RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

UNIVERSITE ECHAHID HAMMA LAKHDAR - EL OUED
FACULTÉ DES SCIENCES EXACTES
Département D'Informatique

Mémoire de Fin D'étude
Présenté pour l'obtention du Diplôme de

MASTER ACADEMIQUE

Domaine : **Mathématique et Informatique**
Filière : **Informatique**
Spécialité : **Systèmes D'informations**

Thème

Amélioration de la résolution d'image à l'aide de l'apprentissage en profondeur

Présenté par :

- HMIDI IMANE
- BENNEDJAI Hadjer

Suivie par :

- M. ZAIZ FAOUZI

Soutenue le :26 -09- 2023

Devant le jury:

M.	LEJDEL BRAHIM	MAA	Président
M.	KHABECHE MOHIBEDDINE	MAA	Rapporteur

Année Universitaire: 2022/2023

Remerciements

Tout d'abord, nous remercions Dieu de nous avoir donné la force et la patience d'achever ce travail en temps voulu.C'est notre plaisir ainsi que notre devoir de remercier toutes les personnes qui ont rendu cela possible.Contribuez directement ou indirectement à la création de ce projet, qui nous a aidés et soutenus et réalisez ce travail. Ainsi, nous exprimons notre gratitude et tenons à remercier Dr.« Zaiz Faouzi » qui est notre encadreur, qui n'a ménagé aucun effort pour nous guider afin que nous puissions mettre en place ce projet.

Nous adressons également nos sincères remerciements à Professeur Dr.« LAOUID Abdelkader », qui nous a soutenu plus que quiconque , Difficile de terminer ce travail si et ses précieux conseils.

Nous tenons également à remercier les membres du jury pour le temps précieux qu'ils ont consacré à l'étude notre mémoire.

Enfin, nous remercions chaleureusement nos parents et nos sœurs, qui nous ont toujours fait confiance et ils nous ont soutenus dans notre étude.

Dédicace

Je dédie ce projet

Quant à ma mère, qui m'a soutenu et encouragé durant ces années d'études,

vous y trouverez peut-être le témoignage de ma profonde gratitude.

A mes frères et à ceux qui ont partagé avec moi tous les moments d'émotion

durant la réalisation de ce travail.

Ils m'ont chaleureusemen soutenu et encouragé tout au long de mon parcours.

À ma famille, mes proches et ceux qui me donnent de l'amour et de l'énergie.

A tous mes amis qui m'ont toujours encouragé et à qui je souhaite encore plus.

A tous ceux que j'aime.

Imane

Dédicace

Je dédie cet humble travail à mes chers parents qui n'ont cessé de m'encourager tout aulong de mes études et depuis.

Ils ont été avec moi toute ma vie

Pour me donner une vie meilleure, que Dieu les protège.

A mes frères Pour leur soutien moral et leurs précieux conseils de tous les instants tout au long de mes études.

A toute ma famille, à mes chers amis.

Pour les aider et les soutenir dans les moments difficiles.

Enfin, je dédie ce travail à tous ceux qui me connaissent de près ou de loin.

Hadjer

Résumé:

Lors de l'utilisation d'images numériques dans des applications, il est toujours recommandé d'utiliser des images haute résolution, car elles peuvent .Ils contiennent des détails importants pour différentes applications par rapport à leurs homologues basse résolution. Dans ce travail, nous utilisons une méthode basée sur l'apprentissage profond pour convertir une image basse résolution en une image haute résolution.

Cette méthode utilise un réseau adversaires génératifs de super résolution améliorés grâce à un traitement séquentiel et un traitement parallèle. Le réseau est capable de produire une image haute résolution, tout en prenant une image basse résolution en guise d'entrée.

***Mots Clés*:** Super-résolution,interpolation,l'apprentissageenprofondeur,ESRGAN, SRGAN.

Abstract:

When using digital images in applications, it is always recommended to use high-resolution images because they can contain important details for different applications compared to their low-resolution counterparts. In this work, we use a deep learning based method to convert a low resolution image into a high resolution image.

This method uses a super-resolution generative adversarial network enhanced through sequential processing and parallel processing. The network is capable of producing a high resolution image, while taking a low resolution image as input.

***Keywords* :**Super-resolution ,Interpolation ,Deep-learning ,ESRGAN,SRGAN.

الملخص:

عند استخدام الصور الرقمية في التطبيقات ، يوصى دائمًا باستخدام صور عالية الدقة قدر الإمكان. تحتوي على تفاصيل مهمة لتطبيقات مختلفة مقارنة بنظيراتها منخفضة الدقة. في هذا العمل ، نستخدم طريقة قائمة على التعلم العميق لتحويل صورة منخفضة الدقة إلى صورة عالية الدقة.

تستخدم هذه الطريقة شبكة خصومة فائقة الدقة محسّنة من خلال المعالجة المتسلسلة والمعالجة المتوازية. الشبكة قادرة على إنتاج صورة عالية الدقة ، مع التقاط صورة منخفضة الدقة كمدخل.

كلمات المفتاحية: دقة الفائقة , الإستفياء , التعلم العميق, شبكات الخصومة التوليدية فائقة الدقة.

TABLE DES MATIÈRES

TABLE DES FIGURES

Liste des abréviations :

SR : Super Resolution.

HR : High Resolution.

LR : Low Resolution.

IA : Intelligence Artificielle.

ML : Machine Learning.

DL : Deep Learning.

CNN : Convolutional Neural Network.

SRCNN : Super-Resolution Convolutional Neural Networ.

FRCNN : Fast Super-Resolution Convolutional Neural Network.

VDSR : Very Deep Super Résolution.

GAN : Generative Adversarial Networks.

ESRGAN :Enhanced Super-Resolution Generative Adversarial Networks.

VGG : Visual Geometry Group

PSNR : Peak Signal-to-Noise Ratio.

MSE : Mean Squared Error.

SSIM : The Structural SIMilarity.

DRCN : Le Deep Recursive Convolutional Network.

LapSRN : Laplacian Pyramid Super-Resolution Network.

CMSC : Cascaded Multi-Scale Cross-Network.

Open Cv : Open Source Computer Vision Library.

RaGAN : Relativistic Average GAN.

Sklearn : Scikit-learn.

RRDB : le Résiduel-dans-Résiduel proposé dense Block.

RLU : Rectified Linear Unit.

Listes d'équations :

INTRODUCTION GÉNÉRALE

Dans notre vie d'aujourd'hui, l'image est une source importante d'échange d'informations pour les utilisateurs, car elle se présente sous toutes les formes, elle a donc souvent besoin d'être traitée. Le traitement d'images est une discipline qui étudie les images numériques et leurs transformations dans le but d'en améliorer la qualité ou d'en extraire des informations fiables. Cette étude a commencé dans les années 1920 et a évolué rapidement avec les progrès de l'informatique et la naissance de machines informatiques de plus en plus sophistiquées.Les technologies modernes de traitement d'image se sont traditionnellement concentrées sur l'optimisation, la restauration, la compression, l'extraction, la transformation et la super-résolution des images. Dans les systèmes de vision industrielle. Avec le développement des technologies de l'information, la super résolution a fait de grands progrès dans le traitement des images.

La super résolution d'image (SR) est le processus de reconstruction d'images haute résolution (HR) à partir d'images basse résolution (LR). Il s'agit d'une catégorie importante des techniques de traitement d'image et de vision par ordinateur, c'est-à-dire qu'il s'agit de prendre une image d'entrée et d'augmenter sa taille, ce qui signifie l'afficher et l'agrandir avec le moins de dégradation possible dans sa qualité. , mais en fait il y a un problème c'est que quand tout le monde essaie d'agrandir et de réduire l'image, il est nécessaire et courant de la redimensionner, et ils trouvent que ça devient moins bien. Une fois réduite, il est difficile de la reconstruire par mise à l'échelle du fait de la perte d'informations, on est donc amené à refixer l'image si on veut délibérément la redimensionner, pour rendre ces deux opérations plus compatibles. Les travaux en cours ont été exécutés. Basé sur l'application d'un modèle de haute précision basé sur l'apprentissage en profondeur.

Il s'est concentré sur la durée d'exécution et a toujours maintenu une haute qualité visuelle des images agrandies. Ces dernières années, de grands progrès ont été réalisés. Cependant, les modèles deviennent de plus en plus gros ; Du premier réseau à 3 couches réussi au dernier réseau à 160 couches. Ainsi, les modèles étaient très gourmands en calcul, volumineux et lents à exécuter. Avec l'avènement des téléphones mobiles et de l'Internet des objets (IoT), il existe un intérêt croissant pour la construction de modèles plus rapides et plus petits. Ceci est essentiel pour appliquer des solutions d'apprentissage en profondeur ultra-précises à des applications du monde réel où les ressources de mémoire et de calcul sont limitées. Actuellement, la plupart de ces solutions sont encore à l'étude.

Cette projet peut être divisée en quatre chapitres : état d'art,travaux similaires, conception, et implementation. Le but de état d'art est d'analyser les architectures de réseau de neurones les plus efficaces pour atteindre une plus grande précision et de concevoir un réseau petit et efficace. Le chapitre deuxième nous mentionnerons certains des travaux précédents de super résolution et verrons ce que vous obtenez ,Le chapitre troisième traite de la description de l'architecture réseau utilisée. Nous comprenons également mieux pourquoi le réseau fonctionne comme il le fait. C'est un domaine d'apprentissage en profondeur qui est généralement négligé. Le chapitre quatrème est destiné à évaluer le succès du modèle proposé. Elle sera évaluée à l'aide des procédures de reconstruction d'image les plus courantes.Cependant, juger de la qualité de l'image est intrinsèquement subjectif. Par conséquent, une évaluation qualitative, c'est-à-dire visuelle, est également effectuée.

CHAPITRE 1

ÉTAT D'ART

1.1 Introduction :

Dans ce chapitre, nous présenterons des concepts importants liés à la super-résolution, certaines méthodes traditionnelles, L'apprentissage automatique (ML) est un terme générique désignant la résolution de problèmes pour lesquels le développement d'algorithmes par des programmeurs humains serait d'un coût prohibitif. Au lieu de cela, les problèmes sont résolus en aidant les machines à « découvrir » leurs « propres » algorithmes,sans avoir besoin de les résoudre. se faire dire explicitement quoi faire par tout algorithme développé par l'homme.Récemment, les réseaux de neurones artificiels génératifs ont pu surpasser les résultats de nombreuses approches précédentes.Les approches d'apprentissage automatique ont été appliquées aux grands modèles de langage, à la vision par ordinateur, à la reconnaissance vocale, au filtrage des e-mails, à l'agriculture et à la médecine, où il est trop coûteux de développer des algorithmes pour effectuer les tâches nécessaires,et nous parlerons des modèles de super-résolution intégrés à l'apprentissage profond pour résoudre cette tâche.

1.2 Super resolution :

1.2.1 Définition :

Le rôle de la super-résolution en vision par ordinateur est d'augmenter la résolution d'une image ou d'un clip vidéo en reproduisant les détails haute fréquence manquant dans l'entrée basse résolution. Cette idée est devenue très populaire au fil des années et est particulièrement utile aujourd'hui. Cependant, de nombreuses méthodes sont développées sur une base similaire : elles utilisent des algorithmes spécifiques pour récupérer les pixels manquants des images basse résolution. Supposons que l'image ait une résolution de 64 x 64 pixels et une haute résolution de 256 x 256 pixels. Dans ce cas, le processus est appelé suréchantillonnage 4x car les dimensions spatiales (hauteur et largeur de l'image) sont mises à l'échelle quatre fois[16].

1.2.2 Pourquoi la super résolution :

L'objectif principal de la super-résolution (SR) est de générer une image de sortie avec une résolution plus élevée que l'image d'entrée tout en préservant le contenu et la structure d'origine. Les images haute résolution offrent une densité de pixels élevée et donc plus de détails sur la scène originale. Le besoin d'une haute résolution est courant dans les applications de vision par ordinateur afin d'obtenir de meilleures performances en matière de reconnaissance

de formes et d'analyse d'images. De nombreuses applications nécessitent une adaptation à des régions d'intérêt spécifiques dans les images où la haute résolution devient critique, comme les applications de surveillance, d'investigation et d'imagerie satellitaire.

1.3 Techniques traditionnelles de super-résolution :

1.3.1 Interpolation :

Définition l'interpolation :

L'interpolation d'une image est le processus de recherche des valeurs à des points inconnus à l'aide de données connues points.L'interpolation se produit chaque fois que nous redimensionnons ou redimensionnons pour zoomer ou réduire, pour faire pivoter une image et même pour améliorer la résolution d'une image, ce qui entraîne l'ajout ou la suppression de pixels dans une image. Parfois, l'interpolation devient nécessaire pour obtenir une qualité plus détaillée, nette et haute résolution d'une image.

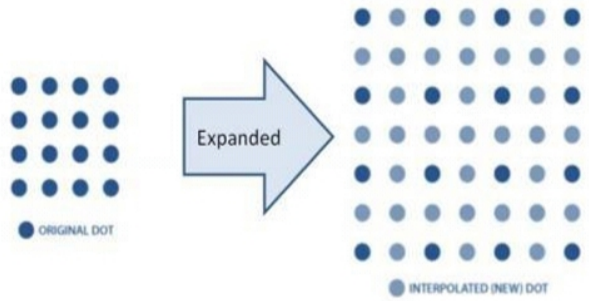


FIGURE 1.1 – Idée de base de l'interpolation.

En général, l'interpolation suit l'étape illustrée à la figure 1.2. Tout d'abord, on cible la position du pixel à interpoler, puis on accède au pixel voisin pour trouver une valeur moyenne de pixel. Cela se fait à la fois horizontalement et verticalement, et enfin, la valeur de pixel interpolée sera obtenue dans une image[17].

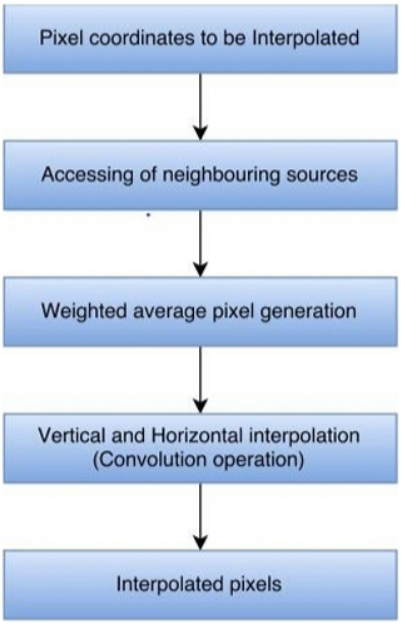


FIGURE 1.2 – Flux de base de l'interpolation.

1.3.2 Méthode d'interpolation :

Les techniques d'interpolation dans le cadre de méthodes non adaptatives impliquent de manipuler directement les pixels existants sans tenir compte d'aucun autre contenu écrit ou caractéristiques de l'image. Elles effectuent une interpolation de pixel un à un.

1-Interpolation du voisin le plus proche :

C'est l'algorithme d'interpolation le plus simple. Chaque pixel inconnu est associé à la même valeur d'intensité que ses pixels voisins. De plus, cette méthode est la mise en œuvre la plus rapide de la technique de détartrage d'image. Par conséquent, les méthodes du plus proche voisin sont utiles lorsque la vitesse est une considération majeure, en particulier lors du zoom sur une petite partie d'une image. Le principe de la mise à l'échelle d'une image est d'avoir une image de référence, et sur la base de cette image, de nouvelles images mises à l'échelle peuvent être construites. Lorsque vous zoomez sur une image, cela crée en fait un espace vide sur l'image de base d'origine. Pour la méthode du voisin le plus proche, les zones vides sont remplacées par les pixels les plus proches.

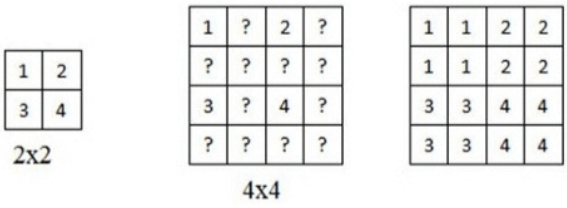


FIGURE 1.3 – Interpolation du plus proche voisin avec l'échelle de 2 [1].

2-Interpolation bilinéaire :

L'interpolation bilinéaire est la moyenne pondérée des 4 pixels voisins les plus proches pour estimer qu'elle est finale valeur interpolée. Ici, nous calculons l'interpolation linéaire 2 fois. Un dans le sens horizontal et l'autre dans le sens vertical.Puisqu'il en résulte une interpolation quadratique avec un champ réceptif de taille 2×2, il montre de bien meilleures performances que l'interpolation au plus proche voisin tout en gardant une vitesse relativement rapide Si toutes les distances de pixels connues sont égales, le pixel interpolé final est simplement leur somme divisée par 4. Le résultat de cette méthode donne une image plus lisse que l'image originale. Le le temps de calcul est également moindre.

3-Interpolation bicubique :

En mathématiques, l'interpolation bicubique est une extension de l'interpolation cubique de points de données. En amélioration d'image, l'interpolation bicubique est généralement choisie plutôt que l'interpolation voisine la plus proche et bilinéaire lorsque la vitesse n'est pas prise en compte. Si auparavant, la méthode précédente utilisait uniquement les quatre pixels les plus proches, mais dans cette méthode, les 16 pixels les plus proches sont utilisés pour créer le pixel intermédiaire, ces 16 pixels connus sont à des distances différentes des pixels inconnus, donc les pixels les plus proches ont reçu plus importance pour les calculs. Cette méthode consomme plus de temps de calcul que les autres méthodes non adaptatives mais peut fournir des images plus détaillées et plus claires.

1.4 Techniques d'évaluation :

Les indices visuels ne suffisent pas pour évaluer et comparer les performances des différentes méthodes SR car ils sont de nature très subjective. Pour comparer équitablement les modèles, une mesure quantitative commune des performances SR est nécessaire, et le rapport signal/bruit de pointe (PSNR) et l'indice de similarité structurelle (SSIM) sont les deux mesures d'évaluation

les plus couramment utilisées pour évaluer les performances SR. Les métriques PSNR et SSIM sont souvent utilisées ensemble pour fournir une évaluation juste des méthodes par rapport à l'état de l'art. Discutons ensuite de ces métriques[16].

1.4.1 PSNR :

Le compatibility signal/bruit de crête ou PSNR est une métrique objective qui mesure la qualité de la recreation d'image d'une change avec perte. Mathématiquement, il est défini standard :

$$PNSR(I_{LR}, I_{HR}) = 10log_{10}(\frac{M^2}{MSE(I_{LR}, I_{HR})}) \tag{1.1}$$

Ici, "MSE" représente l'erreur quadratique moyenne au niveau des pixels entre les pictures, et "M" est la valeur maximale conceivable d'un pixel dans une picture (pour les pictures RVB 8 bits, nous sommes habitués à (M = 255). Furthermore la valeur du PSNR est élevée (en décibels/dB), meilleure est la qualité de la remaking.

1.4.2 SSIM :

L'indice de similarité structurelle (SSIM) est une mesure subjective qui détermine la cohérence structurelle entre deux images (les images LR et HR super-résolues dans ce cas).Cette métrique permet de comparer la qualité perceptuelle de deux images à l'aide de la formule ci-dessous,avec la moyenne μ écart σ,et la corrélation (c) des deux images :

$$SSIM(I_{LR}, I_{HR}) = \frac{(2\mu_{I_{LR}}\mu_{I_{HR}} + C1)(2\sigma_{I_{LR},I_{HR}} + C2)}{(\mu^2_{I_{LR}} + \mu^2_{I_{HR}} + C1)(\sigma^2_{I_{LR}} + \sigma^2_{I_{HR}} + C2)} \tag{1.2}$$

where,
$\mu_{I_{LR}}, \mu_{I_{HR}}$ = the means of I_{LR} and I_{HR}
$\sigma^2_{I_{LR}}, \sigma^2_{I_{HR}}$ = the variances of I_{LR} and I_{HR}
$\sigma_{I_{LR},I_{HR}}$ = the covariance of I_{LR} and I_{HR}
k_1 and k_2 are two variables to stabilize the division with weak denominators.

FIGURE 1.4 –

SSIM est compris entre 0 et 1, où une valeur plus élevée indique une plus grande cohérence structurelle et donc une meilleure capacité SR.

1.4.3 MOS :

Mean Opinion Score est un moyen manuel de déterminer les résultats d'un modèle, où les humains sont invités à évaluer une image entre 0 et 5. Les résultats sont agrégés et le résultat moyen est utilisé comme métrique.

1.5 Comparaison des méthodes l'interpolation :

Il a été prouvé que l'interpolation du plus proche voisin produit une qualité inférieure à celle des autres. Ainsi, cela crée une apparence en bloc et expose les irrégularités, ce qui entraîne une image de mauvaise qualité. Le résultat de l'interpolation binaire est meilleur que celui de la méthode précédente. Cette méthode rend l'image plus lisse que son voisin le plus proche. Cependant, comme le niveau de gris change au cours du processus, l'image peut encore être quelque peu rugueuse et entraîner un flou ou une perte de résolution. L'interpolation cubique est la méthode qui produit une image interpolée de la plus haute qualité. Elle est également très efficace et génère une meilleure image très proche de l'image originale.[18].

1.6 Limites :

Les méthodes basées sur l'interpolation ont souvent des effets secondaires, des résultats peu clairs et des difficultés telles que :

- Complexité informatique et amplification du bruit.
- Ils ne parviennent pas à éliminer de nombreuses erreurs et faits de stress dans les systèmes.
- L'interpolation peut fournir des images assez précises. Il semble flou et est facilement identifiable comme un faux.
- Les méthodes d'interpolation couramment utilisées pour accomplir la tâche, telles que l'interpolation cubique, ne fournissent aucune information supplémentaire pour résoudre le problème de reconstruction.

1.7 Apprentissage en profondeur pour super résolution d'image :

Les techniques d'apprentissage profond ont toujours été un facteur clé dans l'amélioration de la technologie de super-résolution en raison de leur capacité à extraire automatiquement des fonctionnalités.Des efforts récents se sont même concentrés sur la réduction du besoin d'images haute résolution comme vérité terrain pour la formation des réseaux neuronaux.Techniques de super-résolution basées sur un travail d'apprentissage profond utilisant des réseaux de neurones formés sur de grands ensembles de données d'images haute résolution.Ces réseaux apprennent à reconnaître les modèles et les structures sous-jacents de ces images, puis utilisent ces informations pour créer des modèles et des structures similaires dans les images basse résolution introduites dans le réseau.

1.8 Les models de super-résolution :

Il existe de nombreuses méthodes utilisées pour résoudre cette tâche. Nous couvrirons les éléments suivants :

- Super résolution de pré-échantillonnage.
- Super résolution post-suréchantillonnage.
- Modèles génératifs.
- Réseaux de reconstruction progressive.
- Réseaux résiduels.
- Réseaux résiduels multi-étages.
- Réseaux récursifs.
- Réseaux multi-succursales.
- Réseaux basés sur l'attention [19].

1.8.1 Super résolution de pré-échantillonnage :

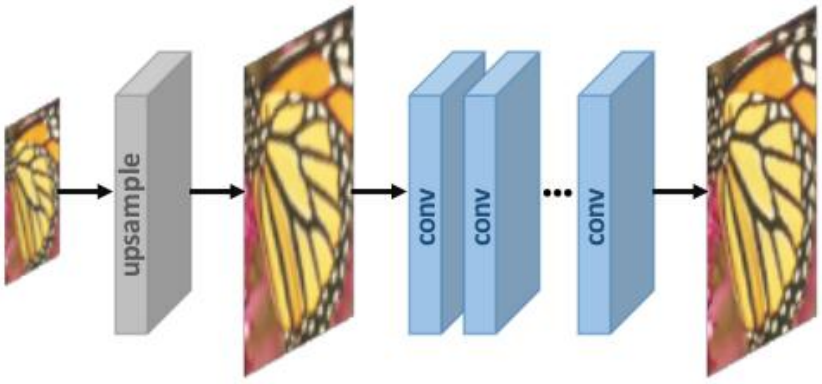


FIGURE 1.5 – Super résolution de pré-échantillonnage [2]

Car cela est considéré comme difficile, nous ne mappageons pas directement les images LR aux images HR. Une solution simple est d'utiliser des algorithmes de suréchantillonnage traditionnels pour obtenir des images à plus haute résolution, puis d'utiliser des réseaux de neurones profonds pour les affiner. Par exemple, une interpolation bicubique est utilisée pour suréchantillonner les images LR en images HR grossières avec la taille souhaitée. Des CNN profonds sont ensuite appliqués à ces images pour créer des images de haute qualité.

• **SRCNN :**

La première méthode d'apprentissage profond pour relever le défi de la RS est SRCNN. Pour fonctionner avec une grande précision, ils se sont fortement appuyés dans leur modélisation CNN sur les méthodes SR traditionnelles basées sur un codage clairsemé qui était bien développé à l'époque. Dans la figure 1.6, une comparaison entre SRCNN et une méthode basée sur un codage clairsemé peut être vue. Basé sur des méthodes de codage clairsemées, CNN était un modèle assez simple qui reposait sur trois étapes pour passer d'une image basse résolution à une image haute résolution. Avant que l'une de ces trois étapes ne soit franchie, l'image basse résolution a d'abord été mise à l'échelle par un algorithme cubique. Puisqu'il s'agit d'une étape de pré-traitement, elle n'est pas incluse dans les trois étapes de l'étape de formation CNN mentionnées ci-dessous :

1- La première des trois étapes est l'extraction et la représentation correctes. Cela se fait par une couche convolutive qui mappe les correctifs de l'image dans un vecteur de la taille des cartes de caractéristiques.

2- Deuxièmement, ils effectuent une cartographie non linéaire, qui utilise une couche convolutive pour mapper chacun des vecteurs de carte de caractéristiques préexistants en vecteurs représentant les patchs haute résolution de l'image.

3- Enfin, ils ont reconstruit l'image en prenant ces corrections de mouillage haute résolution et en en faisant la moyenne, puisque les corrections sont . Ce qui signifie que le modèle est optimisé pour les scores PSNR.

Il est rapporté que SRCNN peut gérer des facteurs d'échelle de 2, 3 et 4. Pour chaque travailleur, un réseau distinct est formé avec la même structure mais des variables différentes.

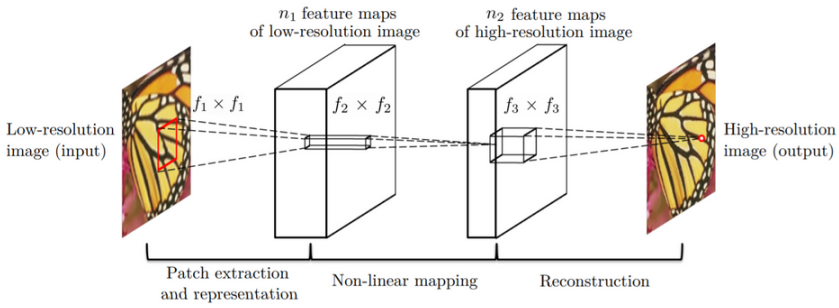


FIGURE 1.6 – Architecture du réseau de neurones convolutif à super résolution (SRCNN).

1.8.2 Super-résolution post-suréchantillonnage :

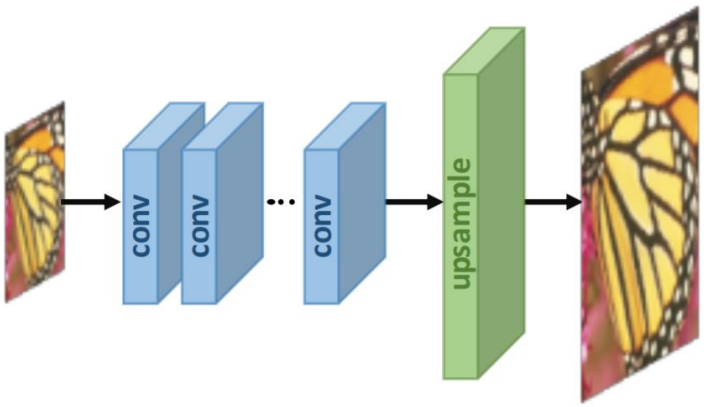


FIGURE 1.7 – Super-résolution post-suréchantillonnage[3]

Notre processus d'extraction de fonctionnalités dans le prééchantillonnage SR se déroule sur un espace de résolution plus élevée et nécessite également une puissance de calcul plus élevée. Le post-rééchantillonnage SR tente de résoudre ce problème en effectuant une extraction de fonctionnalités dans l'espace de résolution inférieure, puis en suréchantillonnant uniquement à la fin, ce qui réduit considérablement la dépense. De plus, pour regrouper les échantillons, un processus d'approximation appris par déconvolution/convolution sous-pixel est utilisé, au lieu d'utiliser une simple interpolation cubique, ce qui rend le réseau entraînable de bout en bout.

Nous discuterons de quelques techniques courantes qui suivent cette structure :

• **FSRCNN :** Comme on peut le voir dans la figure ci-dessus, les principaux changements entre SRCNN et FSRCNN sont : Au début, aucun prétraitement ou suréchantillonnage n'est

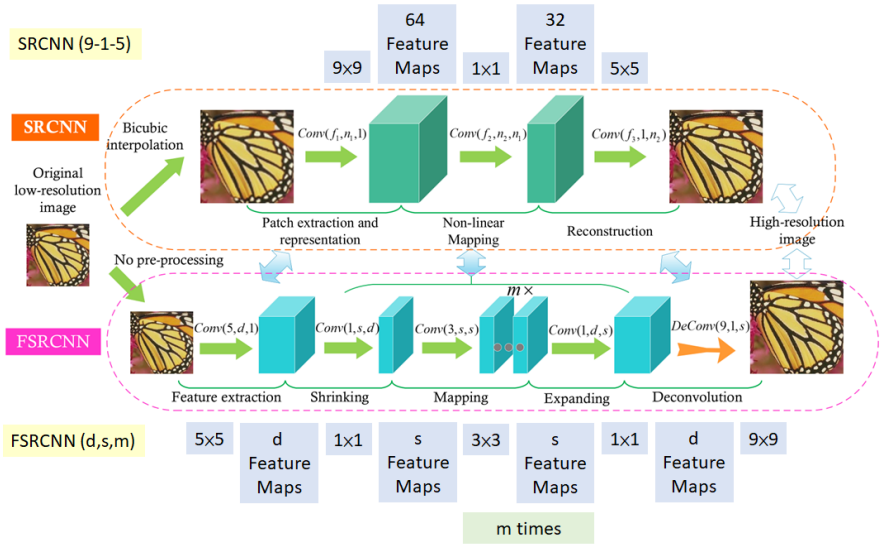


FIGURE 1.8 – Comparaison entre SRCNN et FSRCNN[4]

effectué. Les caractéristiques ont été extraites dans l'espace à basse résolution. De la même manière que le réseau Inception est développé, une convolution 1×1 est utilisée après la convolution initiale 5×5 pour réduire le nombre de canaux, ce qui nécessite moins de calcul et de mémoire. Au lieu d'avoir un grand filtre convolutif, plusieurs convolutions 3×3 sont utilisées, similaire au fonctionnement du réseau VGG en simplifiant l'architecture pour réduire le nombre de paramètres. Le modèle est amélioré en utilisant un filtre déconvolutif appris pour effectuer un suréchantillonnage. Finalement, FSRCNN surpasse SRCNN.

1.8.3 Modèles génératifs :

Jusqu'à présent, les réseaux discutés optimisent la différence de pixels entre les images HR prédites et réelles. Bien que cette métrique fonctionne bien, elle n'est pas idéale ; les gens distinguent les images par la qualité perceptuelle plutôt que par la différence de pixel. Les modèles génératifs (ou GAN) cherchent à optimiser la qualité perceptuelle pour produire des images agréables à l'œil humain. Finalement, regardons quelques architectures liées au GaN.

• **SRGAN :**

GAN apparaît pour la première fois SRGAN était un exemple de super résolution. Un GAN est généralement composé d'un réseau de générateurs et d'un réseau discriminateur. Les images de haute résolution générées et non générées sont distinguées par le discriminateur. La Le but du

générateur est de produire des images qui trompent le discriminateur, ce qui rend Il pense que ce sont des images de haute résolution. Les deux architectures ont été développées en utilisant divers modèles GAN provenant d'autres domaines liés à l'image, étant le premier GAN super résolution. Plus important encore, le réseau de générateurs est composé de nombreux blocs résiduels disposés de manière spécifique, comme suggéré par Gross et WWilbe.Cela génère un réseau extrêmement profond qui est capable de produire beaucoup de détails.La différence avec l'image d'origine n'est presque pas visible à moins de zoomer. Le réseau criminator est un modèle CNN à anticipation assez simple qui utilise une fonction sigmoïde comme couche finale pour générer une valeur entre 0 (image SR) et 1 (image HR). Un changement significatif Selon Leidig et al. Quelle était la raison pour laquelle ils se sont détournés des précédents Étant donné que MSE favorise des scores PSNR élevés, le travail n'utilisait plus MSE comme fonction de perte. Il n'y a pas nécessairement de bons scores perceptifs. Ils ont créé une fonction de perte qui se compose de deux parties : la perte de contenu et la perte contradictoire afin d'obtenir des scores perceptifs plus élevés. La perte de contenu, également appelée perte de contenu VGG car elle est basée sur le réseau VGG, est définie comme la distance euclidienne entre les caractéristiques de l'image de référence et de l'image reconstruite. Le rival Les images d'apparence plus naturelle sont favorisées par la perte en fonction des probabilités que le discriminateur a renvoyées pendant.

La perte de MSE capture la similitude des pixels Perte de similarité perceptuelle, qui est utilisée pour collecter des données de haut niveau à l'aide d'un réseau profond de perte contradictoire du discriminateur Le modèle an obtenu plus de MOS, ce qui signifie une meilleure qualité perceptuelle dans les résultats, malgré des valeurs de PSNR comparativement plus faibles.

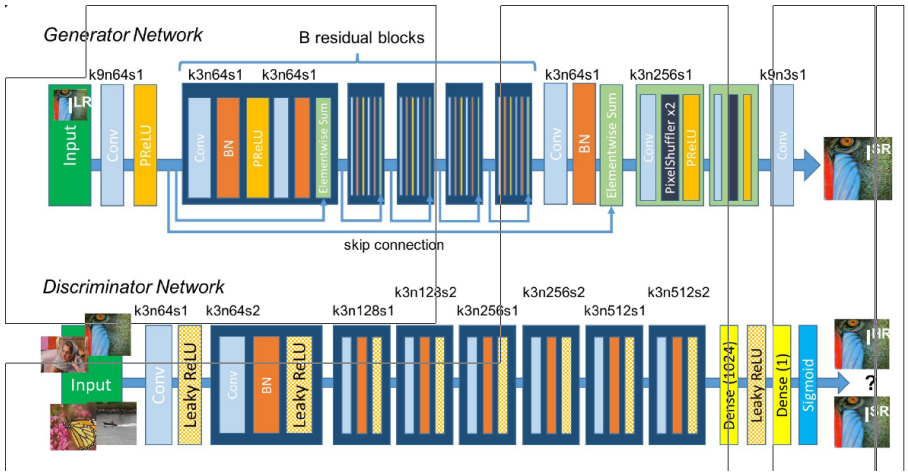


FIGURE 1.9 – Architecture réseau de SRGAN[5]

- **ESRGAN :**

Grâce à ESRGAN, une mise en évidence relative a été ajoutée pour améliorer SRGAN. L'avantage est que le réseau est entraîné non seulement à identifier l'image correcte ou fausse, mais également à rendre les images réelles moins réalistes par rapport aux images générées, ce qui contribue à tromper le discriminateur. De plus, la normalisation par lots est supprimée dans SRGAN et des blocs denses (inspirés de DenseNet) sont utilisés pour améliorer le flux d'informations. Ce type de bloc dense est connu sous le nom de RRDB.

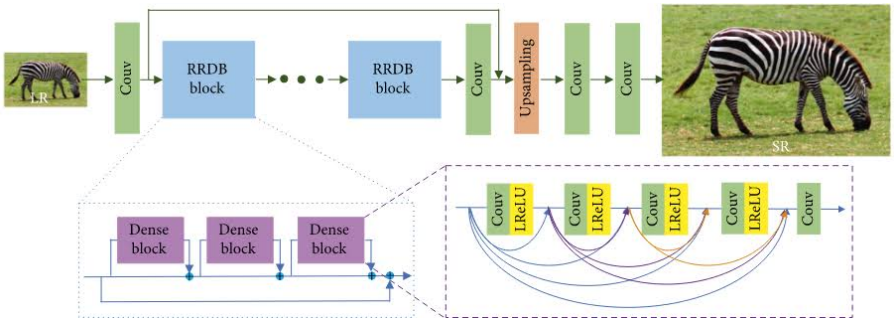


FIGURE 1.10 – Architecture réseau de ESRGAN

1.8.4 Suréchantillonnage progressif :

Nous avons constaté que les méthodes de pré- et post-échantillonnage sont utiles. Cependant, il est confronté à des problèmes qui nécessitent de redimensionner les images LR par des facteurs importants (par exemple 8x), et que le suréchantillonnage soit effectué avant ou après la traversée du réseau SR profond, les résultats peuvent être sous-optimaux. Dans de tels cas, il est plus logique d'agrandir progressivement l'image LR pour éventuellement répondre aux critères de dimension spatiale de la sortie HR plutôt que de la multiplier par 8 d'un coup. Les méthodes qui utilisent cette stratégie d'apprentissage sont appelées méthodes de suréchantillonnage progressif.L'architecture LapSRN (Laplacian Pyramid Super-Resolution Network) est l'un de ces modèles qui reconstruit progressivement les résidus de sous-bande des images HR.Les différences entre l'image suréchantillonnée et l'image HR de vérité terrain au niveau respectif du réseau sont appelées résidus de sous-bande.La présentation ci-dessous présente l'architecture réseau de LapSRN comparée à d'autres architectures conventionnelles.

LapSrn utilise une chaîne CNN. LapSRN prédit progressivement les résidus de sous-bandes grossiers à fins en prenant une image LR en entrée. Pour extraire les différentes cartes de caractéristiques à chaque niveau, LAPSRN applique d'abord une série de couches convolutives. Les

cartes de caractéristiques sont ensuite agrégées à un niveau plus fin à l'aide d'une couche convolutive portée. Enfin, les résidus du sous-domaine sont prédits à l'aide d'une couche convolutive. Des processus de suréchantillonnage et d'addition sont utilisés pour reconstruire efficacement l'image des RH à tous les niveaux. Dans une reconstruction pas à pas basée sur la hiérarchie de Laplace, LapSRN génère plusieurs prédictions SR moyennes.

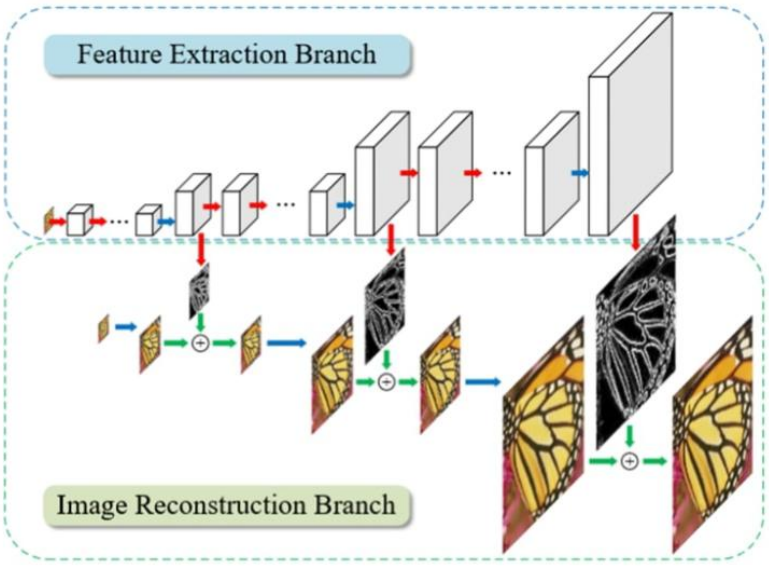


FIGURE 1.11 – l'architecture LapSRN(Laplacian Pyramid Super-Resolution Network)[6].

1.8.5 Réseaux résiduels :

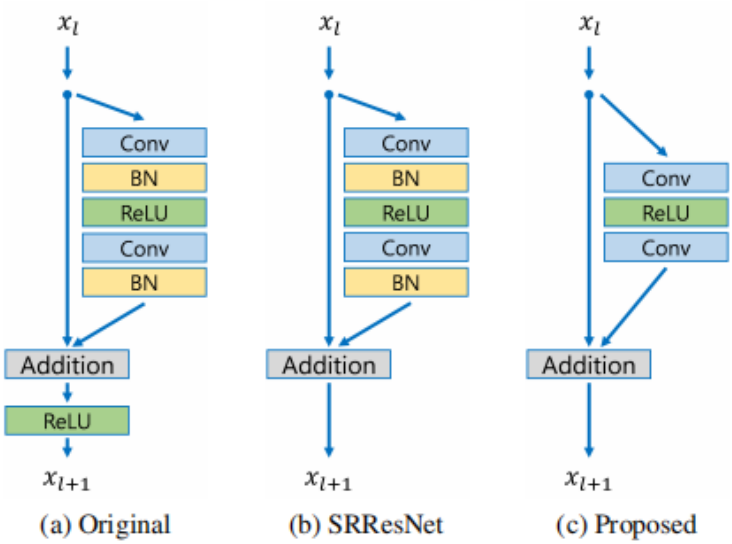


FIGURE 1.12 – Comparaison des blocs résiduels dans l'original ResNet, SRResNet et le nôtre.

L'architecture SRResNet est basée sur plusieurs blocs résiduels. La figure ci-dessus montre le bloc résiduel dans EDSR. La suppression des couches de normalisation par lots est la principale différence avec SRResNet. Selon l'auteur, BN normalise l'entrée, limitant ainsi la portée du réseau ; la suppression de BN améliore la précision. De plus, les couches BN consomment de la mémoire, et leur suppression réduit la mémoire jusqu'à 40 %, ce qui rend la formation du réseau plus efficace.

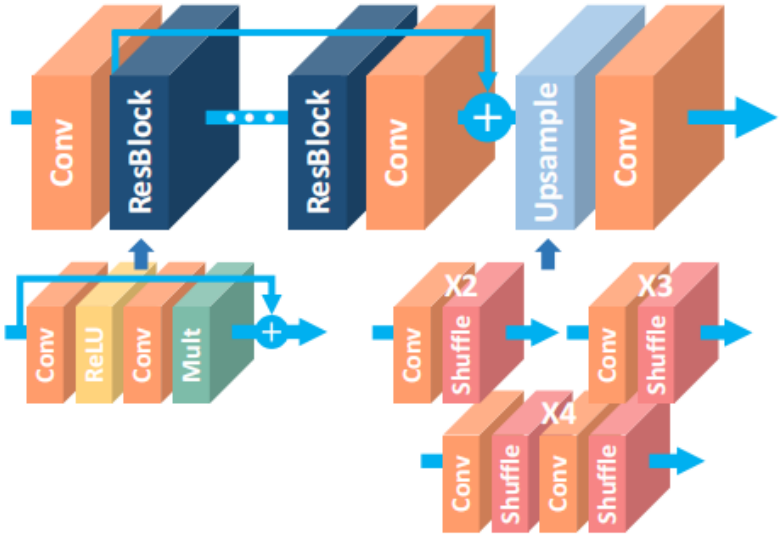


FIGURE 1.13 – L'architecture EDSR[7].

1.8.6 Réseaux résiduels multi-étages :

Quelques architectures envisagent une conception à plusieurs étages pour améliorer leurs performances en traitant la tâche d'extraction de caractéristiques séparément dans l'espace basse résolution et l'espace haute résolution.Les caractéristiques grossières sont prévues par la première étape et améliorées par la dernière.Discutons de l'architecture de l'un de ces réseaux multiétages.

• **BTSRN :**

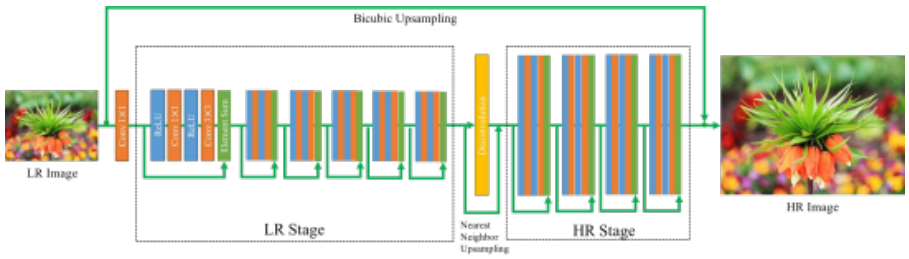
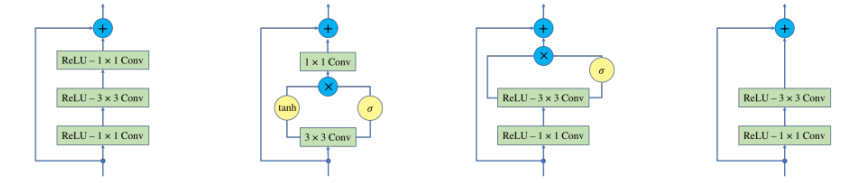


FIGURE 1.14 – Architecture du réseau proposé.



(a) Residual block in PixelCNN (b) Gated conv in PixelCNN (c) Gated conv in PixelCNN++ (d) PConv block in proposed BTSRN model

FIGURE 1.15 – Comparaison de différentes conceptions de blocs résiduels.

Comme on peut le voir sur la figure1.14, BTSRN se compose de deux étages :

- un étage basse résolution (LR).
- un étage haute résolution (HR).

Le stade LR se compose de 6 blocs résiduels, tandis que le stade HR contient 4 blocs résiduels. La convolution dans l'étape HR nécessite plus de calculs que dans l'étape LR, car la taille d'entrée est plus élevée. Le nombre de blocs dans les deux étapes est déterminé de manière à réaliser un compromis entre précision et performance. Avant d'être envoyée à l'étage HR, la sortie de l'étage LR est suréchantillonnée. Cela peut être accompli en ajoutant les sorties de la couche de convolution et l'échantillon usps du voisin le plus proche. Comme indiqué en (d) dans la figure précédente, les auteurs proposent un nouveau bloc résiduel appelé Pconv. Sur la base des résultats, le composant proposé réalise un bon compromis entre précision et efficacité. Comme pour EDSR, la normalisation par lots est évitée pour empêcher le recentrage et la remise à l'échelle. Cela est dû au fait que la super-résolution est une tâche de régression, ce qui signifie que les sorties cibles sont fortement corrélées aux statistiques d'entrée de premier ordre.[19].

1.8.7 Réseaux récursifs :

Les réseaux récursifs utilisent des paramètres de réseau partagés dans des couches convolutives pour réduire leur empreinte mémoire, comme indiqué dans la figure1.16. Discutons de certaines structures qui impliquent des unités répétitives.

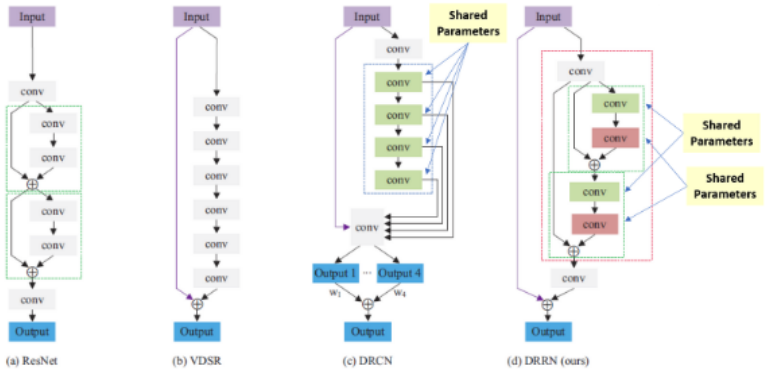


FIGURE 1.16 – des paramètres de réseau partagés dans des couches convolutives.

• **DRCN :**

Le Deep Recursive Convolutional Network (DRCN) utilise la même couche de convolution à plusieurs reprises. Les couches convolutives du bloc résiduel sont partagées, comme le montre la figure 1.17.

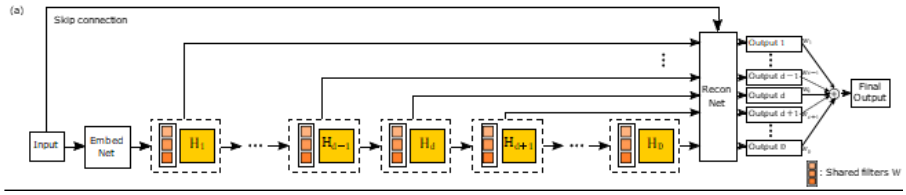


FIGURE 1.17 – Notre modèle final (avancé) avec supervision récursive et saut de connexion.

Les sorties et les entrées de tous les blocs convolutionnels partagés intermédiaires sont envoyées à la couche de reconstruction, qui utilise toutes les entrées pour créer une image haute résolution. Cette architecture peut être considérée comme un ensemble de réseaux car elle utilise plusieurs entrées pour générer la sortie.

1.8.8 Réseaux multi-agences :

Une tendance a maintenant été observée : les réseaux plus profonds produisent de meilleurs résultats. Mais en raison du problème de flux d'informations, la formation de réseaux plus profonds est difficile. Les réseaux résiduels résolvent ce problème en utilisant des connexions de raccourci. En ayant plusieurs branches à travers lesquelles l'information peut passer, les réseaux multi-branches améliorent le flux d'informations, ce qui entraîne la fusion d'informations pro-

venant de plusieurs champs réceptifs et donc une meilleure formation. Discutons des réseaux qui utilisent cette méthode.

• **CMSC :**

Le réseau croisé multi-échelle en cascade (CMSC) possède une couche d'extraction de caractéristiques, des sous-réseaux en cascade et une couche de reconstruction, comme d'autres cadres de super-résolution, comme illustré ci-dessous :

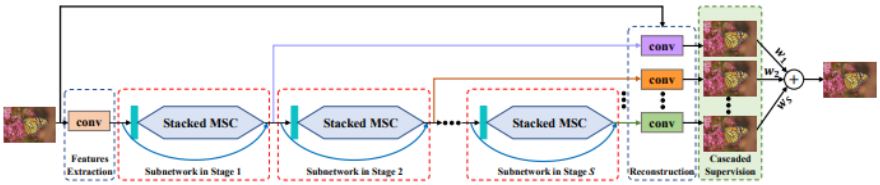


FIGURE 1.18 – (a)L'architecture du réseau CMSC proposé.

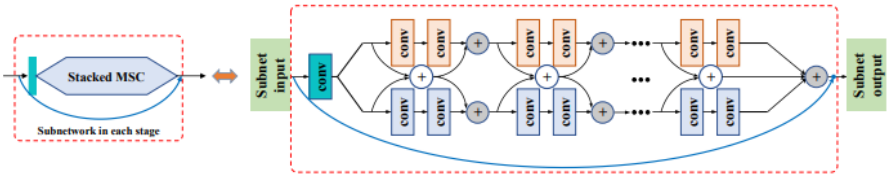


FIGURE 1.19 – (b)L'architecture de chaque sous-réseau en cascade dans le réseau CMSC.

Le sous-réseau en cascade se compose de deux branches, comme on peut le voir en (b).En raison des tailles de filtres différentes de chaque branche, chaque champ récepteur est unique. La fusion d'informations de différents champs réceptifs à travers le module améliore le flux d'informations.Plusieurs blocs MSC sont empilés les uns sur les autres de manière itérative pour réduire progressivement la différence entre la sortie et l'image HR.Pour obtenir la sortie HR finale, les sorties de tous les blocs sont transférées ensemble à un bloc de reconstruction.

1.8.9 Réseaux basés sur l'attention :

Jusqu'à présent, tous les emplacements spatiaux et tous les canaux sont considérés de manière équivalente par les réseaux discutés. En général, accorder une attention sélective à différentes parties d'une image peut donner de bien meilleurs résultats. Maintenant, nous allons parler de quelques architectures qui aident à atteindre cet objectif.

• **SelNet :**

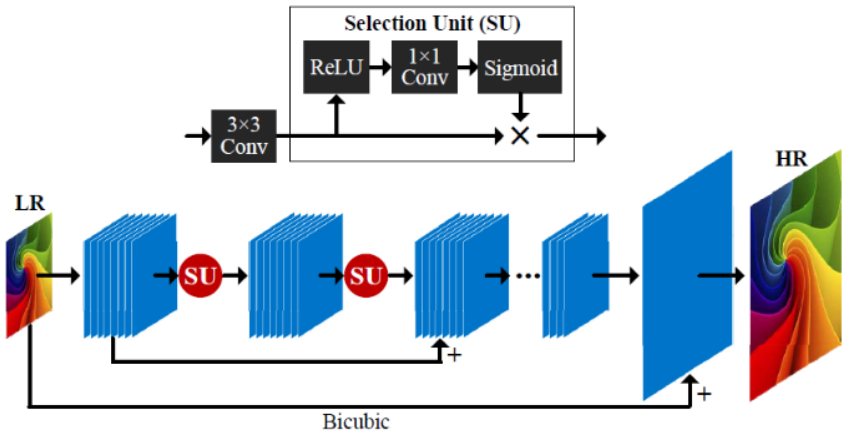


FIGURE 1.20 – structure SelNet [8]

propose une nouvelle unité de sélection à la fin des blocs convolutifs qui aide à sélectionner les informations à transmettre. Une activation ReLu, une convolution 1×1 et un déclenchement sigmoïde constituent le module de sélection. Une Unité de choix est la multiplication d'un module de choix et d'une connexion d'identité.
Pour une mise à l'échelle apprise, une couche de sous-pixel (semblable à ESPCN) est conservée vers la fin du réseau. Le réseau apprend une image HR résiduelle, qui est ensuite ajoutée à l'entrée interpolée pour obtenir l'image HR finale.

1.9 Travaux similaires :

Nous mentionnerons certains des travaux précédents de super résolution et verrons ce que vous obtenez.

1.10 Apprentissage profond pour la super résolution des images :

Ils ont utilisé et formé des keras pour créer un réseau de neurones convolutifs décisionnels (SRCNN).Avec cette technologie proposée, les images de test à basse résolution ont été considérablement améliorées et ils ont obtenu de bons résultats sur les données utilisées[20].

1.11 Image Super-Resolution Using Deep Convolutional Networks :

Ils ont présenté une nouvelle technique d'apprentissage profond pour une seule image à super-résolution (SR). Un réseau de neurones à convolution profonde peut utiliser les techniques SR conventionnelles basées sur un codage parcimonieux. SRCNN est un apprentissage de bout en bout cartographique entre les images basses et hautes résolutions, avec un intervalle Peu de traitement pré et post optimisation. Grâce aux méthodes les plus récentes, le SRCNN an une structure légère et des performances supérieures.Ils pensaient que l'exploration de plus de filtres et de stratégies de formation différentes pourrait être plus efficace.De plus, la structure proposée, ainsi que son annonce Pour d'autres problèmes de vision de bas niveau, tels que la distorsion de l'image floue ou le débruitage SR+ simultané, les avantages de la simplicité et de la durabilité peuvent être utilisés. Enquêter sur un réseau pour gérer la mise à niveau à différents niveaux de facteurs est une autre option[21].

1.12 Enhanced Super-Resolution Generative Adversarial Networks :

Le modèle ESRGAN, qu'ils ont introduit, permet d'obtenir une qualité cognitive constamment meilleure que les méthodes SR précédentes.En termes d'indice perceptuel, la méthode a remporté la première place dans le défi PIRM-SR. Ils ont créé une nouvelle architecture qui comprend plusieurs blocs RDDB sans couches BN. De plus, la formation du modèle profond proposé est facilitée par des techniques utiles telles que la mise à l'échelle résiduelle et une configuration plus petite.Nous avons également introduit l'utilisation du GAN relatif comme discriminateur, qui apprend à juger si une image est plus réaliste que l'autre en dirigeant le générateur pour récupérer des matériaux plus détaillés. De plus, nous avons amélioré la perception sensorielle Perte d'utilisation des fonctionnalités avant l'activation, ce qui permet une surveillance plus forte et restaure une luminosité plus précise et des textures réalistes[22].

1.13 plate-forme de mise à l'échelle d'images haute résolution alimentée par ESRGAN :

L'application basée sur MLOps est conçue pour mettre à l'échelle les images par un facteur de 4 à l'aide d'ESRGAN, une technique basée sur l'apprentissage en profondeur pour la super-résolution d'image. L'application est hébergée sur les services Web de Render et est implémentée en tant qu'API basée sur Flask. Les utilisateurs peuvent télécharger leurs images basse résolution sur l'API, et l'application utilisera ESRGAN pour les mettre à l'échelle jusqu'à quatre fois leur résolution d'origine. L'API est optimisée pour l'évolutivité et peut gérer simultanément de gros volumes de demandes d'images. Avec cette application, les utilisateurs peuvent facilement améliorer la qualité de leurs images et produire des versions haute résolution pour leurs diverses utilisations [22].

1.14 Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network :

Ils décrivent le SRResNet, un réseau résiduel profond qui remet en question les normes publiques. Lorsque les ensembles de données sont évalués à l'aide d'une mesure largement utilisée du PSNR Ils ont mis en évidence certaines limites de la super-résolution d'image utilisée PSNR et ont introduit SRGAN, qui augmente la fonction de perte de contenu avec une perte contradictoire en entraînant le GAN. Confirmez que les reconstructions SRGAN sont significatives en utilisant des tests étendus inexistants. Les facteurs d'amélioration ($4\times$) sont généralement considérés comme une reconstruction plus photoréaliste obtenue à l'aide d'un ensemble de méthodes de référence de l'art[9].

1.15 Fast and Accurate Image Super-Resolution with Deep Laplacian Pyramid Networks :

Un réseau convolutionnel profond a été proposé dans ce travail. Pour une résolution d'image plus rapide et plus précise, créez une pyramide La Placea. Dans les cas où le modèle évolue progressivement, les résidus à haute fréquence sont prédits de manière inexacte, sans tenir compte de la forte perte de Charbonnier Mission. Ils ont utilisé 73 % de paramètres en moins

que notre méthode d'analyse comparative précédente et ont obtenu des performances améliorées en partageant les paramètres à travers et au sein des niveaux hiérarchiques.Pour former des modèles plus profonds, ils ont incorporé des connexions de saut locales dans le réseau.De plus, pour former un modèle à traiter, ils ont utilisé une stratégie de formation à plusieurs échelles. Une variété d'échelles de sténographie. Proposez un dossier complet. Évaluez diverses options de conception et réfléchissez à elles. Une analyse complète est bénéfique pour la société. Il a démontré le LapSRN proposé dans le contexte d'une image haute résolution. Cependant, la conception du réseau est couramment utilisée et peut être Il peut être utilisé pour d'autres conversions d'images et réglages[6].

1.16 Image Super-Resolution Using Very Deep Residual Channel Attention Networks :

Pour les images SR précises, ils proposent des réseaux d'attention de canal résiduel très profonds (RCAN). La structure résiduelle dans résiduelle (RIR) permet à RCAN d'atteindre une profondeur extrêmement élevée en collaboration avec LSC et SSC. Le RIR permet de contourner de nombreuses informations basse fréquence via plusieurs connexions de saut, permettant au réseau principal de se concentrer sur l'apprentissage des informations haute fréquence.
En outre, ils proposent un routage d'attention (CA) pour redimensionner de manière adaptative les fonctionnalités de chaque canal en tenant compte de l'interconnectivité inter-canal afin d'améliorer la capacité du réseau. L'efficacité du RCAN proposé est démontrée par des expériences approfondies sur SR avec les modèles BI et BD. RCAN montre également des résultats prometteurs pour la reconnaissance d'objets[23].

1.17 Résultats de certains modèles :

Nous comparons et voyons certains des résultats des modèles sur plusieurs ensembles de données de référence publics .SRCNN[24], EDSR[25] et RCAN[23] ,et également avec des approches axées sur la perception, notamment SRGAN[26], EnhanceNet[27]et ESRGAN[11].

1.17.1 BSD100 (cinq échantillons aléatoires) Résultats visuels :

FIGURE 1.21 – Résultats pour cinq échantillons aléatoires de BSD100 en utilisant l'interpolation bicubique, SRResNet et SRGAN (4× mise à l'échelle)[9].

1.17.2 Set14 Visual Results :

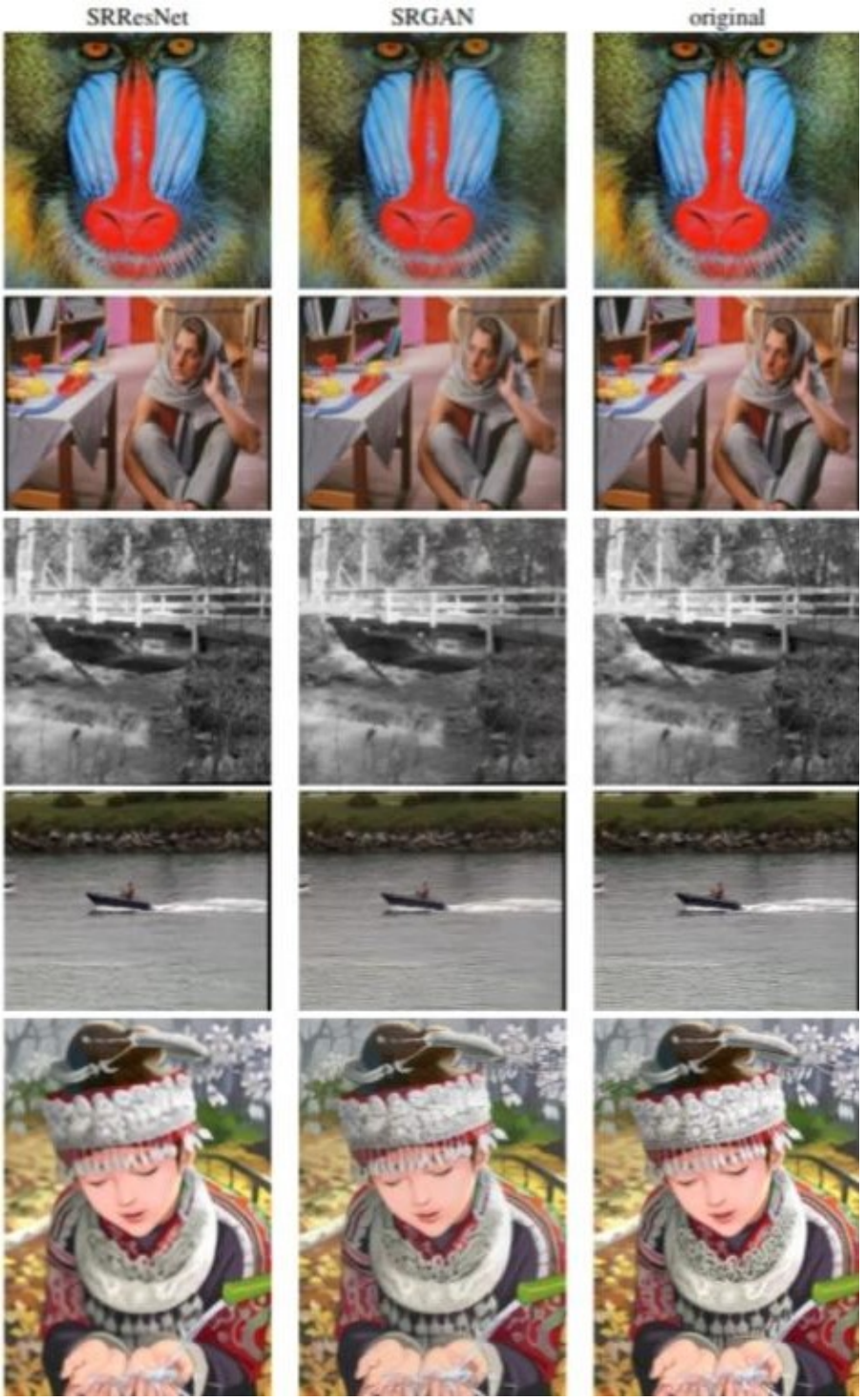


FIGURE 1.22 – Résultats pour cinq échantillons aléatoires de BSD100 en utilisant SRResNet et SRGAN.(4× mise à l'échelle).[9]

En général,plus susceptibles d'introduire des artifacts désagréables, lorsque le réseau est plus profond et plus complexe.La figure2.3 et la figure2.4 . montre que SRGAN produit des artifacts qui affectent la qualité de l'image dans les réseaux de neureaux .ils étaient par conséquent trop

profonds, ce qui entraînait de grands réseaux lents[2]

FIGURE 1.23 – Artifacts-generated-by-the-SRGAN-deep-neural-network[10].

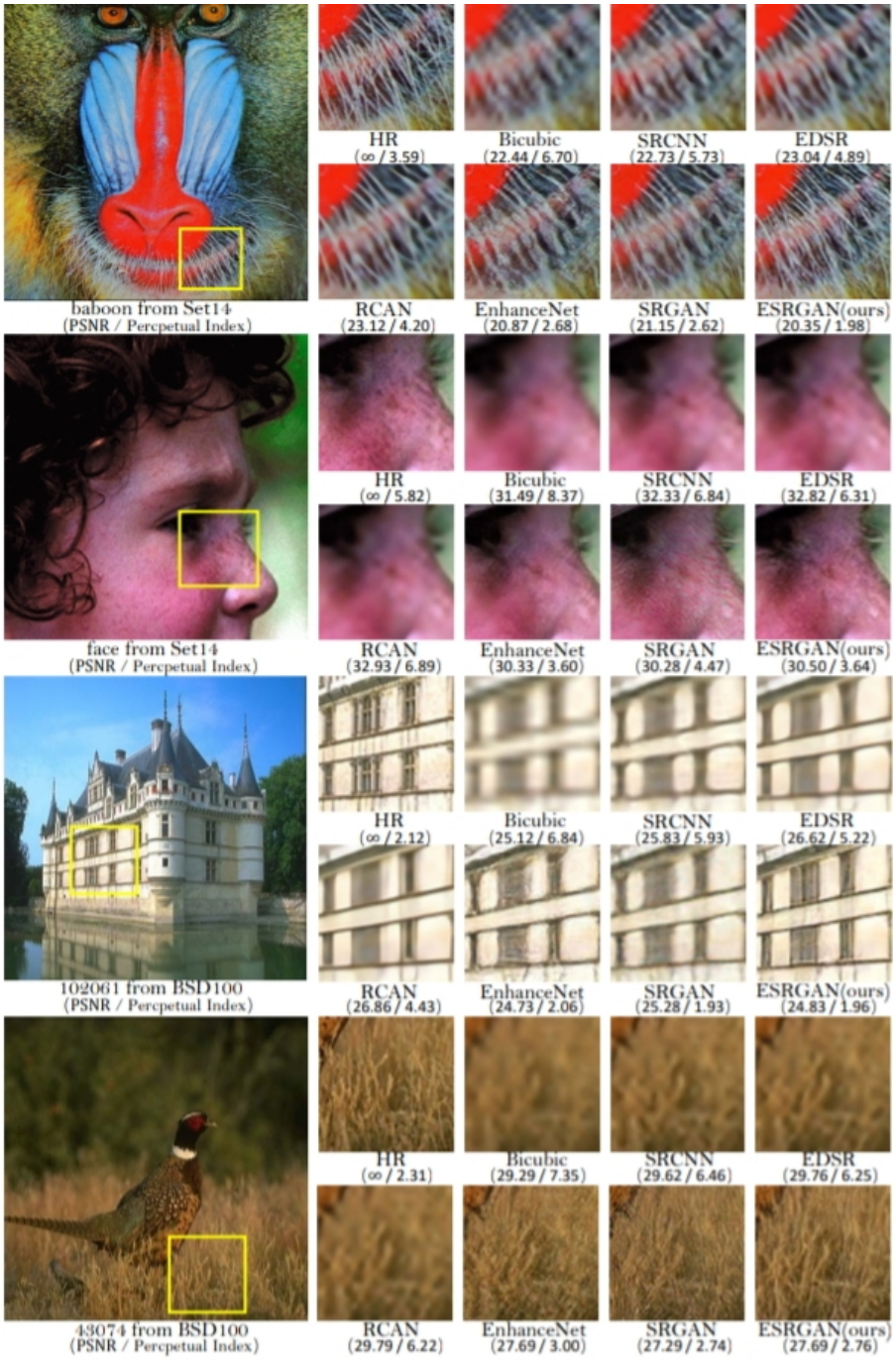


FIGURE 1.24 – artefacts désagréables par SRGAN(Fourrure animale, structure du bâtiment,texture de l'herbe,et le visage).[11]

1.18 conclusion :

Dans ce chapitre, nous avons introduit des concepts importants liés à la super-résolution, et nous avons également parlé de l'intégration de l'apprentissage profond dans les images et des méthodes de super-résolution utilisées pour résoudre cette tâche. En général, les techniques de super-résolution basées sur l'apprentissage profond ont montré des améliorations significatives. en résolution et en qualité d'image.

CHAPITRE 2

CONCEPTION ESRGAN

2.1 Introduction :

Dans le chapitre précédent, nous avons présenté premièrement les défis rencontrés lors de la lots de l'écriture manuscrite cursive. Ensuite, la différence entre la structure physique et celle logique d'un document, puis, nous avons vu le principe et les différentes approches et stratégies de lots. Finalement, nous avons exposé notre première contribution qui se résume en une technique de lots multi-scans.Dans ce chapitre, nous allons voir dans un premier temps quelques méthodes de classification utilisées dans la reconnaissance du manuscrit, parmi lesquelles, on trouve la méthode des machines à vecteurs de support qui a été utilisé dans notre travail. Souvent, la phase de classification est suivie par une phase de post-traitement afin d'améliorer les résultats de classification en utilisant des informations de plus niveaux (syntaxique,sémantique,. . . etc.) non disponible durant la phase de classification. Pour cela, nous allons présenter quelques méthodes de sélection en post-traitement. Enfin, nous allons voir notre deuxième contribution : la technique de sélection à base du puzzle, qui permet de résoudre le problème de coupures et chevauchements dans les lignes des segments des caractères. Principalement, Le système utilise trois phases de traitement, à savoir : création des sous-segments d'images, l'agrandissement de chaque sous-segment d'images et la reconstruction ou le ré-assemblage des segments agrandis. Le schéma suivant illustre en les différentes phases (voir la figure 2.1).

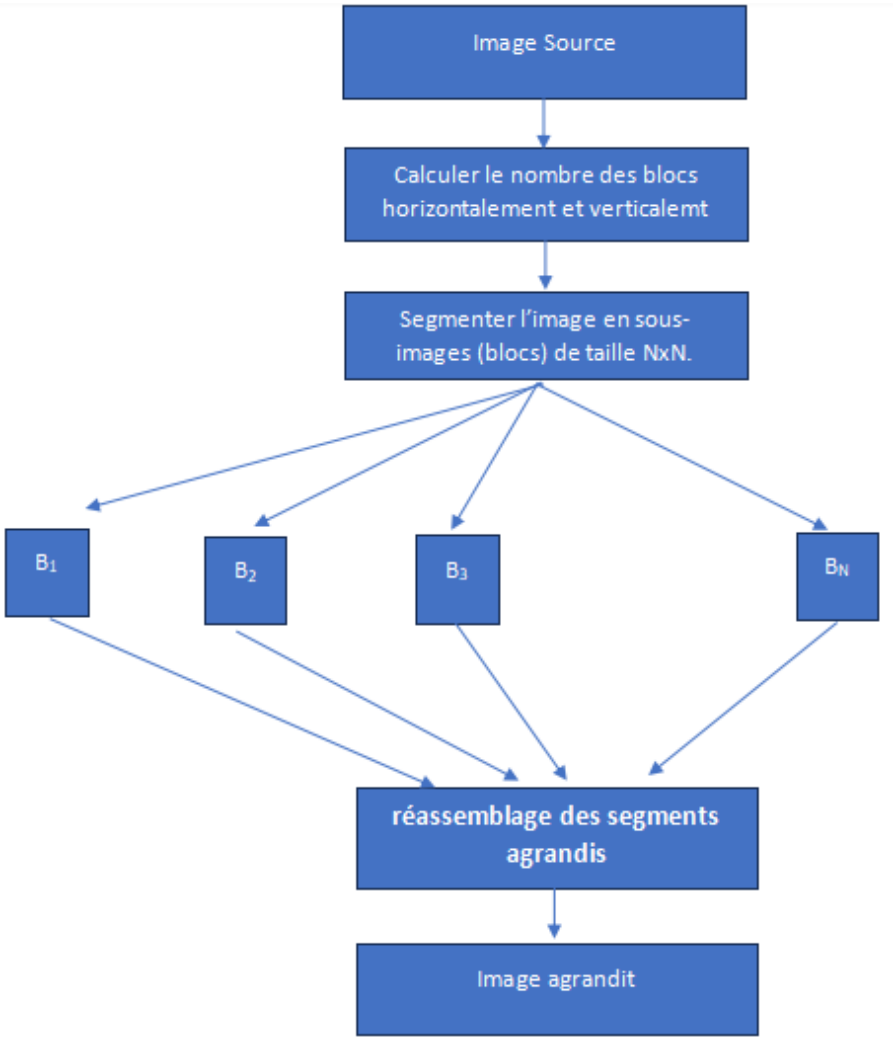


FIGURE 2.1 – Le schéma couches de suréchantillonnage.

2.2 Génération des sous-images :

Dans cette phase, le système subdivise l'image source en un ensemble de sous-images de tailles similaire. par la suite chaque sous-image est agrandit séparément. La figure 2.2 illustre un exemple de génération des sous-images à partir d'une image source.

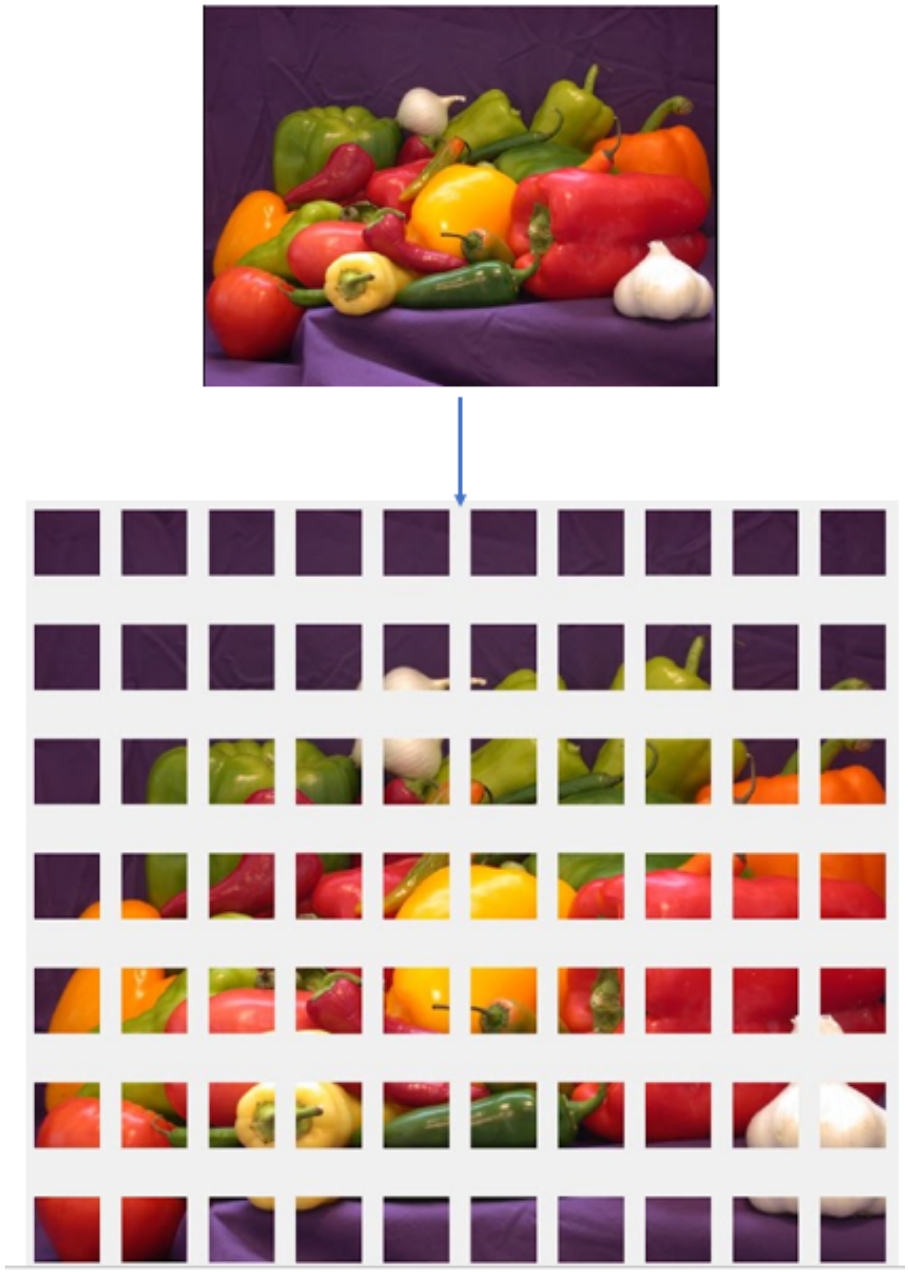


FIGURE 2.2 – Exemple de subdivision de l'image source en sous-images.

2.3 Agrandissement des sous-images :

2.3.1 Couche d'entrée :

Reçoit l'image d'origine.

2.3.2 Le résiduel dans résiduel dense Block (RRDB) :

C'est la première grande modification de l'architecture du générateur G. C'est une amélioration de l'architecture ResNet.La figure (3.2) montre que ESRGAN est en fait basé sur cette

architecture ou sur des types de réseaux résiduels similaires. Il allie densité et réseau résiduel multi-niveaux pour rendre l'apprentissage plus facile et plus efficace. Les couches de convolution et les fonctions d'activation composent le bloc restant. comme indiqué dans la figure(3.3). Un "raccord" (ou une connexion ignorée qui contourne une ou plusieurs couches d'un bloc).Un raccourci ajoute l'entrée d'origine à la sortie du bloc, permettant au réseau d'identifier facilement les différences (résiduelles) entre les images de bas niveau et les images haute résolution souhaitées. Parce que les gradients deviennent très faibles lors de la rétro-propagation et rendent la formation difficile pour les réseaux très profonds, cette action permet au réseau d'éviter le problème du déplacement des gradients. De plus, les blocs résiduels facilitent la formation de réseaux beaucoup plus complexes sans sacrifier les performances. parce qu'il peut capturer plus d'images et réduire le bruit gênant. utilisé pour renforcer la capacité de création de détails. Comme le montre la figure (3.4), restauration précise des tissus dans les images et augmentation de la capacité du réseau. Toutes les couches de normalisation par lots (BN) ont été supprimées dans la deuxième modification :Il a été observé que la suppression des couches dans de nombreuses architectures de réseau augmente les performances, économise les ressources informatiques et réduit la complexité des calculs et l'utilisation de la mémoire.De plus, lorsque le réseau est plus profond et plus complexe, les modèles avec des couches BN sont plus susceptibles d'introduire des artifacts désagréables.Après avoir supprimé le lot de couche de normalisation, ESRGAN décompose les résidus en multipliant une constante entre 0 et 1 avant de les ajouter au chemin principal pour réduire les résidus, éviter l'instabilité du réseau et empêcher leur apparition. Le paramètre de mise à l'échelle des résidus est constant.De plus, une initialisation plus petite est utilisée, ce qui facilite la formation de la structure résiduelle lorsque la variance des paramètres d'initialisation diminue[10].

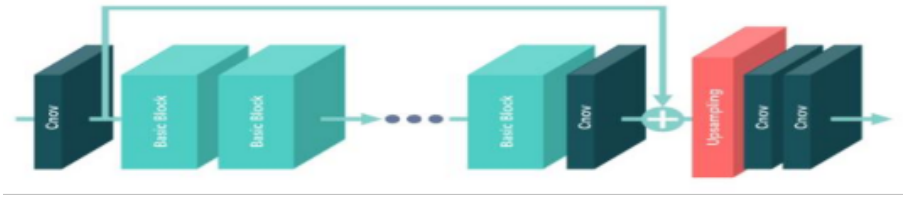


FIGURE 2.3 – Structure de base de SRResNet.

FIGURE 2.4 – Diagramme schématique du modèle de bloc dense.

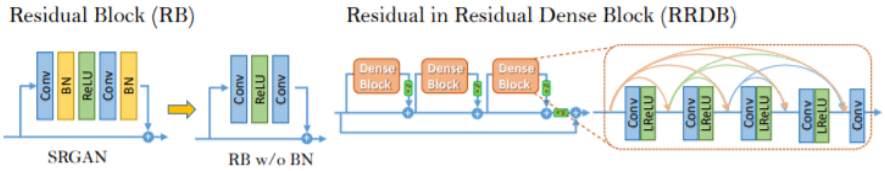


FIGURE 2.5 – Schéma global du modèle RRDB.

3.7.2.1 Couches convolutives :

En général, il s'agit des couches de base du réseau de générateurs. Ces couches constituent la base de tout modèle d'apprentissage profond, car elles sont utilisées pour extraire des caractéristiques et des informations importantes de l'image d'entrée basse résolution. En modifiant régulièrement l'image[28].

3.7.2.2 La fonction Leaky ReLU :

La fonction d'activation de l'unité linéaire rectifiée avec fuite est appelée « LreLU ». Le Leaky ReLU est une version modifiée de la fonction d'activation standard ReLU. La fonction d'activation Leaky ReLU réduit le problème du « ReLU » en autorisant un petit gradient non nul lorsque l'entrée est inférieure à zéro. Vous disposez d'un paramètre alpha (alpha) qui détermine la pente de la fonction pour les entrées négatives dans la variante "LreLU". Cela peut être défini mathématiquement comme[29] :

$$F(x) = \begin{cases} x & \text{si } x > 0 \\ x & \text{si } x =< 0 \end{cases} \tag{2.1}$$

L'utilisation du Leaky ReLU dans des modèles comme ESRGAN est destinée à introduire un certain degré de non-linéarité dans le modèle, lui permettant d'apprendre des fonctionnalités complexes et de mieux gérer les problèmes de gradient de disparition qu'un ReLU standard. Il est fréquemment utilisé dans ESRGAN pour activer certaines couches du réseau neuronal afin

de faciliter la formation et d'améliorer la qualité des images super-résolues créées par le modèle. La valeur spécifique d' peut varier en fonction des données d'entraînement et de l'architecture du modèle. En général, c'est un petit nombre positif comme 0,02.

2.3.3 Couches de suréchantillonnage(upsampling) :

Ces couches sont utilisées pour mettre à l'échelle les caractéristiques basse résolution extraites par le réseau afin de générer une sortie haute résolution. Le système utilise trois phases de traitement, à savoir : création des sous-segments d'images, l'agrandissement de chaque sous-segment d'images et la reconstruction ou le réassemblage des segments agrandis. Le schéma suivant illustre en les différentes phases.

2.3.4 Perte adverse :

Le discriminateur D qui utilise la notion de GAN relatif moyen (RAGAN) a été amélioré en plus de la structure améliorée du générateur. RaGAN est utilisé car le discriminateur doit également réduire la probabilité que "les vraies données soient correctes" et augmenter la probabilité que "les fausses données soient correctes". En d'autres termes, nous substituons le discriminant standard au discriminateur moyen relatif observé par Dra. Comme le montre la figure3.6,Il essaie de prédire la probabilité si la vraie image x_r est plus réaliste que la fausse image x_f, plutôt que “si l'image est vraie ou fausse”. Surligneur standard dans SRGAN peut être exprimé comme $D(x) = \sigma(C(x))$, où (σ) est la fonction sigmoïde et C(x) est le produit non transformé du discriminateur. Alors le RaD est pour formulé comme : $Dra(x_r, x_f) = \sigma(C(x_r) - Ex_f[C(x_f)])$, où $Ex_f[.]$ représent l'opération de prise de moyenne pour toutes les fausses données du mini-lot.Le discriminateur la perte est alors définie comme :

$$\mathrm{L}_D^{Ra} = -E_{x_r}\left[log(D_{Ra}(x_r, x_f))\right] - E_{x_r}\left[log(1 - D_{Ra}(x_f, x_r))\right] \tag{2.2}$$

La perte contradictoire pour le générateur est symétrique de :

$$\mathrm{L}_G^{Ra} = -E_{x_r}\left[log(1 - D_{Ra}(x_r, x_f))\right] - E_{x_r}\left[log(D_{Ra}(x_f, x_r))\right] \tag{2.3}$$

Où $\mathrm{x}_f = \mathrm{G}_{x_i}$ et x_i représente l'image LR d'entrée. On observe que la perte contradictoire pour le générateur contient à la fois x_r et x_f . Par conséquent, notre générateur bénéficie des gradients

des données générées et des données réelles en cas de confrontation formation, tandis que dans SRGAN, seule la partie générée prend effet, Pendant ce temps, l'utilisation de RaGAN aide à apprendre des bords plus nets et des textures plus fines.

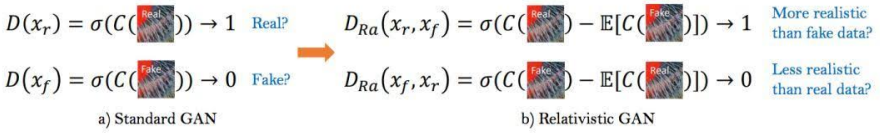


FIGURE 2.6 – la différence entre le discriminateur standard et le discriminateur relativiste[11].

2.3.5 Perte de perception :

La perte de perception est introduite en super-résolution pour optimiser le modèle de super-résolution dans l'espace des caractéristiques plutôt que dans l'espace des pixels. Dans l'ESRGAN, la perte de perception est améliorée en utilisant les fonctionnalités avant l'activation, ce qui peut entraîner une cohérence de la luminosité et une récupération de la texture. Avant l'activation, au lieu d'après l'activation comme dans SRGAN, la perte de perception est mise en œuvre en utilisant les fonctionnalités VGG. ESRGAN utilise les fonctionnalités avant activation pour les deux raisons suivantes[4] :

1- Une faible quantité de fonctionnalités activées permet une faible supervision et entraîne des performances inférieures.

2- Les fonctionnalités activées provoquent une luminosité incohérente par rapport à l'image de vérité au sol.

• Calculez la perte de perception à l'aide du modèle VGG créé pour optimiser les fonctionnalités de haut niveau d'ESRGAN et combinez le réseau discriminant ESRGAN pour obtenir un PSNR plus élevé. Cependant, dans le but d'obtenir un résultat visuel plus évident. ESRGAN utilise la perte perçue pour générer une perte de réseau, de sorte que le résultat de l'image traitée peut répondre aux exigences de l'indice de perception et obtenir le résultat perceptuel optimal. La fonction de perte du réseau généré est LPI, la fonction de perte de perception est LPI et la fonction de perte de pixel par pixel est L_1.

$$L_G = L_{percep} + \lambda \mathrm{L}_G^{Ra} + \eta L_1 \tag{2.4}$$

De plus, le suréchantillonnage dans le réseau de génération est effectué à l'aide de couches de convolution sous-pixels. Il y a huit couches convolutives dans la partie discriminante du réseau.

À mesure que le nombre de couches réseau s'approfondit, le nombre de fonctionnalités augmente continuellement et la taille des fonctionnalités diminue. Lorsque la fonction d'activation est choisie comme LeakyReLU, deux couches entièrement connectées et la fonction d'activation sigmoïde finale sont utilisées pour l'obtenir. La probabilité d'une image naturelle.

2.3.6 Couche de sortie :

Produit l'image améliorée finale.

2.4 Fonctions de perte :

Dans cette section, nous discuterons de diverses fonctions de perte qui peuvent être utilisées pour entraîner les réseaux [19].

2.4.1 Perte de pixels :

il s'agit du type de fonction de perte le plus simple et le plus courant utilisé dans la formation des réseaux à super-résolution. L2, L1 ou une métrique de différence est utilisée pour évaluer le modèle. L'entraînement avec perte de pixels optimise le PSNR, mais n'optimise pas directement la qualité de perception et génère donc des images qui pourraient ne pas plaire à l'œil humain.

2.4.2 Perte de perception :

La perte de perception tente de faire correspondre les caractéristiques de haut niveau d'une image générée à une image de sortie HR donnée. Ceci est accompli en utilisant un pré-réseau formé, comme VGG, et en utilisant les différences de sorties de fonctionnalités entre les sorties en tant que perte et les images prédites. SRGAN intègre cette fonction de perte.

2.4.3 Perte Charbonnier :

Cette fonction de perte est utilisée dans LapSRN à la place de la perte générique L2. Les résultats montrent que la perte Charbonnier traite mieux les valeurs aberrantes et produit des images plus nettes que celles générées avec la perte L2, qui sont généralement plus lisses. *Perte de texture : introduite dans EnhanceNet, cette fonction de perte tente d'optimiser la matrice Gram des sorties de fonctionnalités inspirée de la fonction de perte Style Transfer. Cette fonction de perte entraîne le réseau à capturer les informations de texture dans une image HR.

2.4.4 Perte contradictoire :

La perte contradictoire est utilisée dans toutes les architectures liées au GAN et produit généralement des images qui ont une meilleure qualité de perception. En utilisant le discriminateur relativiste, ESRGAN ajoute une variante en demandant au réseau non seulement de rendre les fausses images plus réelles mais également de rendre les vraies images plus fausses.

2.5 Conclusion :

Les techniques d'apprentissage en profondeur sont devenues très attrayantes en vision par ordinateur et dans divers autres domaines. Les réseaux de neurones comphiniques, qui sont très simples à mettre en œuvre, nous nous efforçons de fournir des résultats très prometteurs. En fait, les réseaux de neurones profonds tentent de le faire pour extraire les caractéristiques les plus importantes pour réduire la perte entre le signal estimé et la vérité de base, et ainsi, offrir des performances exceptionnelles.

CHAPITRE 3

IMPLÉMENTATION

3.1 Introduction :

Nous expliquerons les langages et les bibliothèques Programmation utilisée pour mettre en œuvre notre projet, puis nous passons au code.

3.2 Outils de programmation utilisés :

Le Deep Learning nécessite un matériel spécial et une très puissante (GPU) dont on ne dispose pas, c'est pour cela nous avons utilisé la plateforme google colab pour pouvoir implémenter et valider notre travail :

4.2.1 Google colab :

est un Cloud pour la communauté de la science des données de Google d'améliorer les compétences de codage en langage de Programmation Python. Nous donne la possibilité de créer et permet d'entraîner des modèles de machine learning et de deep learning complexes et lourds sans avoir à utiliser les ressources limitées de notre machine.[2].

Les avantages :

• Ecrire et exécuter du code en Python Vous pouvez également documenter votre propre code et vos équations mathématiques.
• Créer / télécharger/ partager un notebook Jupyter.
• Importer / enregistrer vers et depuis Google Drive.
• Importer/publier des blocs-notes Jupyter à partir de GitHub .
• Importation d'ensembles de données externes, par exemple depuis GitHub,Kaggle.
• Prend en charge de nombreuses bibliothèques, telles que : OpenCV, Keras,TensorFlow,PyTorch.

4.2.2 Python :

est un langage de programmation populaire pour la programmation. Guido van Rossum travaille sur ce projet depuis 1989.Il offre des constructions qui permettent une programmation claire à grande et à petite échelle. Il an un système de type dynamique et une gestion automatique de la mémoire. Il prend en charge tous les paradigmes de programmation (procédural,

orienté objet, fonctionnel et logique) et dispose d'une bibliothèque standard vaste et complète. Les interpréteurs Python peuvent être utilisés par de nombreux systèmes d'exploitation.

FIGURE 3.1 – Logo du langage python[12].

Caractéristiques du langage :

Nous détaillons un peu les principales caractéristiques de Python, plus précisément, les implantations actuelles :

• Python est un langage portable.

• Bien que Python soit gratuit, il peut être utilisé librement dans des projets commerciaux.

• La syntaxe de Python est très simple et fonctionne bien avec des types de données évolués comme les listes et les dictionnaires.

• Python est (optionnellement) multi-threadé[2].

3.3 Importation les packages et bibliothèques utilisés :

Nous importerons les bibliothèques et les packages que nous utiliserons dans ce projet :

4.3.1 OpenCV :

est une bibliothèque open source de Fonctions de programmation principalement destinées à la vision par ordinateur en temps réel qui possède une vaste collection d'excellents algorithmes.Développé à l'origine par Intel, il a ensuite été soutenu par Willow Garage puis Itseez (qui a ensuite été acquis par Intel). La bibliothèque est multiplateforme . Depuis l'une des dernières fusions, il contient une interface facile à utiliser pour la mise en œuvre de la super résolution (SR) basée sur des méthode d'apprentissage en profondeur. L'interface contient des

modèles pré-formés qui peuvent être utilisés pour l'inférence très facilement et efficacement. cela fonctionne avec C++ et Python.

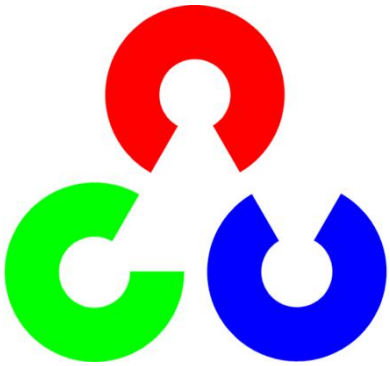


FIGURE 3.2 – Logo d'Open Cv.[13]

Les avantages OpenCV :

• La bibliothèque est gratuite sur plusieurs plateformes.

• Comme c'est l'une des dernières intégrations, il dispose d'une interface facile à utiliser pour gérer et travailler sur la Super-Résolution (SR) basée sur la méthode de formation approfondie.

• Les modèles préformés de l'interface peuvent être utilisés pour l'inférence très facilement et efficacement.

• Cela fonctionne avec C++ et Python.

4.3.2 TensorFlow :

une bibliothèque de logiciels open-source et gratuite. C'est une bibliothèque mathématique symbolique utilisée pour des applications d'apprentissage automatique comme les réseaux neuronaux. Elle est utilisée dans l'application Google Search et Production. L'équipe de Google Brain l'a créée pour une utilisation interne de Google. Le 9 novembre 2015, il est sorti sous la licence Apache 2.0.

FIGURE 3.3 – Logo TensorFlow.[14]

4.3.3 Keras :

Il s'agit d'une bibliothèque open source basée sur python qui offre la possibilité d'interagir avec les algorithmes de machine learning et de réseaux de neurones profonds, tels que Tensorflow[30].

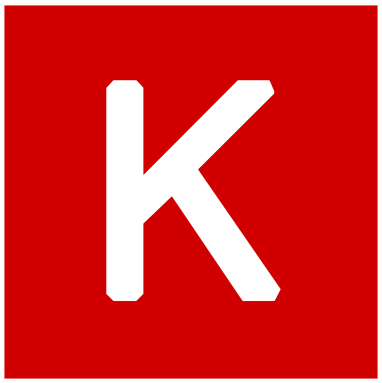


FIGURE 3.4 – keras.[15]

Les avantages Keras :

- Vitesse de diffusion et facilité de compréhension.
- Le déploiement multiplateforme et de modèle est facile.
- Prise en charge de plusieurs GPU.
- Grand soutien de la communauté.

4.3.4 NumPy :

est le package de base de Python pour le calcul scientifique.Travis Oliphant l'a fondé en 2005. Vous pouvez l'utiliser librement car c'est un projet open source.Il s'agit d'une bibliothèque Python qui fournit un objet tableau multidimensionnel, une variété d'objets dérivés (tels que des tableaux masqués et des matrices) et une variété de routines pour des opérations rapides sur des tableaux, telles que les mathématiques, la logique, la manipulation de la forme, la sélection, les transformées discrètes de Fourier, l'algèbre linéaire de base, les opérations statistiques de base, la simulation aléatoire.

Les advantage :

- NumPy vise à fournir un objet tableau jusqu'à 50 fois plus rapide que les listes Python traditionnelles.

FIGURE 3.5 – Logo Numpy

• L'objet tableau dans NumPy s'appelle ndarray, il fournit de nombreuses fonctions de support qui rendent le travail avec ndarray très facile.

• Les tableaux sont très fréquemment utilisés en science des données, où la vitesse et les ressources sont très importantes.

4.3.5 Sklearn :

est la bibliothèque la plus utile et la plus robuste pour l'apprentissage automatique en Python. Il fournit une sélection d'outils efficaces pour l'apprentissage automatique et la modélisation statistique, notamment la classification, la régression, le regroupement et la réduction de la dimensionnalité via une interface de cohérence en Python. Cette bibliothèque, qui est en grande partie écrite en Python, est construite sur NumPy, SciPy et Matplotlib[31].

FIGURE 3.6 – Logo Sklearn

Les aventage :

- Grande accessibilité.
- Facilité de prise en main.
- Liberté d'utilisation.
- Large catalogue d'algorithmes d'apprentissage statistique.

4.3.6 Matplotlib :

Matplotlib est une bibliothèque de traçage complète qui peut être utilisée pour créer des visualisations en python statiques, animées et interactives. Les choses faciles sont faciles à faire et les choses difficiles sont possibles à faire.

FIGURE 3.7 – Logo Matplotlib

Les avantages :

- Créez des tracés de qualité de publication.
- Créez des figures interactives qui peuvent zoomer, faire un panoramique, mettre à jour.
- Personnalisez le style visuel et la mise en page.
- Exportez vers de nombreux formats de fichiers.
- Intégré dans JupyterLab et les interfaces utilisateur graphiques.
- Utilisez un large éventail de packages tiers construits sur Matplotlib[32].

3.4 Data set DIV2K :

est un ensemble de données à image unique de haute résolution très populaire qui contient 1000 images de différentes scènes et est divisé en 800 pour la formation et 100 pour la validation. Il a été collecté pour les défis de la super-résolution afin d'encourager la recherche sur les images à haute résolution avec une dégradation plus réaliste[33].

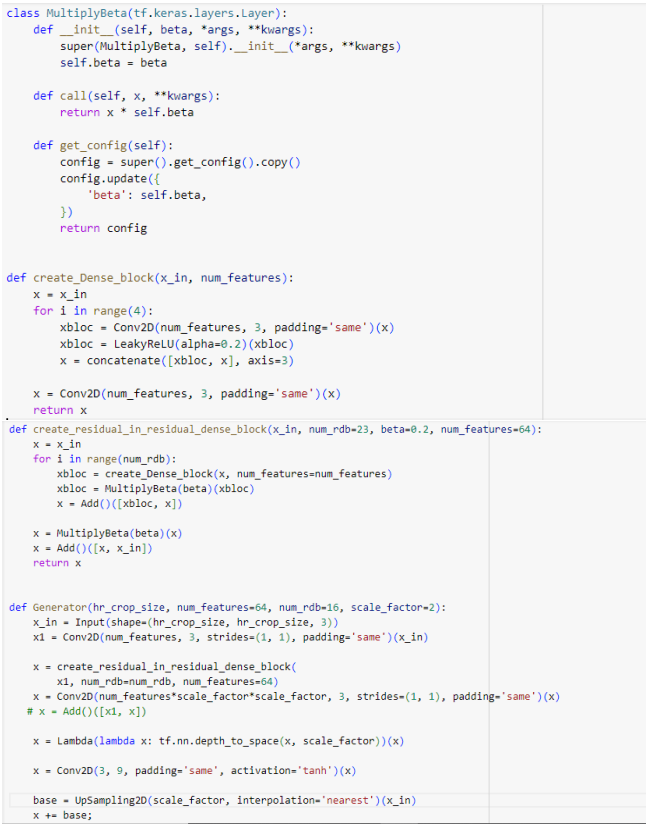
3.5 Détails de la formation :

ESRGAN met également à l'échelle l'image basse résolution (LR) en image haute résolution (HR) de 45 x 45 à 90 x 90,Nous avons formé le modéle 5000 fois.La formation et la vérification de l'expérience sont basées sur la plateforme Google Collab. Le modèle GPU est NVIDIA T4 Tensor Core GPU, la mémoire est 8 Go et le cadre de programmation est keras, basé sur Python.Pendant la formation, Nous définissons $\alpha = 0.2$ pour la fonction d'activation, LeakyReLU avec batch_size = 16. Dans le réseau de générateurs, nous utilisons l'optimiseur Adam et fixons le taux d'apprentissage initial à $1e^{-4}$. et définissons le taux d'apprentissage initial sur $1e^{-4}$. Dans le réseau générateur.

3.6 Explication du code :

-Générateur :

Son rôle est d'essayer de créer des images proches de l'image originale. Nous avons créé une fonction de 23 rrdb contenant une fonction de bloc dense. Nous avons créé 16 fonctions de bloc dense à partir des couches convolutives et relu quatre fois la fonction d'activation qui fuit, puis lorsqu'elle sort d'un bloc dense, vous la multipliez par bêta. Il les combine, et le résultat est que le bloc précédent entre dans un bloc dense et le processus est répété dans d'autres blocs denses. Ensuite, nous avons une variable lambda qui équilibre la perte contradictoire et la perte de contenu dans le processus de formation du réseau, et après cela, le processus de « suréchantillonnage » a lieu.Finalement, il renvoie le formulaire.



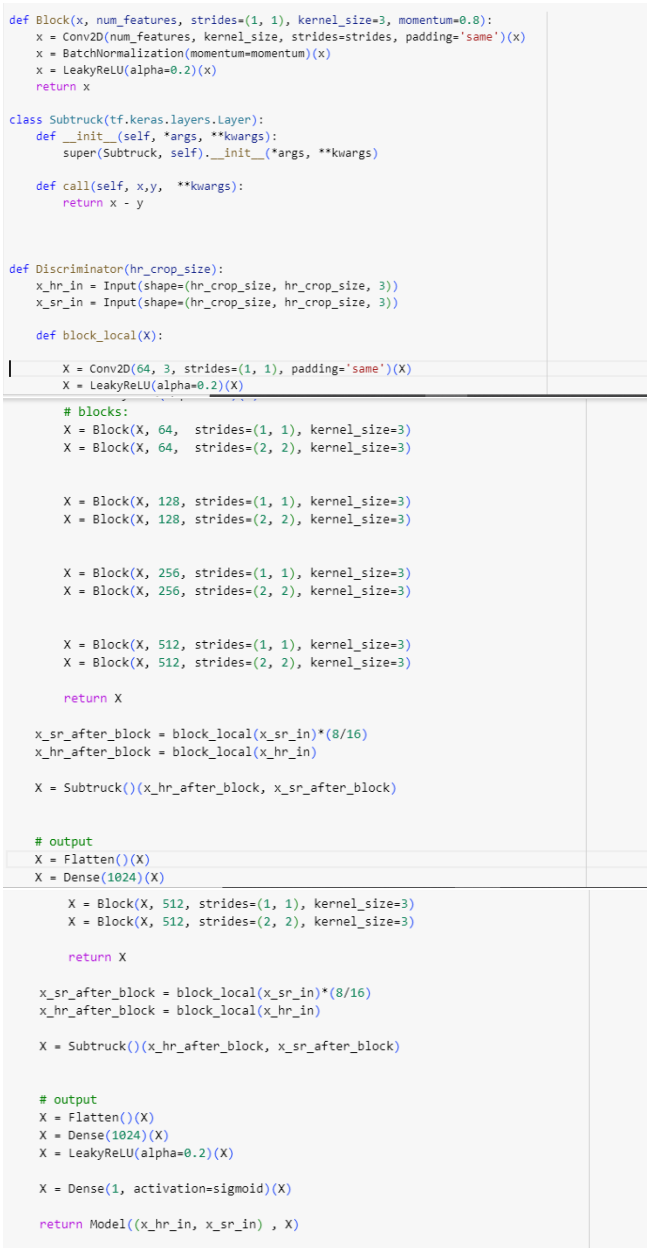
```
class MultiplyBeta(tf.keras.layers.Layer):
    def __init__(self, beta, *args, **kwargs):
        super(MultiplyBeta, self).__init__(*args, **kwargs)
        self.beta = beta

    def call(self, x, **kwargs):
        return x * self.beta

    def get_config(self):
        config = super().get_config().copy()
        config.update({
            'beta': self.beta,
        })
        return config


def create_Dense_block(x_in, num_features):
    x = x_in
    for i in range(4):
        xbloc = Conv2D(num_features, 3, padding='same')(x)
        xbloc = LeakyReLU(alpha=0.2)(xbloc)
        x = concatenate([xbloc, x], axis=3)

    x = Conv2D(num_features, 3, padding='same')(x)
    return x
```

```
def create_residual_in_residual_dense_block(x_in, num_rdb=23, beta=0.2, num_features=64):
    x = x_in
    for i in range(num_rdb):
        xbloc = create_Dense_block(x, num_features=num_features)
        xbloc = MultiplyBeta(beta)(xbloc)
        x = Add()([xbloc, x])

    x = MultiplyBeta(beta)(x)
    x = Add()([x, x_in])
    return x


def Generator(hr_crop_size, num_features=64, num_rdb=16, scale_factor=2):
    x_in = Input(shape=(hr_crop_size, hr_crop_size, 3))
    x1 = Conv2D(num_features, 3, strides=(1, 1), padding='same')(x_in)

    x = create_residual_in_residual_dense_block(
        x1, num_rdb=num_rdb, num_features=64)
    x = Conv2D(num_features*scale_factor*scale_factor, 3, strides=(1, 1), padding='same')(x)
  # x = Add()([x1, x])

    x = Lambda(lambda x: tf.nn.depth_to_space(x, scale_factor))(x)

    x = Conv2D(3, 9, padding='same', activation='tanh')(x)

    base = UpSampling2D(scale_factor, interpolation='nearest')(x_in)
    x += base;
```

FIGURE 3.8 – code Générateur.

-Discriminateur :

Son rôle est de distinguer les images réelles des fausses images. C'est une fonction dont les entrées sont une image haute résolution et une image super résolution et elle renvoie la réalité du pourcentage de la fausse image. Il contient 8 couches convolutives et des fonctions d'activation Leaky relu qui fuient et une normalisation par lots, lorsqu'une image haute résolution passe. Sur les blocs précédents, ainsi que l'image haute résolution, nous soustrayons les sorties d'image via subtruck, et enfin nous avons une fonction sigmoïde qui donne nous le rapport qui est limité à 0 et 1.



```
def Block(x, num_features, strides=(1, 1), kernel_size=3, momentum=0.8):
    x = Conv2D(num_features, kernel_size, strides=strides, padding='same')(x)
    x = BatchNormalization(momentum=momentum)(x)
    x = LeakyReLU(alpha=0.2)(x)
    return x

class Subtruck(tf.keras.layers.Layer):
    def __init__(self, *args, **kwargs):
        super(Subtruck, self).__init__(*args, **kwargs)

    def call(self, x,y,  **kwargs):
        return x - y



def Discriminator(hr_crop_size):
    x_hr_in = Input(shape=(hr_crop_size, hr_crop_size, 3))
    x_sr_in = Input(shape=(hr_crop_size, hr_crop_size, 3))

    def block_local(X):

        X = Conv2D(64, 3, strides=(1, 1), padding='same')(X)
        X = LeakyReLU(alpha=0.2)(X)
        # blocks:
        X = Block(X, 64,  strides=(1, 1), kernel_size=3)
        X = Block(X, 64,  strides=(2, 2), kernel_size=3)


        X = Block(X, 128, strides=(1, 1), kernel_size=3)
        X = Block(X, 128, strides=(2, 2), kernel_size=3)


        X = Block(X, 256, strides=(1, 1), kernel_size=3)
        X = Block(X, 256, strides=(2, 2), kernel_size=3)


        X = Block(X, 512, strides=(1, 1), kernel_size=3)
        X = Block(X, 512, strides=(2, 2), kernel_size=3)

        return X

    x_sr_after_block = block_local(x_sr_in)*(8/16)
    x_hr_after_block = block_local(x_hr_in)

    X = Subtruck()(x_hr_after_block, x_sr_after_block)


    # output
    X = Flatten()(X)
    X = Dense(1024)(X)
        X = Block(X, 512, strides=(1, 1), kernel_size=3)
        X = Block(X, 512, strides=(2, 2), kernel_size=3)

        return X

    x_sr_after_block = block_local(x_sr_in)*(8/16)
    x_hr_after_block = block_local(x_hr_in)

    X = Subtruck()(x_hr_after_block, x_sr_after_block)


    # output
    X = Flatten()(X)
    X = Dense(1024)(X)
    X = LeakyReLU(alpha=0.2)(X)

    X = Dense(1, activation=sigmoid)(X)

    return Model((x_hr_in, x_sr_in) , X)
```

FIGURE 3.9 – code Discriminateur

-loss function :

Ce code fourni définit trois classes : 'perceptual_loss', 'DiscriminatorLoss' et 'PSNR'. 1. Catégorie « Perte cognitive » : - Cette classe est utilisée pour calculer la fonction de perte du réseau de générateurs dans un modèle basé sur GAN, où les pertes de perception et de contenu sont prises en compte. - Le constructeur ('___init___') accepte plusieurs paramètres, notamment 'lambda_value', 'n' et 'vgg_num_layers'. Initialise le modèle VGG19 (généralement utilisé pour les tâches cognitives), extrait les sorties d'une couche donnée et prépare les fonctions d'erreur quadratique moyenne et de perte d'entropie binaire. La méthode Calculate calcule la perte totale du générateur, qui est une combinaison de perte de perception ('Lpercep'), de perte de

générateur ('Lg') et de perte de contenu ('L1'), chacune multipliée par des poids spécifiques. -La méthode calculate_generator_loss calcule la perte du générateur en utilisant l'ent.

```
class Perceptual_loss():
    def __init__(self, lambda_value =5e-3, n=1e-2, vgg_num_layers=20):

        self.layer_5_4 = vgg_num_layers
        self.lambda_value = lambda_value
        self.n = n

        vgg = VGG19(input_shape=(None, None, 3), include_top=False)
        self.vggModel = Model(
            vgg.input, vgg.layers[self.layer_5_4].output)

        self.mean_squared_error = MeanSquaredError()
        self.binary_cross_entropy = BinaryCrossentropy()

    def calculate(self, hr, sr, r_d_hr, r_d_sr):
        Lpercep = self.__calculate_percep(hr, sr)
        Lg = self.__calculate_generator_loss(r_d_hr, r_d_sr)
        L1 = self.calculate_content_loss(hr, sr)

        return Lpercep + self.lambda_value * Lg + self.n * L1
```

```
   def __calculate_generator_loss(self, r_d_hr, r_d_sr ):
       x1 =  self.binary_cross_entropy(tf.zeros_like(r_d_hr), r_d_hr)
       x2 =  self.binary_cross_entropy(tf.ones_like(r_d_sr), r_d_sr)
       return (8/16) * (x1 + x2)

   def calculate_content_loss(self, hr, sr):
       return 0.5 * tf.norm(sr - hr)

   def __calculate_percep(self, hr, sr):
       sr = preprocess_input(sr)
       hr = preprocess_input(sr)
       sr_map = self.vggModel(sr)/12.75
       hr_map = self.vggModel(hr)/12.75

       return self.mean_squared_error(hr_map, sr_map)

class DiscriminatorLoss():

   def __init__(self):
       self.BinaryCrossentropy = BinaryCrossentropy()

   def calculate(self, r_d_hr, r_d_sr):
       hr_loss = self.BinaryCrossentropy(tf.ones_like(r_d_hr), r_d_hr)
       sr_loss = self.BinaryCrossentropy(tf.zeros_like(r_d_sr), r_d_sr)
       return   (8/16) * ( hr_loss + sr_loss)
```

```
class DiscriminatorLoss():

    def __init__(self):
        self.BinaryCrossentropy = BinaryCrossentropy()

    def calculate(self, r_d_hr, r_d_sr):
        hr_loss = self.BinaryCrossentropy(tf.ones_like(r_d_hr), r_d_hr)
        sr_loss = self.BinaryCrossentropy(tf.zeros_like(r_d_sr), r_d_sr)
        return   (8/16) * ( hr_loss + sr_loss)

class PSNR(Loss):
    def __init__(self, denormalize , *args, **kwargs):
        super().__init__(*args, **kwargs)
        self.denormalize = denormalize

    def call(self, real_y, pridcted_y):
        hr, sr = self.denormalize(real_y,pridcted_y)
        return tf.image.psnr(hr, sr, max_val=255)
```

FIGURE 3.10 – code loss function.

3.7 Résultats :

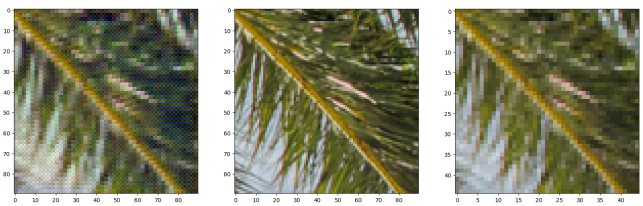


FIGURE 3.11 – Avant la formation.

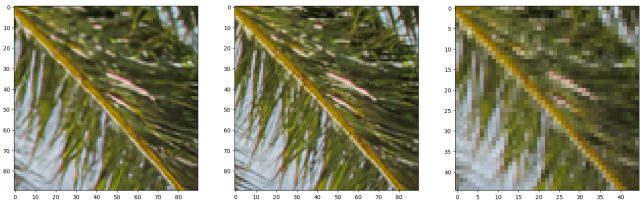


FIGURE 3.12 – Après la formation.

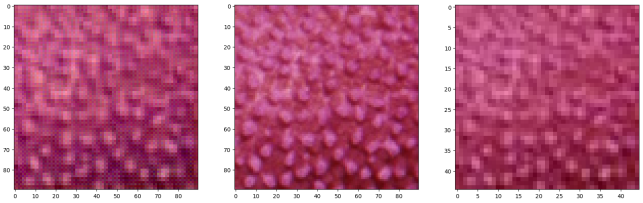


FIGURE 3.13 – Avant la formation.

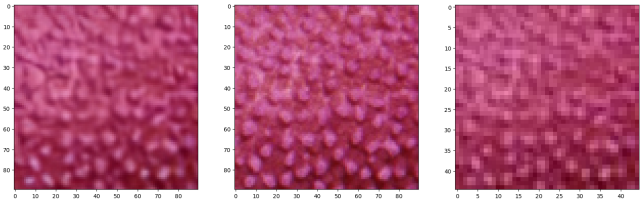


FIGURE 3.14 – Après la formation.

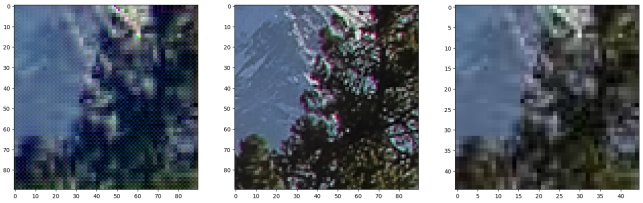


FIGURE 3.15 – Avant la formation.

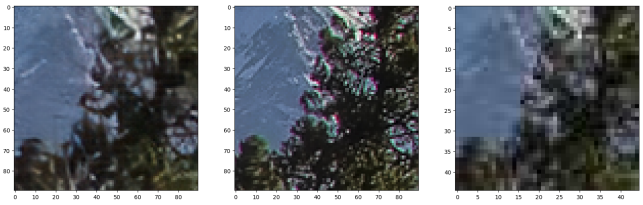


FIGURE 3.16 – Après la formation.

3.8 Comparaison entre Traitmment séquntiel et Traitement parallèle :

Quant au traitement séquentiel, après avoir divisé une image, il produit l'image divisée aux dimensions de 45 x 45. Ensuite, le générateur l'agrandit, puis nous la plaçons dans la nouvelle image agrandie aux dimensions de 90 x 90. Ensuite, il passe au diviser l'image à côté, et ce processus est effectué ainsi jusqu'à ce qu'elle atteigne la fin de la ligne, puis elle passe à la ligne suivante après 45, et ainsi de suite jusqu'à ce que nous atteignions la fin de l'image.
Quant au traitement en parallèle, après avoir divisé une image, il produit l'image divisée aux dimensions de 45 x 45. Ensuite le générateur l'agrandit puis la place dans la nouvelle image agrandie aux dimensions de 90 x 90. Traitement de la partie précédente s'arrête puis se déplace vers l'image divisée à côté, et ce processus continue ainsi jusqu'à ce qu'il atteigne la fin de la ligne, puis passe à la ligne suivante à une distance de 45, par exemple, puis nous traitons toutes les parties de l'image ensemble.

	time
0	89.77465653
1	51.31742668
2	56.776999
3	44.15533853
4	58.34727621
5	50.76667333
6	40.73214602
7	51.69942474
8	51.64430571
9	59.67822099

FIGURE 3.17 – Temps d'exécution traitement parallèle.

	time
0	45.67882824
1	40.1349473
2	46.73900342
3	34.15987444
4	46.13288093
5	40.32331133
6	33.22402835
7	40.16545486
8	40.19511962
9	45.92150092
10	40.27904654

FIGURE 3.18 – Temps d'exécution traitement séquntiel.

FIGURE 3.19 – Image haute résolution en parallèle et séquntiel.

FIGURE 3.20 – photo super-resolution en parallèle et séquntiel.

FIGURE 3.21 – photo (super-resolution,higt,low) en parallèle.

FIGURE 3.22 – photo (super-resolution,higt,low) en séquntiel.

```
v = valid_ds.as_numpy_iterator()
```

```
v_image = v.next()
index_image_v = 0
image_low = v_image[0][index_image_v]
```

```
def thread_function(name, yi, xi, y_offset, x_offset, scale, newScale):
    print("Thread ",name,"  starting")

    ROI = image_low[yi:yi + scale, xi:xi + scale]
    zoomedROI = esrgan.generator(np.expand_dims(ROI, axis=0))
    zoomedROI = np.squeeze(zoomedROI, axis=0)
    zoomedImg[y_offset:y_offset + newScale, x_offset:x_offset + newScale] = zoomedROI

    print("Thread ",name,"  finishing")

start = time.time();
srcH = image_low.shape[0]
srcW = image_low.shape[1]
zoomedImg = np.zeros((srcH * dataset_config.scale, srcW * dataset_config.scale, 3), dtype=np.float16)

thread_id= 0
```

```
threads_list = []
xhi = 0
yhi = 0
newScale = dataset_config.scale * lr_crop_size

for xli in range(0, srcW - lr_crop_size, lr_crop_size):
    yhi = 0
    for yli in range(0, srcH - lr_crop_size, lr_crop_size):

        x = threading.Thread(target=thread_function, args=(thread_id, yli, xli, yhi, xhi, lr_crop_size, newScale,))
        x.start()
        threads_list.append(x)
        thread_id = thread_id + 1
        yhi = yhi + newScale

    xhi = xhi + newScale


for thread in threads_list:
    thread.join()


end = time.time()
```

```
print("The time of execution of above program is :", (end - start) * 10 ** 3, "ms")
```

```
The time of execution of above program is : 50187.7121925354 ms
```

```
index_image_v = 0
image_out_v = denormalize_m11(zoomedImg,image_low )
image_out_v += denormalize_m11(v_image[1][index_image_v], v_image[1][index_image_v])
fig, ax = plt.subplots(ncols=3, figsize=(20,20))
for idx in range(3):
    print(image_out_v[idx].shape)
    ax[idx].imshow(image_out_v[idx])
```

```
xxx = np.reshape(np.array(image_out_v[0]), image_out_v[0].shape)
```

```
image = im.fromarray(xxx)
```

```
image.save('sr.png')
```

FIGURE 3.23 – code Traitement parallèle.

```
v = valid_ds.as_numpy_iterator()
v_image = v.next()
index_image_v = 0
image_low = v_image[0][index_image_v]
```

```
start = time.time();
srcH = image_low.shape[0]
srcW = image_low.shape[1]
zoomedImg = np.zeros((srcH * dataset_config.scale, srcW * dataset_config.scale, 3), dtype=np.float16)
# cv2.imshow("blank Image",zoomedImg)

index = 1
xhi = 0
yhi = 0
newScale = dataset_config.scale * lr_crop_size
for xli in range(0, srcW - lr_crop_size, lr_crop_size):
    yhi = 0
    for yli in range(0, srcH - lr_crop_size, lr_crop_size):
        ROI = image_low[yli:yli + lr_crop_size, xli:xli + lr_crop_size]
        zoomedROI = esrgan.generator(np.expand_dims(ROI, axis=0))
        zoomedROI = np.squeeze(zoomedROI, axis=0)
        zoomedImg[yhi:yhi + newScale, xhi:xhi + newScale] = zoomedROI
        yhi = yhi + newScale
    xhi = xhi + newScale
```

```
end = time.time()
```

```
print("The time of execution of above program is :", (end - start) * 10 ** 3, "ms")
```

```
index_image_v = 0
image_out_v = denormalize_m11(zoomedImg,image_low )
image_out_v += denormalize_m11(v_image[1][index_image_v], v_image[1][index_image_v])
fig, ax = plt.subplots(ncols=3, figsize=(20,20))
for idx in range(3):
    print(image_out_v[idx].shape)
    ax[idx].imshow(image_out_v[idx])
```

```
xxx = np.reshape(np.array(image_out_v[0]), image_out_v[0].shape)
```

```
image = im.fromarray(xxx)
```

```
image.save('sr.png')
```

FIGURE 3.24 – code Traitmment séquntiel.

3.9 Mesure de qualité d'image :

Pour évaluer les performances de ce réseau, nous utilisons, via une fonction compare images, métrique de qualité d'image : le rapport signal/bruit de crête(PSNR)meilleure est la qualité de l'image.

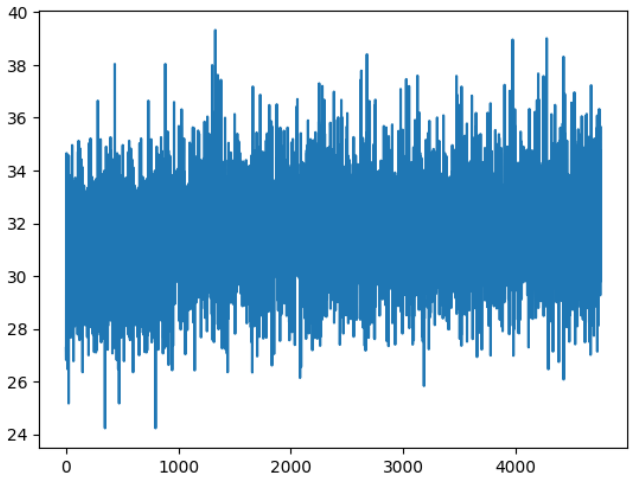


FIGURE 3.25 – Le grqphe de psnr.

3.10 discussion :

- Même si ESRGAN est formé sur des périodes plus courtes, il a bien fonctionné.
- La proposition de perte de perception résout le problème des images trop fluides causées par la perte de pixels et améliore la qualité visuelle des images, mais elle produit également inévitablement un grand nombre d'artefacts et de distorsions dans les images.
- Le choix de la méthode d'interpolation peut affecter la qualité globale de l'image super-résolution . Utilisez l'interpolation du voisin le plus proche pour l'étape finale de mise à l'échelle . Il a tendance à produire des résultats en blocs ou pixellisés par rapport aux méthodes d'interpolation plus avancées telles que le ré-échantillonnage cubique ou Lanczos .

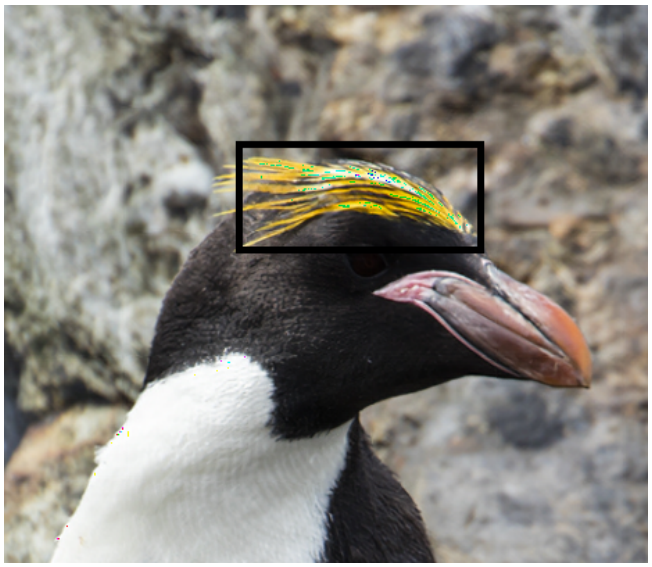


FIGURE 3.26 –

3.11 Conclusion :

Dans ce chapitre la partie pratique de notre projet est présentée en montrant la base d'images utilisée ainsi que notre application. De plus, nous avons présenté les résultats obtenus à partir de notre modèle et analysé ceux que nous avons jugés acceptables mais qui pourraient être améliorés.

CONCLUSIONS ET ORIENTATIONS FUTURES:

La finalité de ce travail est de reconstruire des images haute résolution (HR) à partir d'images basse résolution (LR),plus récemment, les GAN viennent ici à la rescousse.Nous avons obtenu de bons résultats sur les données utilisées, les images de test de basse résolution ont été nettement et visiblement améliorées en utilisant la technique proposée.

Comme on peut le constater, beaucoup de travail a été fait dans le domaine de la RS, mais il existe encore de nombreux défis qui doivent être surmontés. L'un des principaux problèmes réside dans le développement de méthodes SR réellement capables de fonctionner sur des images du monde réel dans un environnement aveugle où aucune information sur les dégradations susceptibles d'affecter une image n'est supposée.Il peut y avoir de nombreux artefacts sur l'image de sortie et les résultats sont encore loin d'être parfaits. Nous Avon's sûr à 100 pour cent que plusieurs années plus tard, de nouveaux algorithmes fourniront de bien meilleurs résultats.

BIBLIOGRAPHIE

[1] . M.B Hisham1, SShahrulNizamYaakob, R. R.A.A2, . A.B.A. Nazren1, and N.M.Wafi1, "An analysis of performance for commonly used interpolation method," Ph.D. dissertation, School of Computer and Communication Engineering, University Malaysia Perlis, 01000 Kangar, Perlis, Malaysia.

[2] A. hanane and D. Yamine, "Détection de cancer de la peau par l'utilisation d'apprentissage profond," *Université Elchahid Hamma Lakhdar El-oued*, 2021.

[3] Z. Wang, J. Chen, and S. Hoi, "Deep learning for image super-resolution : A survey," *arXiv preprint arXiv :1707.02921.*

[4] [Online]. Available : https://medium.com/analytics-vidhya/esrgan-enhanced-super-resolution-gan-96a28821634

[5] [Online]. Available : https://www.analyticsvidhya.com/blog/2021/05/deep-learning-for-image-super-resolution/

[6] Wei-Shengi, J.-B. Huang, N. Ahuja, and M.-H. Yang, "Fast and accurate image super-resolution with deep laplacian pyramid networks," 2017.

[7] J. Moses and Selvathi, "A survey on non-adaptive image interpolation algorithm based on quantitative measures," *International Journal.*

[8] Chaofeng, W. Z. Li, and J. Shi, "Lightweight image super-resolution with adaptive weighted learning network," *Shanghai University Shanghai, China.*

[9] C. Ledig, L. Theis, F. Huszar, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," 25 may 2017.

[10] K. Lia, Y. Liangb, S. Yanga, J. Jiaa, J. Huanga, and X. Wanga, "Single image super resolution based on generative adversarial networks," *Department of Computer Technology and Application, Qinghai University, China*, 04 Jun 2022.

[11] X. Wang1, S. Wu2, J. Gu3, Y. Liu4, C. Dong2, C. C. Loy5, Y. Qiao2, and X. Tang1, "Esrgan : Enhanced super-resolution generative adversarial networks," 1 Sep 2018.

[12] [Online]. Available : https://logowik.com/python-vector-logo-4431.html

[13] [Online]. Available : https://fr.wikipedia.org/wiki/OpenCV

[14] [Online]. Available : https://www.tensorflow.org/about

[15] [Online]. Available : https://www.google.com/search?q=logo%20Keras&oq=logo%20Keras%20&aqs=chrome..69i57j0i22i30l14.3996j0j4&client=ms-android-oppo-rvo3&sourceid=chrome-mobile&ie=UTF-8&shem=canimgc&source=sh%2Fx%2Fim%2Fcan%2F2#vhid=vvZYFI1h4TOLsM&vssid=l

[16] [Online]. Available : https://www.v7labs.com/blog/image-super-resolution-guide/

[17] P. C. kadasur, "Investigation on image zooming interpolation," *KAUNAS*, 2017.

[18] A. Magnusson and A. Sandahl, "Artifact-free image upscaling : An evaluation of aa artifact-free color interpolation algorithm with respect to visual quality," *2022-06-13*.

[19] [Online]. Available : https://blog.paperspace.com/image-super-resolution/

[20] Wissem, *Apprentissage profond pour la super Résolution des images*, 2022.

[21] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep," *Convolutional Networks 31 juil 2015*, 2015.

[22] [Online]. Available : https://medium.com/aiskunks/pixel-perfect-esrgan-powered-high-resolution-image-upscaling-platform-2c3ae0934d53

[23] Zhang, Li, K., Li, Wang, and Z. Fu, "Image super-resolution using very deep residual channel attention networks," in *ECCV*, 2018.

[24] Dong, Loy, H. K, and T. X, "Learning a deep convolutional network for image super-resolution," in *ECCV*, 2014.

[25] Szegedy, S. V. C.and Ioffe, and V., "Inception-v4, inception-resnet and the impact of residual connections on learning," *arXiv preprint arXiv :1602.07261*, 2016.

[26] Ledig, C, Theis, L, H. F, Caballero, J., Cunningham, A. A. A.and Acosta, T. A, Totz, J, Wang, and Z. et al, "Photo-realistic single image super-resolution using a generative adversarial network."

[27] M. Haris, Shakhnarovich, and Ukita, "Deep backprojection networks for super-resolution," in *CVPR*, 2018.

[28] [Online]. Available : https://medium.com/analytics-vidhya/2d-convolution-using-python-numpy-43442ff5f381

[29] [Online]. Available : https://www.mygreatlearning.com/blog/relu-activation-function/

[30] [Online]. Available : https://coursee.org/article/keras-lessons-what-is-keras

[31] [Online]. Available : https://www.tutorialspoint.com/scikit_learn/scikit_learn_introduction.htm

[32] [Online]. Available : https://matplotlib.org/

[33] [Online]. Available : https://data.vision.ee.ethz.ch/cvl/DIV2K/