN sont constitués de deux réseaux de neurones qui se font concurrence lors de l'entraînement pour produire des images de haute qualité : un discriminant et un générateur. Nous pouvons améliorer le processus de génération d'images pour obtenir de meilleurs résultats en modifiant les paramètres du générateur et du discriminateur.
Les auto-encodeurs, l'apprentissage par renforcement et les algorithmes evolutifs sont

d'autres méthodes d'optimisation de la génération d'images.

2.2.1 Methodes d'amelioration de la generation d'image et leur lien avec l'optimisation

2.2.1.1 Auto-encodeurs

Les auto-encodeurs variationnels (VAEs) sont un autre type de réseau neuronal qui peut être utilisé pour générer des images. Les VAE fonctionnent en encodant d'abord une image dans un espace latent, puis en décodant cet espace latent en une nouvelle image [3]. L'espace latent est une représentation de plus basse dimension de l'image, et il peut être utilisé pour contrôler les propriétés de l'image générée. Par exemple, l'utilisateur peut spécifier la couleur, la texture ou la forme souhaitée de l'image générée, et le VAE tentera de générer une image qui correspond à ces spécifications.
Les auto-encodeurs et l'optimisation sont deux concepts étroitement liés dans le domaine de l'apprentissage automatique. Les auto-encodeurs sont des outils polyvalents utilisés dans diverses tâches telles que la réduction de dimensionnalité, la détection de bruit dans les images et la détection d'anomalies. Dans le contexte de la réduction de dimensionnalité, les auto-encodeurs sont employés pour apprendre une représentation de dimension réduite des données d'entrée. Cette approche s'avère utile pour des tâches telles que la visualisation et la compression de données. En ce qui concerne la détection de bruit dans les images, les auto-encodeurs permettent d'apprendre une représentation propre de l'image en partant d'une version bruitée [13]. Cela est particulièrement bénéfique pour des applications de restauration et d'amélioration d'images. Enfin, pour la détection d'anomalies, les auto-encodeurs sont utilisés pour apprendre une distribution normale des données d'entrée, permettant ainsi d'identifier des valeurs aberrantes qui peuvent indiquer une activité anormale ou frauduleuse.

2.2.1.2 Apprentissage par renforcement

L'apprentissage par renforcement (RL) offre plusieurs avantages pour la génération d'images. Dans un premier temps, il permet de former des réseaux antagonistes génératifs (GAN), qui sont une classe de réseaux neuronaux utilisés pour générer des images réalistes [57]. Les GAN consistent en deux réseaux neuronaux qui s'opposent : un générateur et un discriminateur. Le générateur cherche à produire des images qui sont indiscernables des images réelles, tandis que le discriminateur essaie de distinguer les images réelles des images générées. Au fur et à mesure que ces deux réseaux se confrontent, ils s'améliorent mutuellement au fil du temps.
Dans l'apprentissage par renforcement (RL), l'agent apprend à prendre des actions qui conduisent à la récompense la plus élevée possible par essais et erreurs. L'agent commence par explorer l'environnement et prendre des actions aléatoires. Au fil du temps, l'agent apprend à associer certaines actions à certaines récompenses. Cela permet à l'agent de prendre de meilleures décisions quant aux actions à entreprendre à l'avenir.

2.2.1.3 Algorithmes evolutifs

Les algorithmes évolutionnaires (EAs) sont une forme d'algorithme d'optimisation métaheuristique largement applicable, y compris pour la génération d'images. Les EAs opèrent en effectuant une recherche itérative au sein d'une population de solutions afin de trouver la

meilleure solution possible. Chaque solution de la population est représentée par un chromosome, qui est une séquence de gènes. La qualité d'une solution est évaluée en fonction de sa concordance avec la fonction objectif du problème [58].
Dans la génération d'images, la fonction objectif consiste généralement à générer une image aussi similaire que possible à une image cible donnée. L'EA commence avec une population de chromosomes générés de manière aléatoire. Chaque chromosome représente une image potentielle. La qualité de chaque chromosome est ensuite évaluée en le comparant à l'image cible. Les chromosomes avec la meilleure qualité sont sélectionnés pour se reproduire. Les descendants de ces chromosomes subissent ensuite des mutations, ce qui signifie que certains de leurs gènes sont modifiés. Les nouveaux chromosomes sont ensuite évalués et le processus se répète.
Ce processus se poursuit jusqu'à ce que l'EA converge vers une solution de haute qualité. La solution vers laquelle l'EA converge est l'image générée.

La relation entre les algorithmes évolutifs (EAs) et l'optimisation réside dans le fait que les EAs peuvent être utilisés pour trouver des solutions optimales ou quasi-optimales à une grande variété de problèmes. Les EAs sont souvent utilisés pour résoudre des problèmes difficiles ou impossibles à résoudre à l'aide d'algorithmes d'optimisation traditionnels, tels que les problèmes NP-difficiles.

2.2.2 Bref historique et évolution des méthodes d'optimisation

Au fil des ans, les techniques d'optimisation de la génération d'images ont subi des changements constants, de nouvelles approches apparaissant à mesure que la demande d'images générées de haute qualité augmente. Un aperçu du développement des techniques d'optimisation de la génération d'images est donné ci-dessous :

— **Optimisation aléatoire :** Dans les années 1970 et 1980, le domaine de l'optimisation stochastique a commencé à émerger, il se concentrait sur le développement d'algorithmes d'optimisation utilisant des méthodes probabilistes pour guider la recherche de solutions optimales. L'un des algorithmes d'optimisation stochastique les plus influents était le recuit simulé, qui a été développé par Kirkpatrick, Gelatt et Vecchi en 1983 [59]. Le recuit simulé consiste à perturber de manière aléatoire la solution actuelle et à accepter la perturbation avec une probabilité qui diminue avec le temps, permettant à l'algorithme d'explorer l'espace de recherche de manière plus approfondie.
— **Méthodes de traitement d'image traditionnelles** (années 1970 à 1990) : Au début, les images étaient améliorées à l'aide de techniques de traitement d'image classiques telles que la segmentation, la détection des contours et le filtrage d'image. Ces méthodes n'avaient pas la capacité de créer de nouvelles photos ; au lieu de cela, ils se sont concentrés sur l'amélioration des attributs des photographies existantes.
— **Algorithmes évolutionnaires** ont d'abord été utilisés pour créer des images à l'aide de modèles mathématiques dans les années 1990 et 2000. Les exemples incluent les algorithmes génétiques et l'optimisation des essaims de particules [60]. Bien que ces algorithmes puissent améliorer les attributs visuels, les photos résultantes étaient souvent de mauvaise qualité.
— **Autoencodeurs variationnels (VAE) (années 2010)** les VAE ont été introduits pour la première fois en tant que technique de production d'images de haute qualité

au début des années 2010 [61]. Les images peuvent être apprises à être encodées dans un espace de dimension inférieure par les VAE, qui peuvent ensuite échantillonner à partir de cet espace pour produire de nouvelles images.

— **Réseaux génératifs contradictoires (GAN) (2014)** Les GAN ont été utilisés pour la première fois en 2014 et ont immédiatement pris les devants en tant que technique la plus largement utilisée pour produire des photos de haute qualité [31]. Les GAN sont constitués de deux réseaux de neurones qui se font concurrence pour produire d'excellentes images : un discriminateur et un générateur. L'explication précédente est dansLa figure 1.12.

— **Transfert de style neuronal (2015)** Le transfert de style neuronal est une technique permettant de créer de nouvelles images en fusionnant le contenu d'une image avec le style d'une autre. Il a été introduit pour la première fois en 2015 [62]. Cette méthode convertit le style d'une image artistique en une photographie à l'aide d'algorithmes d'apprentissage en profondeur.

2.2.3 Importance de la génération des images en utilisant l'optimisation

Dans un certain nombre de disciplines, y compris la vision par ordinateur, les graphiques et le traitement d'images, la production des images de haute qualité en utilisant l'optimisation est une tâche cruciale. Afin de créer une image qui imite étroitement une image de référence particulière, une image de départ est utilisée et ses valeurs de pixel sont optimisées.

Voici quelques justifications de l'importance de produire des photos de haute qualité en utilisant l'optimisation :

— **Vision par ordinateur** la génération de photos de haute qualité améliorer le fonctionnement des algorithmes de vision par ordinateur [63]. L'utilisation de techniques d'optimisation peut être très bénéfique dans ce contexte. Par exemple, en utilisant des algorithmes d'optimisation tels que les réseaux génératifs antagonistes (GAN), nous pouvons générer des images réalistes et de haute qualité d'objets spécifiques. Ces images synthétiques peuvent ensuite être utilisées pour augmenter la taille et la diversité des données d'entraînement. En entraînant un réseau de neurones avec ces images générées, nous pouvons améliorer sa capacité à reconnaître et à comprendre les caractéristiques de l'objet cible dans une image réelle.

— **Graphisme** Dans le domaine du graphisme, produire des photographies de haute qualité est essentiel pour produire des modèles 3D visuellement attrayants et réalistes [64]. Nous pouvons créer des images qui correspondent étroitement aux éléments réels en utilisant l'optimisation, qui peuvent ensuite être utilisées comme textures ou matériaux dans les modèles 3D.

— **Traitement d'image** La génération d'images de haute qualité peut également être utile pour les activités de traitement d'image telles que la restauration ou l'amélioration d'image [65]. Par exemple, nous pouvons créer des photos de haute qualité de photographies détériorées en utilisant l'optimisation, qui peut ensuite être appliquée pour restaurer l'image d'origine.

2.3 Comprendre l'optimisation pour la génération d'images

2.3.1 Les principes fondamentaux de l'optimisation pour la génération d'images

Les principes fondamentaux de l'optimisation pour la génération d'images sont importants car ils peuvent contribuer à améliorer la qualité et la vitesse du processus de génération d'images. En suivant ces principes, nous pouvons entraîner un générateur d'images capable de produire des images de haute qualité. De plus, ces principes peuvent également aider à tromper les générateurs adversaires dans le cadre des GAN (Generative Adversarial Networks). Voici quelques-uns des principes de base de l'optimisation pour la génération d'images :

— **Utiliser un ensemble de données de haute qualité :** La qualité de l'ensemble de données utilisé pour entraîner un générateur d'images est essentielle pour la qualité des images générées. Un ensemble de données de haute qualité contiendra une grande variété d'images représentatives du type d'images que le générateur est censé produire [66].

— **Choisir le bon modèle :** Il existe de nombreux types différents de générateurs d'images disponibles, chacun ayant ses propres forces et faiblesses. Le bon modèle pour une application particulière dépendra de la qualité souhaitée des images générées, de la quantité de données disponibles et des ressources informatiques disponibles [66].

— **Entraîner le modèle pendant suffisamment longtemps :** L'entraînement d'un générateur d'images peut prendre beaucoup de temps, en particulier si l'ensemble de données est volumineux ou si le modèle est complexe. Il est important de former le modèle pendant suffisamment longtemps pour s'assurer qu'il converge vers une bonne solution [33].

— **Utiliser un bon algorithme d'optimisation :** L'algorithme d'optimisation utilisé pour entraîner le générateur d'images peut avoir un impact significatif sur la qualité des images générées. Un bon algorithme d'optimisation sera capable de trouver le minimum global de la fonction de perte, ce qui donnera les meilleures images générées possibles [67].

— **Régulariser le modèle :** La régularisation est une technique qui peut être utilisée pour éviter que le modèle ne surajuste les données d'entraînement. Le surajustement se produit lorsque le modèle apprend le bruit dans les données d'entraînement plutôt que les motifs sous-jacents. La régularisation peut être utilisée pour éviter le surajustement en ajoutant une pénalité à la fonction de perte qui décourage le modèle d'apprendre trop des données d'entraînement [68].

— **Utiliser un ensemble de validation :** Un ensemble de validation est un ensemble de données qui est exclu de l'ensemble de données d'entraînement et utilisé pour évaluer les performances du modèle. L'ensemble de validation peut être utilisé pour déterminer quand arrêter l'entraînement du modèle, ainsi que pour sélectionner les meilleurs hyperparamètres pour le modèle [68].

2.3.2 Algorithmes d'optimisation pour la génération d'images

Il est possible de générer des images en utilisant une variété de stratégies d'optimisation. Les algorithmes fréquemment utilisés sont illustrés dans la La figure 2.1 :

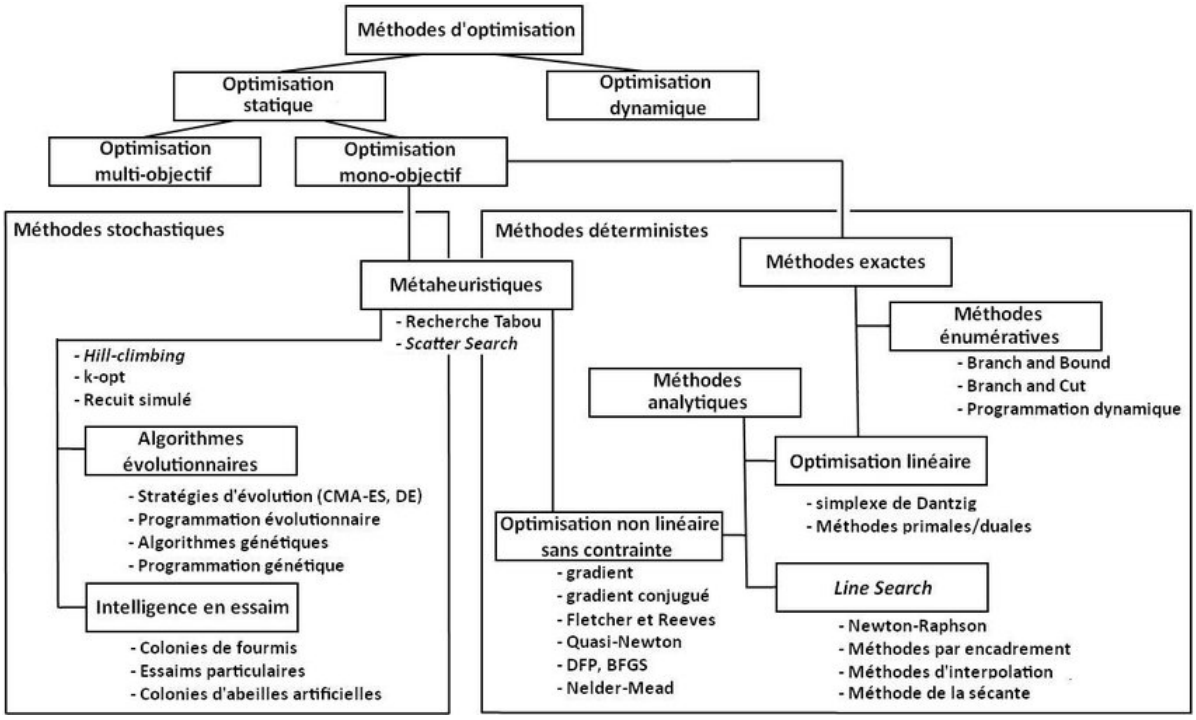

FIGURE 2.1 – Différents algorithmes d'optimisation [69]

2.3.2.1 Déscente du gradient (Gradient Descent)

La descente de gradient stochastique (SGD) est un algorithme d'optimisation populaire pour l'entraînement des modèles d'apprentissage automatique, y compris les modèles de génération d'images. La SGD fonctionne en mettant à jour de manière itérative les paramètres du modèle dans la direction du gradient négatif de la fonction de perte. Le gradient est une mesure de la variation de la fonction de perte en réponse à un petit changement dans les paramètres. En mettant régulièrement à jour les paramètres dans la direction du gradient négatif, le modèle peut améliorer progressivement ses performances [70].

La SGD présente plusieurs avantages par rapport à d'autres algorithmes d'optimisation, tels que :

— Elle est simple à mettre en œuvre et efficace pour l'entraînement avec de grands ensembles de données.
— Elle peut être utilisée pour entraîner des modèles avec des fonctions de perte non convexes, qui sont souvent utilisées pour la génération d'images.
— Elle peut être utilisée pour entraîner des modèles avec un grand nombre de paramètres.

Cependant, la SGD présente également certains inconvénients, tels que :

— Elle peut être sensible au choix des hyperparamètres, tels que le taux d'apprentissage.
— Elle peut être lente à converger, en particulier pour les modèles avec un grand nombre de paramètres.
— Elle peut être sensible au bruit dans les données d'entraînement.

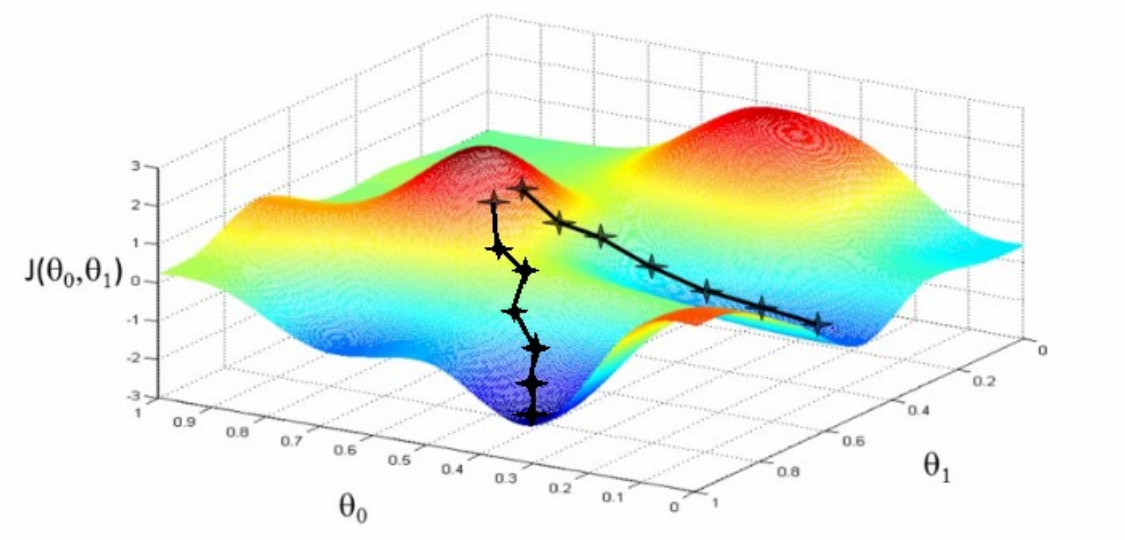

FIGURE 2.2 – Optimisation descente de gradient [71]

2.3.2.2 Adam

Adam est un algorithme d'optimisation populaire pour l'entraînement des modèles d'apprentissage profond. Adam combine les avantages de l'inertie et de l'AdaGrad en maintenant une moyenne mobile des gradients et des carrés des gradients pour chaque paramètre, puis en utilisant ces moyennes pour ajuster le taux d'apprentissage [72]. Adam s'est avéré efficace pour diverses tâches d'apprentissage automatique, y compris la génération d'images. En particulier, Adam a démontré sa capacité à entraîner de manière plus efficace des modèles génératifs profonds, tels que les réseaux adverses génératifs (GAN), par rapport à d'autres algorithmes d'optimisation.
Voici quelques avantages de l'utilisation d'Adam pour la génération d'images :

— Adam est plus efficace que d'autres algorithmes d'optimisation, tels que SGD avec inertie.
— Adam est plus robuste à l'ajustement des hyperparamètres.
— Adam peut entraîner de manière plus efficace des modèles génératifs profonds par rapport à d'autres algorithmes d'optimisation.

Voici quelques défis de l'utilisation d'Adam pour la génération d'images :

— Adam peut être plus coûteux en termes de calcul que d'autres algorithmes d'optimisation.
— Adam peut être moins stable que d'autres algorithmes d'optimisation.
— Adam peut être plus difficile à déboguer que d'autres algorithmes d'optimisation.

2.3.2.3 Momentum

Le momentum est une technique qui peut être utilisée pour améliorer la stabilité et la convergence de la descente de gradient stochastique (SGD). Elle consiste à ajouter un terme de momentum à la règle de mise à jour, ce qui permet de lisser les mises à jour et d'éviter que l'algorithme ne reste coincé dans des minima locaux. Le momentum s'est avéré efficace pour diverses tâches d'apprentissage automatique, y compris la génération d'images. En particulier, le momentum a montré sa capacité à entraîner de manière plus efficace des modèles génératifs profonds, tels que les réseaux adverses génératifs (GAN), par rapport à SGD sans momentum [73].

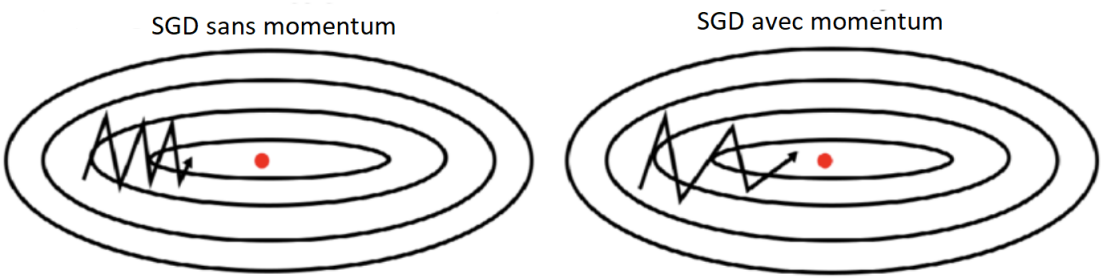

FIGURE 2.3 – Optimisation de SGD sans momentum et SGD avec momentum [71]

Avantages :
— Convergence améliorée : Le momentum peut contribuer à améliorer la convergence des modèles de génération d'images en les aidant à éviter de rester coincés dans des minima locaux.
— Entraînement plus rapide : Le momentum peut accélérer l'entraînement des modèles de génération d'images en les rendant plus efficaces pour trouver la solution optimale.
— Entraînement plus stable : Le momentum peut rendre l'entraînement des modèles de génération d'images plus stable en évitant qu'ils oscillent de manière excessive autour de la solution optimale.

Défis :
— Nécessite un réglage : L'hyperparamètre de momentum doit être réglé avec soin pour obtenir les meilleurs résultats.
— Peut entraîner un surajustement : Le momentum peut parfois conduire à un surajustement, en particulier si le modèle n'est pas régularisé correctement.
— Peut être coûteux en termes de calcul : Le momentum peut être plus coûteux en termes de calcul que d'autres algorithmes d'optimisation, tels que la descente de gradient stochastique (SGD).

2.3.2.4 Algorithmes génétiques

La famille des algorithmes d'optimisation appelés algorithmes génétiques a été développée dans le sillage de la sélection naturelle [74]. Ils impliquent la génération d'une population de solutions candidates et la sélection des meilleures en fonction d'une fonction de fitness. Les solutions choisies sont ensuite utilisées pour effectuer des procédures de croisement et de mutation afin de créer une nouvelle population.

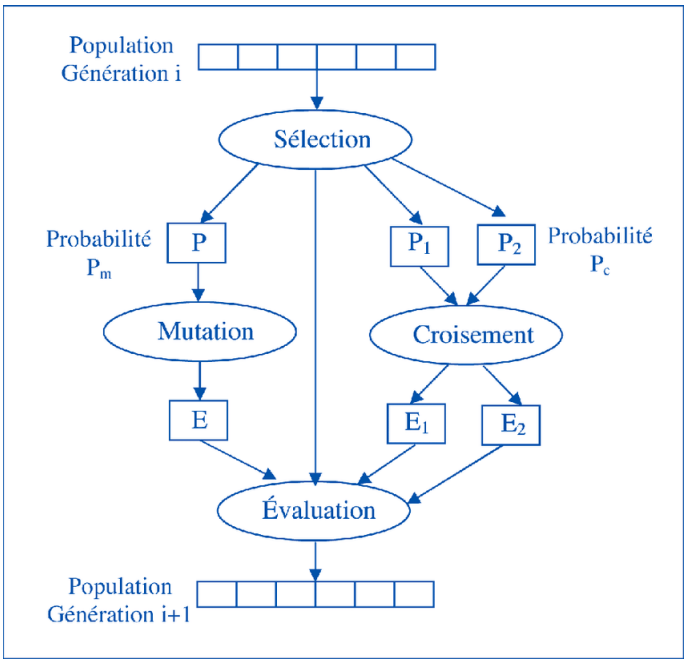

FIGURE 2.4 – Optimisation de algorithmes génétiques [71]

2.3.2.5 Recuit simulé

Inspiré de la procédure de recuit métallurgique, le recuit simulé est un algorithme d'optimisation probabiliste. Afin d'examiner en profondeur l'espace de recherche, il commence par une température élevée et l'abaisse progressivement au fil du temps [75]. Une solution adjacente est choisie au hasard, et si elle améliore la fonction de perte, elle est acceptée.

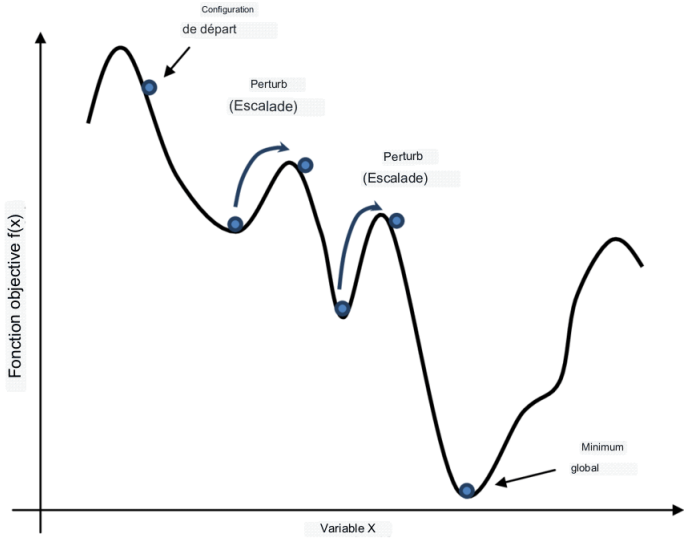

FIGURE 2.5 – Optimisation de Recuit simulé [76]

2.3.2.6 Stratégies d'évolution

un ensemble d'algorithmes d'optimisation utilisant l'évolution naturelle comme source d'inspiration principale [77]. Il s'agit de créer une population de solutions potentielles

puis de les mettre à jour en combinant les méthodes de mutation et de sélection.

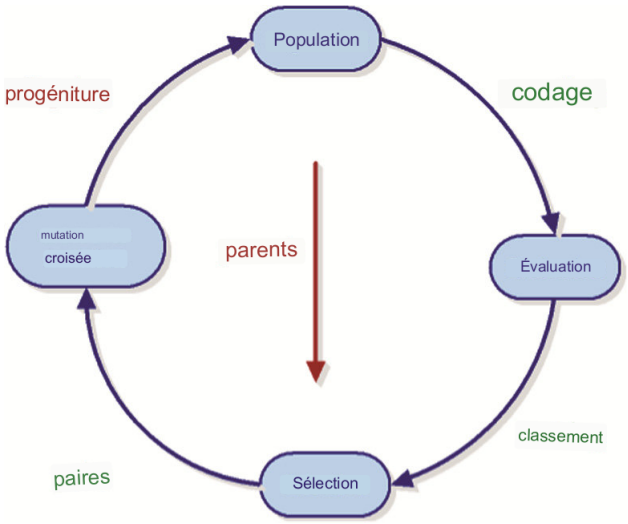

FIGURE 2.6 – Optimisation de Stratégies d'évolution [76]

2.3.2.7 L'optimisation aléatoire

L'optimisation aléatoire est une technique qui peut être utilisée pour générer des images en variant aléatoirement les paramètres d'un modèle génératif. Cela peut servir à créer de nouvelles et intéressantes images, ou à explorer l'espace des images possibles qui peuvent être générées par le modèle [78].Il existe différentes façons de mettre en œuvre l'optimisation aléatoire pour la génération d'images. Une approche courante consiste à utiliser un algorithme génétique et un algorithme d'optimisation bayésienne. L'optimisation aléatoire est une technique puissante qui peut être utilisée pour générer des images. Cependant, il est important de noter que l'optimisation aléatoire peut être chronophage. Il peut également être difficile de trouver une solution de qualité suffisante.
Voici quelques avantages de l'utilisation de l'optimisation aléatoire pour la génération d'images :
— Elle peut être utilisée pour créer de nouvelles et intéressantes images.
— Elle peut être utilisée pour explorer l'espace des images possibles qui peuvent être générées par le modèle.
— Elle peut être utilisée pour trouver des images qui répondent à des critères spécifiques.
Voici quelques défis liés à l'utilisation de l'optimisation aléatoire pour la génération d'images :
— Cela peut prendre du temps.
— Il peut être difficile de trouver une solution de qualité suffisante.
— Il peut être difficile de contrôler la qualité des images générées.

De toutes ces méthodes, l'optimisation aléatoire est la plus importante pour la génération d'images adverses. Cela s'explique par le fait que les réseaux antagonistes génératifs (GAN) sont réputés difficiles à entraîner et que l'optimisation aléatoire peut contribuer à améliorer la stabilité et la convergence du processus d'entraînement. En générant aléatoirement de nouvelles solutions, l'optimisation aléatoire peut aider à éviter que le GAN ne se retrouve bloqué dans des minima locaux. De plus, l'optimisation aléatoire peut favoriser la diversité des images générées, ce qui peut conduire à des résultats plus

réalistes et créatifs.

Voici quelques détails supplémentaires sur l'importance de l'optimisation aléatoire pour la génération d'images adverses :

Stabilité : Les GAN sont souvent instables lors de l'entraînement et peuvent facilement rester coincés dans des minima locaux. L'optimisation aléatoire peut contribuer à améliorer la stabilité du processus d'entraînement en évitant que le GAN ne reste bloqué dans un minimum local.

C**onvergence :** Les GAN peuvent mettre du temps à converger et il peut arriver qu'ils ne convergent pas du tout. L'optimisation aléatoire peut aider à améliorer la convergence du processus d'entraînement en introduisant de nouvelles solutions dans l'espace de recherche.

Diversité : Les GAN peuvent parfois générer des images répétitives ou ennuyeuses. L'optimisation aléatoire peut aider à introduire de la diversité dans les images générées en générant aléatoirement de nouvelles solutions.

En résumé, l'optimisation aléatoire est un outil important pour la génération d'images adverses. Elle peut contribuer à améliorer la stabilité, la convergence et la diversité des images générées.

2.4 Défis et orientations futures

2.4.1 Défis liés à l'optimisation pour la génération d'images

2.4.1.1 Data scarcity

La pénurie de données est un concept qui se réfère à la situation où il y a une insuffisance de données disponibles pour entraîner un modèle d'apprentissage automatique [79]. Cela peut engendrer plusieurs problèmes, notamment :

— Inexactitude : les modèles entraînés sur de petits ensembles de données ont tendance à être moins précis que ceux entraînés sur de grands ensembles de données.
— Biais : les modèles entraînés sur de petits ensembles de données peuvent être influencés par les caractéristiques des données sur lesquelles ils ont été entraînés, ce qui peut entraîner des problèmes tels que la discrimination et l'injustice.
— Surapprentissage : les modèles entraînés sur de petits ensembles de données sont plus susceptibles de surapprendre les spécificités des données plutôt que les motifs sous-jacents.

Il existe plusieurs approches pour résoudre ce problème de pénurie de données, notamment :

— Augmentation des données : il s'agit d'une technique consistant à créer de nouvelles données à partir des données existantes. Cela peut être réalisé en appliquant des transformations aux données, telles que le recadrage, la rotation et le retournement.
— Transfert d'apprentissage : cette technique consiste à utiliser un modèle préalablement entraîné sur une tâche pour entraîner un nouveau modèle sur une autre tâche. Les

poids du modèle préalablement entraîné peuvent servir de point de départ pour l'entraînement du nouveau modèle.
— Crowdsourcing de données : cette technique implique la collecte de données auprès d'un grand nombre de personnes. Cela peut être réalisé à l'aide de sondages, de tâches en ligne et d'autres méthodes.

2.4.1.2 Complexité informatique (Computational complexity)

La complexité computationnelle est un domaine de l'informatique qui se consacre à l'étude des exigences temporelles et spatiales des algorithmes (125). Son objectif est de classer les problèmes en fonction de leur difficulté intrinsèque et de développer des algorithmes efficaces pour résoudre des problèmes dans des classes de complexité spécifiques [80].
Il existe deux types principaux de complexité computationnelle :
— La complexité temporelle : elle mesure le temps nécessaire à l'exécution d'un algorithme. Elle est généralement évaluée en termes du nombre d'opérations élémentaires effectuées par l'algorithme.
— La complexité spatiale : elle mesure la quantité de mémoire utilisée par un algorithme. Elle est généralement mesurée en termes du nombre de bits de mémoire utilisés par l'algorithme.
Il existe différentes classes de complexité qui regroupent des problèmes de difficulté similaire. Parmi les classes de complexité les plus importantes, on trouve :
— P : l'ensemble des problèmes pouvant être résolus en temps polynomial.
— NP : l'ensemble des problèmes pour lesquels une solution peut être vérifiée en temps polynomial.
— NP-complet : l'ensemble des problèmes qui sont au moins aussi difficiles que n'importe quel problème dans NP.
— NP-difficile : l'ensemble des problèmes qui sont au moins aussi difficiles que n'importe quel problème dans NP-complet.
Le problème P contre NP est l'un des problèmes non résolus les plus significatifs en informatique. Il pose la question de savoir si tous les problèmes de la classe NP peuvent également être résolus en temps polynomial (P). Si P=NP, cela signifierait que tout problème vérifiable en temps polynomial peut également être résolu en temps polynomial. Cette égalité aurait de nombreuses implications dans le domaine de l'informatique et d'autres domaines connexes.

2.4.1.3 Non-convexité (Non-convexity)

La non-convexité fait référence aux fonctions qui ne possèdent pas un unique minimum global, mais plutôt plusieurs minima locaux, ce qui rend la recherche de la solution optimale plus difficile [81]. Les fonctions non convexes sont fréquentes en apprentissage automatique, où elles peuvent être présentes dans différentes tâches telles que la classification, la régression et le regroupement.
L'optimisation non convexe présente plusieurs défis, notamment :
— Minima locaux : Ces points de la fonction sont inférieurs aux points environnants, mais ne sont pas le minimum global. Si un algorithme d'optimisation se retrouve coincé dans un minimum local, il ne pourra pas trouver la solution optimale.
— Points de selle : Les points de selle ne sont ni des minima ni des maxima de la fonction. Ils peuvent être difficiles à identifier et peuvent entraîner le blocage de

l'algorithme d'optimisation.
— Multiples minima locaux : Comme mentionné précédemment, les fonctions non convexes peuvent avoir plusieurs minima locaux. Cela rend la recherche du minimum global difficile, car l'algorithme d'optimisation peut converger vers un minimum local différent selon le point de départ.

Il existe différentes techniques pour résoudre les défis de l'optimisation non convexe, telles que :
— Redémarrages aléatoires : Cette approche consiste à lancer l'algorithme d'optimisation à partir de plusieurs points différents. Cela permet d'éviter de rester bloqué dans des minima locaux.
— Algorithmes gloutons : Les algorithmes gloutons prennent des décisions localement optimales à chaque étape. Ils peuvent être utiles pour trouver de bonnes solutions dans les problèmes d'optimisation non convexe, mais ne garantissent pas la découverte du minimum global.
— Métaheuristiques : Les métaheuristiques sont des algorithmes qui utilisent des heuristiques pour rechercher des solutions aux problèmes d'optimisation. Les heuristiques sont des règles empiriques qui ne garantissent pas la solution optimale, mais elles peuvent être utiles pour trouver de bonnes solutions dans un délai raisonnable.

2.4.1.4 Haute dimensionnalité (High dimensionality)

La grande dimensionalité est un terme utilisé pour décrire des ensembles de données qui possèdent un grand nombre de caractéristiques [82]. Cela peut rendre l'analyse des données difficile, car les méthodes statistiques traditionnelles peuvent ne pas être efficaces. La grande dimensionalité présente plusieurs défis, notamment :
— Malédiction de la dimensionalité : À mesure que le nombre de caractéristiques augmente, le volume de l'ensemble de données augmente de manière exponentielle. Cela peut rendre difficile la détection de motifs dans les données, car le signal peut être noyé par le bruit.
— Sparcité : De nombreux ensembles de données à grande dimensionalité sont clairsemés, ce qui signifie que la plupart des caractéristiques sont nulles. Cela peut rendre difficile l'estimation des paramètres d'un modèle, car il n'y a pas suffisamment de données pour le faire.
— Surapprentissage : Les ensembles de données à grande dimensionalité sont plus susceptibles de surapprendre, ce qui signifie que le modèle apprend le bruit dans les données plutôt que les motifs sous-jacents.

Il existe plusieurs techniques qui peuvent être utilisées pour relever les défis de la grande dimensionalité, telles que :
— Sélection de caractéristiques : Cette approche consiste à choisir un sous-ensemble de caractéristiques qui sont les plus pertinentes pour la tâche en cours. Cela permet de réduire la dimensionalité de l'ensemble de données et d'améliorer les performances du modèle.
— Régularisation : La régularisation est une technique qui pénalise la complexité du modèle. Cela permet de prévenir le surapprentissage et d'améliorer les performances de généralisation du modèle.
— Réduction de dimensionalité : La réduction de dimensionalité consiste à réduire le nombre de caractéristiques dans un ensemble de données tout en préservant les informations essentielles. Cela peut être réalisé à l'aide de diverses techniques, telles que l'analyse en composantes principales (ACP) et l'ACP avec noyau.

2.4.1.5 Problème de Gradients

C'est la difficulté rencontrée dans l'entraînement des réseaux de neurones artificiels à l'aide des méthodes d'apprentissage par gradient et la rétropropagation. Dans de telles méthodes, chaque poids de réseau de neurones reçoit une mise à jour lors de chaque itération d'apprentissage proportionnelle à la dérivée partielle de la fonction d'erreur par rapport au poids courant. Le problème est que dans certains cas, la pente est très faible, ce qui empêche effectivement les poids de changer. Dans le pire des cas, cela peut complètement perturber la construction du réseau de neurones.

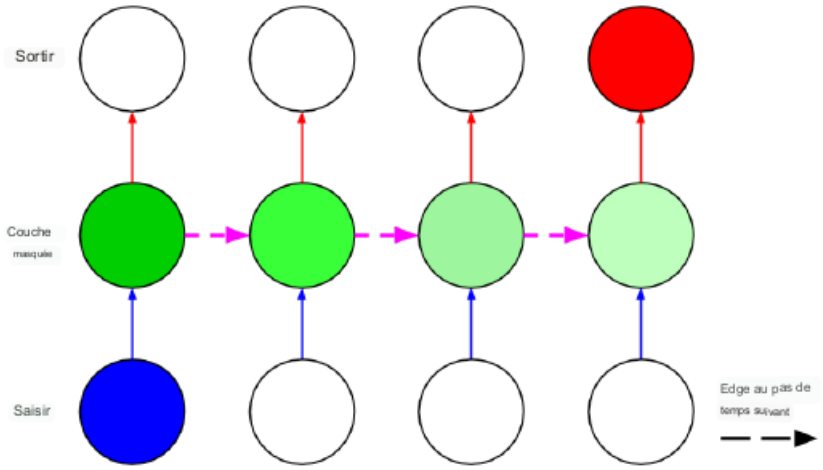

FIGURE 2.7 – Problème de gradient sur d'optimisation [83]

2.4.2 Résolution des problèmes liés aux algorithmes d'optimisation

2.4.2.1 Problème de Gradients

Le travail de Sepp Hochreiter [84] a mis en évidence un défi dans l'entraînement des réseaux de neurones artificiels utilisant des méthodes basées sur les gradients et la rétropropagation, connu sous le nom de "Problème du gradient qui disparaît". Ce problème survient lorsque le gradient (signal d'erreur) devient extrêmement petit, ce qui empêche efficacement les poids du réseau de se mettre à jour. Dans certains cas, cela peut même entraîner l'arrêt complet de l'entraînement du réseau neuronal.
Pour résoudre ce problème, Hochreiter a proposé une solution appelée réseau à mémoire à court terme et à long terme (LSTM). Les réseaux LSTM sont une forme de réseaux de neurones récurrents capables d'apprendre des dépendances à long terme. Ils y parviennent en utilisant des portes qui contrôlent le flux d'informations à travers le réseau. Ces portes permettent de conserver des informations importantes provenant des étapes temporelles précédentes, même lorsque les gradients sont très petits.
Les réseaux LSTM se sont avérés efficaces dans diverses tâches, telles que le traitement du langage naturel, la reconnaissance vocale et la traduction automatique. Ils sont

désormais largement utilisés dans la recherche et les applications de l'apprentissage profond.

Le défi du problème des gradients qui survient lorsque les gradients de la fonction de perte par rapport au poids d'un réseau neuronal deviennent extrêmement petits. Ce phénomène se produit souvent lorsque les fonctions d'activation utilisées dans le réseau ont une plage limitée de sorties. Lorsque les mises à jour des poids deviennent également très petites, ce qui peut rendre difficile l'apprentissage efficace du réseau. Bengio et al [85] ont proposé plusieurs techniques pour résoudre ce problème des gradients qui disparaissent. Ces techniques comprennent l'utilisation de fonctions d'activation plus appropriées, telles que la fonction ReLU, le rognage des gradients, la normalisation par lots, des techniques de régularisation comme la régularisation L2 et le dropout, ainsi que l'initialisation de Xavier. Le travail de Bengio sur le problème des gradients qui a disparu a eu une grande influence dans le domaine de l'apprentissage profond. Ses contributions ont permis de rendre les réseaux neuronaux profonds plus pratiques et ont conduit à leur adoption pour un grand événement d'applications.

Zhang et al [86]examinent la problématique des gradients et proposent l'utilisation de la descente de gradient comme solution. Ils démontrent que la régression est un algorithme d'optimisation itératif qui met à jour les paramètres d'un réseau neuronal en utilisant le gradient de la fonction de perte. Le gradient représente la mesure du changement de la fonction de perte en réponse à une modification des paramètres. En effectuant des mises à jour fréquentes des paramètres dans la direction opposée du gradient, la descente de gradient contribue à réduire la valeur de la fonction de perte.

2.4.2.2 Problème de la complexité informatique

Complexité computationnelle : L'optimisation des réseaux neuronaux peut être exigeante en termes de puissance de calcul, en particulier pour les modèles de grande taille [87]. Cependant, il existe des solutions pour surmonter ce défi, telles que l'entraînement distribué et la compression du modèle.
Dans l'article de Bishop et al [88], il est mentionné que l'entraînement d'un réseau neuronal profond a une complexité computationnelle de $O(n\hat{2})$, où n représente le nombre de paramètres du réseau. Cela est dû à l'algorithme de descente de gradient utilisé pour optimiser le réseau, qui nécessite le calcul du gradient de la fonction de perte par rapport à tous les paramètres. Ce processus peut être très chronophage, surtout pour les modèles de grande envergure. Heureusement, plusieurs techniques peuvent être employées pour réduire la complexité computationnelle de l'entraînement des réseaux neuronaux profonds, notamment :

— Utilisation de l'entraînement distribué : Cette approche implique la répartition des données d'entraînement sur plusieurs machines, permettant ainsi de former le modèle en parallèle sur chaque machine. Cela peut considérablement réduire le temps nécessaire à l'entraînement, en particulier pour les ensembles de données volumineux.
— Utilisation de la compression du modèle : Cette méthode vise à réduire le nombre de paramètres du réseau sans affecter significativement la performance du modèle. Elle peut être réalisée en utilisant des techniques telles que l'élagage (pruning) et la quantification.

Les travaux du chercheur Kingma et al [72] dans Adam, C'est une technique d'optimisation largement utilisée pour le développement des réseaux neuronaux. Il représente une variation de la descente de gradient stochastique (SGD) qui adapte les taux d'apprentissage individuels pour chaque paramètre. Adam s'est avéré extrêmement efficace pour entraîner des réseaux neuronaux sur de vastes ensembles de données.
Pour appliquer Adam et résoudre les problèmes de complexité computationnelle, il est nécessaire de formuler initialement le problème en tant que problème d'optimisation. Cela implique de définir une fonction de perte qui mesure l'écart entre la sortie désirée et la sortie réelle du modèle, ainsi qu'un objectif qui vise à minimiser cette fonction de perte.
Une fois que le problème est formulé comme un problème d'optimisation, Adam peut être utilisé pour entraîner un réseau neuronal à résoudre ce problème. Il mettra à jour de manière itérative les paramètres du réseau neuronal en suivant le gradient négatif de la fonction de perte. Les taux d'apprentissage de chaque paramètre seront adaptés en fonction des gradients précédents et des gradients au carré, ce qui permet une adaptation fine de l'apprentissage à chaque paramètre du réseau.

2.4.2.3 Problème de non-convexité

Un problème d'optimisation non convexe se réfère à un type de problème d'optimisation où la fonction objectif ou la région réalisable ne présente pas de propriété de convexité. Une fonction est considérée comme convexe lorsqu'un segment de ligne reliant deux points quelconques sur son graphique reste entièrement situé au-dessus ou en dessous du graphique lui-même. Cette propriété indique une certaine régularité de la fonction. D'autre part, la région réalisable désigne un ensemble de points qui respectent toutes les contraintes imposées par le problème d'optimisation [89].

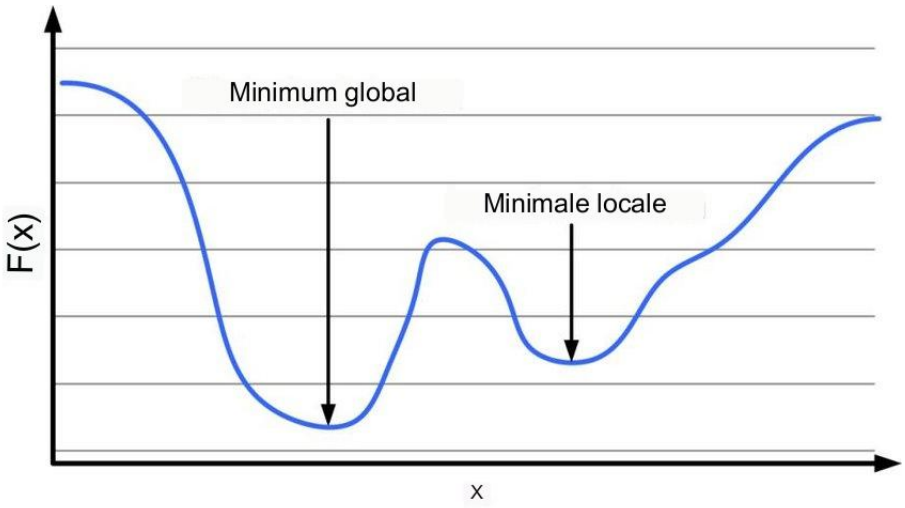

FIGURE 2.8 – Optimisation non-convexité [90]

Jorge Nocedal et Stephen Wright [91], dans leur ouvrage intitulé "Optimisation numérique", soulignent les difficultés rencontrées lors de la résolution des problèmes d'optimisation non convexes en raison de la présence de multiples optima locaux, ce qui rend la recherche de l'optimum global complexe. Pour aborder ces défis, Nocedal et Wright proposent différentes approches, notamment l'utilisation de heuristiques, de métaheuristiques et du

calcul parallèle. Les heuristiques sont des techniques visant à trouver des points de départ favorables pour les algorithmes d'optimisation, tandis que les métaheuristiques sont des algorithmes d'optimisation généraux, applicables à une grande variété de problèmes. Le calcul parallèle permet d'accélérer le traitement des problèmes d'optimisation non convexes.

2.4.3 Impact de l'utilisation de l'optimisation pour la génération d'images

Le domaine de la génération d'images évolue rapidement et de nouvelles stratégies et approches sont créées en permanence. En ce qui suit quelques, nous décrivons quelques impacts de l'utilisation de l'optimisation pour la génération des images.

2.4.3.1 Stabilité améliorée de l'entraînement

L'obtention d'une convergence stable lors de l'entraînement des modèles génératifs constitue l'un des principaux défis à relever. Ce défi est d'autant plus complexe car les réseaux génératifs antagonistes (GAN) sont souvent sujets à l'instabilité et peuvent présenter des problèmes tels que l'effondrement des modes, où ils se contentent de générer un nombre restreint d'images de sortie [7]. Des études récentes ont exploré diverses techniques visant à améliorer la stabilité de l'entraînement. Parmi celles-ci figurent l'utilisation de l'entraînement adversarial avec la distance de Wasserstein, l'application d'une pénalité de gradient et l'emploi d'un tampon de répétition [92]. Ces approches innovantes contribuent à surmonter les obstacles liés à la stabilité de l'entraînement des modèles génératifs [93].

2.4.3.2 Réalisme accru

Un autre objectif essentiel de la génération d'images consiste à produire des images qui soient plus réalistes et indiscernables des images réelles. Toutefois, cela s'avère souvent difficile car les modèles génératifs ont du mal à saisir les détails subtils présents dans les images réelles. Pour relever ce défi, des études récentes ont exploré diverses techniques visant à améliorer le réalisme des images générées [94]. Parmi ces techniques figurent l'utilisation de réseaux plus profonds et plus complexes, l'augmentation de la quantité de données utilisées lors de l'entraînement, ainsi que l'application de l'entraînement adversarial en utilisant un discriminateur spécialement conçu pour distinguer les images réelles des images générées [95]. Ces approches innovantes contribuent à améliorer la qualité et la réalisme des images générées, se rapprochant ainsi davantage des images réelles [96].

2.4.3.3 Amélioration de la contrôlabilité

Il est souhaitable de développer des modèles génératifs offrant un meilleur contrôle, permettant ainsi aux utilisateurs de spécifier les caractéristiques souhaitées pour les images générées. Cela pourrait être accompli en utilisant différentes techniques, telles que l'apprentissage par renforcement, qui permettrait de guider le processus de génération en fonction de récompenses définies par l'utilisateur [97]. Une autre approche consiste à utiliser une interface texte-vers-image, où l'utilisateur pourrait fournir des descriptions textuelles des images souhaitées, et le modèle génératif serait alors capable de produire

des images correspondantes. Ces techniques innovantes visent à offrir aux utilisateurs un plus grand contrôle sur le processus de génération d'images, leur permettant ainsi de spécifier leurs préférences et de créer des images répondant à leurs besoins spécifiques [98].

2.4.3.4 Nouvelles applications

Les modèles génératifs offrent de nombreuses possibilités d'application, notamment dans des domaines tels que l'édition d'images, la synthèse vidéo et la génération artistique. À l'avenir, il est envisageable d'explorer de nouvelles applications pour les modèles génératifs, comme leur utilisation pour générer des modèles 3D réalistes ou pour créer des formes de divertissement interactif novatrices. Ces pistes de recherche pourraient ouvrir de nouvelles perspectives en repoussant les limites de la créativité numérique et en explorant de nouvelles façons d'engager les utilisateurs dans des expériences visuelles et interactives [99].

2.5 Conclusion

En conclusion, l'optimisation de la génération d'images est un domaine de l'apprentissage automatique qui offre la possibilité de générer des images de haute qualité grâce à des algorithmes d'optimisation. Cette technologie trouve son utilité dans de nombreux secteurs tels que les jeux vidéo, la réalité virtuelle, la publicité, le design et la médecine. L'objectif principal est de produire des images réalistes en utilisant des modèles génératifs ainsi que des techniques et des outils d'optimisation avancés.
Dans ce contexte, l'outil d'optimisation d'image joue un rôle essentiel pour améliorer les images contradictoires que nous cherchons à générer. L'objectif est de rendre ces images totalement similaires à l'image d'origine, tout en rendant leur reconnaissance difficile pour le réseau. Grâce à cette approche, nous pouvons obtenir des résultats surprenants qui renforcent la qualité et la fiabilité de notre modèle.
Avec des recherches plus poussées, ces résultats offrent des perspectives prometteuses pour le développement futur des technologies d'amélioration d'image. Cela ouvre la voie à de nouveaux développements dans des domaines tels que l'augmentation des données, où les images contradictoires sont essentielles au lancement de notre projet d'augmentation des données et de formation du réseau.

Chapitre 3

Architecture, implémentation, et résultats

3.1 Introduction

Ce chapitre est consacré à la présentation de la conception, de la mise en œuvre et des résultats obtenus par En utilisant le système proposé. La première partie consiste à introduire la structure générale de notre système. La deuxième partie décrit les détails de mise en œuvre, y compris les outils et le matériel utilisés pour créer le système proposé. La dernière partie décrit les résultats obtenus et une discussion.

3.2 Architecture du système proposé

Dans cette partie, nous allons exposer la conception de notre système. Tout d'abord, nous aborderons l'architecture globale qui mettra en avant les différentes parties du notre système. Ensuite, nous détaillerons l'architecture de chaque partie individuellement.

3.3 Architecture générale

Notre système de génération d'image contradictoire en utilisant l'optimisation passe par les étapes illustrées dans la figure 3.1. La structure générale de notre système a été construite en plusieurs étapes, toutes incluses dans l'unité principale qui elle-même est divisée en deux parties avec une entrée et deux sorties Pour comparer la première partie avec la deuxième partie :

Dans cette étape, nous formons le réseau sur les données et produisons un modèle de formation qui sera utilisé comme discriminateur dans la prochaine étape. Dans cette étape, nous utlilisons les images contradictoires optimisée pour générer de nouvelles images qui vont être plus tard utilisées pour entraîner le réseaux profond. Les différentes étapes de cette deuxième partie se résume comme suit :

— **Génération d'images :** Étape de génération d'images contradictoires à l'aide de GAN.
— **optimisation :** L'optimisation aléatoire est utilisée pour sélectionner les meilleures images parmi toutes les images générées.
— **Préparation des données :** Cette étape consiste à préparer les données qui vont être utilisées par la suite pour entraîner à nouveau le réseau. La base de données augmentée va donc contenir les images originales ainsi que les nouvelles images obtenues à partir de l'étape précédente.
— **Entraînement du CNN :** dans cette étape, nous allons entraîner le réseau sur la nouvelle base de données.

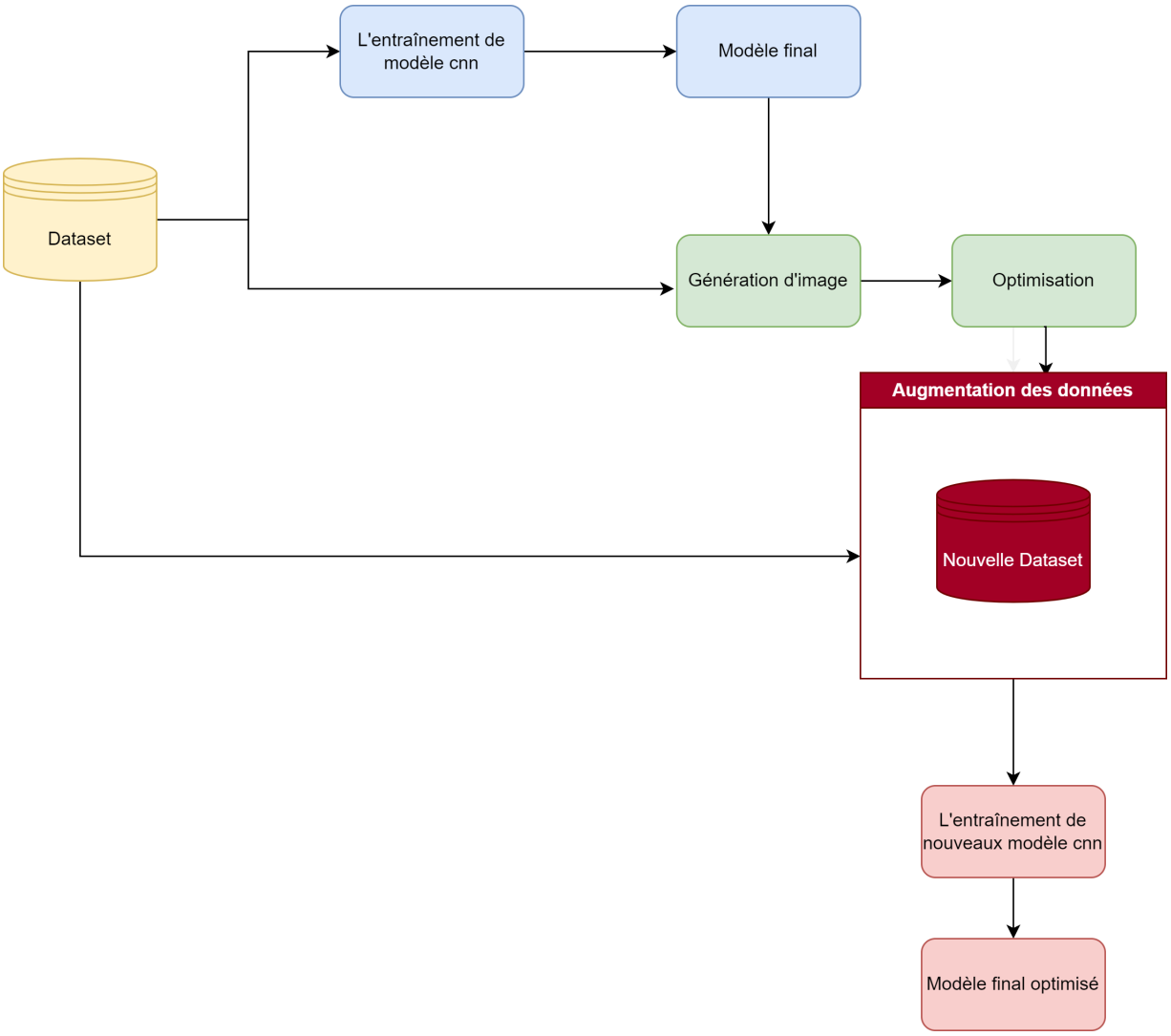

FIGURE 3.1 – L'architecture générale de notre système.

3.4 Architecture détaillée

Nous exposons dans cette partie les éléments constituants notre système de manière détaillée.

3.4.1 Dataset

Nous avons utilisé la base de données (flower photos) [100]. Cette base contient 3670 images de fleurs. La collecte de données est basée sur les données Flickr, vous pouvez utiliser cet ensemble de données pour identifier les plantes à partir de l'image. Elle contient des images divisées en cinq catégories : camomille, lavande, rose, tournesol, pissenlit. Pour chaque chapitre, il y a environ 800 images. Les images ne sont pas de très haute résolution, autour de 180 x 180 pixels.

3.4.2 Phase de génération des images

Cette phase consiste à générer de nouvelles images contradictoires. Les différentes étapes sont décrite comme suit.

3.4.2.1 Préparation de données d'entrée

Pour générer des images contradictoires à partir d'une image, nous utilisant l'image elle même et une image obtenue à partir de celle ci en appliquant l'algorithme IBM.

3.4.2.2 Déterminer le degré de bruit (epsilon)

La valeur d'epsilon est déterminée comme l'amplitude maximale de perturbation qui sera responsable du niveau de bruit ajouté à une image. Sa valeur est très faible, rendant le bruit imperceptible à l'œil nu. Cependant, cela a un impact significatif sur la convergence des réseaux de neurones lors de la classification des images.

3.4.2.3 Génération d'image

Le GAN (Generative Adversarial Network) joue un rôle central dans le processus de génération d'images. Il se compose de deux parties principales, à commencer par le générateur. Le générateur combine l'image d'origine avec de nombreuses distorsions générées. Dans notre cas, nous avons défini 50 distorsions différentes pour chaque image, ce qui aboutit à la génération de 50 images contradictoires.

3.4.2.4 La sélection des images

Le deuxième composant du GAN est le discriminateur. Si le discriminateur classe mal une image, elle est considérée comme réussie et l'image est conservée. En revanche, si le discriminateur classe bien l'image, il est considéré comme un échec.

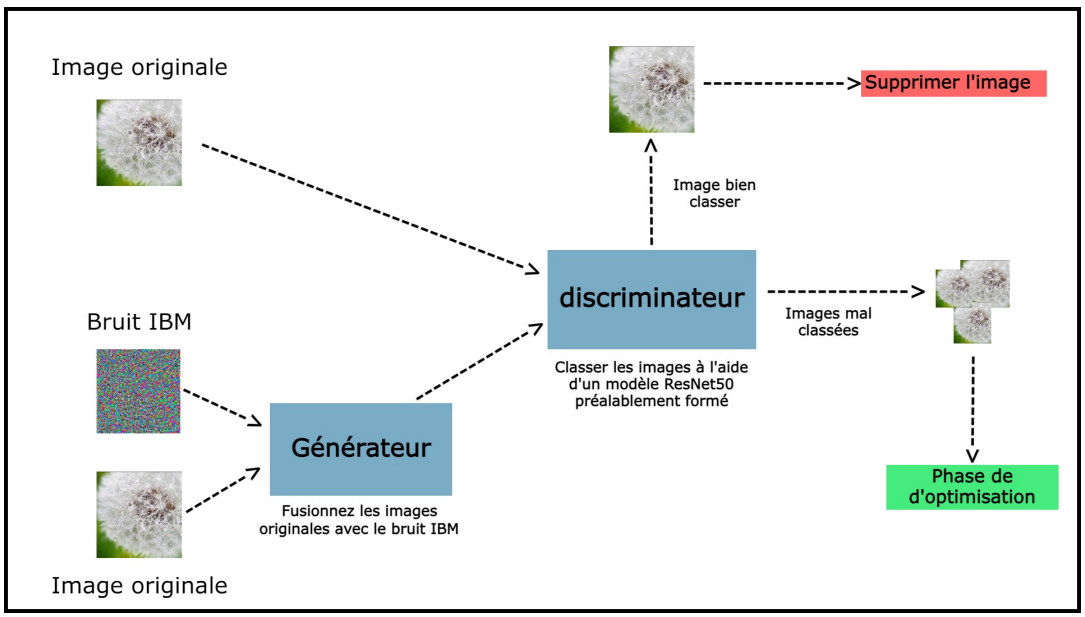

FIGURE 3.2 – Génération d'images

3.4.3 Phase de d'optimisation de l'image

Cette étape consiste à trouver les meilleures images parmi les images sélectionnées lors de l'étape précédente. Dans ce qui suit, nous décrivons le mode de fonctionnement de l'algorithme d'optimisation utilisé.

3.4.3.1 Déterminer les règles de sélection des images

Dans cette étape, nous prenons les images générées par le GAN et les soumettons à l'optimisation. Nous utilisons la méthode d'optimisation aléatoire pour sélectionner les meilleures images contradictoires, en appliquant les deux critères suivants :

La première condition est de choisir des images avec de petites valeurs epsilon, qui indiquent un niveau de distorsion plus faible.

Le deuxième critère est basé sur le degré de confiance de la classification. Le degré de confiance représente la précision de classification erroné. L'image avec un degré de confiance plus grands a plus de chances d'être sélectionnée. Cela signifie que CNN est très confiant dans sa classification incorrecte de l'image contradictoire. Cette relation s'explique comme suit :

$$\mathbf{Ce{=}Ct\text{-}Cc} \tag{3.1}$$

Tel que :
— **Ce est confiance de l'erreur.**
— **Ct est confiance totale.**
— **Cc est confiance correcte.**

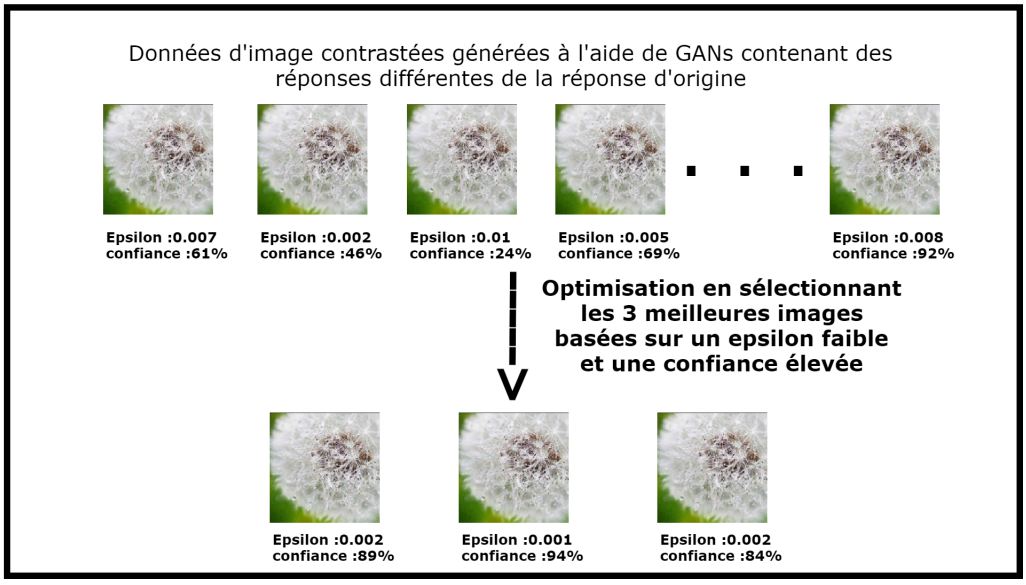

FIGURE 3.3 – L'utilisation de l'optimisation pour la sélection des meilleures images.

3.4.4 Phase d'entraînement en utilisant la base de données augmentée

Dans notre système, nous avons utilisé ResNet-50 est un modèle de réseau neuronal convolutif (CNN) de 50 couches de profondeur. ResNet-50 est un modèle populaire pour la classification d'images et a été utilisé pour obtenir des résultats de pointe sur une variété de références de classification d'images. ResNet-50 est construit sur le concept de blocs résiduels [101]. Un bloc résiduel est un bloc de construction composé de deux couches convolutionnelles, suivies d'une connexion de raccourci. La connexion raccourcie permet d'ajouter la sortie de la première couche convolutive à la sortie de la deuxième couche convolutive. Cela permet au réseau d'apprendre les fonctions d'identité, ce qui peut aider à empêcher le réseau de sur-adapter les données de formation.

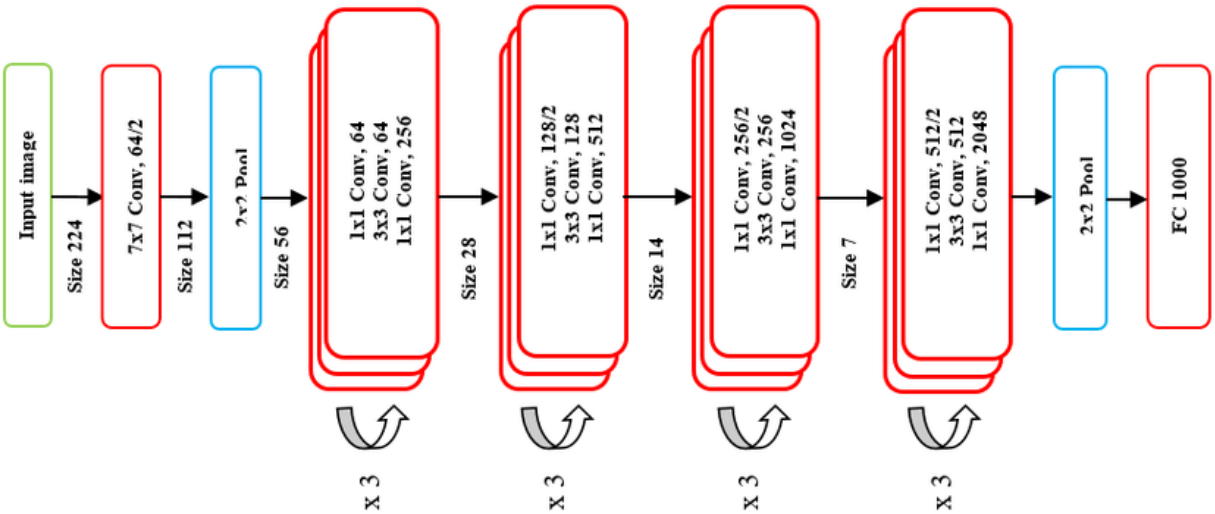

FIGURE 3.4 – Architecture du modèle resnet50

Ce système est une version open source disponible que nous avons utilisée dans notre système, où nous avons créé un modèle de réseau de neurones en utilisant l'architecture ResNet50, un modèle pré-entraîné que nous avons utilisé pour classer les différents types de image dans la base de données, Des couches supplémentaires sont ensuite ajoutées au modèle pour adapter la classification à cinq types de fleurs spécifiques : (roses, tulipes, marguerites, pissenlits et tournesols).

Le pipeline de cette partie est donné comme suivant :

3.4.4.1 Collecter la dataset

Les données sont la clé la plus importante du processus d'apprentissage, car c'est la première étape pour commencer à construire le modèle. En d'autres termes, pour entraîner un modèle d'apprentissage automatique, il est essentiel d'avoir accès à des données pertinentes et de qualité. La sélection et la collecte de ces données appropriées pour la formation et les tests sont souvent considérées comme une tâche difficile. Dans notre travail, nous avons rencontré une difficulté particulière lors de l'obtention de données brutes. Ces données peuvent être collectées de manière manuelle, en effectuant des mesures ou des observations, ou à l'aide de données provenant d'Internet. Cependant, il est important de noter que l'utilisation de données provenant d'Internet nécessite

de respecter les conditions d'utilisation autorisées, car la plupart des données ne sont pas disponibles gratuitement. De plus, pour notre projet, nous avons spécifiquement besoin d'images au format RVB, ce qui ajoute une contrainte supplémentaire lors de la recherche de données appropriées.

3.4.4.2 Préparation de données

Le prétraitement des données joue un rôle crucial dans la préparation des données. Il comprend plusieurs étapes essentielles. Le pipline présenté dans la figure 3.5 illustre le pipeline utilisé pour le prétraitement de notre jeu de données dans notre système.

FIGURE 3.5 – La structure de préparation de données

— **Importation des bibliothèques nécessaires :** Le premier pas dans le prétraitement des données consiste à utiliser des bibliothèques. Ces bibliothèques sont spécialement conçues pour stocker des routines couramment utilisées, ce qui facilite grandement le traitement de nos ensembles de données. Les bibliothèques fournies par Python nous permettent également de charger nos ensembles de données de manière simple et efficace.

— **Importation du jeu de données :** Il existe plusieurs méthodes d'importation de jeux de données, en fonction du format des fichiers de données tels que dataset.csv, etc. Chaque format de jeu de données peut être chargé à l'aide d'une bibliothèque spécifique.

— **Le fractionnement de jeu de données :** Lors de l'entraînement de tout modèle d'apprentissage automatique, il est essentiel de diviser le jeu de données en données d'entraînement et en données de test, indépendamment du type de jeu de données utilisé.

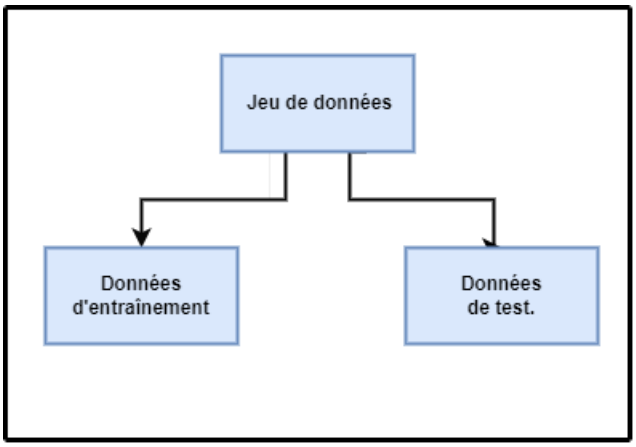

FIGURE 3.6 – Fractionnement de dataset.

L'entraînement du modèle se fait en utilisant les données d'entraînement, où le modèle est exposé aux sorties attendues. Les données de test sont ensuite utilisées pour évaluer la précision du modèle. Généralement, nous réservons 70% ou 80% des données pour l'entraînement, laissant les 30% ou 20% restants pour les tests. Dans notre travail, nous avons adopté un ratio de 80% pour l'entraînement du modèle.

3.4.4.3 Entrainement du modèle

Après avoir préparé les données et créé notre modèle, nous avons remplacé ce modèle par le processus d'entraînement et intégré les ensembles de données d'entraînement et de validation. À chaque itération ou époque, notre modèle apprend à partir de ces ensembles de données fournis. Les meilleurs poids générés sont enregistrés, et en fonction de ces poids, le modèle est automatiquement ajusté pour effectuer la tâche souhaitée. Cela permet d'améliorer progressivement les performances du modèle et de le rendre plus apte à la tâche de classification des fleurs.

3.4.4.4 Evaluation du modèle

À ce stade, il est nécessaire de procéder à l'évaluation de notre modèle formé en utilisant de nouvelles données qui lui sont inconnues. Grâce à son processus d'apprentissage, le modèle devrait être en mesure de classer les images correctement. Toutefois, si notre modèle ne se comporte pas de manière satisfaisante avec ces nouvelles données, il peut être nécessaire de revoir l'architecture du modèle et de le retrainer en utilisant de nouveaux paramètres.

3.4.4.5 La prédiction

La phase finale de notre travail consiste à effectuer des prédictions en utilisant notre modèle entraîné. Durant cette étape, nous chargeons le modèle et procédons à une opération de prédiction pour évaluer s'il est correctement formé et prêt à être utilisé dans des applications pratiques.

3.5 Implémentation et résultats obtenus

Cette section décrit les détails de l'implémentation de notre système. Nous présenterons l'environnement et les outils logiciels et matériels nécessaires pour la réalisation de notre travail, ensuite une série d'expériences et les résultats obtenus.

3.5.1 Environnements et outils

Pour atteindre l'objectif de notre travail, nous avons besoin de différents environnements et outils : langages de programmations, API, bibliothèques, IDE, etc

3.5.1.1 Configuration matérielle

— Pc portable : lenovo thinkpad
— Microprocesseur : Intel(R) Core(TM) i7-4600U CPU @ 2.69GHz
— Fréquence du processeur : 2.69 GHz.
— RAM : 12.0 GB.
— Système d'exploitation : Windows 10, 64 bits.

3.5.1.2 Environnement de développement :

Python
C'est nn langage de programmation interprété de haut niveau qui a été créé par Guido Van Rossum en 1991. La philosophie de conception de Python met l'accent sur la lisibilité du code grâce à l'utilisation de grands espaces blancs. Le langage offre des constructions et une approche orientée objet pour aider les programmeurs à écrire du code clair et logique pour des projets de petite et grande envergure. En somme, Python est un langage de programmation populaire en raison de sa simplicité et de sa facilité d'utilisation, ce qui le rend adapté pour une variété de projets [102].

FIGURE 3.7 – Logo de python

Colab Google Colab est un environnement de développement en ligne basé sur le cloud qui permet aux utilisateurs d'écrire, d'exécuter et de partager des notebooks Jupyter en utilisant le navigateur web. Il est basé sur l'infrastructure de Google et fournit un accès gratuit à une machine virtuelle avec des processeurs et des GPUs de haute performance. Google Colab prend en charge plusieurs langages de programmation, notamment Python, R et Scala, et offre une variété de bibliothèques de machine learning préinstallées pour faciliter le développement de modèles de deep learning. Les utilisateurs peuvent également partager leurs notebooks avec d'autres utilisateurs et collaborer en temps réel sur des projets [103].

FIGURE 3.8 – Logo de colab

OpenCv (Open Source Computer Vision Library) est une bibliothèque gratuite de logiciels de vision par ordinateur et d'apprentissage automatique. OpenCV a été développé pour aider les produits commerciaux à intégrer plus rapidement la perception artificielle et pour offrir une base standard pour les applications de vision par ordinateur [104].

FIGURE 3.9 – Logo de openCv

NumPy NumPy est une bibliothèque Python open source utilisée pour effectuer des calculs scientifiques et mathématiques complexes. Elle fournit des structures de données de tableau multidimensionnel hautement optimisées et des fonctions pour effectuer des opérations mathématiques de base, telles que l'addition, la soustraction, la multiplication et la division. NumPy est largement utilisé en science des données, en apprentissage automatique et en traitement du signal. Les tableaux NumPy sont plus rapides et plus efficaces que les listes Python ordinaires, ce qui les rend particulièrement adaptés pour les tâches nécessitant des calculs sur de grandes quantités de données [105].

FIGURE 3.10 – Logo de numpy

Google drive Google Drive est un service de stockage en ligne fourni par Google

qui permet aux utilisateurs de stocker, de partager et d'accéder à des fichiers et des dossiers depuis n'importe quel appareil connecté à Internet. Les utilisateurs peuvent télécharger des fichiers, tels que des documents, des images et des vidéos, sur Google Drive et y accéder à tout moment via un navigateur web ou l'application Google Drive. Ils peuvent également partager des fichiers avec d'autres personnes en leur envoyant un lien ou en leur donnant l'accès pour afficher ou modifier les fichiers. Google Drive offre un espace de stockage gratuit limité et un espace de stockage payant supplémentaire pour ceux qui ont besoin de plus d'espace [106].

FIGURE 3.11 – Logo de google drive

TensorFlow TensorFlow est une bibliothèque open source développée par Google pour la création et l'entraînement de modèles de machine learning, notamment des réseaux de neurones. Il permet aux développeurs de concevoir et de construire des modèles de machine learning de manière flexible et facilement extensible. TensorFlow fournit des outils pour créer des graphiques computationnels qui décrivent le flux de données entre les différentes opérations mathématiques, et exécuter ces opérations sur des processeurs graphiques ou des unités de traitement de tenseur (TPU) pour accélérer l'entraînement des modèles. TensorFlow est utilisé dans une variété de domaines, tels que la reconnaissance d'image, la traduction automatique, le traitement du langage naturel et le traitement du signal [107].

FIGURE 3.12 – Logo de TensorFlow

Matplotlib Matplotlib est une bibliothèque Python open source utilisée pour créer des visualisations de données en deux dimensions, telles que des graphiques, des histo-

grammes, des nuages de points et des diagrammes en boîte. Elle fournit une interface pour créer des graphiques de qualité professionnelle avec une personnalisation détaillée, telle que la couleur, la police, la taille et les étiquettes d'axe. Matplotlib peut être utilisé dans des applications interactives et non interactives, et permet de sauvegarder des graphiques dans divers formats d'image, tels que PNG, PDF, SVG et EPS. Elle est couramment utilisée en science des données, en visualisation de données, en recherche scientifique et en éducation pour représenter des données de manière claire et compréhensible [108].

FIGURE 3.13 – Logo de Matplotlib

Keras C'est une bibliothèque open source de haut niveau pour la création et l'entraînement de modèles de machine learning en Python. Elle fournit une interface simple et conviviale pour la création de modèles de réseaux de neurones artificiels, tels que les réseaux de neurones convolutifs (CNN) et les réseaux de neurones récurrents (RNN), en utilisant des blocs de construction prédéfinis appelés couches. Keras permet aux développeurs de créer des modèles de machine learning rapidement et facilement, avec une personnalisation limitée pour les utilisateurs plus avancés. Il peut être utilisé avec plusieurs bibliothèques de backend, telles que TensorFlow, Theano et CNTK, pour accélérer l'entraînement des modèles de machine learning. Keras est utilisé dans une variété de domaines, tels que la vision par ordinateur, le traitement du langage naturel, la reconnaissance de la parole et la recommandation de produits [109].

FIGURE 3.14 – Logo de keras

3.5.2 Processus de l'implémentation

Cette étape se compose de trois parties : La première partie consiste à générer les images, Et la deuxième partie pour optimise et sélectionner les meilleures images générées dans le groupe, La troisième partie consiste à pratiquer les images générées et optimise.

3.5.3 Implémentation de Phase de préparation de datast

a. Lien entre Google Drive et bloc-notes Colab : Dans cette étape, nous allons enregistrer les fichiers de la base de données sur la plate-forme Google Drive. À l'aide de la bibliothèque d'accès aux fichiers Google Drive, nous importons les fichiers dans Google Colab et montons le dossier Google Drive dans le système de fichiers du bloc-notes Colab. Cela crée un lien entre les deux. Cette étape est importante car elle permet aux utilisateurs d'accéder facilement aux fichiers stockés dans leur compte Google Drive à partir de leur bloc-notes Colab.La figure 3.15 montre le lien entre les plateformes Google Drive et bloc-notes Colab.

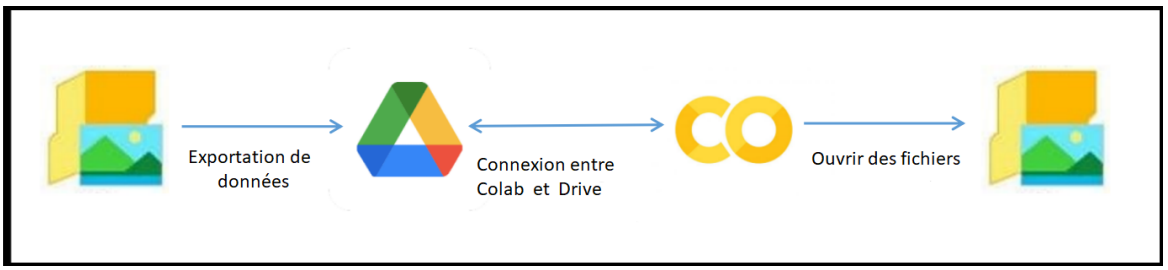

FIGURE 3.15 – Importation des fichiers de datast.

b. Lire et copier des dataset Dans cette étape, nous vérifions la validité des fichiers et fournissons leur contenu d'images que le réseau de neurones va apprendre, et faisons une copie du contenu du fichier "dataset" qui contient les images à étudier, et mettons une copie dans le nouveau fichier "attack dataset".

c. Définissez des chemins de fichiers et liez-les à leur classe Dans cette étapee, nous spécifions le chemin du fichier pour chaque classe de fleur (roses, tulipes, marguerite pissenlit, tournesols, etc.), par exemple, un fichier qui contient des "roses". Nous spécifions et définissons la catégorie de ce fichier comme un fleur du classe roses.

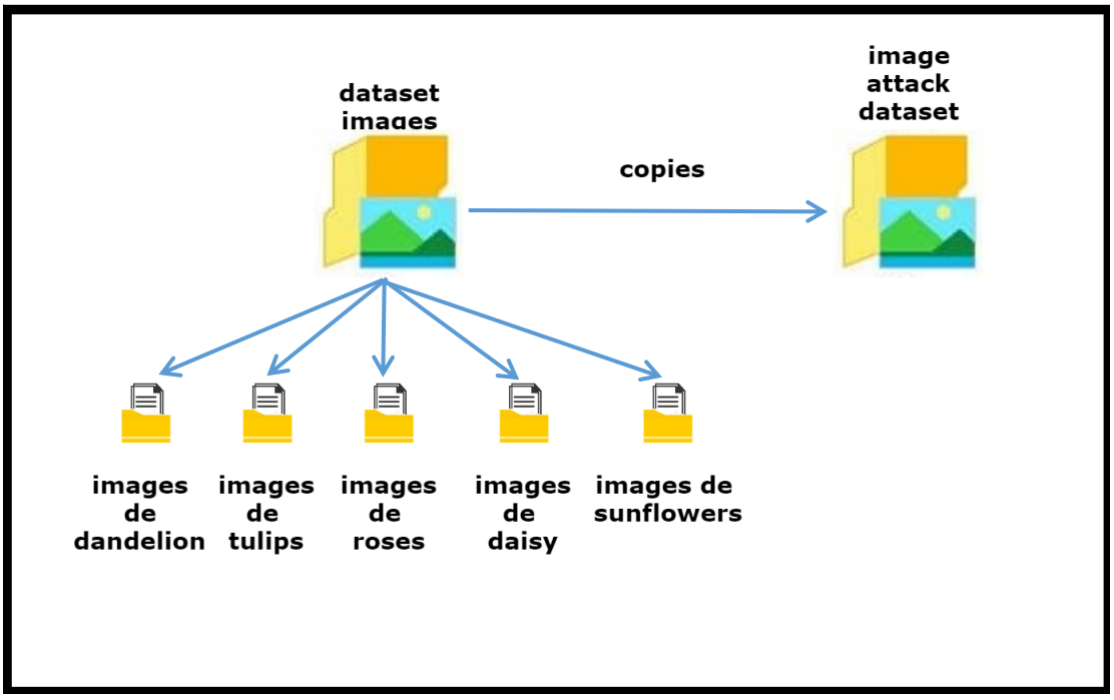

FIGURE 3.16 – Copier et définissez des chemins de fichiers.

3.5.3.1 Implémentation de la phase de la génération des images

a. Télécharger et éditer des images Nous chargeons une image à partir du chemin du fichier "attack dataset", qui est une copie exacte des fichiers de l'ensemble de données d'origine, la redimensionnons à 180 x 180 pixels, agrandissons ses dimensions pour qu'elles correspondent à la forme d'entrée du modèle ResNet50.

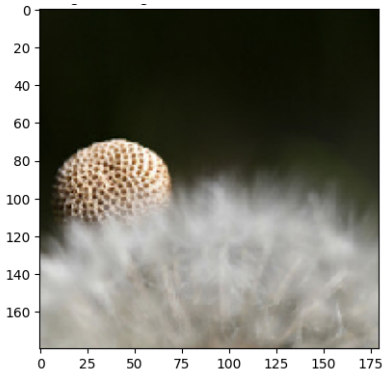

FIGURE 3.17 – Éditeur de l'image

b. Générer une image contradictoire par la fonction IBM (Générateur) Pour implémenter l'attaque, nous avons besoin de quelques entrées : Image de l'entrée d'origine extraite du fichier "attack dataset".

— et amplitude de perturbation maximale (epsilon) et (epsilon) aléatoire compris entre 0, 001 et 0, 01.
— Le nombre d'étapes à suivre dans le processus itératif.
— Et la taille du pas pour chaque perturbation.
— Il initialise l'image contradictoire comme une copie de l'image originale, puis la met à jour de manière itérative en ajoutant de la turbulence, en la découpant dans la plage de pixels valides et en ajoutant du bruit aléatoire pour briser les symétries. A chaque étape, il calcule le gradient d'une fonction de perte (entropie catégorielle) par rapport à l'image adverse à l'aide du mécanisme GradientTape de TensorFlow, et l'utilise pour calculer le désordre à ajouter. Enfin, il renvoie l'image contradictoire sous la forme d'un tenseur TensorFlow.

Vous trouverez ci-dessous une explication des équations permettant d'appliquer du bruit à une image.

Initialisation de l'image modifiée en tant que copie de l'image d'origine.

$$\mathbf{x^0_{adv}} = \mathbf{x} \tag{3.2}$$

À l'intérieur de la boucle itérative (pour chaque étape de 1 à num steps) :
Calcul des gradients de la perte par rapport à l'image modifiée à l'étape précédente en utilisant la fonction tape.gradient().

$$\mathbf{gradient} = \nabla_{\mathbf{x^{(i-1)}_{adv}}} \mathbf{loss}(\mathbf{model}, \mathbf{x^{(i-1)}_{adv}}, \mathbf{target\_label}) \tag{3.3}$$

Calcul des gradients signés en appliquant la fonction signe (sign) aux gradients.

$$\mathbf{signedgrad} = \text{sign}(\mathbf{gradient}) \tag{3.4}$$

Calcul de la perturbation à ajouter à l'image en multipliant les gradients signés par la taille du pas (step size).

$$\mathbf{perturbation} = \mathbf{step\_size} \times \mathbf{signed\_grad} \tag{3.5}$$

Mise à jour de l'image modifiée en ajoutant la perturbation.

$$\mathbf{x_{adv}^{(i)}} = \mathbf{x_{adv}^{(i-1)}} + \mathbf{perturbation} \tag{3.6}$$

Restriction des valeurs des pixels de l'image entre 0 et 255 en utilisant la fonction "clip" pour maintenir la plage de couleurs correcte.

$$\mathbf{x_{adv}^{(i)}} = \text{clip}(\mathbf{x_{adv}^{(i)}}, \mathbf{0}, \mathbf{255}) \tag{3.7}$$

Ajout d'un bruit aléatoire à la perturbation pour rompre les éventuelles symétries dans l'attaque.

$$\mathbf{noise} = \epsilon \times \text{normalize}(\mathbf{random\_noise}) \tag{3.8}$$

Encore une fois, restriction des valeurs des pixels de l'image pour maintenir la plage de couleurs correcte.

$$\mathbf{x_{adv}^{(i)}} = \mathbf{x_{adv}^{(i)}} + \mathbf{noise} \tag{3.9}$$

l'image modifiée après l'attaque est renvoyée en tant que résultat final.

$$\mathbf{x_{adv}^{(i)}} = \text{clip}(\mathbf{x_{adv}^{(i)}}, \mathbf{0}, \mathbf{255}) \tag{3.10}$$

$$\mathbf{x_{adv}^{(num\_steps)}} \tag{3.11}$$

Les résultats de l'ajout de bruit à une image sont affichés dans La figure 3.18 montre Générer une image contradictoire par la fonction IBM

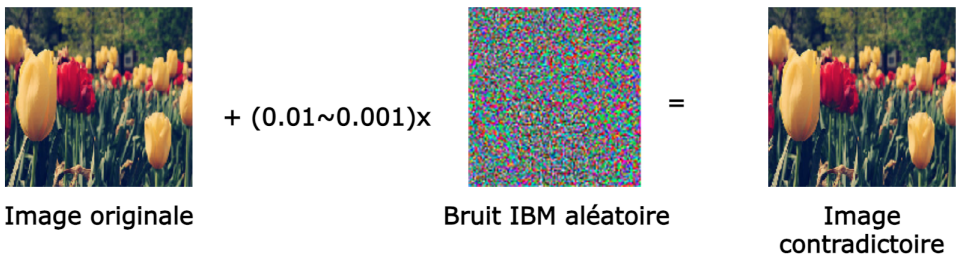

FIGURE 3.18 – Générer une image contradictoire par l'algorithme IBM.

c. L'étape d'assurer le succès des attaques et de mémoriser les images contradictoires réussies (discriminateur)

À ce stade, nous prenons l'original et l'image contradictoire comme entrée, et le chemin du fichier pour enregistrer l'image contradictoire. Utilisation du modèle ResNet50 pour faire des prédictions sur les deux images et stocker les étiquettes de classe et les probabilités prédites dans des dictionnaires. Si les étiquettes prédites pour les deux images

différent, il affiche les deux images avec les étiquettes prédites et les probabilités, Les images passent à une étape d'optimisation.

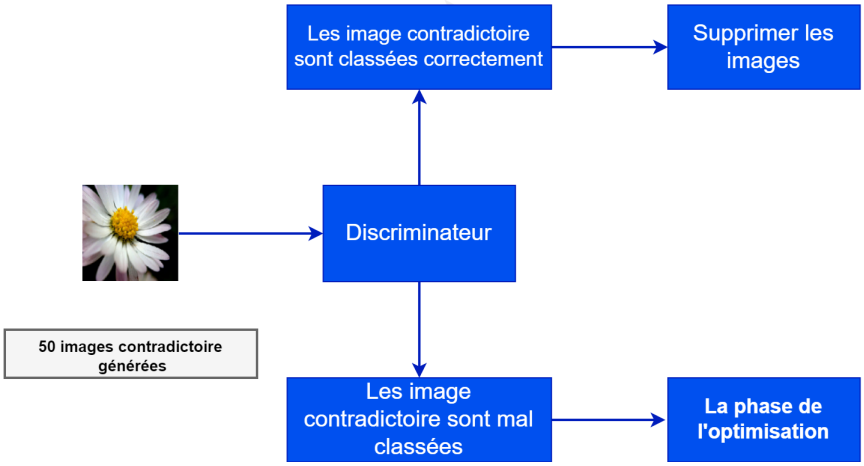

FIGURE 3.19 – Distinguer les images réussies et échouées pour la classification à l'aide du discriminateur

3.5.3.2 Implémentation de la phase de d'optimisation de l'image

A ce stade, nous améliorons les images contradictoire en utilisant une approche d'optimisation aléatoire et en définissant les règles nécessaires pour sélectionner les images à manipuler. Dans notre cas, la sélection est déterminée par deux critères. Tout d'abord, nous identifions les images ayant une valeur epsilon plus faible, ce qui indique une distorsion minimale. Cette distorsion réduite et efficace permet à l'image contradictoire d'être déformée précisément aux endroits appropriés, empêchant ainsi le réseau de neurones convolutif (CNN) de la classifier correctement. Deuxièmement, nous prenons en compte les images qui sont mal classées par le CNN avec un niveau de confiance élevé. Cela signifie que le CNN est très sûr de sa classification incorrecte de l'image contradictoire, exprimant ainsi un haut degré de confiance dans cette erreur de classification.La figure 3.20 montre les étapes d'optimisation d'une image.

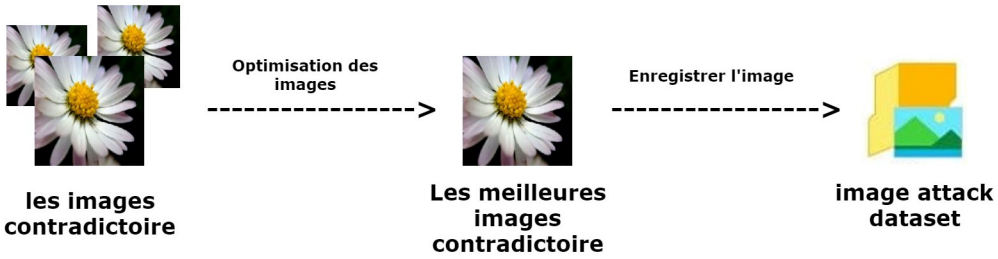

FIGURE 3.20 – Optimisation de l'image

Les résultats d'optimisation sont pour les images comme indiqué dans La figure 3.21.

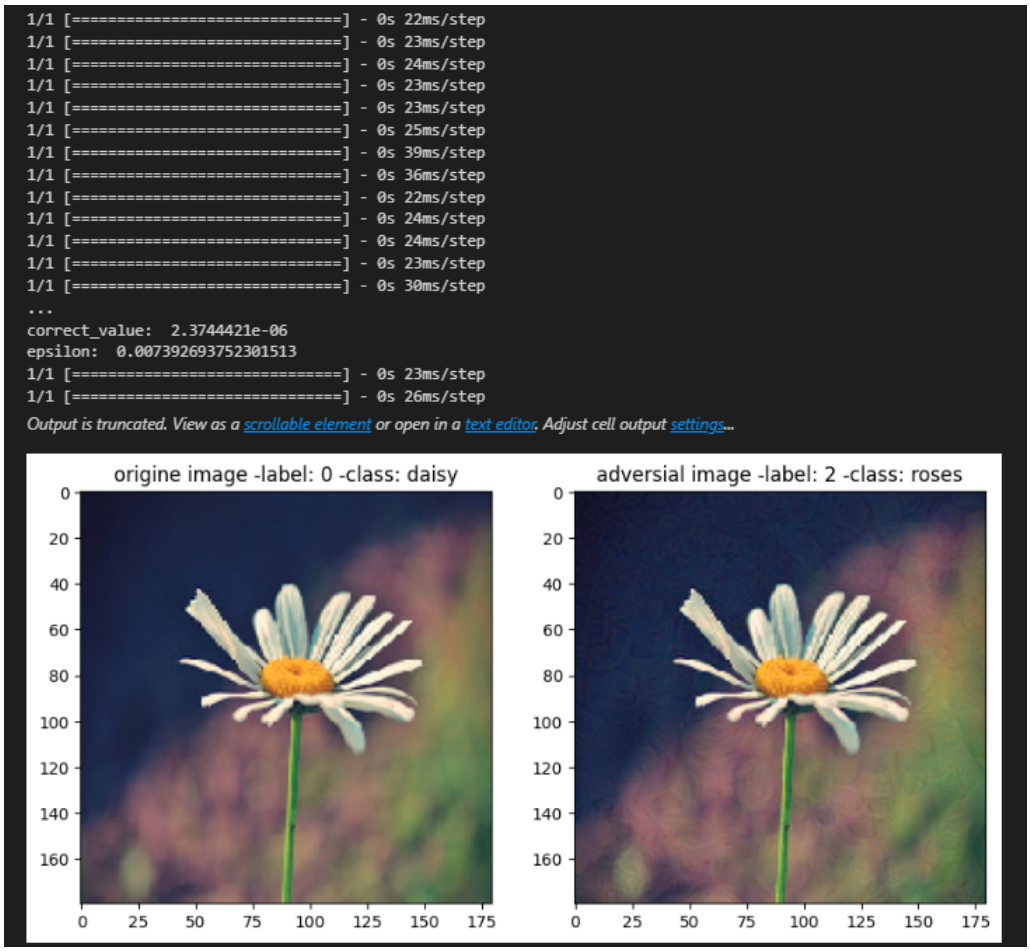

FIGURE 3.21 – Résultats de optimisation de l'image

3.5.3.3 Augmentation automatique des données

De cette manière, les 3 meilleures images sur 50 images contradictoire sont sélectionnées afin que leurs résultats soient bons et efficaces.Ce que nous remarquons, c'est que nous avons bénéficié de la formation d'agrandissement et d'augmentation des données.Nous pouvons multiplier les images à un nombre illimité d'images, mais cela coûte cher en calcul, mais cela reste des données efficaces. Un exemple de ceci : La base de données de départ pour le projet était de 3679 images après que les images aient été générées à un taux de 5% à partir de la base de données, soit 184 des images originales qui ont été déterminées pour produire des images contradictoire. projet, j'ai choisi les 3 meilleures images générées sur 50. Le résultat est 552 images générées, ce qui signifie que les données du projet sont devenues 4231 images qui peuvent être entraînées à l'aide du modèle CNN.

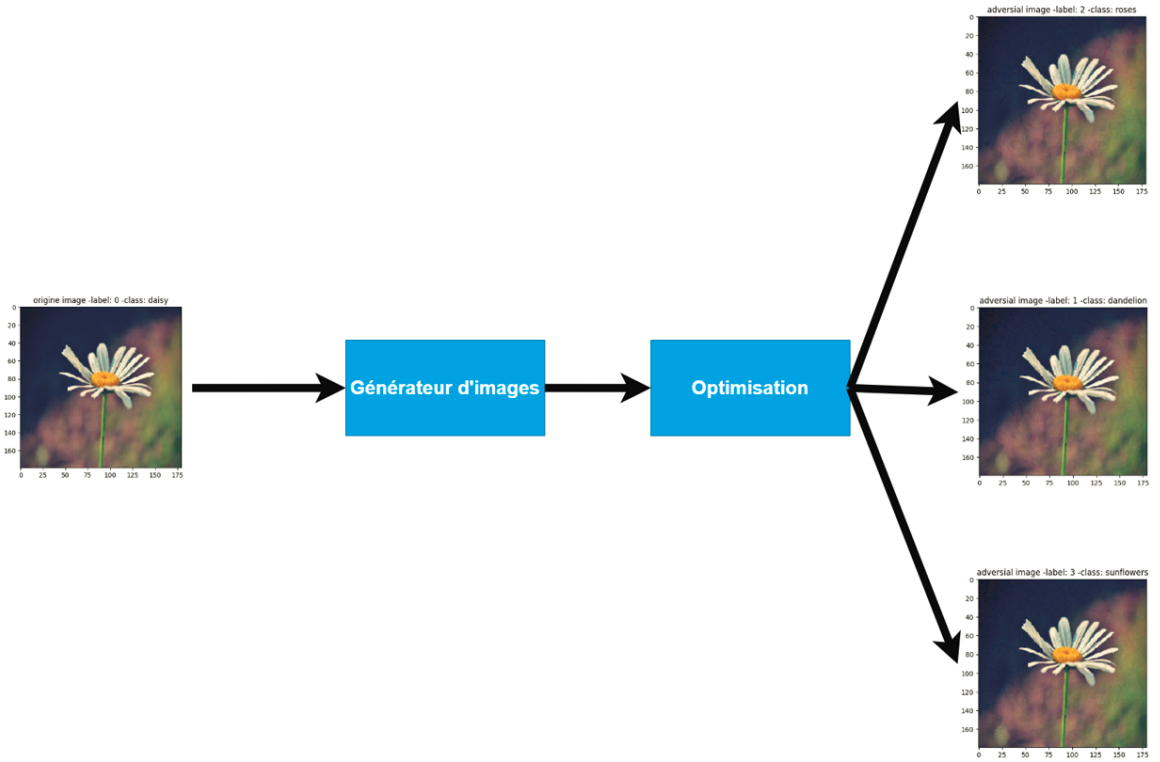

FIGURE 3.22 – Augmentation des données

3.5.3.4 Implémentation de la phase de entraînement de CNN sur les image contradictoir

Comme nous l'avons mentionné précédemment, les données qui ont été stockées dans le chemin du fichier "attack dataset" sont les données qui seront formées selon notre modèle ResNet50 précédemment formé.
La première étape consiste à créer un modèle de séquence à l'aide de la classe de séquence Keras. Le modèle ResNet50 pré-entraîné est chargé avec ses couches de convolution. Nous ajoutons nos propres couches de classification. Par conséquent, les couches ResNet50 sont gelées (non entraînables) pour éviter tout retour en arrière. propagation du gradient et mise à jour des poids pendant l'entraînement. La couche d'entrée dans notre modèle se compose d'une cellule avec des dimensions (180, 180, 3) où 180 est la dimension de l'image et 3 montrent les couches RVB et les paramètres de la couche de sortie se composent de 5 neurones (un pour chaque classe).

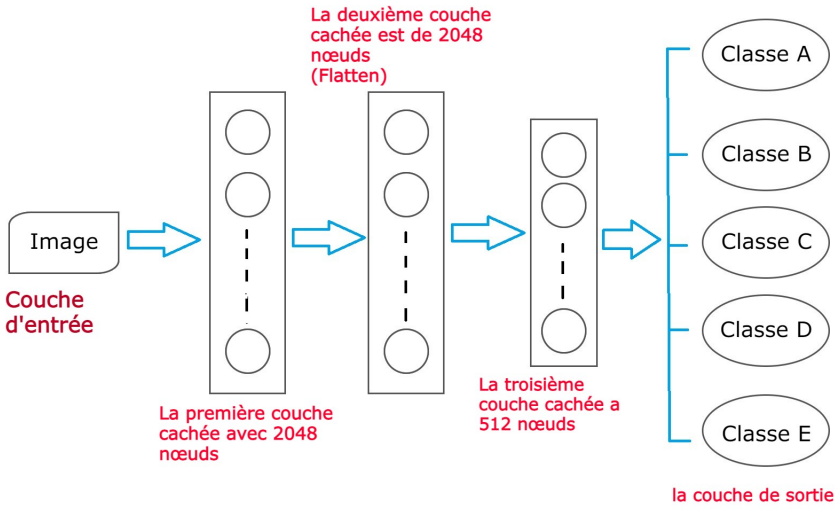

FIGURE 3.23 – La structure de notre modèle d'apprentissage

La figure 3.24 montre en détails l'architecture utilisée.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 resnet50 (Functional)       (None, 2048)              23587712

 module_wrapper (ModuleWrapp  (None, 2048)            0
 er)

 module_wrapper_1 (ModuleWra  (None, 512)             1049088
 pper)

 module_wrapper_2 (ModuleWra  (None, 5)               2565
 pper)

=================================================================
Total params: 24,639,365
Trainable params: 1,051,653
Non-trainable params: 23,587,712
_________________________________________________________________
```

FIGURE 3.24 – Aperçu sur notre modèle

C. L'entraînement après de la configuration de notre modèle en passe à l'étape d'entrainement.

```
Epoch 1/10
92/92 [==============================] - 339s 4s/step - loss: 0.6999 - accuracy: 0.7704 - val_loss: 0.4426 - val_accuracy: 0.8408
Epoch 2/10
92/92 [==============================] - 13s 140ms/step - loss: 0.2625 - accuracy: 0.9069 - val_loss: 0.3563 - val_accuracy: 0.8748
Epoch 3/10
92/92 [==============================] - 15s 155ms/step - loss: 0.1657 - accuracy: 0.9460 - val_loss: 0.4612 - val_accuracy: 0.8558
Epoch 4/10
92/92 [==============================] - 13s 136ms/step - loss: 0.0961 - accuracy: 0.9711 - val_loss: 0.4026 - val_accuracy: 0.8830
Epoch 5/10
92/92 [==============================] - 13s 134ms/step - loss: 0.0577 - accuracy: 0.9840 - val_loss: 0.4152 - val_accuracy: 0.8748
Epoch 6/10
92/92 [==============================] - 14s 142ms/step - loss: 0.0337 - accuracy: 0.9929 - val_loss: 0.3980 - val_accuracy: 0.8762
Epoch 7/10
92/92 [==============================] - 14s 142ms/step - loss: 0.0164 - accuracy: 0.9990 - val_loss: 0.4031 - val_accuracy: 0.8912
Epoch 8/10
92/92 [==============================] - 13s 140ms/step - loss: 0.0078 - accuracy: 0.9997 - val_loss: 0.4101 - val_accuracy: 0.8830
Epoch 9/10
92/92 [==============================] - 16s 158ms/step - loss: 0.0043 - accuracy: 1.0000 - val_loss: 0.4159 - val_accuracy: 0.8830
Epoch 10/10
92/92 [==============================] - 15s 155ms/step - loss: 0.0030 - accuracy: 1.0000 - val_loss: 0.4214 - val_accuracy: 0.8912
```

FIGURE 3.25 – Résultat d'entrainement de notre modèle

3.5.4 Résultats

L'évaluation est la phase durant laquelle nous vérifions si le modèle s'adapte correctement aux données nouvelles et non publiées. Le framework Keras propose une fonctionnalité d'évaluation permettant de valider et de vérifier la formation du modèle. Nous avons utilisé des images augmentées pour entraîner un modèle. Le résultat de cette formation est un modèle final de classification d'images, que nous comparons avec un modèle formé à partir des deux déclarations d'origine illustrées dans la figure 3.26.

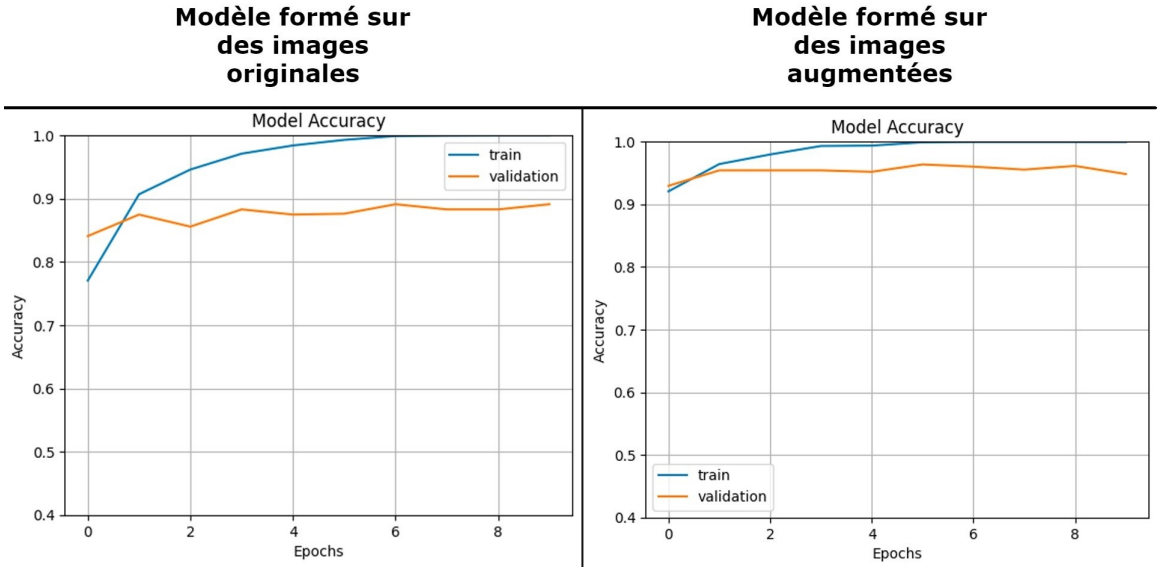

FIGURE 3.26 – Résultats de l'évaluation du premier modèle et du deuxième modèle

Les images suivantes représentent les résultats obtenus grâce à notre application après avoir généré des images, leur avoir appliqué une optimisation et les avoir stockées dans leur base de données.

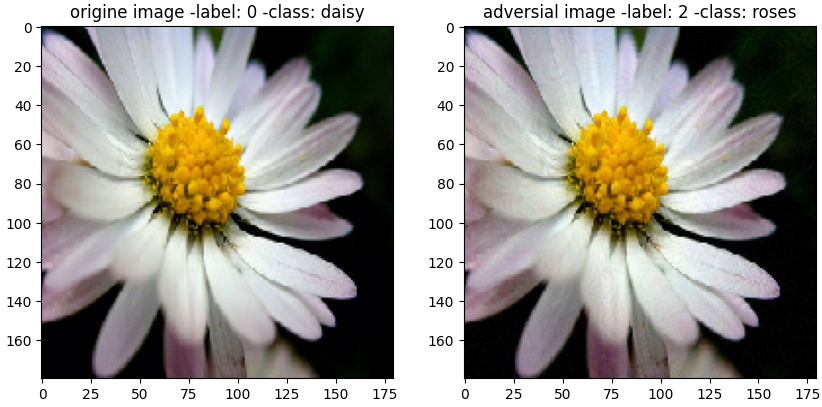

FIGURE 3.27 – Comparaison de l'image générée et de l'image d'origine dans sa classification, de sorte que la fleur daisy a été classée comme une roses.

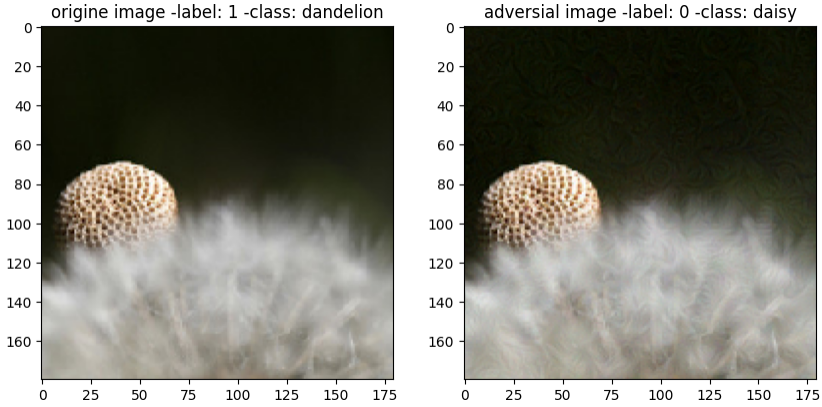

FIGURE 3.28 – Comparaison de l'image générée et de l'image d'origine dans sa classification, de sorte que la fleur de dandeline a été classée comme une daisy.

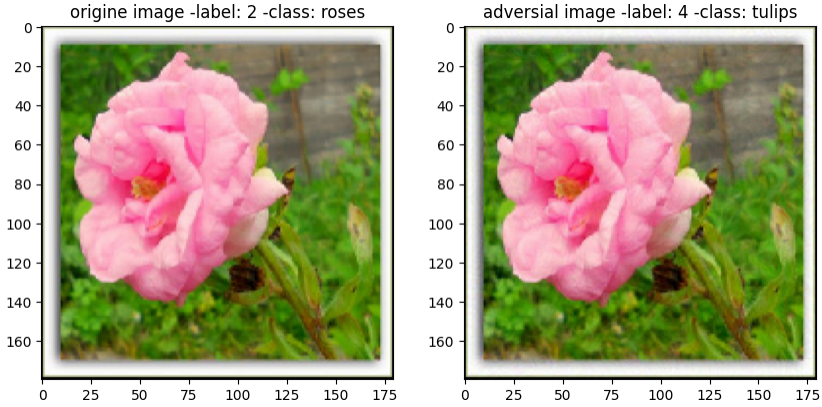

FIGURE 3.29 – Comparaison de l'image générée et de l'image d'origine dans sa classification, de sorte que la rose a été classée comme une tulips.

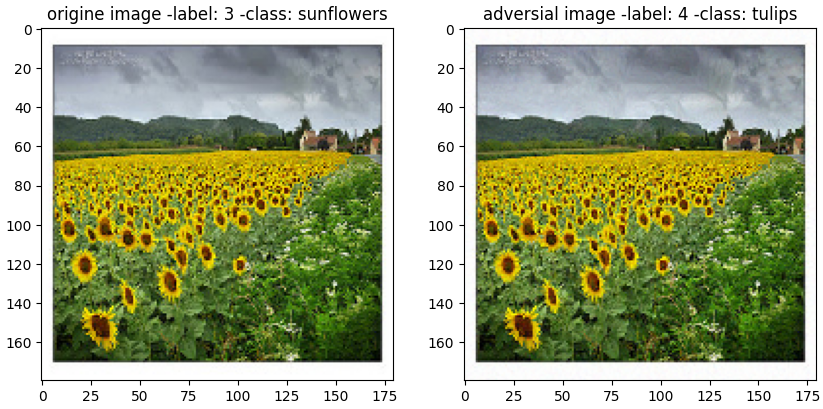

FIGURE 3.30 – Comparaison de l'image générée et de l'image d'origine dans sa classification, de sorte que la fleur de sunflowers a été classée comme une tulips.

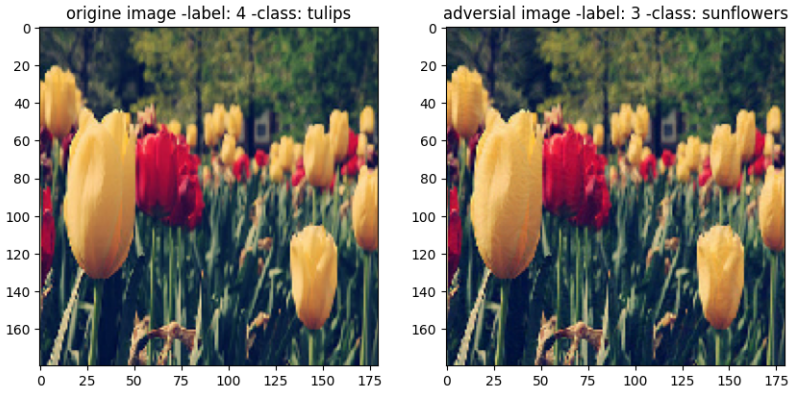

FIGURE 3.31 – Comparaison de l'image générée et de l'image d'origine dans sa classification, de sorte que la fleur de tulips a été classée comme une sunflowers.

3.5.5 Discussions

Selon les résultats que nous avons pu observer, nous avons d'abord obtenu une base de données qui a augmenté d'environ 15%. Tout cela a été réalisé lors de l'étape de génération d'images en appliquant le processus d'optimisation illustré dans la figure 3.22. Nous avons également réalisé une amélioration significative de l'évaluation, se traduisant par une augmentation de 6% de la précision, comme le montre la figure 3.26. Grâce à cette augmentation de précision, notre modèle final est résistant aux images contradictoires. Ces résultats découlent de notre travail basé sur le Générateur d'images GAN et l'utilisation de techniques d'optimisation aléatoire pour produire des images contradictoires. Il est donc possible de dire que la génération d'images contradictoires et leur utilisation pour entraîner un réseau de neurones convolutif ouvre de nouvelles perspectives dans le domaine de l'apprentissage profond, auquel nous contribuons à développer.

3.6 Conclusion

Dans ce chapitre, nous avons décris la conception globale, la mise en œuvre et les résultats de notre système utilisé pour la génération et l'optimisation d'images. Nous avons également présenté les différents détails techniques de notre implantation, ainsi que les résultats obtenus.

Conclusion générale

Ce mémoire de fin d'études a mis en évidence l'importance de l'utilisation de la génération d'images via les réseaux antagonistes génératifs (GAN) et l'optimisation pour améliorer les performances des réseaux profonds en vision par ordinateur. L'exploration des techniques d'augmentation de données à l'aide des GAN a permis d'élargir l'ensemble de données disponibles, augmentant ainsi la variété et la diversité des données d'entraînement. Cette approche s'est avérée extrêmement efficace pour renforcer les capacités d'apprentissage en profondeur des modèles en leur fournissant un large éventail d'exemples à partir desquels ils peuvent apprendre. De plus, l'utilisation de processus d'optimisation sur les images contradictoires a permis de sélectionner les meilleures images générées, renforçant ainsi la puissance et les capacités de généralisation des modèles profonds. Cette approche a joué un rôle essentiel dans l'amélioration des performances des réseaux profonds en leur permettant de gérer des scénarios complexes et difficiles dans les tâches de vision par ordinateur, telles que la reconnaissance, la segmentation et la détection d'objets. Les résultats de cette étude ont été prometteurs, démontrant des améliorations significatives de la précision des modèles utilisant l'augmentation de données par les GAN et l'optimisation des images contradictoires. Ces avancées ouvrent de nouvelles perspectives pour le développement de modèles d'apprentissage en profondeur plus robustes et polyvalents, capables de relever des défis dans différentes applications de vision par ordinateur. En conclusion, nous pouvons dire que l'utilisation de la génération d'images via les réseaux antagonistes et l'optimisation constitue une voie prometteuse pour améliorer les performances des réseaux profonds en vision par ordinateur, ouvrant ainsi de nouvelles possibilités pour la recherche future et l'application pratique dans ce domaine en constante évolution.

Bibliographie

1. Schumacher, D. *AI Image Generation* https://devinschumacher.com/ai-image-generators/ (2023).
2. Metri, O. & Mamatha, H. in *Generative Adversarial Networks for Image-to-Image Translation* 235-262 (Elsevier, 2021).
3. mathworks. *Train Variational Autoencoder (VAE) to Generate Images* https://www.mathworks.com/help/deeplearning/ug/train-a-variational-autoencoder-vae-to-generate-images.html.
4. Zhao, Z., Zhang, Z., Chen, T., Singh, S. & Zhang, H. Image augmentations for gan training. *arXiv preprint arXiv :2006.02595* (2020).
5. Wang, T.-C. *et al. High-resolution image synthesis and semantic manipulation with conditional gans* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), 8798-8807.
6. Brownlee, J. *A Gentle Introduction to CycleGAN for Image Translation* https://machinelearningmastery.com/what-is-cyclegan/ (2023).
7. Arjovsky, M., Chintala, S. & Bottou, L. *Wasserstein generative adversarial networks* in *International conference on machine learning* (2017), 214-223.
8. Karras, T. *et al. Analyzing and improving the image quality of stylegan* in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), 8110-8119.
9. Zhang, D. & Khoreva, A. Progressive augmentation of gans. *Advances in Neural Information Processing Systems* **32** (2019).
10. Tomczak, J. & Welling, M. *VAE with a VampPrior* in *International Conference on Artificial Intelligence and Statistics* (2018), 1214-1223.
11. Mezheritsky, T. *Modélisation du mouvement tridimensionnel du foie à partir d'images échographiques 2D par auto-encodeurs convolutionnels* thèse de doct. (Polytechnique Montréal, 2021).
12. Wang, Z. *et al.* Pianotree vae : Structured representation learning for polyphonic music. *arXiv preprint arXiv :2008.07118* (2020).
13. Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I. & Frey, B. Adversarial autoencoders. *arXiv preprint arXiv :1511.05644* (2015).
14. Felhi, G. Interpretable Sentence Representation with Variational Autoencoders and Attention. *arXiv preprint arXiv :2305.02810* (2023).

15. Lafont, P. Ouverture programmatique et inscription de la VAE dans le champ de la recherche comparative en éducation. *Institutionnalisation et internationalisation des dispositifs de reconnaissance et de validation des acquis de l'expérience, vecteur de renouvellement des relations entre univers de formation et de travail ?- Tome II : Politiques publiques et privées pour la mise en oeuvre des dispositifs de validation des acquis de l'experience et leur internationalisation* **2,** 327 (2014).

16. You, T., Kim, S., Kim, C., Lee, D. & Han, B. Locally Hierarchical Auto-Regressive Modeling for Image Generation. *Advances in Neural Information Processing Systems* **35,** 16360-16372 (2022).

17. Van Den Oord, A., Kalchbrenner, N. & Kavukcuoglu, K. *Pixel recurrent neural networks* in *International conference on machine learning* (2016), 1747-1756.

18. Ahmed, F. Generative models for natural images (2018).

19. Van den Oord, A., Kalchbrenner, N., Espeholt, L., Vinyals, O., Graves, A. *et al.* Conditional image generation with pixelcnn decoders. *Advances in neural information processing systems* **29** (2016).

20. Oord, A. *et al. Parallel wavenet : Fast high-fidelity speech synthesis* in *International conference on machine learning* (2018), 3918-3926.

21. Huang, C.-W., Krueger, D., Lacoste, A. & Courville, A. *Neural autoregressive flows* in *International Conference on Machine Learning* (2018), 2078-2087.

22. Kingma, D. P. & Dhariwal, P. Glow : Generative flow with invertible 1x1 convolutions. *Advances in neural information processing systems* **31** (2018).

23. Han, X., Hu, X., Huang, W. & Scott, M. R. *Clothflow : A flow-based model for clothed person generation* in *Proceedings of the IEEE/CVF international conference on computer vision* (2019), 10471-10480.

24. Weng, L. *Flow-based Deep Generative Models* `https://lilianweng.github.io/posts/2018-10-13-flow-models/`.

25. Dinh, L., Sohl-Dickstein, J. & Bengio, S. Density estimation using real nvp. *arXiv preprint arXiv :1605.08803* (2016).

26. Brunet, H. & Rocca-Serra, J. Model for a glow discharge in flowing nitrogen. *Journal of applied physics* **57,** 1574-1581 (1985).

27. Haxholli, E. & Lorenzi, M. Augmented Neural ODE Flows (2022).

28. Papamakarios, G., Pavlakou, T. & Murray, I. Masked autoregressive flow for density estimation. *Advances in neural information processing systems* **30** (2017).

29. Kingma, D. P. *et al.* Improved variational inference with inverse autoregressive flow. *Advances in neural information processing systems* **29** (2016).

30. Goodfellow, I. *et al.* Generative adversarial networks. *Communications of the ACM* **63,** 139-144 (2020).

31. Goodfellow, I. *et al.* Generative adversarial networks. *Communications of the ACM* **63,** 139-144 (2020).

32. Hayes, J., Melis, L., Danezis, G. & De Cristofaro, E. LOGAN : Evaluating Privacy Leakage of Generative Models Using Generative Adversarial Networks (2017).
33. Creswell, A. *et al.* Generative adversarial networks : An overview. *IEEE signal processing magazine* **35,** 53-65 (2018).
34. Goodfellow, I. Nips 2016 tutorial : Generative adversarial networks. *arXiv preprint arXiv :1701.00160* (2016).
35. Mendoza, P. Alpha-Beta Pruning Algorithm : The Intelligence Behind Strategy Games (2022).
36. TechTarget, L. R. *Réseau antagoniste génératif (GAN)* `https://www.lemagit.fr/definition/Reseau-antagoniste-generatif-GAN` (2023).
37. Li, Z. *et al.* A comprehensive survey on data-efficient GANs in image generation. *arXiv preprint arXiv :2204.08329* (2022).
38. Pan, Z. *et al.* Recent progress on generative adversarial networks (GANs) : A survey. *IEEE access* **7,** 36322-36333 (2019).
39. Ganegedara, T. *Intuitive Guide to Neural Style Transfer* `https://towardsdatascience.com/light-on-math-machine-learning-intuitive-guide-to-neural-style-transfer-ef88e46697ee`.
40. *Transfert de style avec CycleGAN* `https://datascientest.com/transfert-de-style-avec-cyclegan` (2023).
41. Ledig, C. *et al. Photo-realistic single image super-resolution using a generative adversarial network* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 4681-4690.
42. Brownlee, J. *18 Impressive Applications of Generative Adversarial Networks (GANs)* `https://machinelearningmastery.com/impressive-applications-of-generative-adversarial-networks/` (2023).
43. Tu, J., Mei, G., Ma, Z. & Piccialli, F. SWCGAN : Generative Adversarial Network Combining Swin Transformer and CNN for Remote Sensing Image Super-Resolution. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing* (2022).
44. Cha, D. & Kim, D. *DAM-GAN : Image Inpainting using Dynamic Attention Map based on Fake Texture Detection* in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), 4883-4887.
45. Ravi, S., Dinh, T., Lokhande, V. & Singh, V. Constrained Deep Learning using Conditional Gradient and Applications in Computer Vision (2018).
46. Elharrouss, O., Almaadeed, N., Al-ma'adeed, S. & Akbari, Y. Image Inpainting : A Review. *Neural Processing Letters* **51** (2020).
47. Zhang, H. *et al. Stackgan : Text to photo-realistic image synthesis with stacked generative adversarial networks* in *Proceedings of the IEEE international conference on computer vision* (2017), 5907-5915.
48. Bergmann, U., Jetchev, N. & Vollgraf, R. Learning Texture Manifolds with the Periodic Spatial GAN (2017).

49. Creswell, A. *et al.* Generative Adversarial Networks : An Overview. *IEEE Signal Processing Magazine* **35** (2017).

50. Lu, K., Qian, Z., Wang, M., Wang, D. & Ma, P. Shift-invariant universal adversarial attacks to avoid deep-learning-based modulation classification. *International Journal of Communication Systems* (2023).

51. Liu, Y., Mao, S., Mei, X., Yang, T. & Zhao, X. *Sensitivity of adversarial perturbation in fast gradient sign method* in *2019 IEEE symposium series on computational intelligence (SSCI)* (2019), 433-436.

52. Göpfert, J., Artelt, A., Wersing, H. & Hammer, B. in, 235-247 (2020).

53. Ilanchezian, I. *et al.* Maximal adversarial perturbations for obfuscation : Hiding certain attributes while preserving rest. *arXiv preprint arXiv :1909.12734* (2019).

54. Harder, P., Pfreundt, F.-J., Keuper, M. & Keuper, J. *Spectraldefense : Detecting adversarial attacks on cnns in the fourier domain* in *2021 International Joint Conference on Neural Networks (IJCNN)* (2021), 1-8.

55. Göpfert, J., Artelt, A., Wersing, H. & Hammer, B. in, 235-247 (2020).

56. Sun, J. *et al.* Generative adversarial networks with mixture of t-distributions noise for diverse image generation. *Neural Networks* **122,** 374-381 (2020).

57. Kaelbling, L. P., Littman, M. L. & Moore, A. W. Reinforcement learning : A survey. *Journal of artificial intelligence research* **4,** 237-285 (1996).

58. Bartz-Beielstein, T., Branke, J., Mehnen, J. & Mersmann, O. Evolutionary algorithms. *Wiley Interdisciplinary Reviews : Data Mining and Knowledge Discovery* **4,** 178-195 (2014).

59. Bapst, V., Foini, L., Krzakala, F., Semerjian, G. & Zamponi, F. The quantum adiabatic algorithm applied to random optimization problems : The quantum spin glass perspective. *Physics Reports* **523,** 127-205 (2013).

60. Tenenbaum, J. B., Silva, V. d. & Langford, J. C. A global geometric framework for nonlinear dimensionality reduction. *science* **290,** 2319-2323 (2000).

61. Kingma, D. P. & Welling, M. Auto-encoding variational bayes. *arXiv preprint arXiv :1312.6114* (2013).

62. Gatys, L. A., Ecker, A. S. & Bethge, M. A neural algorithm of artistic style. *arXiv preprint arXiv :1508.06576* (2015).

63. Cao, Y.-J. *et al.* Recent advances of generative adversarial networks in computer vision. *IEEE Access* **7,** 14985-15006 (2018).

64. Bouville, C., Brusq, R., Dubois, J.-L. & Marchal, I. *Generating high quality pictures by ray-tracing* in *Computer Graphics Forum* **4** (1985), 87-99.

65. Sunkavalli, K., Joshi, N., Kang, S. B., Cohen, M. F. & Pfister, H. Video snapshots : Creating high-quality images from video clips. *IEEE transactions on visualization and computer graphics* **18,** 1868-1879 (2012).

66. Wang, Z., Bovik, A. C., Sheikh, H. R. & Simoncelli, E. P. Image quality assessment : from error visibility to structural similarity. *IEEE transactions on image processing* **13,** 600-612 (2004).

67. Rumelhart, D. E., Hinton, G. E. & Williams, R. J. *Learning internal representations by error propagation* rapp. tech. (California Univ San Diego La Jolla Inst for Cognitive Science, 1985).
68. Bishop, C. M. & Nasrabadi, N. M. *Pattern recognition and machine learning* **4** (Springer, 2006).
69. Gardeux, V. *Conception d'heuristiques d'optimisation pour les problèmes de grande dimension. Application à l'analyse de données de puces à ADN.* thèse de doct. (2011).
70. Radford, A., Metz, L. & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv :1511.06434* (2015).
71. Zhou, X. Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagation. *Journal of Physics : Conference Series* **1004** (2018).
72. Kingma, D. P. & Ba, J. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980* (2014).
73. Jegadeesh, N. & Titman, S. Momentum. *Annu. Rev. Financ. Econ.* **3,** 493-509 (2011).
74. Johnson, J. M. & Rahmat-Samii, Y. *Genetic algorithm optimization and its application to antenna design* in *Proceedings of IEEE Antennas and Propagation Society International Symposium and URSI National Radio Science Meeting* **1** (1994), 326-329.
75. Bandyopadhyay, S., Saha, S., Maulik, U. & Deb, K. A simulated annealing-based multiobjective optimization algorithm : AMOSA. *IEEE transactions on evolutionary computation* **12,** 269-283 (2008).
76. Dimara, A. & Anagnostopoulos, C.-N. in, 76-94 (2018).
77. Wierstra, D. *et al.* Natural evolution strategies. *The Journal of Machine Learning Research* **15,** 949-980 (2014).
78. Bergstra, J. & Bengio, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* **13,** 281-305 (2012).
79. Chawla, N. V. Data mining for imbalanced datasets : An overview. *Data mining and knowledge discovery handbook,* 875-886 (2010).
80. Arora, S. & Barak, B. *Computational complexity : a modern approach* (Cambridge University Press, 2009).
81. Boyd, S. P. & Vandenberghe, L. *Convex optimization* (Cambridge university press, 2004).
82. Grossman, R. L., Kamath, C., Kegelmeyer, P., Kumar, V. & Namburu, R. *Data mining for scientific and engineering applications* (Springer Science & Business Media, 2013).
83. Murphy, K. P. *Machine learning : a probabilistic perspective* (MIT press, 2012).
84. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9,** 1735-1780 (1997).

85. Glorot, X. & Bengio, Y. *Understanding the difficulty of training deep feedforward neural networks* in *Proceedings of the thirteenth international conference on artificial intelligence and statistics* (2010), 249-256.
86. Zhang, C. *et al.* Gradient descent optimization in deep learning model training based on multistage and method combination strategy. *Security and Communication Networks* **2021,** 1-15 (2021).
87. Szegedy, C. *et al.* Intriguing properties of neural networks. *arXiv preprint arXiv :1312.6199* (2013).
88. Dean, J. *et al.* Large Scale Distributed Deep Networks. *Advances in neural information processing systems* (2012).
89. Boyd, S. P. & Vandenberghe, L. *Convex optimization* (Cambridge university press, 2004).
90. Christensen, J. *Topology Optimisation of Structures Exposed to Large (nonlinear) Deformations* thèse de doct. (2015).
91. Wright, J. N. S. J. *Numerical optimization* 2006.
92. Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V. & Courville, A. C. Improved training of wasserstein gans. *Advances in neural information processing systems* **30** (2017).
93. Salimans, T. *et al.* Improved techniques for training gans. *Advances in neural information processing systems* **29** (2016).
94. Radford, A., Metz, L. & Chintala, S. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv :1511.06434* (2015).
95. Donahue, J., Krähenbühl, P. & Darrell, T. Adversarial feature learning. *arXiv preprint arXiv :1605.09782* (2016).
96. Miyato, T., Kataoka, T., Koyama, M. & Yoshida, Y. Spectral normalization for generative adversarial networks. *arXiv preprint arXiv :1802.05957* (2018).
97. Salimans, T. *et al.* Improved techniques for training gans. *Advances in neural information processing systems* **29** (2016).
98. Isola, P., Zhu, J.-Y., Zhou, T. & Efros, A. A. *Image-to-image translation with conditional adversarial networks* in *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), 1125-1134.
99. Reed, S. *et al. Generative Adversarial Text to Image Synthesis* in (2016).
100. *flower photos* `https://creativecommons.org/licenses/by/2.0/`.
101. Jahromi, M. *et al.* Privacy-Constrained Biometric System for Non-Cooperative Users. *Entropy* **21,** 1033 (2019).
102. Python, W. Python. *Python Releases Wind* **24** (2021).
103. Bisong, E. & Bisong, E. Google colaboratory. *Building machine learning and deep learning models on google cloud platform : a comprehensive guide for beginners,* 59-64 (2019).
104. Bradski, G., Kaehler, A. *et al.* OpenCV. *Dr. Dobb's journal of software tools* **3** (2000).

105. McKinney, W. *Python for data analysis : Data wrangling with Pandas, NumPy, and IPython* (" O'Reilly Media, Inc.", 2012).

106. Quick, D. & Choo, K.-K. R. Google Drive : Forensic analysis of data remnants. *Journal of Network and Computer Applications* **40,** 179-193 (2014).

107. Developers, T. TensorFlow. *Zenodo* (2022).

108. Bisong, E. & Bisong, E. Matplotlib and seaborn. *Building Machine Learning and Deep Learning Models on Google Cloud Platform : A Comprehensive Guide for Beginners,* 151-165 (2019).

109. Gulli, A. & Pal, S. *Deep learning with Keras* (Packt Publishing Ltd, 2017).