République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Mémoire de Master

Présenté à l'Université de Tébessa
Faculté des Sciences Exactes et Sciences de la Nature et de la Vie

Département de : **Mathématiques et Informatique**
Spécialité : **Informatique**
Option : **Systèmes et Multimédias**

Par :
KERDOUD Fateh

Un système de compression d'images basé sur les réseaux de neurones profonds

JURY

Encadreur	M.C.A	**MENASSEL Rafik**	Université de Tébessa
Président	M.C.A	**LAIMECHE Lakhdar**	Université de Tébessa
Examinateur	M.C.A	**GATTAL Abdeljalil**	Université de Tébessa

2020/2021

بسم الله الرحمن الرحيم

Dédicaces

Je dédie mon travail de thèse à ma famille et à mes nombreux amis. Un sentiment particulier de gratitude à l'égard de mes parents Abdelkrim Allah yarhmou qui je l'aime tant et toujours, à ma superbe mère Nouara, dont les mots d'encouragement et de poussée à la ténacité résonnent dans mes oreilles. Ma chère femme, mes merveilleuses filles Zineb, Meriem et Rokia n'ont jamais quitté mes côtés, qui ont beaucoup sacrifiées pour moi et sont très spéciales.

Je dédie également cette thèse à toute ma famille (frères et sœurs), mes nombreux amis spécialement Hamza, Mahmoud et à la famille du groupe C1.1 printemps 2021 CCF Constantine.

Vous avez tous été mes meilleurs supporters.

KERDOUD Fateh

Remerciements

C'est avec un grand plaisir que je réserve ces quelques lignes en signe de gratitude et de profonde reconnaissance à tous ceux qui de près ou de loin, ont contribué à la réalisation et l'aboutissement de ce travail

*Tout d'abord, je tiens à remercier **Allah** tout puissant, de me permettre de mener à bien ce mémoire, et de m'orienter au chemin du savoir.*

*Ensuite, je remercie sincèrement Mr. **MENASSEL Rafik**, Maitre de conférences au sein du département des mathématiques et informatique de la faculté des sciences exactes et sciences de la nature et de la vie à l'université de Tébessa, pour son encadrement, son assistance, son soutien, sa disponibilité et ses précieux conseils.*

*Mes vifs remerciements s'adressent àDr. **LAIMECHE Lakhdar et** Dr. **GATTAL Abdeljalil**, pour l'honneur qu'ils m'ont accordé en acceptants de juger mon travail.*

Enfin, j'adresse mes chaleureux remerciements à mes enseignants pour la qualité de l'enseignement qu'ils ont bien voulu me prodiguer durant mes études afin de me fournir une formation efficace.

Merci à toutes et tous.

Résumé

La compression des images est un besoin qui se fait sentir dans de nombreuses circonstances. Malheureusement, chaque fois qu'un algorithme de compression avec perte est utilisé, des artefacts se manifestent. Les artefacts d'image, causés par la compression, ont tendance à éliminer les détails à haute fréquence et dans certains cas peuvent ajouter du bruit ou de petites structures d'image.

Plusieurs outils ont été utilisés dans le cadre de l'optimisation des méthodes de compression d'images dans le but de réduire l'impact causé par ces dernières sur l'appréciation des images par l'utilisateur humain ou bien logiciel.

Parmi les techniques introduite afin d'améliorer la qualité des images comprimées on peut citer : les métaheuristiques d'optimisation et ***les technique de machine Learning***

Dans ce mémoire de Master, nous avons essayé d'étudier la façon d'utiliser l'une des techniques de machine Learning (Deep Learning) à savoir les réseaux de neurones convolutifs pour la réduction des artéfacts de compression d'images (débrouillage).

En effet, on a utilisé une approche basée sur l'apprentissage profond pour réduire les artefacts de compression d'images.Plus précisément, on a proposé un réseau neuronal convolutif (CNN) efficace et fondé sur des principes pour cette tâche qui nous a permis de débrouiller ces images. On a surnommé le réseau proposé pour le débrouillage (DBRCNN) comme acronyme de Débrouillage CNN.

Notre DBRCNN est capable de produire des images avec plus de détails que les méthodes génériques ainsi que les autres méthodes profondes vues dans les travaux connexes.

Mots-clés : Compression d'images, Apprentissage Automatique, Apprentissage Profond, Débrouillage d'image, Qualité d'image.

Abstract

Compression of images is a need that arises in many circumstances. Unfortunately, anytime a lossy compression algorithm is used, artifacts will occur. Image artifacts, caused by compression, tend to eliminate high frequency detail and in some cases can add noise or small image structures.

Several tools have been used in the optimization of image compression methods in order to reduce the impact caused by them on the appreciation of images by the human or software user.

Among the techniques introduced to improve the quality of compressed images are: optimization metaheuristics and machine learning techniques

In this Master thesis, we tried to study how to use one of the machine learning techniques (Deep Learning) namely convolutional neural networks for the reduction of image compression artifacts (descrambling).

Indeed, a deep learning based approach has been used to reduce image compression artifacts. Specifically, an efficient and principle-based convolutional neural network (CNN) was proposed for this task that allowed us to unravel these images. The Proposed deblurring Network (DBRCNN) has been nicknamed as an acronym for CNN deblurring.

Our DBRCNN is capable of producing images with more detail than the generic methods as well as the other deep methods seen in related work.

Keywords: Image Compression, Machine Learning, Deep Learning, Deblurring images Image Quality.

ملخص:

يحقق ضغط الصور المفقود معدلات ضغط عالية عن طريق إزالة المعلومات التي لا تساهم أو تساهم بأقل قدر ممكن في إدراك الإنسان للصور. نظرًا لقيود النظام البصري البشري ، قد يكون فقدان المعلومات هذا مقبولًا في العديد من السيناريوهات ، لكن المشغولات المرئية المقدمة تصبح غير مقبولة عند معدلات ضغط أعلى. تركز هذه المقالة على طريقة ضغط JPEG.

حيث يقدم هذا العمل طريقة جديدة لاستعادة الصورة (ازالة الضبابية) باستخدام الشبكات التلافيفية التي تمثل تقدمًا كبيرًا مقارنة بالطرق الأخرى. نقترح نهجًا نسميه (DBRCNN) حيث تقبل الشبكة التلافيفية صورة متدهورة كمدخلات وتنتج صورة عالية الجودة.

توضح هذه المقالة أنه من الممكن تدريب الشبكات العصبية التلافيفية الكبيرة والعميقة (CNNs) لتقليل فك ضغط JPEG ، وأن هذا النهج يمكن أن يوفر جودة إعادة بناء أفضل بشكل ملحوظ من الشبكات المقترحة في الأعمال ذات الصلة ، وكذلك أي شبكات متقدمة أخرى.

الكلمات المفتاحية: استرجاع الصور، فك الترميز ، تقليل ضبابية الصور ، ضغط الصور ، التعلم الآلي ، التعلم العميق ، جودة الضغط

Table des matières

Liste des images

Liste des tables

Introduction générale

La compression des images est un besoin qui se fait sentir dans de nombreuses circonstances. Malheureusement, chaque fois qu'un algorithme de compression avec perte est utilisé, des artéfacts se manifestent. Les artefacts d'image, causés par la compression, ont tendance à éliminer les détails à haute fréquence et dans certains cas peuvent ajouter du bruit ou de petites structures d'image. Ce phénomène présente deux inconvénients majeurs.

- Premièrement, les images apparaissent beaucoup moins agréables à l'œil humain.
- Ensuite, les algorithmes de vision par ordinateur, tels que les détecteurs d'objets, peuvent être gênés et leurs performances réduites. L'élimination de ces artefacts implique la récupération de l'image originale à partir d'une version perturbée de celle-ci.

Plusieurs outils ont été utilisés dans le cadre de l'optimisation des méthodes de compression d'images dans le but de réduire l'impact causé par ces dernières sur l'appréciation des images par l'utilisateur humain ou bien logiciel.

Parmi les techniques introduite afin d'améliorer la qualité des images comprimées on peut citer : les métaheuristiques d'optimisation et ***les techniques de machine Learning***

Dans ce mémoire de Master, nous allons essayer d'étudier la façon d'utiliser l'une des techniques de machine Learning (Deep Learning) à savoir les réseaux de neurones convolutifs pour la réduction des artéfacts de compression d'images (débrouillage).

Nous allons essayer de concevoir une architecture du réseau qui consiste en une séquence de couches convolutionnelles pour effectuer la réduction des artéfacts et l'extraction des caractéristiques.

Donc quelle est la structure CNN optimale pour traiter cette réduction?

Pour répondre à cette question on a vu nécessaire d'organiser ce mémoire de la manière suivante :

Tout d'abord, nous commençons ce mémoire par une introduction afin de se mettre dans le contexte d'étude, ainsi poser la problématique traitée dans ce mémoire.

Dans le premier chapitre, on va se focaliser sur les techniques de compression d'image, les différentes distorsions des images, les différents métriques d'évaluation de la compression ainsi en introduisant les principes de base du *machine Learning*, en se concentrant sur les techniques que nous exploiterons plus particulièrement dans nos développements, qui sont le ***Deep Learning*** en particulier les réseaux de neurones convolutionnels (CNN – *Convolutional Neural Networks*), particulièrement adaptés aux traitements des images.

Le chapitre suivant s'intéresse sur les travaux abordés dansle contexte de l'intégration des algorithmes d'apprentissage profonds dans la réduction des artéfacts de compression d'image ainsi qu'une présentation de l'approche proposée qui se base sur le CNN.

Enfin le dernier chapitre va étaler les différents résultats (obtenus après l'application de notre approche sur le Dataset intitulé UTKFace) et les comparaisons de ces derniers avec les travaux connexes

Eton clôt ce mémoire de thèse, avec une conclusion sur le travail réalisé et une discussion sur les différentes pistes d'améliorations futures.

Chapitre01:

Notions de base sur la compression d'image et l'apprentissage profond

1- Introduction :

Ces dernières années, le développement et la demande de produits multimédias augmentent de plus en plus rapide, ce qui contribue à l'insuffisance de la bande passante du réseau et du stockage des mémoire.

La compression d'images est très importante pour réduire l'espace disque utilisé ainsi que la quantité de bande passante Internet utilisée lors du chargement des images. Dans ce chapitre, on va se focaliser sur cette technique ainsi que l'apprentissage profond qui peut être utilisé pour compresser les images afin d'améliorer ses performances.

2- Définition d'une image :

Une image est une représentation visuelle de quelque chose : comme une ressemblance d'un objet produite sur un support photographique ou une image produite sur un affichage électronique (tel qu'un écran de télévision ou d'ordinateur)

Une image est stockée en mémoire sous forme de points de base appelés pixels. On peut penser à une image numérique comme une page de nombres organisée dans un tableau ou une matrice de telle sorte que chaque nombre représente les caractéristiques du pixel [9].

On peut représentée la position de chaque pixel par les coordonnées sur l'axe X horizontal et l'axe Y vertical figurant dans l'image suivante.

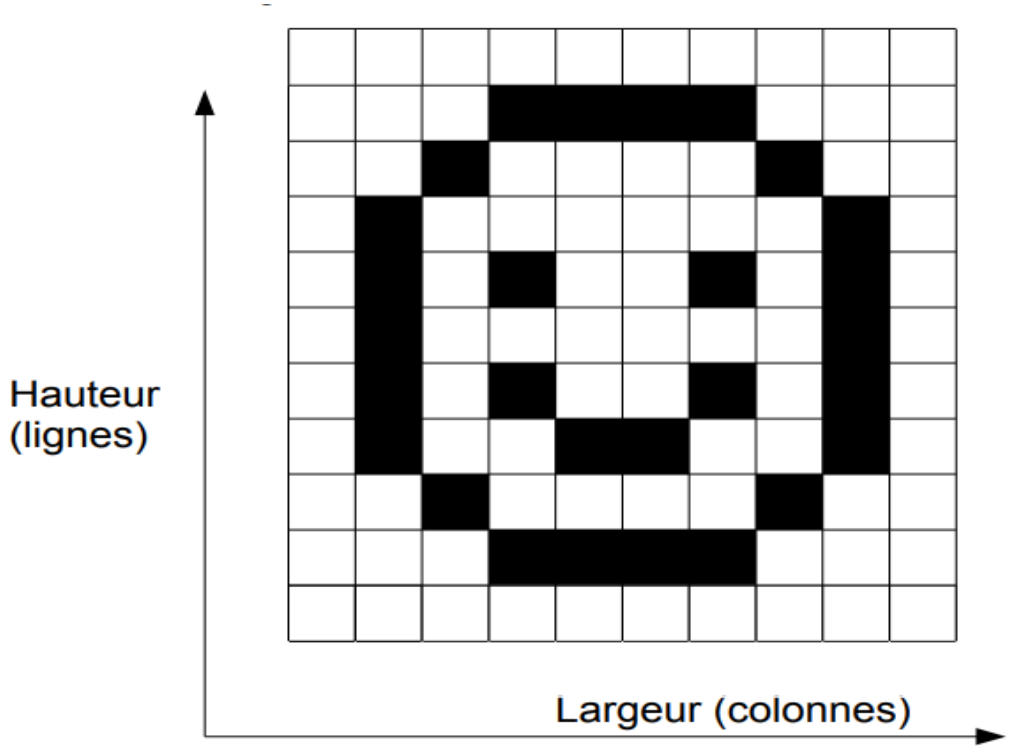


Image1 : une image sous forme de pixels

3- Les types d'image :

Les pixels d'une image dépendent de type de celle-ci qu'on peut distinguer trois types :

3-1-Les images a deux niveaux :

Une image à deux niveaux ou binaire est une image composée de pixels qui peuvent avoir que edeux couleurs, généralement le noir et le blanc. Les images binaires sont également appelées "bi-level" ou "two-level". Cela signifie que chaque pixel est stocké sous la forme d'un seul bit, c'est-à-dire un 0 ou un 1. Les noms noir et blanc, N&B, monochrome ou monochromatique sont souvent utilisés pour ce concept.

La plupart des images binaires se compriment bien avec des schémas simples de compression de longueur d'exécution[9].

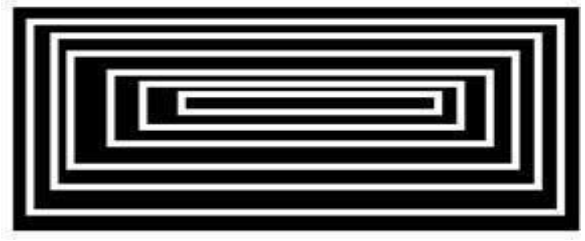


Image2: une image a deux niveaux

3-2- Les images a plusieurs niveaux de gris :

Dans ce type on peut distinguer plusieurs niveaux de gris, Il est communément appelé image en niveaux de gris (Grayscale image) ,les couleurs sont généralement codées sur huit bits c'est-à-dire qu'elle comporte 256 nuances de couleurs différentes.
La gamme des couleurs en 8 bits varie de 0 à 255. Où 0 représente le noir, 255 représente le blanc et 127 représente la couleur grise.

Ce format a été utilisé initialement par les premiers modèles des systèmes d'exploitation UNIX et les premiers Macintosh couleur[9],une image en niveaux de gris d'Einstein est présentée ci-dessous :

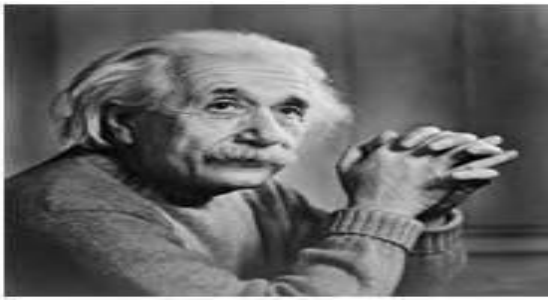


Image3: une image a plusieurs niveaux de gris d'Einstein

3-3 : Les images couleurs :

La spécification fournit une méthode de stockage et de représentation des informations d'image utilisant l'espace de couleur RVB de telle sorte qu'un très grand nombre de nuances, de couleurs et de teintes peuvent être définies dans une image, ce qui donne des images et des graphiques d'une qualité et d'une complexité très élevées.

En général, nous disposons de huit bits pour coder une composante, soit 24 bits bits pour l'espace couleur RVB, ce qui donne 16 777 216 variations de couleur

pour coder la valeur d'un pixel. la couleur peut être codée, soit par composition de couleurs primaires, soit par composition d'informations de luminance et de chrominance [10].

-une couleur peut être représentée par un ensemble de trois coordonnées, c'est-àdire elle peut être reproduite par la superposition de trois couleurs primaires comme la montre la figure suivante :

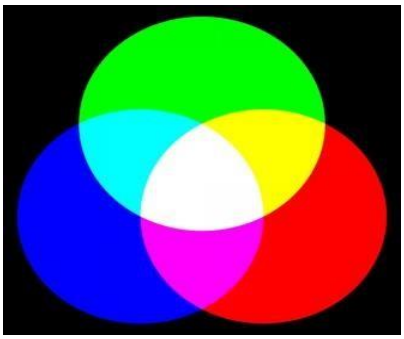


Figure 04 :Superposition des trois couleurs : rouge, vert et bleu (RGB)

4- Les caractéristiques d'une image :

4-1-Le pixel :

En imagerie numérique, un pixel (ou élément d'image) est le plus petit élément d'information d'une image. Les pixels sont disposés dans une grille bidimensionnelle, représentée par des carrés. Chaque pixel est un échantillon d'une image originale, où plus d'échantillons fournissent généralement des représentations plus précises de l'original [17].

4-2-La définition :

La définition de l'image est le nombre de pixels qui la composent. Elle est donnée en indiquant le nombre de pixels sur une ligne, et le nombre sur une colonne[17].

4-3-La résolution :

La résolution d'une image est le nombre de pixels par pouce qu'elle contient (1 pouce = 2.54 centimètres). Elle est exprimée en "PPP" (points par pouce) ou DPI (dots per inch). Plus il y a de pixels (ou points) par pouce et plus il y aura d'information dans l'image (plus précise). Par exemple, une résolution de 300dpi signifie que l'image

comporte 300 pixels dans sa largeur et 300 pixels dans sa hauteur, elle est donc composée de 90 000 pixels (300x300 ppp). Grâce à cette formule, il est facile de connaître la dimension maximale d'un tirage.

Il est généralement admis qu'une résolution de 300 ppp pour une image est largement suffisante avant impression. Cette résolution peut-être revue à la baisse dans le cas d'impressions devant être visualisées à une distance plus ou moins éloignée de l'observateur (par conséquent liée au pouvoir séparateur de l'oeil humain).

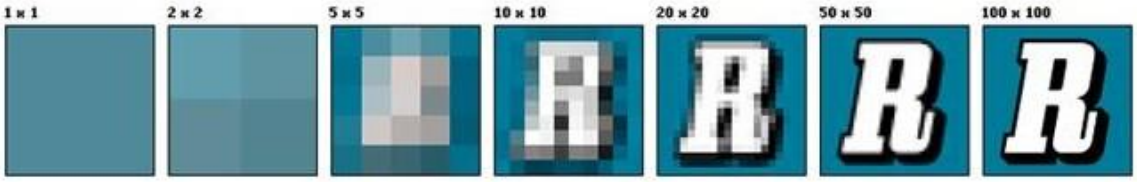


Figure 05 :différentes résolution d'une image

4-4-La profondeur d'une image :

La profondeur d'une image est la quantité d'informations de couleur disponibles par pixel. L'image a une profondeur de 1 bit par pixel, les pixels prennent deux valeurs possibles: noir et blanc.
- Les images d'une profondeur de 8 bits par pixel calculeront 256 valeurs possibles. Une image en mode Niveaux de gris avec une profondeur de 8 bits par pixel a 256 valeurs de gris possibles [16].

4-5-La taille d'une image :

la taille d'une image non compressée est calculée en multipliant la définition par le nombre d'octets par pixel [18].

4-6-La fidélité :

La fidélité se réfère à la notion de conservation de l'information. Il s'agit de mesurer la similarité entre un stimulus original et une version à évaluer [18].

4-7-La qualité d'image :

Evaluer la qualité d'une image c'est lui associer un ou plusieurs qualificatifs permettant de situer sa position relative dans un référentiel défini par notre sens et selon l'application envisagée [18].

5- Quelques distorsions des images :

Nous présentons en premier lieu quelques distorsions connues pour ensuite introduire les notions liées à la qualité d'image.

5-1-Effet de bloc : Les effets de bloc font référence a l'apparition de bloc dans l'image. Cette dégradation est due à une quantification indépendante des blocs conduisant ainsi a des discontinuités au voisinage de ces blocs.

Figure 06 :image qui a un effet de bloc

5-2-Effet mosaïque :

L'origine de cette dégradation est la perte partielle ou totale du contenu d'un bloc de l'image due a la compression bas débit par exemple, ou encore au faible débit du canal de transmission. Cet effet de mosaïque est plus visible dans les zones texturées

Figure 07 :image a un effet mosaïque

5-3-Effet d'escalier :

Due a l'approximation des contours diagonaux par des contours horizontaux et verticaux, cette dégradation se traduit visuellement par la formation d'escaliers.

Figure 08 :image a effet d'escalier

5-4-Le flou :

Cet artefact se manifeste par une perte de finesse et de visibilité des détails résultant d'une diminution du contraste, comme illustrée par la figure

Figure 09 :exemple d'une image flou

5-5 -Faux contours :

Cette distorsion fait émerger de faux contours dans l'image. Sa visibilité est amplifiée dans les zones homogènes

Figure 10 :image a faux contours

6- La compression :

Malgré l'augmentation de la taille des medias, qu'ils soient disques durs internes au PC ou externes, CD/DVD, ou mémoires USB, on cherche toujours à réduire la taille des fichiers. Ceci est encore plus utile pour envoyer des fichiers à un correspondant, que ce soient des photos, des scans ou des saisies d'écran.

Lorsqu'un fichier est comprimé, il occupe moins d'espace disque qu'une version non comprimée et peut être transféré plus rapidement vers d'autres systèmes. La compression est donc souvent utilisée pour économiser de l'espace disque et réduire le temps nécessaire au transfert de fichiers sur Internet[10].

6-1-Classification des algorithmes de compression :

Dans le domaine de la compression, il existe plusieurs façons de comparer les types de compression. Pour cette raison, nous allons voir comment classifier les algorithmes de compression.

6-1-1-Compression symétrique / asymétrique :

6-1-1-1-compression symétrique :

Technique de compression de données qui prend à peu près le même temps pour compresser que pour décompresser[11].

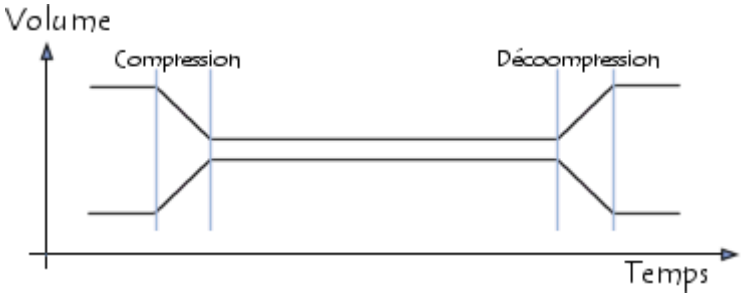


Image 11- schéma de la compression symétrique :

Ce type de compression est le type utilisé dans la transmission de données.

6-1-1-2-compression asymétrique :

Technique de compression des données qui prend généralement plus de temps à compresser qu'à décompresser. Certaines méthodes de compression asymétrique sont plus longues à décompresser, ce qui convient aux fichiers de sauvegarde qui sont constamment compressés et rarement décompressés [11].

6-1-2-Compression physique et logique :

6-1-2-1-Compression physique :

Physique Effectué sur des données indépendamment de l'information qu'elles contiennent n Traduit une série de bits en une autre série de bits n ou bien regarder les données redondantes d'un train de bits à un autre [11].

6-1-2-2-Compression logique :

C'est le fait de remplacer un symbole alphabétique, numérique ou binaire en un autre donc il est Basé sur la connaissance en substituant une information par une information équivalente. on Change par exemple le Royaume-Uni en UK [11] .

6-1-3-Compression avec perte et sans perte :

6-1-3-1-Compression sans perte :

Les techniques de compression sans perte, comme leur nom l'indique, n'impliquent aucune perte d'information. Si les données ont été compressées sans perte, les données originales peuvent être récupérées exactement à partir des données compressées après un cycle de compression/expansion. La compression sans perte est généralement utilisée pour les données dites "discrètes", telles que les enregistrements de bases de données, les feuilles de calcul, les fichiers de traitement de texte, et même certains types d'images et d'informations vidéo[14].

La compression de texte est un domaine important pour la compression sans perte. Il est très important que la reconstruction soit identique au texte original, car de très petites différences peuvent donner lieu à des déclarations ayant des significations très différentes.

Image 12- Comparaison de la compression sans perte

6-1-3-1-1-Méthodes de compression sans perte :

Ces algorithmes sont utilisés pour compresser tous types de données : des données textuelles, des images, du son, des programmes, etc.

On peut citer les trois méthodes suivantes :

- Codage de Huffman
- Codage Lempel-Ziv et ses variantes qui est un type de codage par dictionnaire.
- Le Codage par plage (Run length Encoding)

Un Texte d'exemple pour la compression sans perte (codage par dictionnaire):

La compression des données suit conceptuellement trois principes :

- Trouver des motifs répétitifs dans un fichier.
- Remplacer ces motifs par une référence à une entrée de dictionnaire.
- Créer un dictionnaire des motifs répétitifs.

6-1-3-2-Compression avec perte- Lossy compression- (irréversible)(destructrice) :

Ce terme "avec perte" signifie que le fichier compréssé subit des altérations qui peuvent etre minimes mais irréversibles qu'on ne pourra plus retrouver son état original par la suite[14].

La compression avec perte réduit la quantité de données nécessaires pour stocker un fichier. Cependant, la compression avec perte entraîne également une certaine perte de qualité. Les images JPEG et les fichiers audio MP3 utilisent tous deux la compression avec perte. Malgré une perte de qualité, les utilisateurs sont souvent incapables de déterminer une différence majeure[15].

Lorsqu'une image est capturée à l'aide d'un appareil photo numérique, elle est compressée et enregistrée au format JPEG. La compression des images au format JPEG peut réduire la taille des fichiers de plus de 80 %, sans que l'on puisse constater de changement notable. De même, un fichier MP3 compressé peut avoir une taille dix fois inférieure à celle du fichier audio original et avoir un son presque identique.

Pour une compression avec pertes, ne jamais travailler sur un fichier original, toujours sur une copie.

Image13 :Comparaison de la compression avec perte

6-1-3-2-1-Exemple d'un algorithme de compression avec perte (L'algorithme JPEG) :

Le format de fichier JPG a été l'une des avancées technologiques les plus impressionnantes en matière de compression d'image à voir le jour en 1992. Depuis lors, il a été une force dominante dans la représentation d'images de qualité photo sur l'internet. Et ce pour une bonne raison. Une grande partie de la technologie qui sous-tend le fonctionnement du JPG est exceptionnellement complexe et nécessite une bonne compréhension de la façon dont l'œil humain s'adapte à la perception des couleurs et des bords [15].

Les JPGs sont considérés comme un format de fichier "avec perte", ce qui signifie que pendant la procédure de compression, les blocs redondants sont définitivement supprimés. Plus vous comprimez un fichier, plus vous perdez de données et, par conséquent, plus votre image finale aura l'air mauvaise après plusieurs itérations de l'algorithme.

6-1-3-2-1-1-Processus de compression décompression du JPEG:

Le schéma de compression JPG est décomposé en plusieurs phases. L'image ci-dessous les décrit à un haut niveau, et nous allons parcourir chaque phase ci-dessous.

Le processus de compression et de décompression JPEG irréversibles comporte six étapes principales représentées :

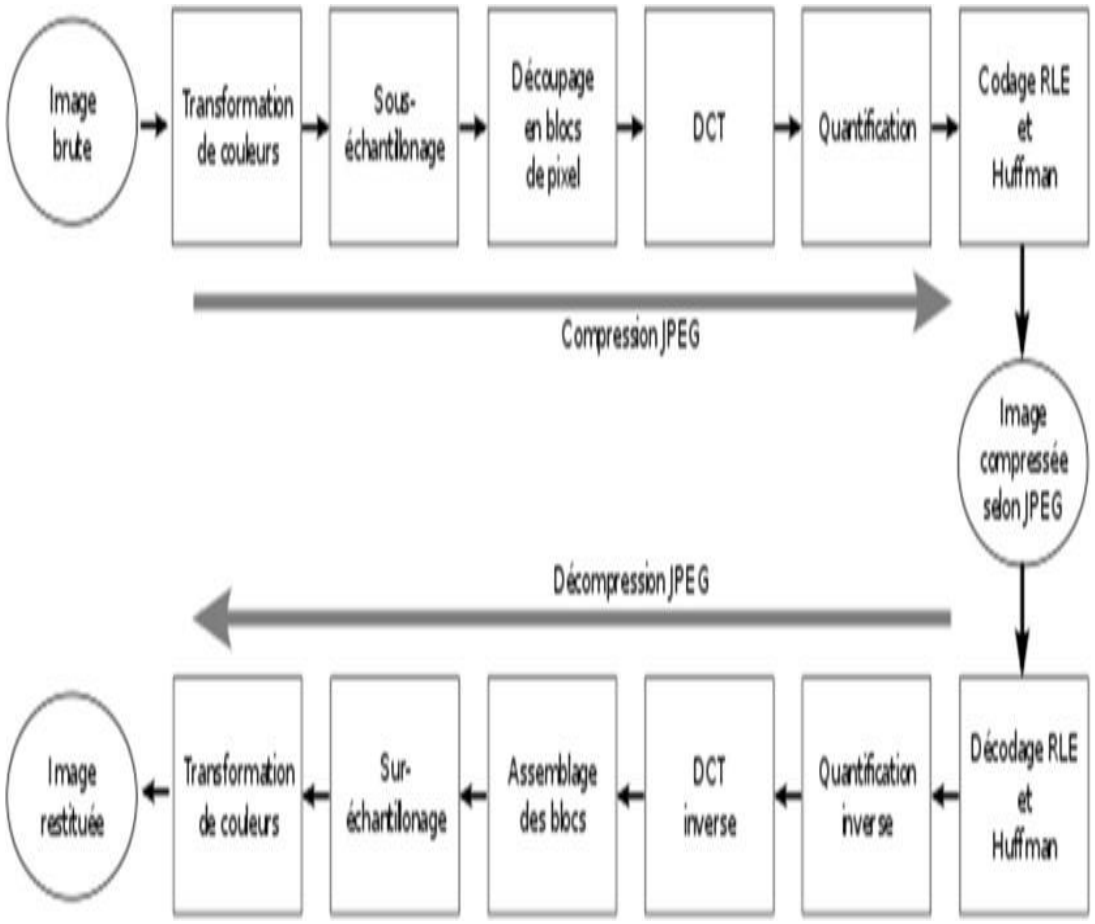


Image 14 : schéma de fonctionnement d'une compression JPEG

Étape 1 : Convertir l'espace couleur RGB en YCbCr

Chaque pixel de votre image est stocké sous la forme d'une combinaison additive de valeurs rouge, bleue et verte. Chacune de ces valeurs peut être comprise entre 0 et 255. Ce modèle de couleur est appelé le modèle RVB. Considérons un pixel de couleur kaki. Il sera stocké sous la forme (240, 230, 140).

Nous convertissons donc l'espace couleur RVB en un espace où la luminance est confinée à un seul canal. Cet espace couleur est appelé YCbCr [15].

Ici, Y est la composante de luminance et Cb, Cr sont les composantes de chrominance. Elles représentent respectivement les différences de bleu et de rouge.

Leurs valeurs sont comprises entre 0 et 255.

Les valeurs YCbCr peuvent être calculées directement à partir de RGB comme suit : [15]

Y = 0,299 R + 0,587 G + 0,114 B .....(1)

Cb = - 0,1687 R - 0,3313 G + 0,5 B + 128.....(2)

Cr = 0,5 R - 0,4187 G - 0,0813 B + 128.....(3)

Étape 2 : L'échantillonnage (Sampling) :

Dans cette étape on va Diviser l'image en groupes de 8 × 8 pixels.

Chaque groupe est initialement représenté par **64 octets**.
Après transformation, chaque groupe est représenté par **2 à 20 octets** (par ex.) [15].

Étape 3: La transformation DCT (Discrete Cosinus Transform) :

Cette étape Permet de changer le domaine d'étude (Spatial vers Fréquentiel) de telle sorte que la Formule pour calculer la DCT sur une matrice NxN est la suivante [15].:

$$\mathbf{DCT(i,j)=}\frac{1}{\sqrt{2}}\mathbf{C(i)\ C(j)}\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\boldsymbol{pixel}(x,y)\cos\left(\frac{(2x+1)i\pi}{2N}\right)\cos\left(\frac{(2y+1)j}{2N}\pi\right) \quad \text{.....(3)}$$

$$\mathbf{C(x)=}\frac{1}{\sqrt{2}} \quad \textbf{si x vaut 0, et 1 si x>0 .....(4)}$$

La formule pour calculer la IDCT (Inverse DCT) sur une matrice NxN **:**

$$\mathbf{pixel(x,y)=}\frac{\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}\mathbf{C(i)}\ \mathbf{C(j)}\boldsymbol{DCT}(i,j)\cos\left(\frac{(2x+1)i\pi}{2N}\right)\cos\left(\frac{(2y+1)j}{2N}\pi\right)}{\sqrt{2}}$$

$$\mathbf{C(x)=}\frac{1}{\sqrt{2}} \quad \textbf{si x vaut 0, et 1 si x>0.....(5)}$$

Étape 4: La Quantification :

Cette étape a pour but de diminuer la précision du stockage des entiers de la matrice DCT pour diminuer le nombre de bits occupés par chaque entier.
de telle sorte que le processus de Quantification est le principal responsable de la dégradation de l'image, L'utilisateur choisit au début la qualité de la compression.
Les développeurs du format JPEG ont estimé qu'il ne fallait pas dépasser un facteur de 25, après quoi l'image était vraiment trop dégradée, le gain en terme de nombre de bits utilisés pour coder l'image devient négligeable [15]..

En fonction du facteur de compression, la matrice de quantification (8x8) est calculer comme suite :

Quantification[i][j]=1+(i+j+1)*qualité.....(6)

Exemple pour un facteur de compression (qualité) de 2 :

$$A = \begin{bmatrix} 3 & 5 & 7 & 9 & 11 & 13 & 15 & 17 \\ 5 & 7 & 9 & 11 & 13 & 15 & 17 & 19 \\ 7 & 9 & 11 & 13 & 15 & 17 & 19 & 21 \\ 9 & 11 & 13 & 15 & 17 & 19 & 21 & 23 \\ 11 & 13 & 15 & 17 & 19 & 21 & 23 & 25 \\ 13 & 15 & 17 & 19 & 21 & 23 & 25 & 27 \\ 15 & 17 & 19 & 21 & 23 & 25 & 27 & 29 \\ 17 & 19 & 21 & 23 & 25 & 27 & 29 & 31 \end{bmatrix}$$

Sachant que le Parcours des Coefficients Quantifiés (Lecture en zigzag)

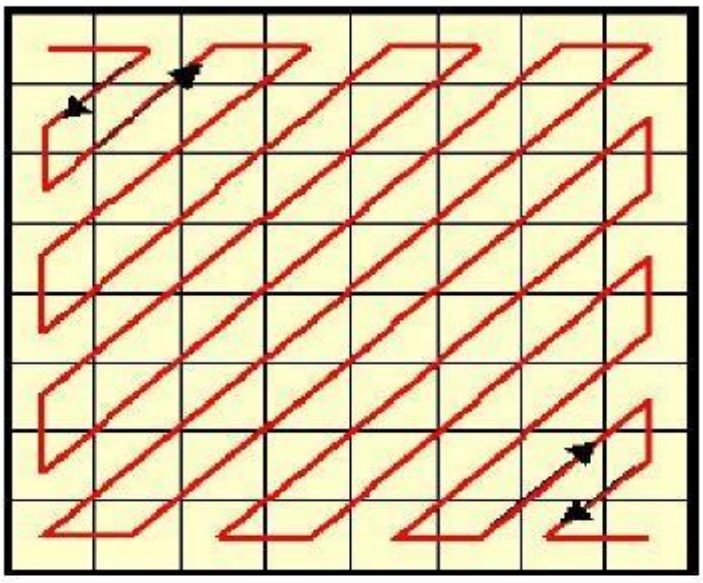


Image 15 : Parcours des Coefficients Quantifiés (Lecture en zigzag)

Etape 5 : le codage :

Un codage type Huffman ou arithmétique continue la compression des données.
Le codage de Huffman est une méthode statistique qui permet d'attribuer à un mot un code binaire aux différents symboles à compresser (pixels ou caractères par exemple).
La longueur de chaque mot de code n'est pas identique pour tous les symboles [15].
Le codage du reste de la matrice DCT quantifiée va se faire en parcourant les éléments par une séquence zigzag
Voici un exemple d'une matrice :

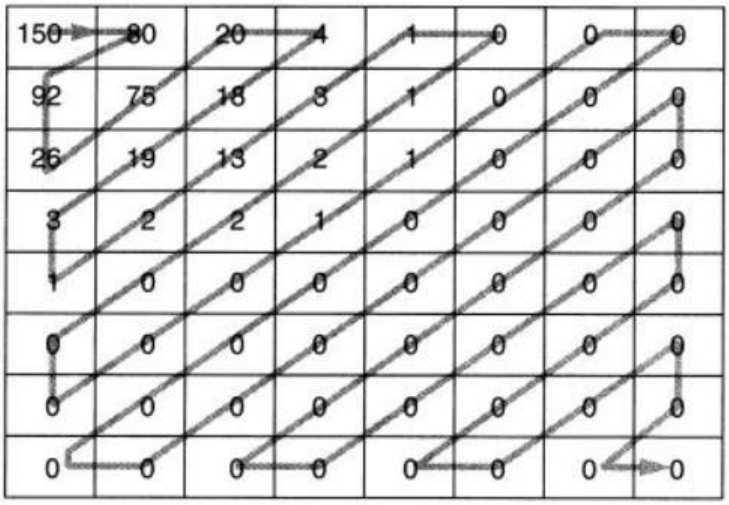
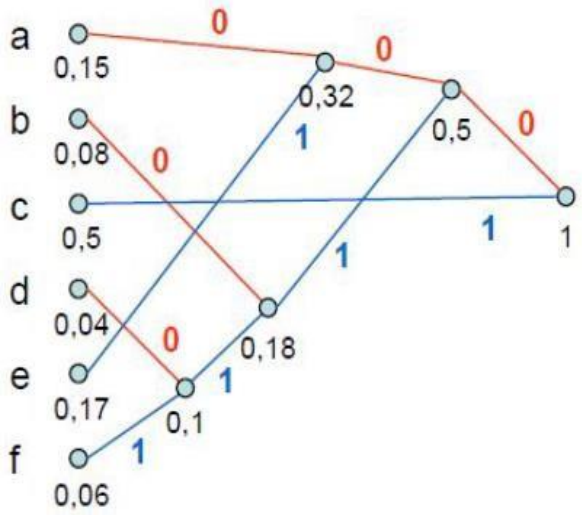


Image 16 : Exemple de (Lecture en zigzag)

Ce qui donne la suite suivante : 150, 80, 92, 26, 75, 20, 4, 18, 19, 3, 1, 2, 13, 3, 1, 0, 1, 2, 2, 0, 0, 0,0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, etc.

Les valeurs non nulles sont codées par le codage de Huffman:



Symbole	Probab.	Code
a	0,15	000
b	0,08	010
c	0,5	1
d	0,04	0110
e	0,17	001
f	0,06	0111

Image 17 : Exemple de codage de Huffman

7- Evaluation de la compression et des pertes :

On peut évaluer la compression par plusieurs métriques notons à titre d'exemple :

7-1-Quotient de la compression :

Le degré de réduction de la taille des données obtenues par une méthode de compression peut être évalué au moyen du quotient défini par la formule[15]. :

Q =(taille initiale/taille finale)

Plus une compression sera forte, plus le quotient de compression sera lui aussi élevé.

7-2-Taux de la compression :

Le taux de compression est une mesure de la performance d'un algorithme de compression de données informatiques [15]. Il est généralement exprimé en pourcentage

T=(1/Q) .....(7)

7-3-Gain de la compression :

Est tout simplement la différence entre la taille initiale et la taille compressée ou la taille finale[15].

G=1-T=(taille_initiale-taille_finale) .....(8)

7-4-l'erreur quadratique moyenne (EQM) :

L' EQM est une mesure de l'erreur moyenne, pondérée par le carré de l'erreur. Elle permet de répondre à la question, « quelle est la magnitude de l'erreur de la prévision », mais n'indique pas la direction des erreurs.

$$\mathbf{EQM}=\frac{1}{M.N}\sum_{i=1}^{M}\sum_{j=1}^{N}((\mathbf{i},\mathbf{j})-\mathbf{I}(\mathbf{i},\mathbf{j}))^2 \quad \text{.....(9)}$$

avec I(i, j) et Id(i, j) le pixel de coordonnées (i, j) de l'image originale et de sa version dégradée, respectivement. N et M représentent respectivement le nombre de lignes et de colonnes de l'image [15].

Un algorithme performant de compression possède un gain de compression maximal et une erreur quadratique moyenne minimale.

7-5-Chi 2 :

Le paramètre nommé *Chi 2* calcule la différence au carré normalisée entre l'image originale et l'image dégradée[15].

$$\mathbf{Chi2}=\frac{1}{M.N}\sum_{i=1}^{M}\sum_{j=1}^{N}\frac{((\mathbf{i},)-\mathbf{Id}(\mathbf{i},\mathbf{j}))^2}{\mathbf{I}(\mathbf{i},\mathbf{j})} \quad \text{.....(1)}$$

.....10

Un algorithme performant de compression possède un gain de compression maximal et une erreur quadratique moyenne minimale;

8- Types d'évaluation :

Il existe essentiellement deux types d'évaluation

8-1-Evaluation subjective :

L'évaluation subjective fait appel à des observateurs humains (utilisateurs finaux) pour évaluer (ou comparer) la qualité d'une image (ou de plusieurs images) selon un protocole bien défini.

Elle est considérée comme le moyen le plus fiable et donc la référence pour comparer les différentes métriques proposées [15]..

8-2-Evaluation objective :

L'évaluation objective fait référence aux méthodes basées sur l'analyse et la mesure quantitative du niveau de dégradation au moyen de métriques directement liées au signal physique.

Les métriques de qualité objectives sont alors évaluées à travers l'étude de la cohérence avec les notes subjectives[15]..

9- Qu'est-ce qu'un réseau neuronal ?

Un réseau neuronal est une série d'algorithmes qui s'efforcent de reconnaître les relations sous-jacentes dans un ensemble de données par un processus qui imite le fonctionnement du cerveau humain. En ce sens, les réseaux neuronaux font référence à des systèmes de neurones, de nature organique ou artificielle. Les réseaux neuronaux peuvent s'adapter à des entrées changeantes ; le réseau génère donc le meilleur résultat possible sans avoir à redéfinir les critères de sortie[13].

Ce sont une série d'algorithmes qui imitent les opérations du cerveau humain pour reconnaître les relations entre de grandes quantités de données.

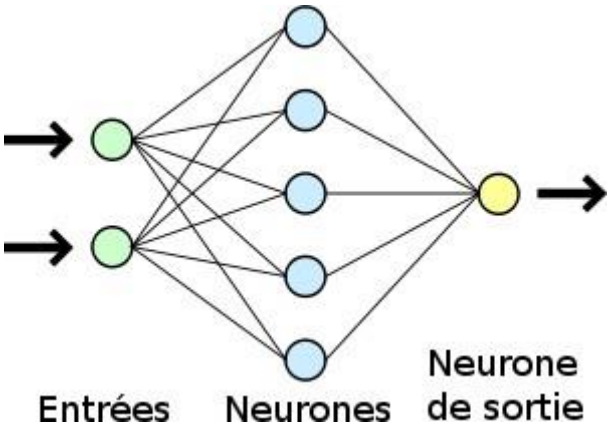


Image 18 : Représentation d'un simple réseau de neurones.

10. Le fonctionnement d'un réseau de neurones artificiels :

Un réseau neuronal artificiel est constitué d'unités de traitement appelées neurones. Un neurone reçoit des entrées multiples provenant de sources différentes, et a une seule sortie.

Un poids est la connexion au signal. Le produit du poids et de l'entrée donne la force du signal

. Les poids Wi sont des valeurs numériques qui représentent soit la valeur de puissance des entrées, soit la valeur de puissance des connexions entre les neurones. Il existe des opérations qui se passent au niveau du neurone artificiel. Le neurone artificiel fera un produit entre le poids (w) et la valeur d'entrée (x), puis ajouter un biais (b), le résultat est transmis à une fonction d'activation (f) qui ajoutera une certaine non-linéarité. **[1]**

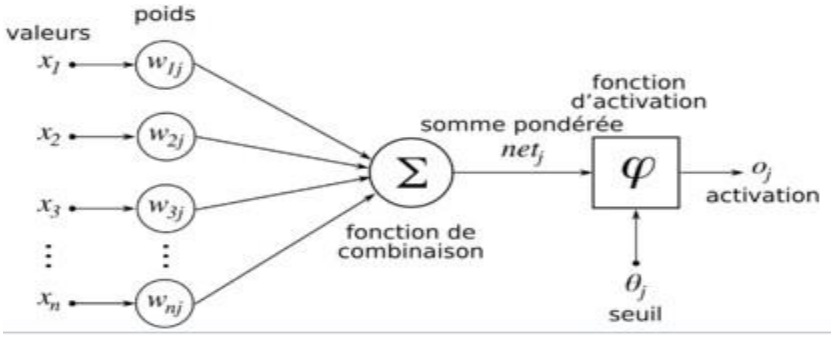


Image 19 : fonctionnement d'un réseau de neurones artificiels.

11. Les fonctions d'activation :

Une fonction d'activation est utilisée à l'introduction une non-linéarité dans le fonctionnement du neurone.Elle présente en général trois intervalles :[2]

1. en dessous du seuil, le neurone est non-actif (souvent dans ce cas, sa sortie vaut 0 ou -1) ;
2. aux alentours du seuil, une phase de transition ;
3. au-dessus du seuil, le neurone est actif (souvent dans ce cas, sa sortie vaut 1).

Il existe différentes fonctions utilisées pour l'activation. L'une des fonctions d'activation les plus couramment utilisées est la fonction sigmoïde

11-1-La fonction sigmoïde :[2]

En mathématiques, la fonction sigmoïde (dite aussi *courbe)* est définie par :

$$F(x)=\frac{1}{1+e^{-x}} \quad \text{.....(11)}$$

Sachant que Somme= $\sum_{i=1} XiWi$
La somme est la somme pondérée des entrées multipliée par les poids entre une couche et la suivante. [2].
L'interconnexion de ces neurones individuels forme le réseau neuronal.[2]

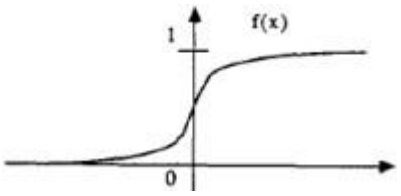


Image 20 : présentation graphique de la fonction sigmoïde.

11-2-La fonction ReLu :[2]

ReLu est une fonction d'activation non linéaire utilisée dans les réseaux neuronaux multicouches ou les réseaux neuronaux profonds. Cette fonction peut être représentée comme suit

f(x)=max(0,x) .....(12)

où x = une valeur d'entrée

Selon l'équation 1, la sortie de ReLu est la valeur maximale entre zéro et la valeur d'entrée. Une sortie est égale à zéro lorsque la valeur d'entrée est négative et à la valeur d'entrée lorsque l'entrée est positive.[2] Ainsi, nous pouvons réécrire l'équation 1 comme suit :

$$f(x)=\begin{cases} 0, si\ x \prec 0 \\ x, si\ x \geq 0 \end{cases} \quad \text{.....(13)}$$

où x = une valeur d'entrée

12. Définition du Deep learning :

Le Deep Learning est basé sur l'idée des réseaux de neurones artificiels et il est dédié pour gérer de larges quantités de données en ajoutant des couches au réseau. Un modèle de deep learning a la capacité d'extraire des caractéristiques à partir des données brutes grâce aux multiples couches de traitement composé de multiples transformations linéaires et non linéaires et apprendre sur ces caractéristiques petit à petit à travers chaque couche avec une intervention humaine minime.[3]

13. Histoire du deep learning : Avant de voir la lumière , le deep learning a progressé chronologiquement comme suit :[4]

Année	Contributeur	Contribution
1943	McCulloch and Pitts	introduction du McCulloch-Pitts (MCP) modèle considérer comme L'ancêtre des réseaux de neurones artificielles
1949	Donald Hebb	considérer comme le père des réseaux de neurones, il introduit la règle d'apprentissage de Hebb qui servira de fondation pour les réseaux de neurones modernes
1958	Frank Rosenblatt	introduction du premier perceptron
1974	Paul Werbos	introduction de la retro propagation
1980		introduction des cartes auto organisatrices
1980	Kunihiko Fukushima	introduction du Neocognitron, qui a inspiré les réseaux de neurones convoluti
1982	John Hopfield	introduction des réseaux de Hopfield
1985	Hilton and Sejnowski	introduction des machines de Boltzmann
1986	Paul Smolensky	introduction de Harmonium, qui sera connu plus tard comme machines de Boltzmann restreintes
1986	Michael I. Jordan définition et	introduction des réseaux de neurones récurrent
1990	Yann LeCun	introduction de LeNet et montra la capacités des réseaux de neurones profond
1997	Schuster and Paliwal	introduction des réseaux de neurones récurrent bidirectionnelles
1997	Hochreiter and Schmidhuber	introduction de LSTM, qui ont résolu le problème du vanishing gradient dans les réseaux de neurones récurrent
2006	Geoffrey Hinton	introduction des Deep belief Network
2009	Salakhutdinov and Hinton	introduction des Deep Boltzmann Machines
2012	Alex Krizhevsky	introduction de AlexNet qui remporta le challenge ImageNet
2014	Ian Goodfellow	inventé la technologie GAN (pour « generative adversarial networks »).

Tableau01 : Histoire du deep learning

14- Applications de l'apprentissage profond :[3]

Les domaines d'application de deep learning est très vaste on peur le trouver dans la visuelle, la détection des fraudes, les soins de santé, la détection des retards de développement chez les enfants, l'ajout de son aux films muets, la traduction automatique, la traduction de texte en image, la synthèse d'image en image, la reconnaissance automatique d'image, la colorisation, prédiction des tremblements de terre, prévision des taux de marché, agrégation de nouvelles et la détection de nouvelles frauduleuses.

15- Approches d'apprentissage profond

Les réseaux neuronaux profonds sont performants dans l'apprentissage supervisé, l'apprentissage non supervisé, l'apprentissage par renforcement, ainsi que l'apprentissage hybride.

15-1 Apprentissage supervisé

Dans l'apprentissage supervisé, les variables d'entrée représentées par X sont mises en correspondance avec les variables de sortie représentées par Y en utilisant un algorithme pour apprendre la fonction de mise en correspondance f.

fonction f.

$$Y = f(X) .$$

L'objectif de l'algorithme d'apprentissage est d'approcher la fonction de correspondance pour

prédire la sortie (Y) pour une nouvelle entrée (X). L'erreur provenant des prédictions effectuées

pendant l'apprentissage peut être utilisée pour corriger la sortie. L'apprentissage peut être arrêté lorsque

l'apprentissage peut être arrêté lorsque toutes les entrées sont entraînées pour obtenir la sortie ciblée **[8].**

15-2 Apprentissage non supervisé :[8]

Dans l'apprentissage non supervisé, nous avons uniquement les données d'entrée et aucune sortie correspondante à cartographier. Cet apprentissage vise à apprendre des données en modélisant la distribution des données.

Les algorithmes peuvent être capables de découvrir la structure passionnante présente dans les données. Les problèmes de regroupement et d'association utilisent l'apprentissage non supervisé

15-3 Apprentissage par renforcement :

L'apprentissage par renforcement utilise un système de récompense et de punition pour entraîner l'algorithme.Dans ce cas, l'algorithme ou un agent apprend de son environnement.

L'agent est récompensé en cas de performance correcte et pénalisé en cas de performance incorrecte. Par exemple, si l'on prend le cas d'une voiture autonome, l'agent reçoit unerécompense pour avoir conduit en toute sécurité jusqu'à sa destination et une pénalité pour avoir fait du hors-piste. De même, dans le cas d'un programme pour jouer aux échecs, l'état de récompense peut être de gagner la partie et la pénalité d'être échec.

L'agent essaie de maximiser la récompense et de minimiser la pénalité. Dans l'apprentissage par renforcement, on ne dit pas à l'algorithme comment effectuer l'apprentissage.

l'algorithme n'est pas informé de la manière d'effectuer l'apprentissage, mais il résout le problème par lui-même [2].

16- Architectures fondamentales d'apprentissage profond :

Les architectures d'apprentissage profond sont plus performantes que les ANN simples, même si le temps de formation des structures profondes est plus élevé que celui des ANN. Cependant, le temps de formation peut être réduit en utilisant des méthodes telles que l'apprentissage par transfert et le calcul par le GPU. L'un des facteurs qui déterminent le succès des réseaux neuronaux est la conception minutieuse de l'architecture du réseau.

Certaines des architectures d'apprentissage profond pertinentes sont examinées ci-dessous.

16-1-Réseaux adversariaux génératifs :

Ils comprennent un réseau générateur et un réseau discriminateur. Le générateur génère le contenu tandis que le discriminateur valide le contenu généré. Le générateur crée des images d'apparence naturelle, tandis que le discriminateur décide si l'image semble naturelle. Les GAN utilisent des réseaux neuronaux convolutionnels et réseaux de neurones [4].

16-2 Réseaux neuronaux convolutifs :

Les réseaux de neurones convolutifs (CNN) sont principalement utilisés pour les images. Ils attribuent

des poids et des biais aux différents objets de l'image et les différencie les uns des autres.

Il nécessite moins de prétraitement par rapport aux autres algorithmes de classification.

CNN utilise des filtres pertinents pour capturer les dépendances spatiales et temporelles dans une image [4,5].

C'est un modèle particulier qui a beaucoup contribué au domaine de la vision par ordinateur et de l'analyse d'images, à savoir les réseaux neuronaux convolutifs (CNN) ou ConvNets.

Leurs applications vont de la reconnaissance d'images et de vidéos à la classification d'images, en passant par l'analyse d'images médicales, la vision par ordinateur et le traitement du langage naturel.

16-2-1-Définition de convolution :

En termes simples, deux images qui peuvent être représentées comme des matrices sont multipliées pour donner une sortie qui est utilisée pour extraire des caractéristiques de l'image [19].

16-2-2Architecture de base :

L'architecture d'un CNN comprend deux parties principales :

-Un outil de convolution qui sépare et identifie les différentes caractéristiques de l'image pour les analyser dans un processus appelé extraction de caractéristiques [19]..

-Une couche entièrement connectée qui utilise la sortie du processus de convolution et prédit la classe de l'image en fonction des caractéristiques extraites lors des étapes précédentes[19]..

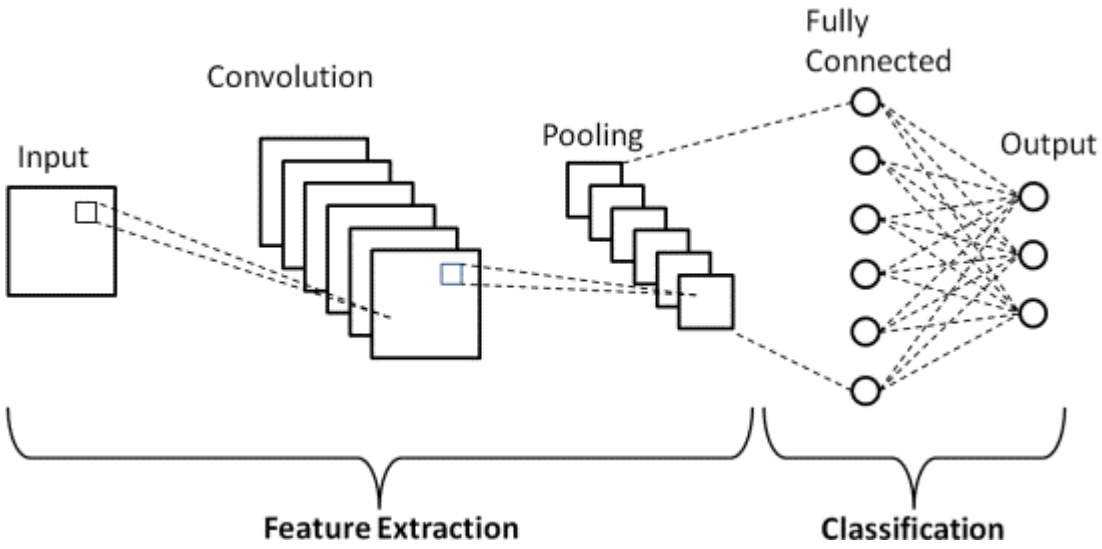


Image 21 : Architecture de base du CNN.

16-2-3-Couches de convolution :

Trois types de couches composent le CNN : les couches de convolution, les couches de mise en commun (pooling layers) et les couches entièrement connectées (Fully Connected). Lorsque ces couches sont empilées, une architecture CNN est formée.

16-2-3-1-Couche convolutionnelle :

Cette couche est la première couche qui est utilisée pour extraire les différentes caractéristiques des images d'entrée. Dans cette couche, l'opération mathématique de convolution est effectuée entre l'image d'entrée et un

filtre d'une taille particulière MxM. En faisant glisser le filtre sur l'image d'entrée, le produit scalaire est pris entre le filtre et les parties de l'image d'entrée par rapport à la taille du filtre (MxM).

La sortie est appelée carte de caractéristiques qui nous donne des informations sur l'image telles que les coins et les bords[19].

16-2-3-2-Couche de mise en commun(pooling layer) :

La couche de mise en commun sert généralement de pont entre la couche convolutive et la couche FC.

Dans la plupart des cas, une couche convolutive est suivie d'une couche de mise en commun. L'objectif principal de cette couche est de diminuer la taille de la carte de caractéristiques convolutionnelle afin de réduire les coûts de calcul. Ceci est réalisé en diminuant les connexions entre les couches et en opérant indépendamment sur chaque carte de caractéristiques. Selon la méthode utilisée, il existe plusieurs types d'opérations de mise en commun (Max pooling, Average pooling and Sum pooling) [19].

16-2-3-3-Couche entièrement connectée:

La couche entièrement connectée (FC) se compose des poids et des biais avec les neurones et est utilisée pour connecter les neurones entre deux couches différentes. Ces couches sont généralement placées avant la couche de sortie et forment les dernières couches d'une architecture CNN.

Dans cette couche, l'image d'entrée des couches précédentes est aplatie et transmise à la couche FC. Le vecteur aplati passe ensuite par quelques couches FC supplémentaires où les opérations de fonctions mathématiques ont généralement lieu. À ce stade, le processus de classification commence à avoir lieu [19]..

16-2-3-4-Fonctions d'activation :

Elle décide quelles informations du modèle doivent être tirées dans la direction avant et lesquelles ne doivent pas l'être à la fin du réseau elle applique une transformation non linéaire sur des données d'entrée pondérées

Elle ajoute une non-linéarité au réseau. Il existe plusieurs fonctions d'activation couramment utilisées, telles que les fonctions ReLU, Softmax, tanH et Sigmoid. Chacune de ces fonctions a un usage spécifique.

Les différentes architectures CNN comprennent LeNet, AlexNet, VGGNet, GoogleNet, ResNet, ZFNet. Les CNN sont principalement utilisés dans des applications telles que détection d'objets, la segmentation sémantique, le sous-titrage [19]..

16-3 Réseaux neuronaux récurrents :[5]

Dans les réseaux neuronaux récurrents (RNN), les sorties des états précédents servent d'entrée à l'état actuel. Les couches cachées des RNN peuvent mémoriser des informations.

L'état caché est mis à jour en fonction de la sortie générée dans l'état précédent.

Le RNN peut être utilisé pour la prédiction de séries temporelles car il peut se souvenir des entrées précédentes. ce qui est appelé mémoire à long et court terme [5].

17- La compression générative :

Les algorithmes traditionnels de compression d'images et de vidéos reposent sur des paires codeur/décodeur (codecs) fabriquées à la main, qui manquent d'adaptabilité et sont agnostiques par rapport aux données à compresser. La compression générative, c'est-à-dire la compression de données à l'aide de modèles génératifs,est une direction à suivre pour produire des reconstructions plus précises et visuellement agréables à des niveaux de compression [6].

La tendance consiste à remplacer les transformées linéaires fabriquées à la main par des réseaux neuronaux artificiels.

18- Les frameworks de Deep learning :

Les frameworks de Deep learning permettent de modéliser un réseau plus rapidement sans entrer dans les détails des algorithmes sous-jacents, chaque framework est construit différemment pour des objectifs différents. Certains frameworks de Deep learning sont présentés ci-dessous et sont résumés dans le tableau suivant.

TensorFlow :

développé par Google brain, supporte des langages tels que Python, C++ et R, il nous permet de déployer nos modèles d'apprentissage profond dans les CPU et les GPU [7] ainsi que dans les GPU [7].

Keras :

Est une API, écrite en Python et exécutée au-dessus de TensorFlow. Elle permet une expérimentation rapide. Elle prend en charge les CNN et les RNN et fonctionne sur les CPU et les GPU [7].

PyTorch :

peut être utilisé pour construire des réseaux neuronaux profonds ainsi que pour exécuter des calculs tensoriels, il fournit un framwork pour créer des graphes de calcul [7].

Caffe :

Caffe se distingue des autres frameworks par sa vitesse de traitement ainsi que par l'apprentissage à partir d'images, Le framwork de Model Zoo de Caffe nous facilite l'accès à des modèles pré-entraînés,

qui nous permettent de résoudre divers problèmes sans effort [7].

Deep Learning Framework	Année de première production	écrit en langage:	Supporte CUDA	Modèles pré-entraînés
TensorFlow	2015	C++, Python	Oui	Oui
Keras	2015	Python	Oui	Oui
PyTorch	2016	Python	Oui	Oui
Caffe	2013	C++	Oui	Oui

Tableau02: Date de production de quelques frameworks du Deep Learning

Références:

[1] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167, 2015.

[2] Palash Goyal, Sumit Pandey, and Karan Jain. Introduction to natural language processing and deep learning. In Deep Learning for Natural Language Processing, pages 1–74. Springer, 2018. doi: 10.1007/978-1-4842-3685-7-1.

[3] Dumitru Erhan, Yoshua Bengio, Aaron Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? Journal of Machine Learning Research, 11(Feb):625–660, 2010.

[4] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in neural information processing systems, pages 2672–2680, 2014.

[5] Filippo Maria Bianchi, Enrico Maiorino, Michael C Kampffmeyer, Antonello Rizzi, and Robert Jenssen. An overview and comparative analysis of recurrent neural networks for short term load forecasting. arXiv preprint arXiv:1705.04378, 2017.

[6] Nathan Hubens. Deep inside: Autoencoders - towards data science, Apr 2018.

[7] Pulkit Sharma. Top 5 deep learning frameworks, their applications, and comparisons!, May 2019..

[8] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. Supervised machine learning: A review of classification techniques. Emerging artificial intelligence applications in computer engineering, 160:3–24, 2007.

Mémoires :

[9] DOUAK Fouzi. (2008). Reconstruction des images compressées en utilisant les réseaux de neurones artificiels et la DCT , thème de Magistère : Université de Batna.

[10] Sébastien Hamis,(20202) Compression de contenus visuels pour transmission mobile sur réseaux de très bas débit, thème de Doctorat : École Doctorale de l'Institut Polytechnique de Paris (ED IP Paris)

Sitographie

[11] *Xmlgraphics* [en ligne]. [consulté le 06 avril 2021 à 00 :30]. Codage et compression d'images et de vidéo Adresse de la page : http://xmlgraphics.apache.org/fop/

[12] *researchgate* [en ligne]. [consulté le 13 avril 2021 à 15 :30]. Deep_Learning_Techniques_An_Overview Adresse de la page : https://www.researchgate.net/publication/341652370_Deep_Learning_Techniques_An_Overview

[13] *researchgate* [en ligne]. [consulté le 13 avril 2021 à 16 :30] AN_INTRODUCTION_TO_ARTIFICIAL_NEURAL_NETWORK

Adresse de la page : https://www.researchgate.net/publication/319903816

[14] *kinsta* [en ligne]. [consulté le 08 avril 2021 à 00 :20]. Pourquoi et comment utiliser la compression avec perte sur vos images WordPress

Adresse de la page : https://kinsta.com/fr/blog/compression-avec-perte/

[15] *ulb* [en ligne]. [consulté le 08 avril 2021 à 19 :20]. Algorithmes de compression - Compression avec pertes : Images fractales

Adresse de la page :
https://www2.ulb.ac.be/cours/acohen/travaux_2006_infodoc/CompressionNumerique/AvecPertesFractales.htm

[16] *merriam-webster* [en ligne]. [consulté le 09 avril 2021 à 13 :15]. Definition of Image

Adresse de la page :

https://www.merriam-webster.com/dictionary/image

[17] *tutorialspoint* [en ligne]. [consulté le 06 avril 2021 à 23 :35]. Definition of Image

Adresse de la page :
https://www.tutorialspoint.com/dip/types_of_images.htm#:~:text=8%20bit%20color%20format%20is,bit%20vary%20from%200%2D255.

[18] *northcoastphoto* [en ligne]. [consulté le 07 avril 2021 à 22 :35]. Digital Image Processing Questions & Answers
Adresse de la page :

https://northcoastphoto.com/digital-explained/

[19] upgrad [en ligne]. [consulté le 09 juin 2021 à 03 :25]. basic-cnn-architecture de la page :

https://www.upgrad.com/blog/basic-cnn-architecture/

Chapitre02:
Débrouillage d'images à l'aide de CNN

1- Introduction :

Dans ce chapitre nous proposons une nouvelle approche d'apprentissage profonde qui traite le débrouillage des images dégradées, nous évaluons l'état de l'art de l'architecture de réseau neuronal convolutif CNN proposée dans les travaux connexes pour le scénario de reconstruction floue.

2- Restauration d'images :

La restauration d'images est une famille de problèmes inverses permettant d'obtenir une image de haute qualité à partir d'une image d'entrée corrompue. La corruption peut être due à l'acquisition, la transmission et la compression des images. Une image corrompue y peut être représentée par :

$$y=H(x)+n \quad [1] \quad \text{.....(14)}$$

où **x** est la version propre de y ;

H est la fonction de dégradation

et **n** est le bruit additif.

En prenant en compte différents types d'opérateurs de dégradation et de distributions de bruit, le même modèle mathématique s'applique à la plupart des problèmes d'imagerie de bas niveau tels que le débrouillage d'images et denoising et la super-résolution.

2-1-Le débruitage(Denoising) :

L'un des défis fondamentaux dans le domaine du traitement d'images et de la vision par ordinateur est le débruitage d'images, où l'objectif sous-jacent est d'estimer l'image originale en supprimant le bruit d'une version de l'image contaminée par le bruit qui peut être causé par différentes conditions intrinsèques (c'est-à-dire le capteur) et extrinsèques (c'est-à-dire l'environnement) qu'il est souvent impossible d'éviter dans des situations pratiques [2].

Image01 : exemple d'une image débruitée

2-2-La super-résolution :

C'est l'opération qui consiste à prendre une image à basse résolution comme entrée et sort l'image à haute résolution comme résultat, elle désigne le processus qui consiste à améliorer la résolution spatiale, c'est-à-dire le niveau de détail, d'une image ou d'un système d'acquisition[3].

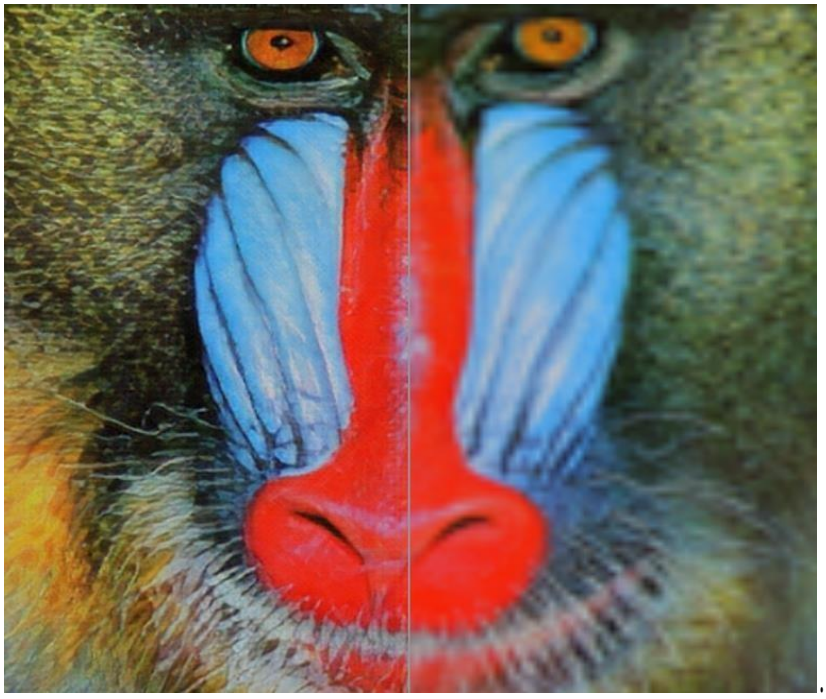


Image02: exemple d'une image subie la super-résolution

2-3-Le débrouillage(Deblurring) :

Le débrouillage est le processus d'élimination des artefacts de flou des images qui peut être causé par de nombreux facteurs [4] :

- Comme résultat de l'application des algorithmes de compression d'image.
- Un mouvement pendant le processus de capture de l'image, par l'appareil photo ou, lorsque des temps d'exposition longs sont utilisés, par le sujet.
- Distorsion de la lumière diffusée et turbulences atmosphériques.

Image03: exemple d'une image débrouillée

Notre travail va se focaliser seulement sur la restauration des images brouillées qu'on va le voir à partir des travaux connexes.

3- Les travaux connexes :

3-1-Dans un article intitulé **Deep Convolutional Neural Network for Image Deconvolution [1]** Les rechercheurs Li Xu de (Lenovo Research & Technology), Jimmy SJ. Ren de (Lenovo Research & Technology), Ce Liu de (Microsoft Research) et Jiaya Jia de (The Chinese University of Hong Kong) ont développé une solution pour le débrouillage qui consiste à établir la connexion entre les schémas traditionnels basés sur l'optimisation et une architecture de réseau neuronal où une structure nouvelle et séparable est introduite comme un support fiable pour une déconvolution robuste contre les artefacts. Notre réseau contient deux sous modules, tous deux formés de manière supervisée avec une initialisation appropriée. Ils donnent performance décente déconvolution d'images non aveugles par rapport aux modèles génératifs précédents. ratif basé sur un modèle de modèles génératifs.

Le réseau complet est formé par la concaténation du module CNN à déconvolution avec un module CNN de débruitage . Le module CNN de débruitage possède deux couches cachées avec 512 cartes de caractéristiques. On fait la convolution de l'image d'entrée avec 512 noyaux de taille 16×16 à introduire dans la couche cachée.

Dans ce système, les deux modules de réseau sont concaténés en combinant la dernière couche de la couche de déconvolution en combinant la dernière couche du CNN de déconvolution avec l'entrée du CNN de débruitage.
Comme indiqué dans l'image suivante :

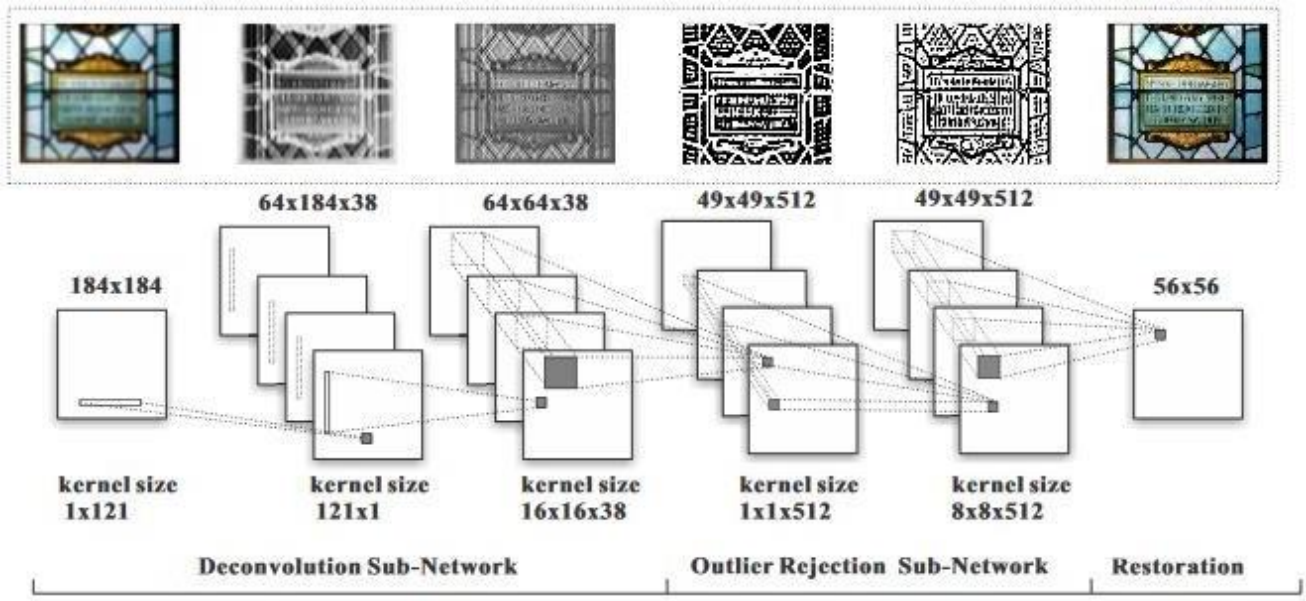


Image04: l' architecture de réseau complète pour la déconvolution profonde

3-2- dans un article intitulé **Blind Deconvolution Using a Normalized Sparsity Measure[2]** Les rechercheurs Dilip Krishnan de (Courant InstituteNew York University), Jimmy SJ. Ren de (Lenovo Research & Technology), Terence Tayde (Chatham Digital) et Rob Fergusde (Courant InstituteNew York University) ont développé une approche qui suppose le modèle de formation d'une image nette u floue par une matrice K ainsi que l'ajout d'un bruit gaussien bruit i.i.d. gaussien **N** : **g = Ku + N**……(15)

Nous observons l'image floue résultante g et notre objectif est de récupérer l'image nette inconnue u et la matrice de flou **K** L'algorithme 1 décrit son approche

Algorithme global

Demande : Image floue observée g, taille maximale du noyau h.

Appliquer des filtres dérivés à g, créant une image de haute fréquence y.

1. Estimation aveugle de la matrice de flou K à partir de y.

Boucle sur les niveaux grossiers à fins :

Alternance :

- Mettre à jour l'image haute fréquence nette x en utilisant la régularisation l1/l2.

- Mettre à jour la matrice de flou K.

-Interpoler la solution au niveau le plus fin comme initialisation.

2. récupération d'image en utilisant l'algorithme non aveugle de .

- Déblouir g en utilisant K pour donner une image nette u.

Retourner l'image nette u.

3-3-Dans un article intitulé **Understanding Blind Deconvolution Algorithms [3]** Les rechercheurs Anat Levin, Yair Weiss, Fredo Durand, and William T. Freeman, Fellow ont développé une solution pour le débrouillage qui analyse les principaux éléments constitutifs des récents algorithmes de déconvolution aveugle. Une classe de solutions implique la détection explicite des bords.
Ils ont collecté des données de flou avec la vérité du terrain et comparé quantitativement les algorithmes existants. Leur comparaison suggère que l'approximation variationnelle de Bayes surpasse de manière significative toutes les alternatives existantes.
Les conclusions de leur analyse sont utiles pour orienter les futures recherches sur la déconvolution aveugle.
Bien qu'il soit possible que la déconvolution aveugle puisse bénéficier de recherches futures sur les statistiques des images naturelles, cet article suggère que de meilleurs estimateurs pour les prieurs existants pourraient avoir plus d'impact sur les futurs algorithmes de déconvolution aveugle. De plus, ils ont observé que l'hypothèse populaire de flou uniforme dans l'espace est généralement irréaliste. Ainsi, il semble que les modèles de flou qui peuvent relâcher cette hypothèse ont un fort potentiel pour améliorer les résultats de la déconvolution aveugle.

3-4-Muhammad Asim de l'université de [Information Technology University Lahore] dans son travail [**CNN-For-End-to-End-Deblurring (Keras)**], non documenté et dépourvu de détails a proposé une structure de 14 blocs pour faire le débrouillage des images du Dataset [CelebA] qui a donné des bons résultats pour la restauration d'images brouillées.

4- Système proposé :

Le débrouillage consiste à récupérer une image nette et claire à partir d'une image d'entrée corrompue et déformée.

L'application du débrouillage se fait dans les domaines de l'astronomie et de l'imagerie médicale. Dans un scénario de la vie réelle, nous devons trouver la version nette et claire d'une image floue donnée. Il peut y avoir différentes façons d'atteindre l'objectif d'éliminer le flou de motricité d'une image dégradée.

Dans ce travail, cet objectif est atteint en utilisant la capacité d'apprentissage adaptatif d'un CNN.

Le système proposé est illustré sur la figure suivante :

Image05 : L'architecture globale de notre système

C'est-à-dire que l'image floue peut être donnée comme entrée au CNN et nous obtiendrons l'image claire comme résultat.

4-1-La structure utilisée :

Pour le débrouillage des images on a utilisé une structure basée sur Le réseau neuronal convolutif composé de sept blocs de couches, dont des couches convolutives, des couches de normalisation et des couches d'activation Relu comme indiqué dans l'image suivvante :

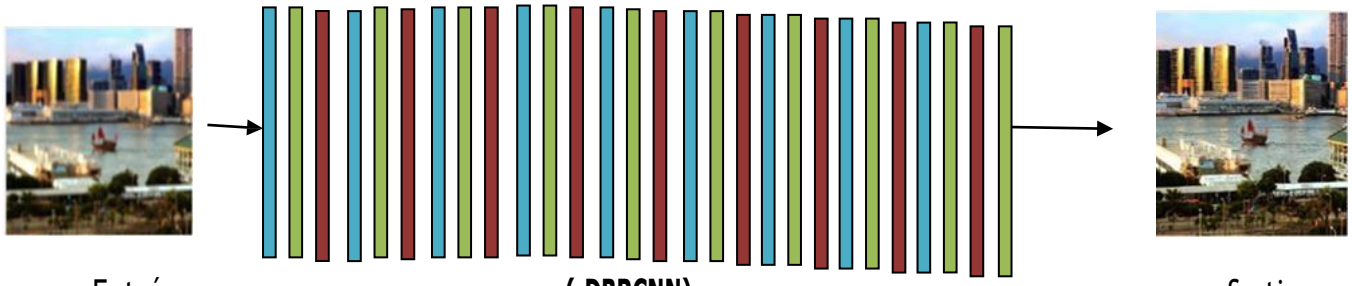


Image06: La structure du système proposé

Couche convolution

Batch Normalisation

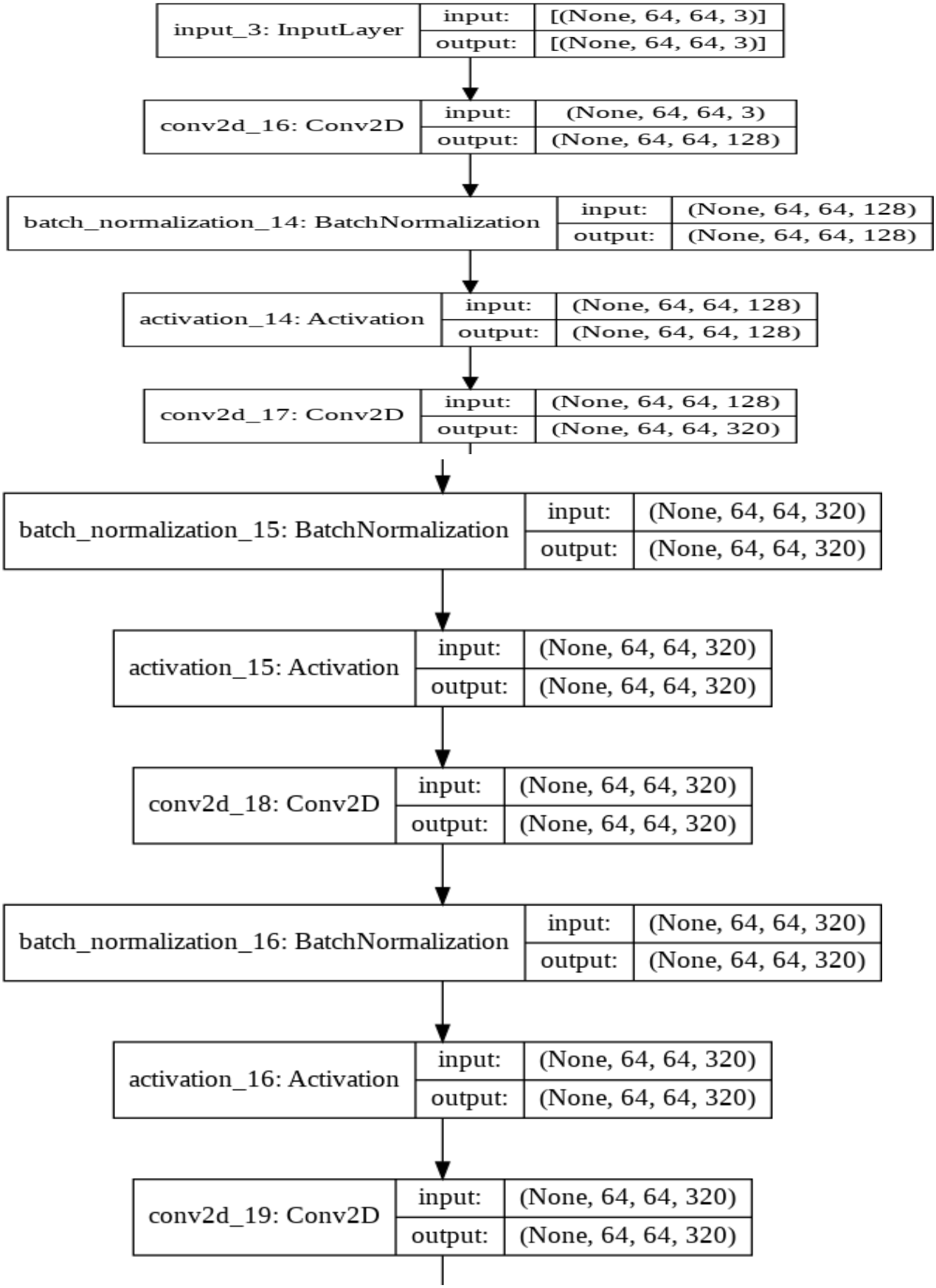
Activation

-Nous proposons une architecture de réseau très profonde pour la restauration d'images. Le réseau est constitué d'une chaîne de couches convolutionnelles symétriques . Les couches convolutionnelles agissent comme des extracteurs de caractéristiques qui éliminent les corruptions.

Le CNN s'adapte à chacune des images floues d'entrée, ce qui permet de minimiser le flou. Les résultats visuels obtenus en utilisant cette méthode sont présentés dans le chapitre suivant.

Dont le détail de cette structure est le suivant (sachant que ce schéma est extrait par le biais de l'instruction

`keras.utils.plot_model).`


input_3: InputLayer
input: [(None, 64, 64, 3)]
output: [(None, 64, 64, 3)]
conv2d_16: Conv2D
input: (None, 64, 64, 3)
output: (None, 64, 64, 128)
batch_normalization_14: BatchNormalization
input: (None, 64, 64, 128)
output: (None, 64, 64, 128)
activation_14: Activation
input: (None, 64, 64, 128)
output: (None, 64, 64, 128)
conv2d_17: Conv2D
input: (None, 64, 64, 128)
output: (None, 64, 64, 320)
batch_normalization_15: BatchNormalization
input: (None, 64, 64, 320)
output: (None, 64, 64, 320)
activation_15: Activation
input: (None, 64, 64, 320)
output: (None, 64, 64, 320)
conv2d_18: Conv2D
input: (None, 64, 64, 320)
output: (None, 64, 64, 320)
batch_normalization_16: BatchNormalization
input: (None, 64, 64, 320)
output: (None, 64, 64, 320)
activation_16: Activation
input: (None, 64, 64, 320)
output: (None, 64, 64, 320)
conv2d_19: Conv2D
input: (None, 64, 64, 320)
output: (None, 64, 64, 320)

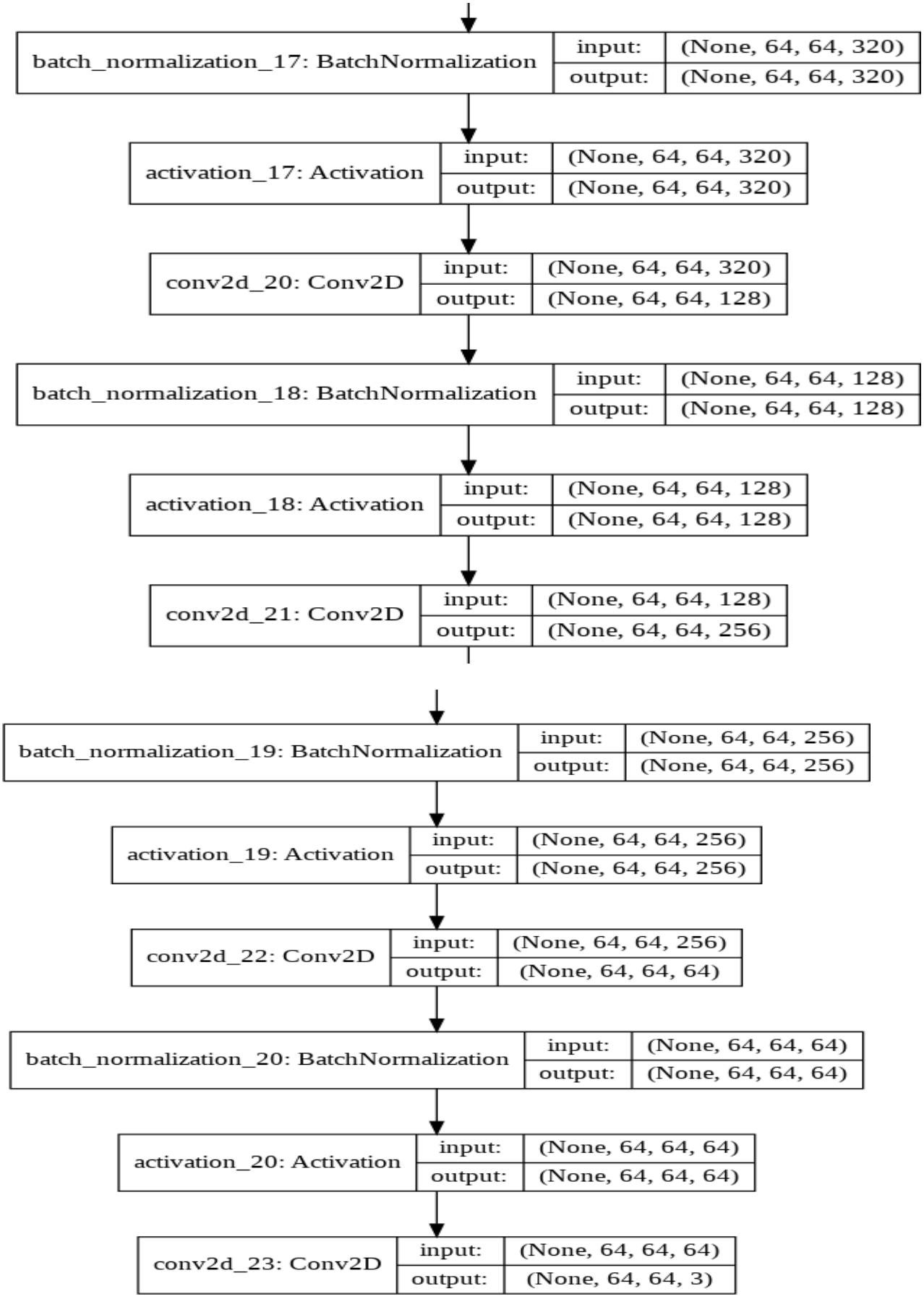


Image07: La structure détaillée du système proposé

4-2-L'algorithme proposé :

Notre objectif est de récupérer une image de qualité nette à partir d'une image d'entrée brouillée, notre modèle travaille selon l'algorithmique suivant :

Algorithme global

---

1. Introduire les imges RGB de taille 64.

2- **Pour i de 1 à 7 (7 blocs) faire :**

1- Appliquer les Convolutions sur l'image y avec un filtre (Kernel) et un pas de déplacement (stride=1)

-Faire un remplissage de bordure de 1 (padding= same) ", pour que , les sorties de la couche auront les mêmes dimensions spatiales que ses entrées.

2- Appliquer une normalisation des lots (BatchNormalization) pour réduire considérablement le nombre d'époques de formation nécessaires pour former des réseaux profonds..

3- Appliquer la fonction d'activation.

Fin pour

3- Appliquer une convolution, un remplissage de 1 et une activation.

Retourner l'image Débrouillée x.

5-Conclusion :

Le présent chapitre nous a permet de connaitre les types de restauration d'images en particulier le Débrouillage qui est l'objet de notre étude, ce qui a nous ramené à étaler quelques travaux connexes et à proposer notre propre algorithme de débrouillage qu'on va le détailler et l'implémenter dans le chapitre suivant.

Références :

[1]K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning forimage recognition," inProc. IEEE Conf. Comp. Vis. Patt. Recogn.,2016.

[3] Xiao-Jiao Mao, Chunhua Shen, Yu-Bin Yang, ''Image Restoration Using ConvolutionalAuto-encoders with Symmetric SkipConnections''

Sitographie :

[2]uwaterloo [en ligne]. [consulté le 07 JUIN 2021]. Image denoising Adresse de la page : https://uwaterloo.ca/vision-image-processing-lab/research-demos/image-denoising

[4] mathworks [en ligne]. [consulté le 07 JUIN 2021]. image-deblurring Adresse de la page : https://se.mathworks.com/help/images/image-deblurring.html

Chapitre03:

Résultats et discussions

1- Introduction :

Nous allons proposer dans cette partie, une approche permettant d'aider à la mise en œuvre d'un modèle, débrouilleur et du coup et d'une façon très synthétique, on va l'expliciter, le faire entrainer, le prédire, l'évaluer et anatomiser les résultats expérimentaux obtenus.

2- Matériel utilisé :

Pour réaliser notre application on a utilisé un micro portable qui a les spécifications suivantes :

Marque	DELL
Modèle	Dell Latitude E6440
Processeur	Intel(R) Core(TM) i7-4610M CPU @ 3.00GHz 3.00 GHz
RAM	12,0 Go
Système utilisé	Windows 10 Professionnel
Cartes Graphiques	AMD Radeon HD 8690M Intel(R) HD Graphics 4600

Image01: Caractéristique du matériel utilisé

3- Présentation des outils de développement :

Les outils d'apprentissage profond permettent aux Datas Scientists de créer des programmes capables d'amener un ordinateur ou une machine à apprendre comme le cerveau humain et à traiter des données et des modèles avant d'exécuter des décisions.

La présentation suivante détaille certains d'outils les plus couramment utilisés et les plus importants pour le développement de notre approche basée sur le CNN.

2-1 :Python :

Python est un langage de programmation interprété, orienté objet, de haut niveau et à sémantique dynamique La syntaxe de Python, simple et facile à apprendre, privilégie la lisibilité et réduit donc le coût de la maintenance des programmes. Python prend en charge les modules et les packages, ce qui encourage la modularité des programmes et la réutilisation du code [1].

2-2-TensorFlow :

Est une plateforme open-source pour la création d'applications d'apprentissage automatique. Il s'agit d'une bibliothèque de mathématiques symboliques qui utilise le flux de données et la programmation différentiable pour effectuer diverses tâches axées sur la formation et l'inférence de réseaux neuronaux profonds. Elle permet aux développeurs de créer des applications d'apprentissage automatique en utilisant divers outils, bibliothèques et ressources communautaires [2].

2-3 :Keras :

Keras offre des fonctionnalités minimales mais très productives grâce à sa bibliothèque d'apprentissage profond. Keras est écrit en Python comme une API d'apprentissage profond, et il fonctionne au-dessus de TensorFlow, une plateforme d'apprentissage automatique. Keras a été développé pour permettre une expérimentation rapide. Il est facile à aborder et offre une interface très productive pour résoudre plusieurs problèmes d'apprentissage automatique en se concentrant sur une approche moderne de l'apprentissage profond. L'avantage essentiel de Kera est qu'il peut prendre l'idée d'un développeur et le guider vers des résultats définitifs[3].

2-4 :Google Colaboratory

Colab est un environnement de notebook Jupyter gratuit qui fonctionne entièrement dans le nuage qui ne nécessite pas de configuration et les notebooks que vous créez peuvent être édités simultanément par les membres de votre équipe - exactement comme vous éditez des documents dans Google Docs [4].

Google Colab prend en charge de nombreuses bibliothèques d'apprentissage automatique populaires qui peuvent être facilement chargées dans votre notebook il vous offre l'exécution basée sur le GPU en évitant la plantation des machines à faibles ressources.

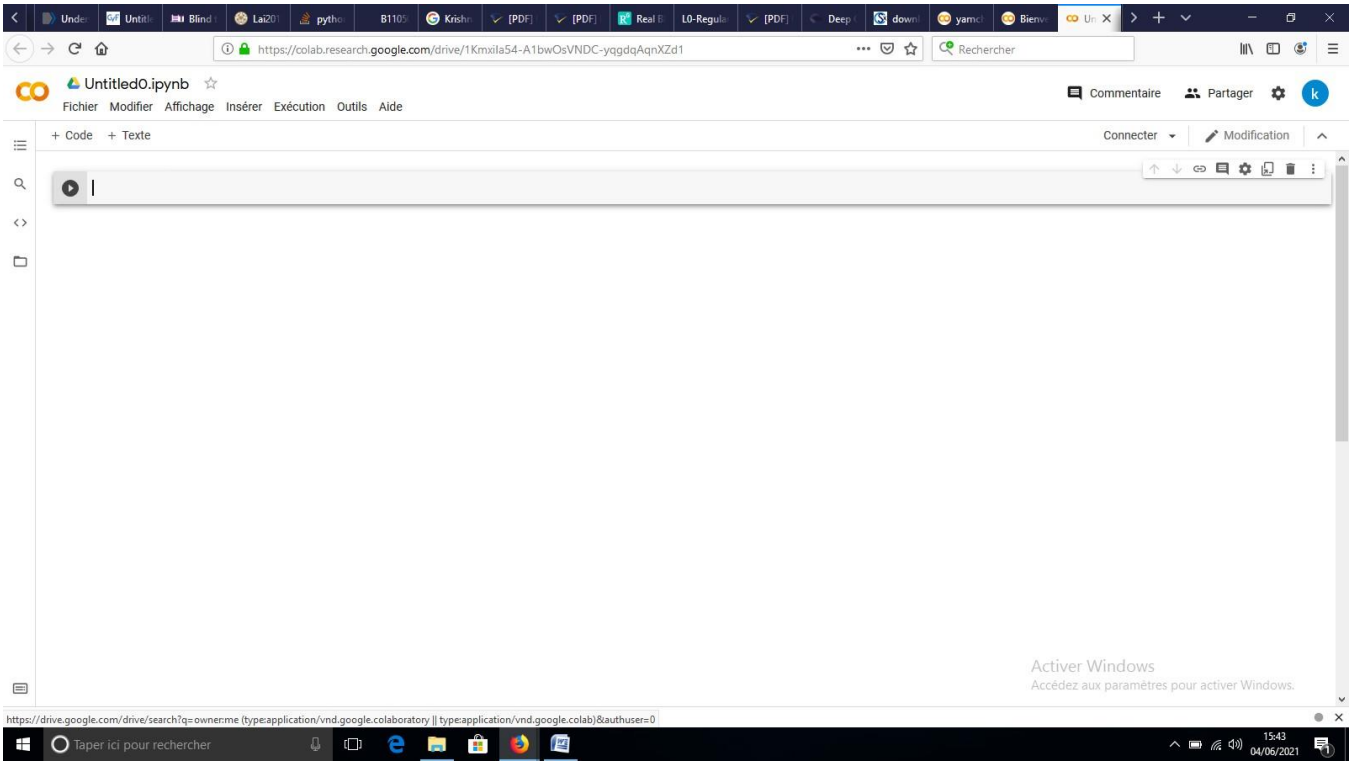


Image02 :Interface du Google Colaboratory

3- Le Dataset :

Le Dataset UTKFace est un Dataset de visages à grande échelle avec une longue fourchette d'âge (de 0 à 116 ans). elle se compose de plus de 20 000 images de visages avec des annotations d'âge, de sexe et d'ethnicité. Les images couvrent une grande variation dans la pose, l'expression faciale, l'illumination, l'occlusion, la résolution, etc. Ce jeu de données pourrait être utilisé pour une variété de tâches, par exemple, la détection de visages, l'estimation de l'âge, la progression/régression de l'âge, la localisation de points de repère, etc. Quelques exemples d'images sont présentés ci-dessous

Image03 : Quelques images du Dataset UTKFace

3-1-Prétraitement sur les images du Dataset:

- Redimensionner les images de notre Dataset de dimension d'origine de 217 x293 à la dimension 64x64. A travers le site 'https://spark.adobe.com/fr-FR/tools/image-resize/'
- Les rendre flou par le biais du site 'https://blur.imageonline.co' en appliquant un taux de de 13%.

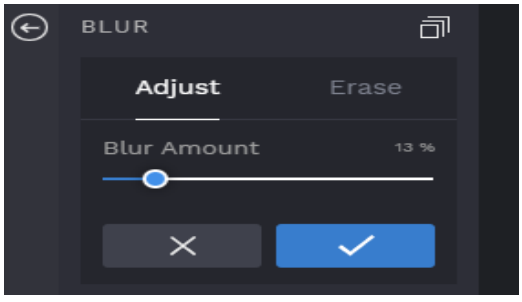


Image04: Ajustement du Brouillage des images de 13%

- Pour chaque pixel, la valeur RGB est divisée par 255.
- Mappage des valeurs RGB [0-255] à la gamme [0-1].

4- La Backpropagation utilisée:

Pour entrainer les hyperparamètres de notre modèle on utilise la backpropagation pour mise à jour la valeur de chaque hyperparamètre. pour que la perte (loss) converge plus rapidement, on a utilisé Adam (Adoptive Moment Estimation).

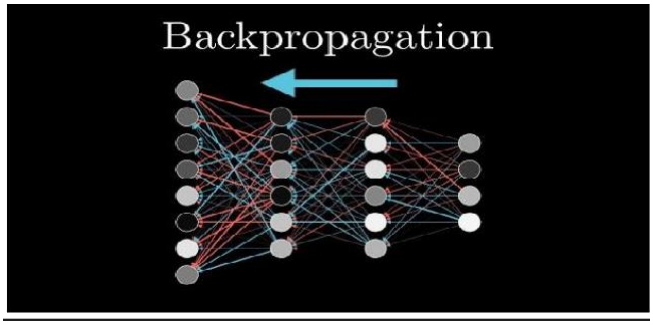


Image05: Fonctionnement de la Backpropagation

5- La fonction de coût utilisée :

Ici on a utilisé un autoencoder de type CNN framework et pour entrainer notre model on a utilisé la fonction de coût MSE (Mean Squared Error Loss) calculée en faisant la soustraction entre les pixels des images originales et les images d'entrainement MSE (images prédictées, images originales) sachant qu'elle a la formule suivante ;

$$\mathbf{EQM}=\frac{1}{M.N}\sum_{i=1}^{M}\sum_{j=1}^{N}((\mathbf{i},\mathbf{j})-\mathbf{I}'(\mathbf{i},\mathbf{j}))^2$$

6- Logique suivie ;

Puisque le CNN est un réseau adaptatif, nous devons l'entraîner à s'adapter à n'importe quelle entrée donnée.

Pour l'entraînement, un ensemble d'images floues et leurs images de base correspondantes sont collectées à partir de notre Dataset.

Afin de donner une image comme entrée à ce CNN, nous devons rendre cette image dans un format d'image particulier, c'est-à-dire, nous devons la compresser comme un fichier JPEG. Plus de nombre d'images utilisées pour la formation, plus grande sera l'efficacité du CNN. Après cela, le CNN est prêt pour le débrouillage. Maintenant l'image floue peut être donnée comme entrée au CNN et nous obtiendrons l'image claire en tant que sortie

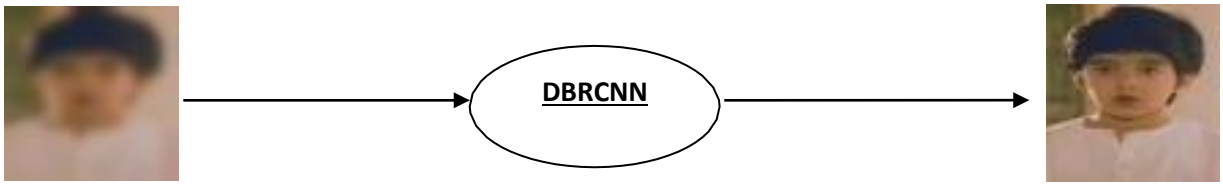


Image floue Mon modèle Image débrouillée

Image06: Fonctionnement global de mon modèle.

7-Les démarches de réalisation de mon application :

Pour réaliser mon application j'ai suivi les étapes suivantes :

1-Importer le Dataset (image floue ,image originale) qui contient des exemples.

2-Développer un modèle (nommé monmodel) au paramètres aléatoires.

3-Compilation du modèle.

3- Développer un algorithme d'apprentissage pour trouver les paramètres de modèle qui minimisent la fonction coût.

4-Développer une fonction coût qui mesure l'erreur entre mon modèle et le Dataset.

5-Mesurer la précision (accuracy) de mon modèle .

6-Evaluer le modèle.

8- Implémentation :

J'ai développé deux implémentations :

8-1-Première implémentation :

Dans cette implémentation j'ai développé un modèle là ou j'ai créé l'entrainement et la prédiction.

8-2-Deuxième implémentation :

Dans cette Deuxième implémentation (DBRCNN2H5) j'ai développé un modèle pré-entrainé de tel sorte que les poids de ce modèle sont pré-sauvegardés dans un fichiers que je l'ai nommé « poids3.h5 » et je contente de le charger sans refaire l'entrainement ni à travers la fonction Load ensuite j'ai fait l'évaluation.

Voici un échantillon de mon code

```
%reset
import numpy as np
import matplotlib.pyplot as plt
import scipy.misc
import cv2
from skimage.io import imread
from glob import glob
import h5py
from tensorflow import keras
from keras.layers import Conv2D, BatchNormalization, Activation
from keras.models import Model, Input
from keras.optimizers import Adam
import keras.backend as K
!pip install -U -q PyDrive
from pydrive.auth import GoogleAuth
from pydrive.drive import GoogleDrive
from google.colab import auth
from oauth2client.client import GoogleCredentials
from google.colab import drive
drive.mount('/content/drive/')
chemin_claire = glob('/content/drive/MyDrive/photodataset/origin/*.jpg')
chemin_floue = glob('/content/drive/MyDrive/photodataset/flou/*.jpg')
```

Image07: Echantillon de mon code

9- Synthèse de notre modèle (DBRCNN):

Voici la synthèse de notre architecture obtenu de l'exécution de l'instruction monmodel.summary()

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 64, 64, 3)]       0
_________________________________________________________________
conv2d (Conv2D)              (None, 64, 64, 128)       38528
_________________________________________________________________
batch_normalization (BatchNo (None, 64, 64, 128)       512
_________________________________________________________________
activation (Activation)      (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 64, 64, 320)       41280
_________________________________________________________________
batch_normalization_1 (Batch (None, 64, 64, 320)       1280
_________________________________________________________________
activation_1 (Activation)    (None, 64, 64, 320)       0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 64, 64, 320)       102720
_________________________________________________________________
batch_normalization_2 (Batch (None, 64, 64, 320)       1280
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 320)       0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 64, 64, 320)       102720
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 320)       1280
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 320)       0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 64, 64, 128)       41088
_________________________________________________________________
batch_normalization_4 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_4 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
conv2d_5 (Conv2D)            (None, 64, 64, 256)       33024
_________________________________________________________________
batch_normalization_5 (Batch (None, 64, 64, 256)       1024
_________________________________________________________________
activation_5 (Activation)    (None, 64, 64, 256)       0
_________________________________________________________________
conv2d_6 (Conv2D)            (None, 64, 64, 64)        802880
_________________________________________________________________
batch_normalization_6 (Batch (None, 64, 64, 64)        256
_________________________________________________________________
activation_6 (Activation)    (None, 64, 64, 64)        0
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 64, 64, 3)         9411
=================================================================
Total params: 1,177,795
Trainable params: 1,174,723
Non-trainable params: 3,072
```

10- Résultats expérimentaux :

10-1- Sur le Dataset UTKFace:

Nous avons réalisé des expériences pour vérifier l'exactitude du schéma proposé, après plusieurs heures d'entrainement (pour 1000 epochs) la perte pour le réseau se converge vers zéro et juste après on a sauvegardé les poids du réseau (dans un fichier .H5 nommé poids2.h5) pour la phase de prédiction dans la deuxième implémentation du code.

Voici quelques expérimentations utilisées :

10-1-1- Expérimentation un :

Pour `Adam(lr= 0.000000000001)` et `epochs=1000` :

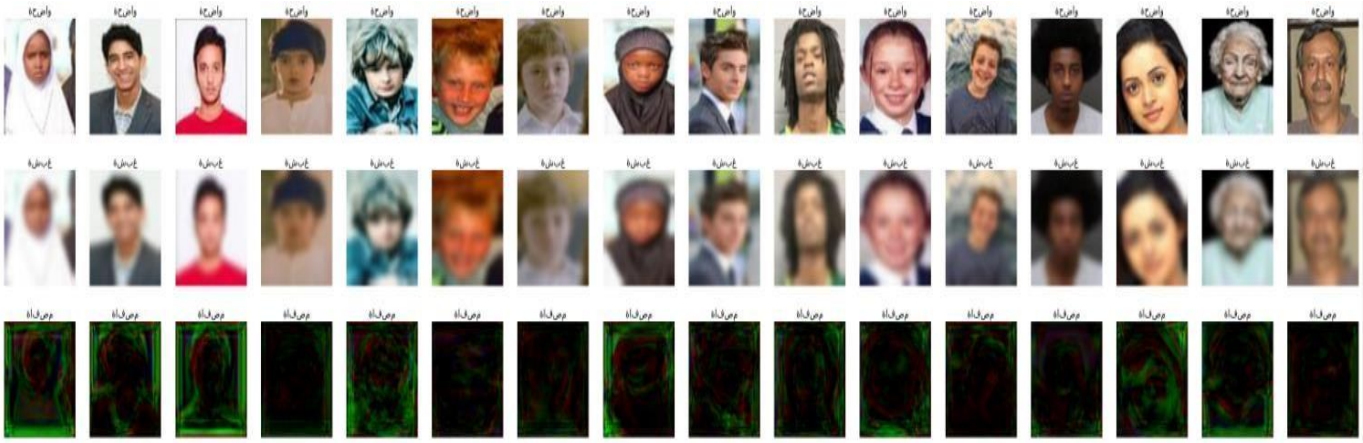


Image08: Résultat de l'application Pour `Adam(lr= 0.000000000001)` et `epochs=1000`

10-1-2-Expérimentation deux :

Pour `Adam(lr= 0.0000001)` et nombre `epochs=10` :

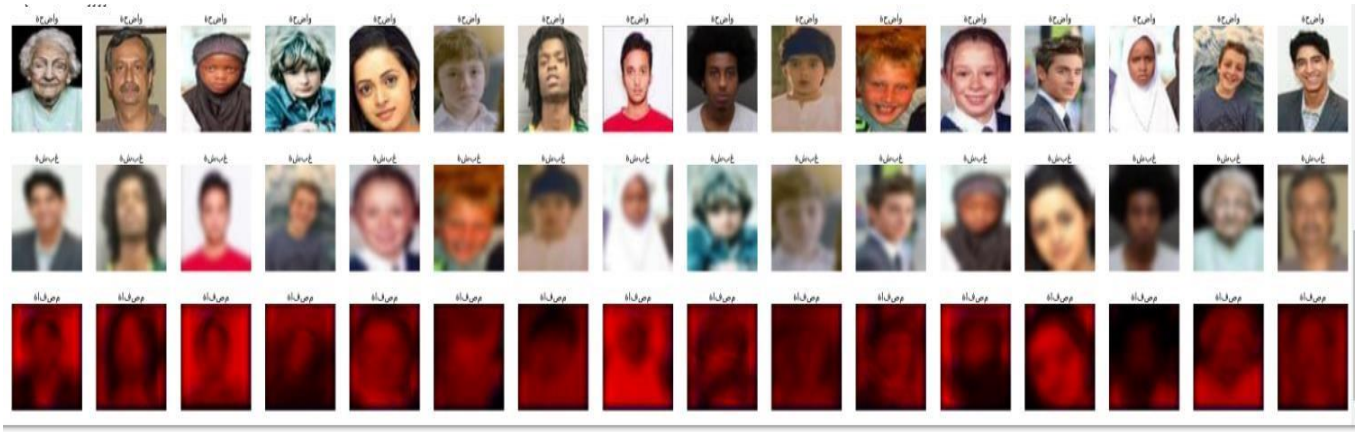


Image09: Résultat de l'application Pour `Adam(lr= 0.0000001)` et `epochs=10`

10-1-3- Expérimentation trois :

Pour `Adam(lr= 0.001) et epochs=50 :`

Image10: Résultat de l'application Pour `Adam(lr= 0.001)` et `epochs=50`

10-1-4- Expérimentation quatre :

Pour Adam `(lr= 0.001)` et epoch= 500 :

Image11: Résultat de l'application Pour `Adam(lr= 0.001)` et `epochs=500`

10-1-5- Expérimentation cinq :

Pour `Adam = Adam(lr= 0.01)` et `epochs=1000` :

Image12: Résultat de l'application Pour `Adam(lr= 0.001)` et `epochs=1000`

10-1-6- La fonction coût:

En appliquant sur notre modèle la fonction MSE en obtenant le résultat suivant :

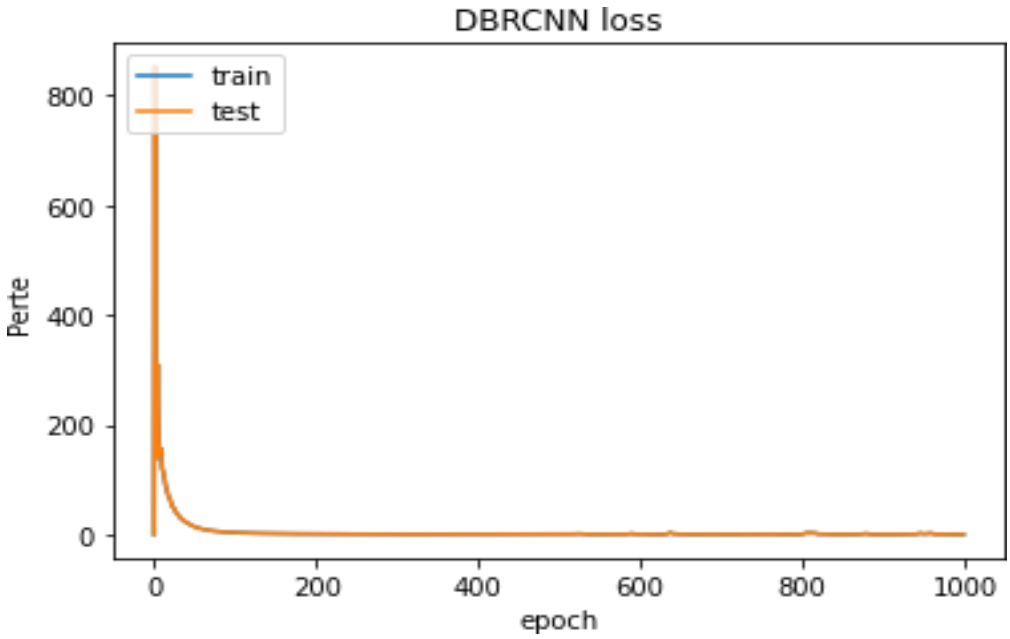


Image13: Evolution de la fonction coût (MSE) en fonction d'epochs.

Le loss qui tend vers zéro qui est très satisfaisant.

10-1-7-L'accuracy (Précision) :

En appliquant sur notre modèle le metrics='accuracy' on obtient la courbe suivante :

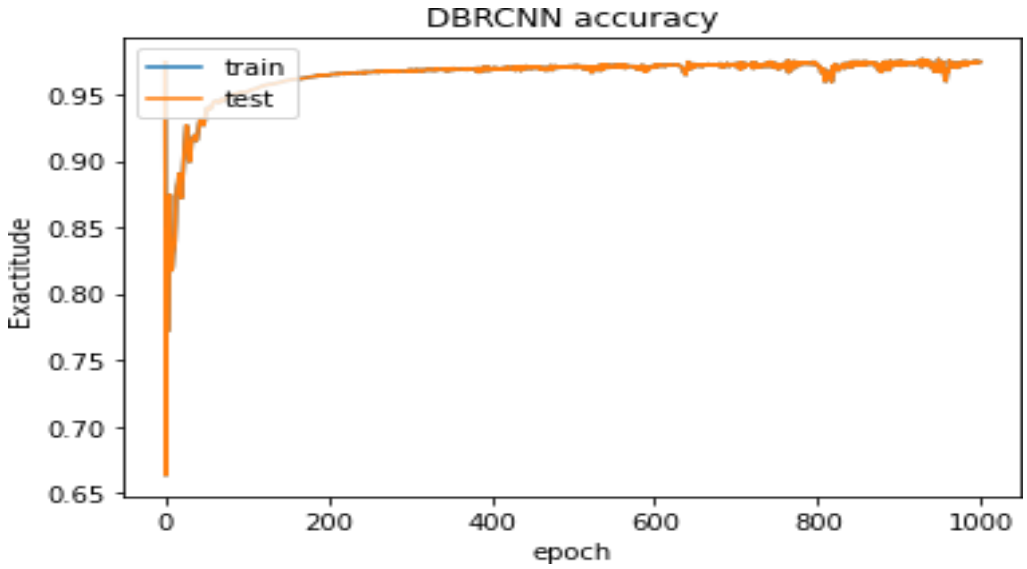


Image14: Evolution de l'accuracy en fonction d'epochs.

L'Accuracy tend vers un qui est très satisfaisant .

10-2- Résultats expérimentaux sur le Dataset des travaux connexes :

En raison du manque de travaux connexes sur le Dataset que nous avons utilisé, ce qui signifie qu'il n'y aura pas de comparaisons des résultats, nous avons dû ré-implémenter l'application sur leur Dataset que nous ne connaissons pas le nom afin d'évaluer notre modèle.

10-2-1-Expérimentation un :

Pour `Adam(lr= 0.00001)` et `epochs=50 :`

Image15: Résultat de l'application Pour `Adam(lr= 0.00001)` et `epochs=50`

10-2-2-Expérimentation deux :

Pour `Adam(lr= 0.001) et epochs=50 :`

Image16: Résultat de l'application Pour `Adam(lr= 0.001)` et `epochs=50`

10-2-3-Expérimentation trois :

Pour `Adam(lr= 0.00001)` et nombre `epochs=200` :

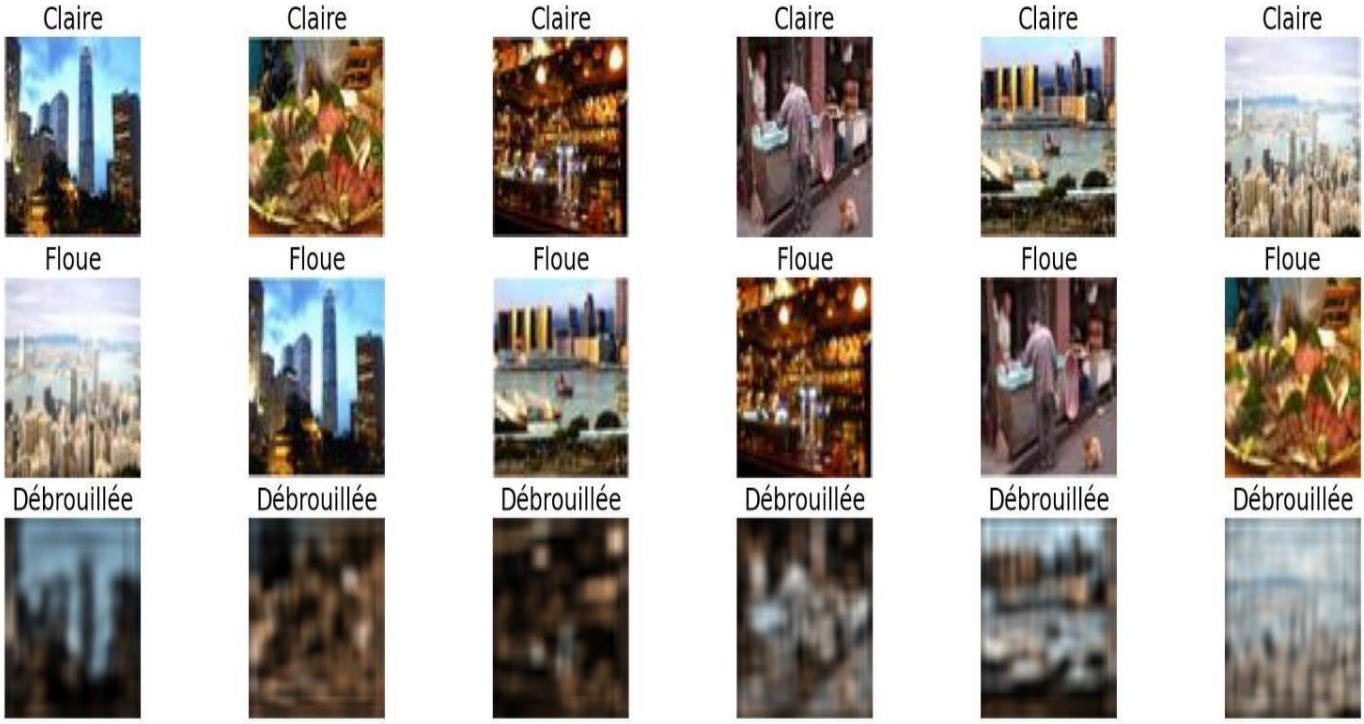


Image17: Résultat de l'application Pour `Adam(lr= 0.00001)` et `epochs=200`

10-2-4-Expérimentation quatre :

Pour Adam `(lr= 0.001)` et epoch= 500 :

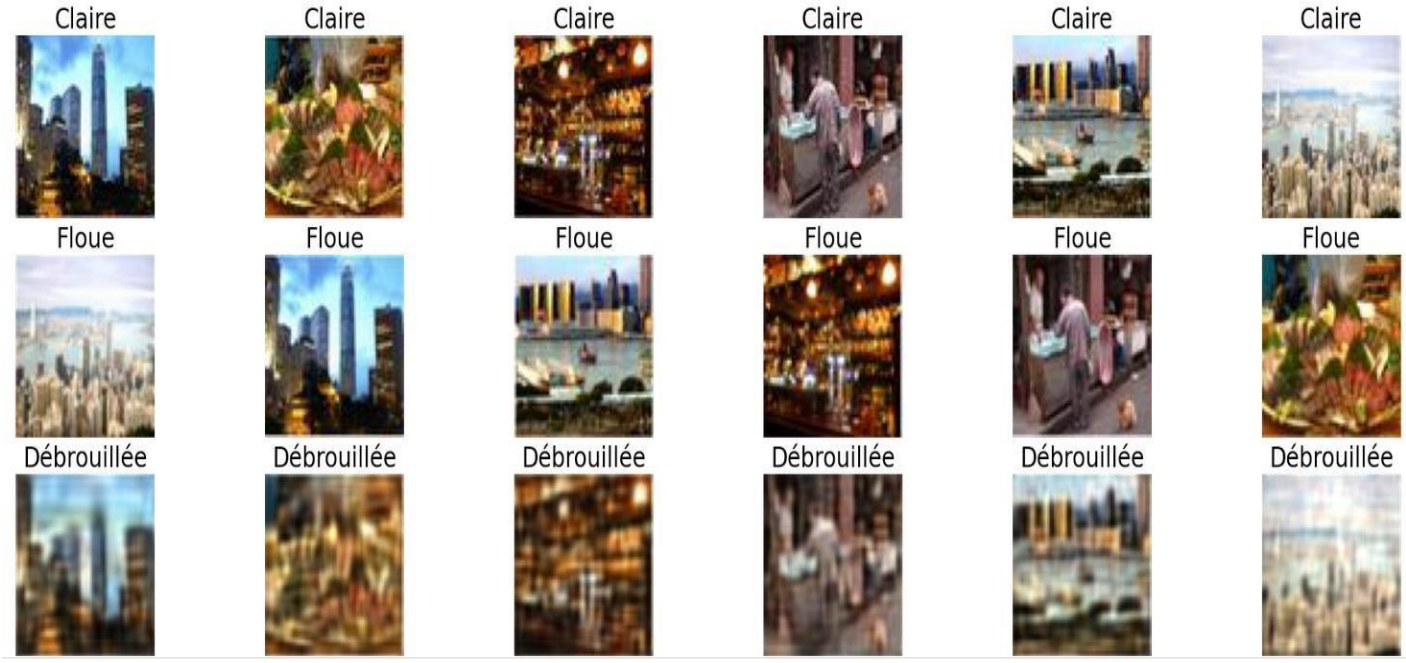


Image18: Résultat de l'application Pour `Adam(lr= 0.001)` et `epochs=500`

10-2-5-Expérimentation cinq :

Pour `Adam = Adam(lr= 0.01)et epochs=1000 :`

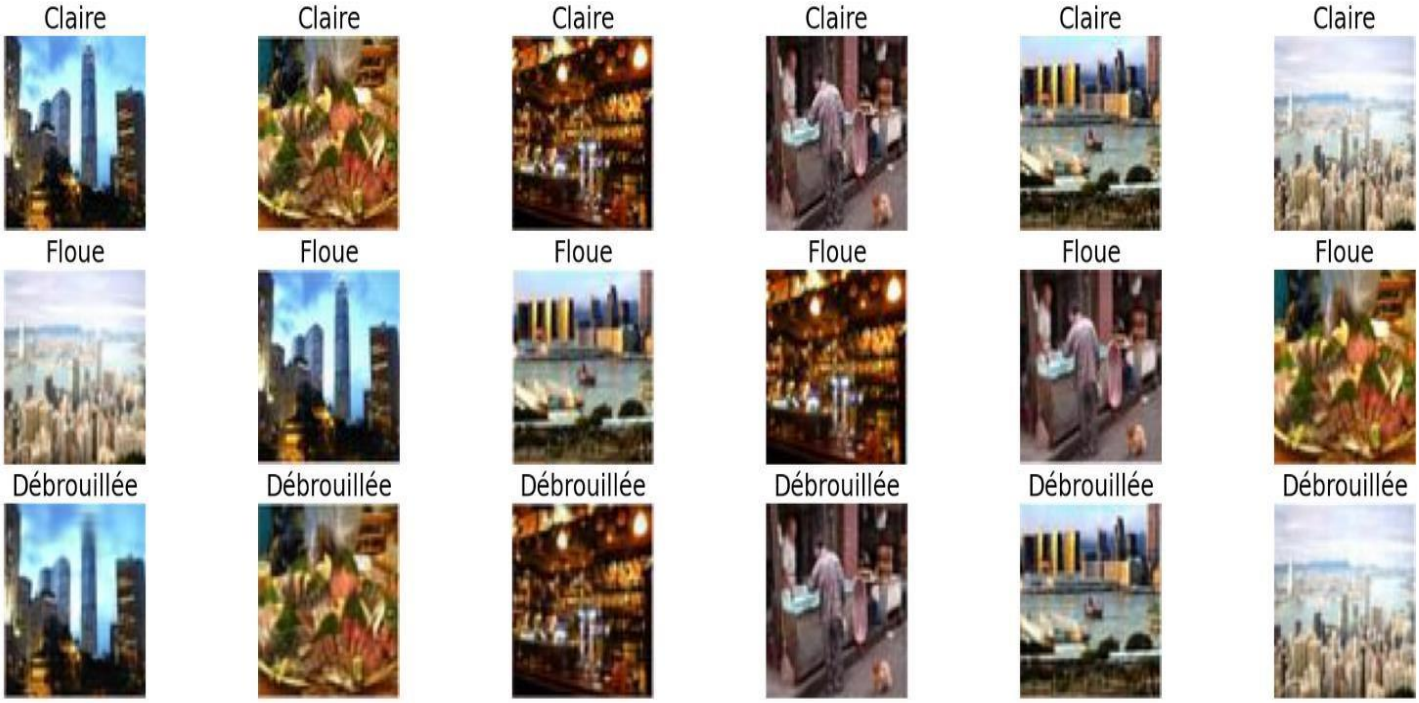


Image19: Résultat de l'application Pour `Adam(lr= 0. 1)` et `epochs=1700`

11-Discussions des résultats obtenus (sur le Dataset des travaux connexes) :

11-1-La fonction coût:

En appliquant sur notre modèle la fonction MSE on obtenant la courbe comme suit :

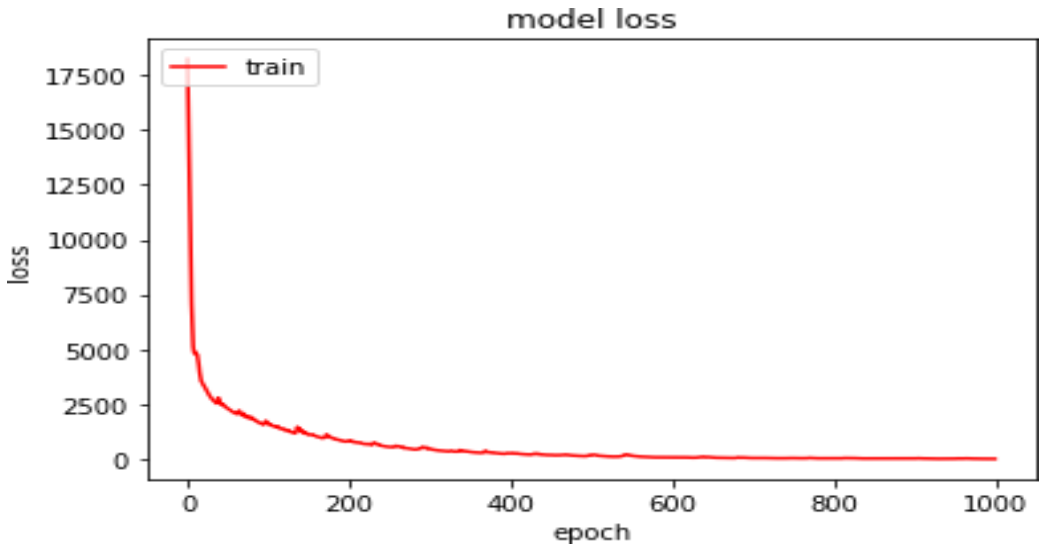


Image20: Evolution de la fonction coût (MSE) en fonction d'epochs.

Le loss tend vers zéro qui est très satisfaisant.

11-2-L'accuracy (Précision) :

En appliquant sur notre modèle le metrics='accuracy' on obtientla courbe suivante :

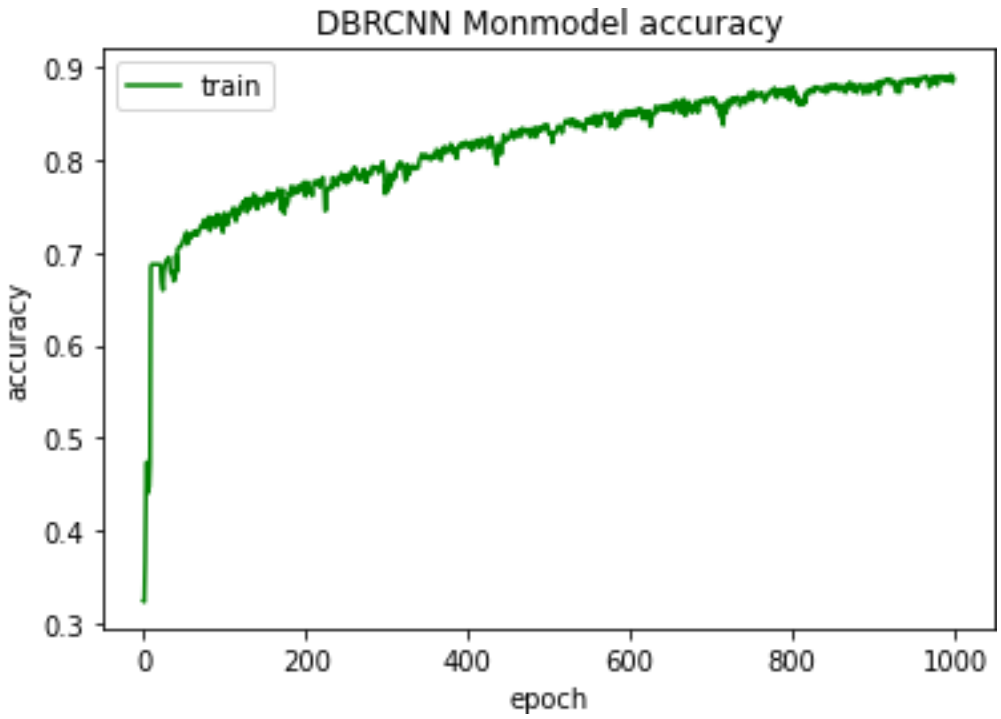


Image21k: Evolution de l'accuracy en fonction d'epochs.

L'accuracy tend vers un qui est très satisfaisant .

12- Analyse et comparaison des résultats avec les travaux connexes:

12-1-Pour la résolution 64X64 :

Le résultat obtenu en utilisant la méthode proposée est très clair par rapport aux techniques de débrouillage existantes. Ce résultat est comparé aux techniques existantes réalisées par Krishnan et al, Levin et al et le débourrage utilisant DCNN. Le facteurs d'évaluation de la performance utilisé pour la comparaison est le rapport signal/bruit (PSNR) par rapport à la vérité du sol[5].

Images	PSNR(Krishnan)	PSNR(Levin)	PSNR(DCNN)	PSNR(DBRCNN) Méthode proposée
Image1	7.3702	6.3370	7.5690	**28.7308**
Image2	10.8856	8.9217	11.3556	**30.1127**
Image3	8.9270	8.1277	9.4881	**30.0834**
Image4	5.9557	5.8330	6.4341	**32.2223**
Image5	10.3725	9.9275	10.9708	**31.3071**
Image6	8.8540	8.3013	9.5322	**31.9864**

Tableau 3 : Comparaison entre les PSNRs des travaux connexes et la technique proposée (DBRCNN) pour la résolution 64X64.

Images	MSE(DCNN)	MSE(DBRCNN) Méthode proposée
Image1	0.2438	**0.0013**
Image2	0.1151	**0.0009**
Image3	0.1771	**0.0009**
Image4	0.3055	**0.0005**
Image5	0.2004	**0.0007**
Image6	0.3562	**0.0006**

Tableau 04 : Comparaison entre le MSE du travail connexe MSE(DCNN) et la technique proposée (DBRCNN) pour la résolution 64X64.

12-2-Pour la résolution 32X32 :

On applique le même modèle sur les images de résolution 32X32 nous obtenons les résultats suivants :

Images	DBRCNN(MSE)	DBRCNN (PSNR)
Image1	**0.0004**	**33.5070**
Image2	**0.0003**	**34.7434**
Image3	**0.0003**	**34.3013**
Image4	**0.0005**	**32.7963**
Image5	**0.0004**	**33.3723**
Image6	**0.0004**	**33.6188**

Tableau 5: Le MSE et le PSNR de la technique proposée (DBRCNN) pour la résolution 32X32.

12-3-Pour la résolution 128X128 :

On applique le même modèle sur les images de résolution 128X128 nous obtenons les résultats suivants :

Images	DBRCNN(MSE)	DBRCNN (PSNR)
Image1	**0.0144**	**18.4141**
Image2	**0.0091**	**20.4095**
Image3	**0.0045**	**23.4579**
Image4	**0.0038**	**24.1465**
Image5	**0.0072**	**21.3766**
Image6	**0.0032**	**24.8268**

Tableau 6 : Le MSE et le PSNR de la technique proposée (DBRCNN) pour la résolution 28X128.

12-4-Pour la résolution 256X256 :

On applique le même modèle sur les images de résolution 256X256 nous obtenons les résultats suivants :

Images	DBRCNN(MSE)	DBRCNN (PSNR)
Image1	**0.0124**	**19.0524**
Image2	**0.0060**	**22.1578**
Image3	**0.0105**	**19.7615**
Image4	**0.0299**	**15.2429**
Image5	**0.0130**	**18.8517**
Image6	**0.0153**	**18.1343**

Tableau 7: Le MSE et le PSNR de la technique proposée (DBRCNN) pour la résolution 256X256.

12-5-Résumés des comparaisonss entre les PSNRs des travaux connexes et les PSNRs de la technique proposée avec ses différentes résolutions:

Images	PSNR des Travaux connexes			PSNR de La méthode Proposée(DBRCNN)			
	PSNR (Krishnan)	PSNR (Levin)	PSNR (DCNN)	PSNR 32X32	PSNR 64X64	PSNR 128X128	PSNR 256X256
Image1	7.3702	6.3370	7.5690	**33.5070**	**28.7308**	**18.4141**	**19.0524**
Image2	10.8856	8.9217	11.3556	**34.7434**	**30.1127**	**20.4095**	**22.1578**
Image3	8.9270	8.1277	9.4881	**34.3013**	**30.0834**	**23.4579**	**19.7615**
Image4	5.9557	5.8330	6.4341	**32.7963**	**32.2223**	**24.1465**	**15.2429**
Image5	10.3725	9.9275	10.9708	**33.3723**	**31.3071**	**21.3766**	**18.8517**
Image6	8.8540	8.3013	9.5322	**33.6188**	**31.9864**	**24.8268**	**18.1343**

Tableau 8 : Comparaison entre les PSNRs des travaux connexes et les PSNRs de la technique proposée avec ses différentes résolutions.

En guise de résumé nous constatons que notre modèle DBRCNN via ses différents PSNR correspondants aux différentes résolutions a de meilleure performance, a de meilleure proximité de l'image restituée par rapport à l'original ainsi que sa qualité de reconstruction est meilleure par rapport aux différentes techniques des travaux connexes.

12-6-Résumé de comparaison entre le MSE(DCNN) du travail connexe et le MSE de la technique proposée (DBRCNN) avec ses différentes résolutions:

Images	MSE(DCNN)	MSE(DBRCNN) 32X32	MSE(DBRCNN) 64X64	MSE(DBRCNN) 128X128	MSE(DBRCNN) 256X256
Image1	0.2438	**0.0004**	**0.0013**	**0.0144**	**0.0124**
Image2	0.1151	**0.0003**	**0.0009**	**0.0091**	**0.0060**
Image3	0.1771	**0.0003**	**0.0009**	**0.0045**	**0.0105**
Image4	0.3055	**0.0005**	**0.0005**	**0.0038**	**0.0299**
Image5	0.2004	**0.0004**	**0.0007**	**0.0072**	**0.0130**
Image6	0.3562	**0.0004**	**0.0006**	**0.0032**	**0.0153**

Tableau 09: Comparaison entre les MSE de la technique existante DCNN et les MSEs de la technique proposée (DBCRCNN) avec ses différentes résolutions.

Les MSE obtenu de l'application de notre modèle sur les différentes résolutions tendent vers le zéro et largement plus inférieurs des MSE du travail connexe (DCNN) ce qui explique la différence minimale entre la qualité des images restituées et les images originales ce qui implique la supériorité d'exactitude de notre modèle

Sachant que le PSNR utilise un terme d'erreur quadratique moyenne (MSE) au dénominateur. Ainsi, plus l'erreur est faible, plus le PSNR est élevé ce qui est confirmé dans notre cas.

13- Conclusion :

On a proposé l'utilisation d'un réseau convolutif profond (DBRCNN) pour le débrouillage, ce modèle qui prouve sa supériorité expérimentalement et donne des résultats convainquant et compétitifs par rapport aux méthodes d'apprentissage non profondes et par rapport aux méthodes profondes vues dans les travaux connexes.

Références:

[5] Image Deblurring Using Convolutional Neural NetworkReshma Vijay V.J., Deepa P.L.(Electronics and Communication, Mar Baselios College of Engineering, India) arXiv preprint arXiv:1502.03167, 2015.

Sitographie :

[1] purushothanandaraja [en ligne]. [consulté le 03Mai 2021]. what-is-python Adresse de la page : https://purushothanandaraja.medium.com/what-is-python-fd890e91a650' 24/05/2021 01 :13

[2] guru99 [en ligne]. [consulté le 03Mai 2021]. what_is_google_colab Adresse de la page : 'https://www.guru99.com/what-is-tensorflow.html' 24/05/2021 01 :20

[3] mindmajix [en ligne]. [consulté le 03Mai 2021]. what_is_Keras Adresse de la page : 'https://mindmajix.com/deep-learning-tools#deep-learning-kit' 24/05/2021 01 :25

[4] tutorialspoint [en ligne]. [consulté le 03Mai 2021]. what_is_google_colab Adresse de la page : 'https://www.tutorialspoint.com/google_colab/what_is_google_colab.htm 24/05/2021 01 :40

Conclusion générale

Diverses approches ont été proposées pour réduire les effets de la compression des images, mais pour utiliser des techniques de compression/décompression normalisées et conserver les avantages de la compression (par exemple, la réduction des coûts de transmission et de stockage), beaucoup de ces méthodes se concentrent sur le "post-traitement", c'est-à-dire le traitement des images lorsqu'elles sont reçues ou visualisées. Aucune technique de post-traitement ne s'est avérée capable d'améliorer la qualité de l'image dans tous les cas ; par conséquent, aucune n'a été largement acceptée, bien que certaines aient été mises en œuvre et soient utilisées dans des systèmes propriétaires. De nombreux programmes de retouche photo, par exemple, intègrent des algorithmes propriétaires de réduction des artefacts JPEG.

Dans ce mémoire, on a utilisé une autre approche basée sur l'apprentissage profond pour réduire les artefacts de compression d'images, plus précisément, on a proposé une structure du réseau neuronal convolutif (CNN) efficace et fondé sur des principes pour cette tâche qui nous a permis de débrouiller ces images. On a surnommé le réseau proposé pour le débrouillage (DBRCNN) comme acronyme de Débrouillage CNN.

L'objectif de notre réseau est de générer des résultats raisonnables à partir des images dégradées (brouillées) d'entrée.

Notre DBRCNN est capable de produire des images avec plus de détails que les méthodes génériques ainsi que les autres méthodes profondes vues dans les travaux connexes.

Dans ce travail on a essayé de proposer une nouvelle procédure d'apprentissage basée sur le CNN à différents couches convolutives et on a abouti à travers les résultats

obtenus que notre approche peut être utilisée comme une étape de prétraitement pour différentes tâches de vision par ordinateur dans le cas où les images sont dégradées par la compression à un point tel que les algorithmes de pointe échouent.

Notre approche proposée basée sur le CNN obtient de meilleures performances en minimisant les artéfacts générés par les différents algorithmes de compression (dans notre cas la réduction des artéfacts produit par JPEG).

Et en répondant à notre problématique citée dans l'introduction qui s'interroge sur la structure CNN optimale pour traiter la réduction d'artéfacts (le débrouillage) on a trouvé plus qu'une structure et on pense qu'il est difficile de trouver la meilleure malgré que notre modèle a montré une large supériorité par rapport aux travaux connexes.

Inspiré par le CNN, on se demande si les Réseaux Neuronaux récurrents (RNN) ou les Réseaux Antagonistes Génératifs (GAN) peuvent aider à obtenir de meilleures performances en restauration ou débrouillage d'images.