République Algérienne Démocratique Et Populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

Université du 8 mai 1945-Guelma-

Faculté des mathématiques, de l'informatique et des sciences de la matière

Département d'informatique

Mémoire de Master

Spécialité : Informatique

Option:

Science et Technologie de l'Information et de la Communication

Thème

L'optimisation de la qualité d'image grâce à une approche innovante

Présenté par : AIMEUR Amina Djihane

Membres du jury :

N°	Nom & prénom	Qualité
1	BRAHIMI Said	Président
2	GUERROUI Nadia	Superviseur
3	GOUASMI Noureddine	Examinateur

soutenue le 25 juin 2025

Remerciements

C'est grâce à Dieu et au courage qu'Il m'a donné pour mener à bien ce mémoire que j'exprime ma profonde gratitude à ma directrice de mémoire, **Dr. Nadia Guerroui** pour son soutien et ses efforts remarquables.

Je remercie sincèrement le **Pr. Mohamed Tahar Kimour, EM-Alger Business School, Algérie**, pour sa disponibilité, son accompagnement et ses conseils éclairés tout au long de ce travail.

Je suis également reconnaissante envers mon collègue **M. Mohamed Chaib**, pour ses encouragements et ses conseils, qui m'ont été d'un grand soutien.

Dédicace

Je dédie ce mémoire à toute ma famille, et plus particulièrement à mes chers parents, pour leur amour et leur soutien indéfectible. J'adresse également ma profonde gratitude à toutes les personnes qui m'ont accompagné et soutenu tout au long de sa réalisation.

تحسين جودة الصورة من خلال نهج مبتكر

الملخص

نقترح من خلال **ResDiff** نهجًا مبتكرًا لأول مرة لتحسين دقة الصور، من خلال دمج تقنيتين متكاملتين ـ الشبكة الالتفافية ESPCN ونموذج الانتشار الاحتمالي DDPM. على عكس الأساليب التقليدية التي تتعامل مباشرة مع الصورة منخفضة الدقة (LR)، يستخدم ResDiff أولاً شبكة ESPCN لإنتاج نسخة أولية عالية الدقة (HR) تركز على المكونات منخفضة التردد. بعد ذلك، يتدخل نموذج DDPM لتحسين هذه التنبؤات من خلال إعادة بناء التفاصيل المفقودة، وخاصة الترددات العالية، على شكل بقايا. ويتم هذا التحسين باستخدام **شبكة U-Net شرطية**، يتم تزويدها بالصورة LR المكبرة، والضجيج المُضاف، وخطوة الزمن في عملية الانتشار، وذلك عبر تضمين زمني مخصص. كما يدمج النظام نهجًا ثانيًا يتمثل في تقنيات موجهة داخل شبكة U-Net، تستند إلى نتائج دالة فقدان هجينة MSE + FFT + DWT تم تطبيقها ضمن ESPCN. وتُثري هذه المعلومات الموجهة عملية التعلم من خلال تقديم خصائص تحليلية خارجية، تجمع بين الإشراف على مستوى البكسل، والمعلومات الترددية العامة، والهياكل متعددة المقاييس. وبهذا، يتمكن ResDiff من إنتاج صور أكثر دقة وجودة عالية، مع الحفاظ على تعقيد حسابي مضبوط.

الكلمات الدالة: تحسين دقة الصورة، الشبكة العصبية التلافيفية ، نماذج الانتشار الاحتمالي، التنبؤ الأولي، الصقل، الجودة البصرية.

L'optimisation de la qualité d'image grâce à une approche innovante

Résumé :

ResDiff constitue une avancée dans la super-résolution d'image, nous proposons une approche novatrice pour la super-résolution d'image, en combinant deux techniques complémentaires : le réseau convolutionnel **ESPCN** et le modèle de diffusion probabiliste **DDPM**. Contrairement aux méthodes classiques qui opèrent directement sur l'image basse résolution (LR), le ResDiff adopte une stratégie en deux étapes. Dans un premier temps, **ESPCN** génère une image haute résolution *(HR)* initiale, axée principalement sur la reconstitution des composantes de basse fréquence. Dans un second temps, *DDPM* intervient pour affiner cette prédiction en réintroduisant les détails manquants — notamment les hautes fréquences — sous forme de **résidu**.

Ce raffinement s'appuie sur un**U-Net conditionnel**, alimenté par l'image LR interpolée, un bruit aléatoire, ainsi que l'étape temporelle du processus de diffusion, encodée via un mécanisme dédié. En parallèle, **ResDiff** intègre une voie d'optimisation supplémentaire, basée sur des mécanismes de guidage injectés dans le U-Net. Ces guidages sont dérivés d'une **fonction de perte hybride** (MSE, FFT et DWT), appliquée au sein du module **ESPCN**, afin d'introduire des caractéristiques analytiques externes.

Cette combinaison permet de concilier supervision pixel-par-pixel, cohérence fréquentielle globale et représentations multi-échelles. En conséquence, **ResDiff** parvient à générer des images haute résolution de qualité supérieure, aux détails plus fidèles, tout en conservant une complexité computationnelle maîtrisée.

Mots clefs : ***Super-résolution d'image, réseau neuronal convolutif CNN, modèles de diffusion probabilistes, prédiction initiale, affinement, qualité visuelle.***

Image quality optimization through an innovative approach

Abstract :

In **ResDiff**, we introduce a groundbreaking method for image super-resolution that harnesses the strengths of two complementary techniques : the convolutional network ESPCN and the probabilistic diffusion model **DDPM**. Unlike conventional approaches that directly process low-resolution (LR) images, **ResDiff** utilizes a two-phase method. First, ESPCN creates an initial high-resolution (HR) image, concentrating on the reconstruction of low-frequency components. Next, DDPM enhances this initial output by reintroducing the missing high-frequency details through a residual process. This combined strategy not only improves the overall quality of the reconstructed image but also provides a more precise depiction of complex details.

This refinement is performed using **a conditional U-Net**, guided by the upsampled LR image, injected noise, and the diffusion timestep, all encoded through dedicated embeddings. Additionally, **ResDiff** incorporates a guided optimization strategy based on **a hybrid loss function** (MSE + FFT + DWT), applied within the ESPCN. This guidance brings external analytical features to the learning process, combining pixel-level supervision, global frequency awareness, and multi-scale structural cues.

As a result, **ResDiff** generates visually faithful, high-quality images while maintaining a controlled computational complexity.

Keywords : ***Super-resolution imaging, convolutional neural network (CNN), probabilistic diffusion models, initial prediction, refinement, visual quality.***

Table des matières

Liste des tableaux

Table des figures

Introduction générale

Contexte de la thèse

La super-résolution d'image (**SISR** : Single Image Super-Resolution), qui consiste à reconstituer une image haute résolution (HR) à partir d'une seule image basse résolution (LR), est une tâche fondamentale en vision par ordinateur. C'est un processus crucial dans de nombreux domaines concrets comme la surveillance, la médecine, l'imagerie satellitaire ou la restauration d'images historiques [Li et al., 2022].

Ces dernières années, l'**apprentissage profond** a transformé la SISR. Les réseaux de neurones convolutifs (CNN), comme le SRCNN ou l'ESPCN, ont considérablement amélioré la qualité des reconstructions en capturant efficacement les structures globales des images. Parallèlement, **les modèles génératifs** tels que les GAN (Generative Adversarial Networks) ont été introduits pour produire des images visuellement plus convaincantes. Plus récemment, **les modèles de diffusion probabilistes (DDPM)** sont apparus comme une alternative très prometteuse, démontrant des performances remarquables dans la génération d'images réalistes, pour la synthèse comme pour la restauration.

Problématique de la thèse

Malgré les avancées, la super-résolution d'image reste un défi.

— **Nature mal posée de la SISR :** La perte des composantes haute fréquence lors de la dégradation rend la tâche difficile, car plusieurs images HR peuvent correspondre à la même image LR.

— **Limites des CNNs :** Bien que performants pour les structures globales, les CNNs peinent souvent à reproduire les **détails fins et les textures réalistes**, pourtant essentiels pour la perception visuelle humaine.

— **Instabilités des GANs :** Malgré leur efficacité visuelle, les GANs souffrent **d'instabilité à l'entraînement** et sont sujets au phénomène de **mode collapse**, ce qui limite la diversité de leurs résultats.

— **Inefficacités des méthodes de diffusion actuelles :** Les modèles de diffusion existants (comme SR3) génèrent les images HR directement à partir d'un bruit aléatoire, en utilisant l'image LR comme simple condition d'entrée. Cette approche contraint le modèle à reconstruire à la fois les basses et hautes fréquences, ce qui **allonge le temps d'apprentissage** et peut entraîner une **perte de finesse** dans les détails générés. Les tentatives

d'amélioration par interpolation bilinéaire ou modules parallèles n'ont pas encore résolu efficacement le problème du guidage du processus de génération.

Contributions de la thèse

Pour relever ces défis, nous proposons des innovations significatives :

— **Introduction du modèle ResDiff** : Nous proposons **ResDiff**, un nouveau modèle de super-résolution basé sur une **structure résiduelle** qui combine un CNN et un modèle de diffusion (DDPM).

— Approche de génération en deux étapes : ResDiff optimise le processus en deux phases :

1. **L'ESPCN** produit **une première estimation de l'image HR**, se concentrant principalement sur **les composantes de basse fréquence**.
2. Le **DDPM** complète ensuite cette prédiction en générant les détails fins sous forme de résidu entre l'image réelle et l'estimation de l'ESPCN. Ce processus est guidé par un U-Net conditionnel intégrant un encodage temporel du bruit.

— **Intégration et valorisation de l'ESPCN** : Nous remplaçons le CNN initialement prévu dans ResDiff par l'architecture ESPCN. Ce choix est motivé par l'extraordinaire efficacité de l'ESPCN, qui est jusqu'à **dix fois plus rapide** que les CNNs classiques tout en offrant des performances comparables, voire supérieures. Sa capacité à traiter des vidéos en haute définition (1080p) en temps réel sur un seul GPU représente un atout majeur pour la super-résolution, et constitue une contribution clé de cette approche.

Organisation de la thèse

Au-delà de cette introduction générale — qui expose le contexte de l'étude, la problématique de recherche, les apports de la thèse et la structure du manuscrit — ce mémoire est articulé autour de trois chapitres, présentés ci-après

Chapitre 1 : **État de l'art.** Il offre au lecteur un aperçu général de la *super-résolution d'image* et du *Deep Learning*.

Chapitre 2 :**Conception d'un CNN spécifique pour la super-résolution d'images.** se focalise sur la conception détaillée et l'analyse approfondie du modèle **ResDiff**, en définissant clairement ses architectures et couches constitutives, et en présentant les divers algorithmes d'apprentissage employés.

Chapitre 3 :**Implémentation et résultats expérimentaux.** Il est dédié à la présentation des essais et à l'évaluation des performances du système, établie sur des mesures objectives et une comparaison avec les méthodes concurrentes.

Enfin, la conclusion générale met en évidence **la valeur de l'approche** et ses **limites**, tout en proposant des **perspectives de recherche future** fondées sur les principaux résultats de ce travail.

Chapitre 1

État de l'art

1.1 Introduction

Le traitement d'image, c'est l'ensemble des techniques utilisées pour modifier, améliorer ou analyser des images à l'aide d'un ordinateur. Il permet, par exemple, d'augmenter la netteté d'une photo floue, de corriger les couleurs, de supprimer du bruit ou encore de détecter des objets dans une image.

On retrouve le traitement d'image dans de nombreux domaines comme la photographie, la médecine (analyses d'IRM, radiographies), la surveillance (reconnaissance faciale, caméras de sécurité) ou encore la vision par ordinateur utilisée par les voitures autonomes. Grâce aux avancées en intelligence artificielle, les algorithmes modernes sont capables d'analyser et d'améliorer les images avec une précision impressionnante.

Dans ce premier chapitre, nous présentons quelques notions concernant la super-résolution d'image, telles que sa définition, ses différentes catégories et théories, ainsi que ses composantes. Ensuite, nous décrivons un état de l'art sur les méthodes développées dans le domaine de la super-résolution d'image, et plus particulièrement, nous abordons la notion de Deep Learning et de ses différentes architectures existantes.

1.2 Concepts généraux et définitions

1.2.1 Super-Résolution d'image

La super-résolution d'image (SR) est une technique qui permet d'améliorer la qualité et la netteté des images. Elle se divise en trois grandes catégories [Reshma and Thomas, 2023] :

— *Super-résolution d'image unique (SISR) :* Cette méthode améliore la qualité d'une seule image en recréant des détails manquants, souvent grâce à des modèles d'intelligence artificielle.

— *Super-résolution multi-images (MISR) :* Ici, plusieurs images sont utilisées ensemble pour affiner les détails et améliorer la netteté, ce qui est particulièrement utile pour les séries d'images présentant des liens temporels forts.

— *Super-résolution vidéo(VSR) :* Cette technique optimise la qualité d'une vidéo en exploitant les liens entre les images successives. Elle utilise des méthodes avancées comme la compensation de mouvement et l'analyse des flux optiques pour obtenir une reconstruction plus fluide et précise.

1.2.1.1 Panorama des Techniques de Super-Résolution

1.2.1.2 Super-Résolution d'Image Unique (SISR)

La Super-Résolution d'Image Unique (SISR) a vu l'émergence de plusieurs approches clés pour reconstituer des images haute résolution à partir d'une seule image basse résolution.

— **Méthodes Basées sur les Réseaux Convolutionnels (CNN)** Les **réseaux convolutifs (CNN)** sont fondamentaux en SISR. Ils sont conçus pour extraire les caractéristiques pertinentes des images à basse résolution (LR) et les transformer progressivement en images à haute résolution (HR). Des architectures pionnières comme le S**RCNN (Super-Resolution Convolutional Neural Network)** et ses nombreuses variantes ont démontré des avancées significatives, offrant des améliorations notables en termes de qualité d'image et de finesse des détails [Dong et al., 2014].

— **Modèles de Diffusion (DPM)** Les modèles de diffusion probabilistes (DPM) représentent une approche plus récente et puissante. Ils fonctionnent par un processus itératif de raffinement d'image, ajustant progressivement le bruit ajouté à chaque étape pour converger vers une image de haute qualité. Le modèle **ResDiff**, que nous mentionnons dans ce contexte, illustre bien cette synergie : il combine un DPM avec un CNN. Sa particularité est de prédire les résidus (les différences) entre les images HR réelles et celles initialement prédites par le CNN, ce qui affine considérablement la qualité de la sortie [Rombach et al., 2022], [Saharia et al., 2022].

— **Approches Basées sur les GAN (Generative Adversarial Networks)** Les **réseaux antagonistes génératifs (GAN)**, comme **ESRGAN**, sont particulièrement prisés pour leur capacité à générer des images très réalistes avec des textures convaincantes. Cependant, leur entraînement peut être délicat ; ces modèles sont sensibles aux problèmes de "mode collapse" (manque de diversité des résultats) et nécessitent souvent un temps de convergence plus long [Molini et al., 2019].

1.2.1.3 Multi-Image Super-Resolution (MISR)

La Super-Résolution Multi-Image (MISR) consiste à synthétiser une seule image haute résolution plus détaillée en exploitant les informations issues de plusieurs images basse résolution. Ce domaine a connu des innovations significatives grâce à l'apprentissage profond :

— **DeepSUM** : Méthode révolutionnaire, DeepSUM a été **la première à employer la convolution 3D** pour fusionner efficacement les données de diverses images d'entrée, établissant ainsi un précédent pour les recherches ultérieures [Molini et al., 2019].

— **RAMS** : Cette technique a intégré **un mécanisme d'attention basé sur la convolution 3D**, conçu pour prioriser les informations à haute résolution. Malgré cela, elle était notablement sensible à la séquence temporelle des images basse résolution [Salvetti et al., 2020].

— **TR-MISR :** En proposant un **module de réarrangement des caractéristiques**, TR-MISR visait à traiter tous les patchs d'image. Cependant, il a été observé qu'il sous-estimait les corrélations spatiales cruciales [An et al., 2022].

— **PIU** : Cette méthode s'est concentrée sur l'**estimation de l'incertitude d'une image super-résolue**, améliorant la qualité en tirant parti de la variabilité inhérente aux images d'entrée. Un inconvénient notable était son omission de l'influence de l'ordre des images.

— **3DWDSRNet MISR-GRU** : Celles-ci représentent des stratégies supplémentaires basées sur l'apprentissage profond qui ont exploré différentes voies pour réaliser la super-résolution multi-image.

— **Le framework ESC-MISR :** Développé pour surmonter les limitations antérieures, ESC-MISR met fortement l'accent sur la correction et l'exploitation des **corrélations spatiales** tout en atténuant simultanément les dépendances temporelles indésirables, grâce à son module MIST et une stratégie de mélange intelligente [Salvetti et al., 2020].

1.2.1.4 Video Super-Resolution (VSR)

La Super-Résolution Vidéo (VSR) vise à améliorer la qualité d'une séquence vidéo en augmentant sa résolution. Pour ce faire, diverses architectures d'apprentissage profond sont employées [Reshma and Thomas, 2023] :

— **Réseaux de neurones convolutifs (CNN)** : Ils sont couramment utilisés pour exploiter les informations à la fois **temporelles et spatiales** contenues dans les vidéos.

— **Réseaux antagonistes génératifs (GAN)** : Ces modèles sont mis à profit pour **générer des informations de haute qualité** et des textures réalistes, contribuant ainsi à une amélioration visuelle significative [Ledig et al., 2017].

— **Spatial-Temporal Transformer (STT)** : Cette approche innovante s'appuie sur une architecture de **Transformers** pour analyser les relations complexes spatio-temporelles au sein d'une séquence vidéo. Cela permet une meilleure compréhension des liens entre les différentes parties de l'image et entre les images successives, améliorant ainsi la reconstruction des détails et la qualité globale.

— Autres architectures notables : Des modèles tels que le **GRU (Gated Recurrent Unit), le RBPN (Recurrent Back-Projection Network)**, et le **TDAN (Temporally-Deformable Alignment Network)** sont également utilisés en VSR, chacun apportant ses propres avantages et s'adaptant à des applications spécifiques [Reshma and Thomas, 2023].

1.2.1.5 Pourquoi choisir la Super-Résolution d'Image Unique (SISR) ?

Le choix de se concentrer sur la Super-Résolution d'Image Unique (SISR), plutôt que sur la super-résolution vidéo ou multi-image, repose sur plusieurs avantages stratégiques [Shang et al., 2023] :

Simplicité de modélisation : La SISR permet une meilleure maîtrise des algorithmes et de leur évaluation. Elle élimine la complexité supplémentaire liée à la cohérence temporelle des vidéos ou à la fusion d'informations hétérogènes provenant de multiples images.

Applicabilité immédiate : La SISR est directement pertinente pour de nombreuses applications pratiques. Pensez à la photographie, la médecine ou la surveillance, où l'amélioration d'une seule image est un besoin fréquent et crucial.

Complexité de données réduite : Contrairement aux vidéos, qui présentent des défis liés aux variations de lumière, aux mouvements et aux occlusions, une image unique offre un cadre de traitement plus stable et moins contraignant.

Optimisation ciblée des modèles : Les algorithmes de SISR peuvent être spécifiquement conçus pour être extrêmement efficaces et rapides, sans la nécessité impérative de traiter des flux de données en temps réel comme c'est le cas pour la vidéo.

Facilitateur de recherche et d'innovation : L'étude de la SISR constitue un terrain idéal pour expérimenter et valider de nouvelles approches. Les avancées réalisées dans ce domaine peuvent ensuite être étendues et adaptées à des scénarios plus complexes, tels que la

super-résolution vidéo.

En somme, la SISR est privilégiée pour sa **simplicité de gestion**, son **utilité directe** et sa capacité à servir de laboratoire efficace pour l'expérimentation et le développement d'algorithmes avant de s'attaquer à des défis plus ardus.

1.2.2 Description du modèle ResDiff pour la Super-Résolution d'Image (SISR)

ResDiff est un modèle de super-résolution d'image novateur qui s'appuie sur les diffusions probabilistes pour générer des images haute résolution (HR) à partir d'images basse résolution (LR) avec une qualité et une efficacité accrues. Sa principale force réside dans son architecture combinée, qui intègre intelligemment un réseau de neurones convolutifs **(CNN)** pour la restauration des composantes de basse fréquence, et un modèle de diffusion probabiliste (DPM). Contrairement aux approches de diffusion classiques, **ResDiff** tire parti de la prédiction initiale du CNN pour orienter le processus de débruitage du **DPM** vers l'espace de résidus, accélérant ainsi la génération et améliorant la qualité des images produites. Pour affiner la restauration des détails, **ResDiff** introduit également une fonction de perte spécifique basée sur le domaine de fréquence, permettant au modèle de se concentrer plus efficacement sur les hautes fréquences [Shang et al., 2023].

Table 1.1 – Résultats d'Évaluation des Modèles de SISR sur Divers Datasets

Méthode	**Dataset/Échelle**	**PSNR↑**	**SSIM↑**	**FID↓**
SRGAN	FFHQ (32 → 128)	17.57	0.688	156.07
ESRGAN	FFHQ (32 → 128)	15.43	0.267	166.36
PULSE	CelebA (64 → 256)	–	–	40.33
SRFlow	CelebA (20 → 160)	25.28	0.720	–
SRDiff	FFHQ (32 → 128)	26.07	0.794	72.36
SRDiff	FFHQ (256 → 1024)	23.01	0.656	56.17
SR3	FFHQ (32 → 128)	25.37	0.778	75.29
SR3	FFHQ (256 → 1024)	22.78	0.647	60.12
SR3	CelebA (20 → 160)	24.89	0.728	83.11
SR3	CelebA (64 → 256)	26.04	0.779	43.27
SRDiff	CelebA (20 → 160)	25.32	0.730	80.98
SRDiff	CelebA (64 → 256)	26.84	0.792	39.16
SR3	DIV2K (40 → 160)	26.17	0.650	111.45
SRDiff	DIV2K (40 → 160)	26.87	0.690	110.32
SR3	Urban100 (40 → 160)	25.18	0.620	61.14
SRDiff	Urban100 (40 → 160)	26.49	0.790	51.37
ResDiff	FFHQ (32 → 128)	**26.73**	**0.818**	**70.54**
ResDiff	FFHQ (256 → 1024)	**23.15**	**0.668**	**53.23**
ResDiff	CelebA (20 → 160)	**25.37**	**0.734**	**78.52**
ResDiff	CelebA (64 → 256)	**27.16**	**0.797**	**38.47**
ResDiff	DIV2K (40 → 160)	**27.94**	**0.720**	**106.71**
ResDiff	Urban100 (40 → 160)	**27.43**	**0.820**	**42.35**

La Table 1.1 résume la comparaison quantitative des mesures d'évaluation, où les valeurs les plus performantes sont indiquées en gras.

1.2.3 Les métriques d'évaluation d'un model de super-résolution (SISR)

a. **Métriques de régression ou de classification**

— *MSE (Mean Squared Error) :* Utilisé en régression, mais en super-résolution,il ne prend pas en compte la perception visuelle ni la structure de l'image.

— *MAE (Mean Absolute Error) :* Similaire à MSE, mais ne capture pas les relations spatiales importantes dans l'image.

— *Accuracy, F1-score, Precision, Recall* : Adaptés à la classification où l'objectif est de prédire des catégories, mais inutiles pour comparer des images reconstruites.

b. **Métriques PSNR ,SSIM ,et FID** [Reshma and Thomas, 2023]

— *PSNR (Peak Signal-to-Noise Ratio)* : Mesure la qualité de la reconstruction en évaluant la différence entre l'image générée et l'image originale (erreur quadratique moyenne, MSE).

— SSIM (Structural Similarity Index) : Évalue la similarité structurelle entre l'image reconstruite et l'originale en tenant compte de la luminance, du contraste et de la structure.

— FID (Fréchet Inception Distance) : Mesure la qualité perceptuelle et la diversité des images générées en comparant les distributions statistiques des caractéristiques extraites par un réseau neuronal (InceptionV3).

c. **Pourquoi ces métriques sont essentielles pour la super-résolution ?**
Ces métriques sont particulièrement adaptées à la super-résolution pour plusieurs raisons fondamentales :

— *La Super-Résolution n'est Pas une Simple Régression :* C'est bien plus qu'un problème de simple minimisation d'erreur au pixel près. Il s'agit de recréer une image visuellement cohérente et agréable.

— *Les erreurs classiques (MSE, MAE) ne reflètent pas bien la perception humaine :* Deux images avec le même MSE peuvent avoir une qualité perçue très différente.

— *Besoin d'une Mesure de la Diversité :* FID est crucial pour évaluer la **variété des images générées**, ce qui est absent dans la régression/classification.

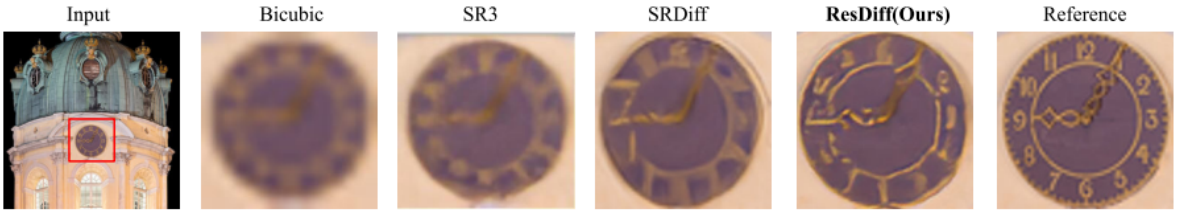

FIGURE 1.1 – Résultats DIV2k 4× [Shang et al., 2023]

En somme, l'évaluation de la super-résolution, comme le confirme la Figure 1.1, nécessite une approche holistique qui combine la fidélité pixel par pixel avec la perception humaine de la qualité et la capacité du modèle à générer des résultats diversifiés et réalistes.

1.2.4 Les algorithmes des différentes catégories de super-résolution (SISR)

Les algorithmes de super-résolution se répartissent en plusieurs catégories distinctes, chacune avec sa propre approche pour transformer les images basse résolution en haute résolution. On retrouve principalement les GANs, les modèles basés sur les Flows, les modèles de Diffusion et les méthodes d'Interpolation (voir la figure 1.2).

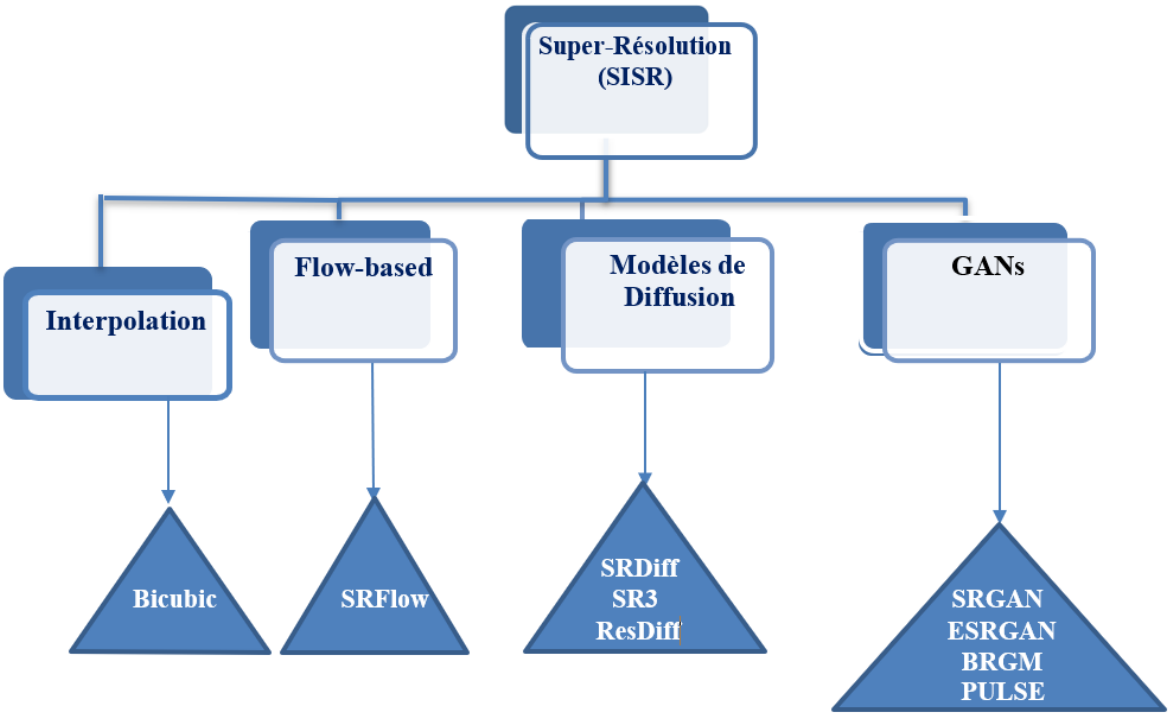

FIGURE 1.2 – Principaux algorithmes utilisés dans la super-résolution (SISR)

Algorithmes Basés sur les GANs (Generative Adversarial Networks) Ces modèles excellent dans la création de détails réalistes grâce à une confrontation entre un générateur et un discriminateur :**SRGAN :** Ce fut le pionnier, le premier GAN appliqué à la super-résolution. Il utilise une **perceptual loss** et une **adversarial loss** pour rendre les images générées plus réalistes, se concentrant sur ce qui est important pour la perception humaine [Ledig et al., 2017]. **ESRGAN :** Une version améliorée de SRGAN, qui intègre un **Residual-in-Residual Dense Block (RRDB)** et une perceptual loss avancée. Ces ajouts lui permettent de produire des textures encore plus fines et réalistes [Wang et al., 2018]. **BRGM :** Ce modèle bayésien se distingue par sa capacité à reconstruire des images HR à partir de **plusieurs images LR**. Il utilise une approche de régularisation hybride pour préserver les contours tout en éliminant les artefacts et le bruit [Dong et al., 2014]. **PULSE :** Ce modèle génère des images HR en optimisant un **espace latent de GAN**. L'objectif est de maximiser la similarité avec l'image basse résolution d'entrée, ce qui le rend particulièrement efficace pour des reconstructions réalistes, notamment de visages [Menon et al., 2020].

Algorithmes Basés sur les Modèles de Diffusion (Diffusion Models) Ces modèles, plus récents, transforment une image bruitée en une image claire par étapes successives :

— **SRDiff :** C'est le premier modèle à avoir appliqué les **diffusions probabilistes à la super-résolution**. Il génère des images HR en partant d'un bruit aléatoire et en le débruitant progressivement [Li et al., 2022].

— **SR3 :** Utilise un **Denoising Diffusion Probabilistic Model (DDPM)** qui est conditionné par l'image LR. Ce conditionnement l'aide à produire des images HR

plus réalistes et fidèles. [Lugmayr et al., 2020]

— **ResDiff :** Le modèle, ResDiff, propose une combinaison innovante entre un **CNN et un modèle de Diffusion**. Le CNN gère la reconstruction des basses fréquences, tandis que le modèle de diffusion se concentre sur la prédiction des résidus (les détails fins) à haute fréquence [Shang et al., 2023].

Algorithmes Basés sur Normalizing Flow Ces modèles apprennent une transformation réversible pour générer des images :

— **SRFlow :** Basé sur les **Normalizing Flow**, ce modèle apprend la distribution des images HR en les conditionnant sur les images LR. Cela lui permet de générer une **plus grande diversité d'échantillons** pour une même image d'entrée [Kawar et al., 2022].

Méthodes d'Interpolation (Baselines) Ces méthodes sont des techniques de base, souvent utilisées comme points de comparaison :

— **Interpolation Bilinéaire :** Méthode simple qui calcule la valeur d'un nouveau pixel en faisant la moyenne des quatre pixels les plus proches. Elle est **efficace en termes de calcul** mais produit des images souvent moins nettes, avec des bords flous.

— **Interpolation Bicubique :** Plus sophistiquée que la bilinéaire, elle prend en compte les 16 pixels les plus proches. Cela permet d'obtenir des images plus fluides et une **meilleure préservation des bords**. Cependant, elle entraîne une perte de détails fins et tend à lisser les textures [Khaledyan et al., 2020].

1.2.5 Réseaux de Neurones Artificiels : Définition, Histoire et Topologies

1.2.5.1 Définition

Les réseaux de neurones artificiels (RNA) sont des systèmes computationnels initialement inspirés par l'analogie avec le fonctionnement des neurones biologiques. Plus qu'une simple copie, ils se sont développés comme une approche statistique distincte [Le Bolzer et al., 2020]. Un RNA est un ensemble de processeurs élémentaires interconnectés (neurones artificiels), opérant en parallèle. Chaque neurone calcule une sortie unique, basée sur les informations qu'il reçoit.

1.2.5.2 Historique

L'histoire des réseaux neuronaux remonte à 1943, lorsque Warren McCulloch et Walter Pitts ont introduit le concept de neurone formel, établissant une équivalence entre le cerveau et une machine de Turing. Cette idée audacieuse a jeté les bases de la cybernétique [McCulloch and Pitts, 1943].

En 1949, Donald Hebb a marqué un tournant avec son livre "The Organization of Behavior", où il a présenté la célèbre règle d'apprentissage de Hebb, qui influence encore de nombreux modèles actuels [Hebb, 2005].

La recherche a connu un essor en 1958 avec l'introduction du Perceptron par Frank Rosenblatt. Inspiré par le système visuel, ce réseau à deux couches (perception et décision) fut la première machine capable d'apprendre par l'expérience [Rosenblatt, 1958]. À la même

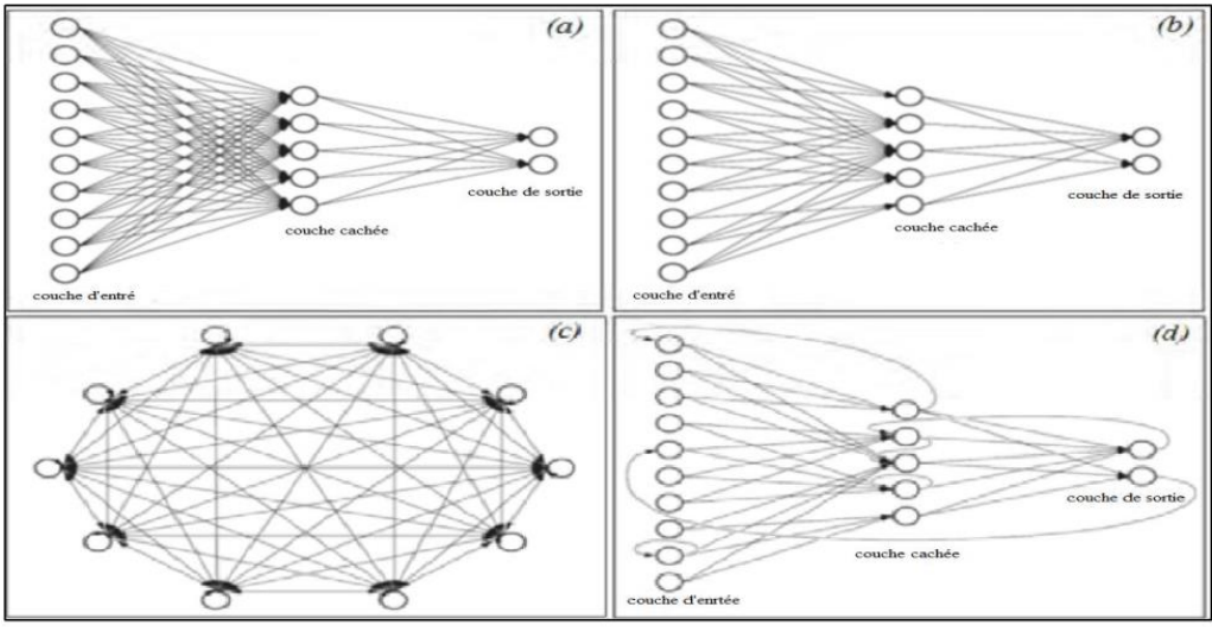

FIGURE 1.3 – Topologie des Réseaux de neurones artificiels [Elsken et al., 2019]

époque, Bernard Widrow a développé l'Adaline, un modèle qui a servi de base aux futurs réseaux multicouches [Alty and Took, 2024].

Cependant, une critique majeure du **Perceptron par Marvin Minsky et Seymour Papert** en 1969 a sévèrement freiné la recherche. Le domaine a connu un ralentissement jusqu'en 1972, avec la publication des travaux de **Teuvo Kohonen** sur les mémoires associatives et leurs applications en reconnaissance de formes [Kohonen, 2012]. Un regain d'intérêt significatif s'est produit en 1982, lorsque John Hopfield a étudié la dynamique des réseaux récurrents complets, relançant ainsi l'exploration des architectures complexes [Hopfield, 1982].

1.2.5.3 Topologies

Aujourd'hui, les réseaux neuronaux sont omniprésents en intelligence et vie artificielles, principalement en raison de leurs **capacités d'apprentissage** et de leur comportement en tant que **systèmes dynamiques** [Dong et al., 2014].

Leur architecture, ou **topologie**, varie considérablement et est cruciale pour leur fonctionnement. On distingue deux grandes catégories de connexion (voir la Figure 1.3).

— **Réseaux non-bouclés (ou statiques)** : Ces réseaux ont un flux d'informations unidirectionnel, sans boucles de rétroaction. Ils sont représentés par les types (a) et (b) [assumant un diagramme externe].

— **Réseaux bouclés (ou dynamiques)** : Ces réseaux intègrent des boucles de rétroaction, permettant à l'information de circuler dans plusieurs directions et de persister dans le temps. Ils sont adaptés aux tâches séquentielles et sont représentés par les types (c) et (d) *(assumant un diagramme externe).*

1.2.6 L'Apprentissage en Profondeur (Deep Learning) : Principes, Évolution et Applications

L'apprentissage en profondeur (Deep Learning, DL) représente un domaine en pleine expansion du Machine Learning (ML), visant à concrétiser l'objectif d'atteindre une véritable intelligence artificielle. Il s'appuie sur des algorithmes inspirés par la structure et le fonctionnement du cerveau humain voir la figure 1.4 [Tahiri, 2024].

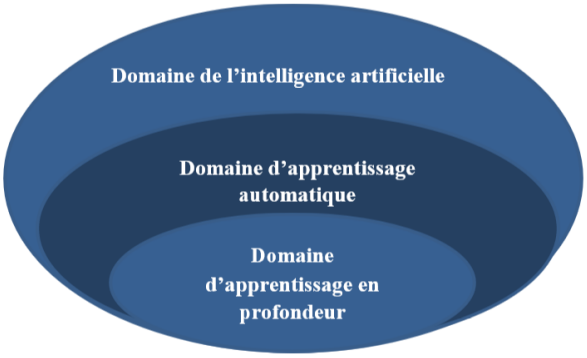

FIGURE 1.4 – La relation entre IA, ML et le Deep learning [Tahiri, 2024]

1.2.6.1 Définition et Fonctionnement

Le Deep Learning regroupe des algorithmes de ML caractérisés par leur capacité à apprendre à travers plusieurs niveaux de représentation, chacun correspondant à un degré d'abstraction croissant. Grâce à ces nombreuses couches de traitement (comme illustré par la figure 1.5), les modèles de DL peuvent extraire automatiquement des caractéristiques complexes à partir de données brutes, en appliquant une série de transformations linéaires et non linéaires. Ce processus itératif affine l'apprentissage progressivement, nécessitant une intervention humaine minimale [Schmidhuber, 2022].

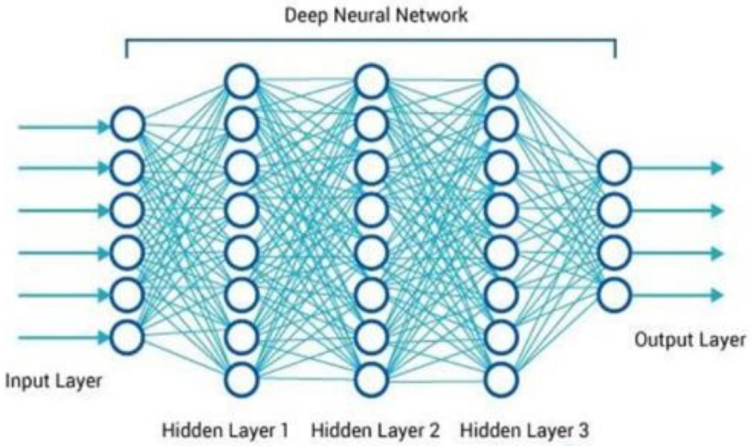

FIGURE 1.5 – Schéma illustratif de DL avec plusieurs couches [Elsken et al., 2019]

1.2.6.2 Historique et Avènement

L'idée du Deep Learning n'est pas nouvelle, ses racines remontant aux années 1980 avec les travaux sur les réseaux de neurones multicouches, notamment ceux du Français Yann Le Cun, aux côtés de Kunihiko Fukushima et Geoffrey Hinton, pionniers du développement **des réseaux de neurones convolutifs (CNN)**. Malgré des promesses initiales, l'approche a été entravée pendant près de deux décennies par les limitations en puissance de calcul et la rareté des grandes bases de données [Deekshith Shetty et al., 2020].

Le renouveau du DL est directement lié aux avancées technologiques et à la disponibilité croissante de vastes volumes de données. Un moment clé fut la création d'**ImageNet** par le Stanford Vision Lab en 2007 (comprenant 15 millions d'images annotées en 2010), qui a fourni les données nécessaires à l'entraînement de modèles complexes [ImageNet, 2025]. Le retour spectaculaire du Deep Learning sur le devant de la scène s'est concrétisé en 2012, lorsqu'un algorithme d'apprentissage profond a remporté l'**ImageNet** Large Scale Visual Recognition Challenge (ILSVRC), surpassant les méthodes traditionnelles et marquant le début de son essor fulgurant [ImageNet, 2025].

1.2.6.3 Justification du Choix du Deep Learning

Le développement du Deep Learning est une réponse directe aux limites des algorithmes de Machine Learning classiques face aux problèmes complexes de l'IA. Son adoption est motivée par plusieurs avantages clés (comme suggéré par la Figure 1.6, comparant ML et DL) :

— **Amélioration des performance**s sur des tâches IA complexes.
— **Capacité à exploiter efficacement les Big Data**.
— **Grande adaptabilité** à une variété de problèmes.
— **Automatisation de l'extraction de caractéristiques**, réduisant le besoin d'ingénierie manuelle [Bengio et al., 2013].

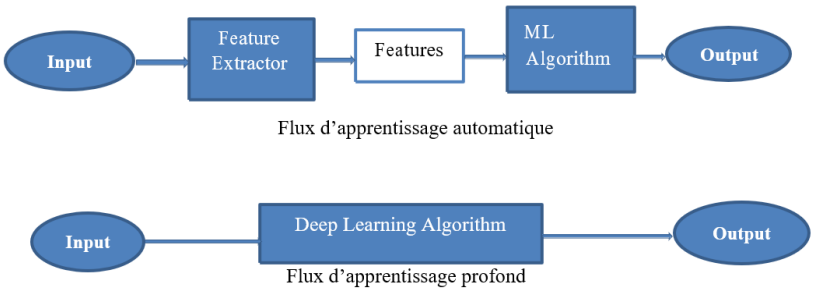

FIGURE 1.6 – Comparativeentre machine Learning et Deep Learning.

Le diagramme dans la figure 1.6 est très clair et efficace pour illustrer la distinction fondamentale entre un flux d'apprentissage automatique (Machine Learning traditionnel) et un flux d'apprentissage profond (Deep Learning).

1.2.7 Architectures Fondamentales du Deep Learning

Le Deep Learning est caractérisé par une diversité d'architectures, bien que leur comparaison directe soit complexe en raison de l'évolution rapide du domaine et des variations dans les jeux de données de test.

Réseaux de Neurones Convolutifs (CNN) Les CNNs sont spécifiquement conçus pour traiter les données structurées en grille, comme les images. Ils sont essentiels pour la reconnaissance, la classification, et la super-résolution d'images et de vidéos, permettant des avancées dans l'identification d'objets, la conduite autonome, et même le traitement du langage naturel (Figure 1.7 pour l'architecture SRCNN). Un CNN typique est un réseau feed-forward composé de couches convolutives et de fonctions d'activation (ex : ReLU), capable de capturer des motifs visuels complexes et d'apprendre automatiquement sans extraction manuelle de caractéristiques [Bengio et al., 2013].

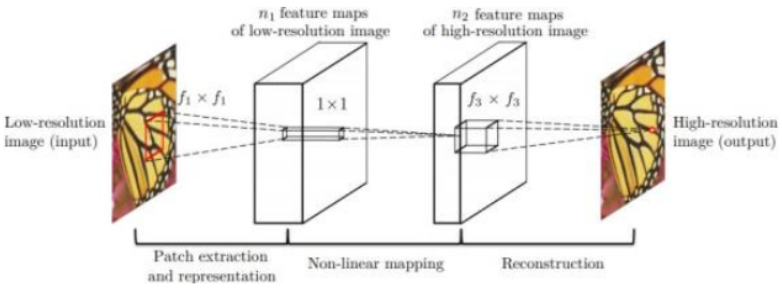

FIGURE 1.7 – Architecture SRCNN [Garber et al., 2020].

Réseaux de Neurones Récurrents (RNN) Les RNNs sont particulièrement adaptés au traitement des données séquentielles (séries temporelles, textes) où l'ordre des éléments est primordial. Grâce à leur "mémoire interne", les RNNs retiennent des informations sur les entrées passées, ce qui les rend efficaces pour la prédiction d'éléments futurs dans une séquence (Figure 1.8 pour l'architecture DSRN). En super-résolution d'image, ils permettent la conception de modèles compacts et efficaces pour le traitement progressif et l'apprentissage à long terme [Kim et al., 2016]. Leurs applications incluent la modélisation linguistique, la traduction automatique, la reconnaissance vocale, et la photographie mobile [Le Bolzer et al., 2020].

Modèles Génératifs Contrairement aux modèles discriminatifs (comme les CNN et RNN) qui prédisent des étiquettes, les modèles génératifs apprennent à produire des données en modélisant leur distribution de probabilité sous-jacente [Kim et al., 2016]. Des exemples notables incluent les Boltzmann Machines et leurs variantes, ainsi que les Generative Adversarial Networks (GANs). Il est important de noter que certains modèles comme SRGAN apprennent une fonction de transformation directe sans modéliser explicitement une distribution de probabilité, la figure 1.9 [Shi et al., 2021] illustre l'architecture (GAN).

1.2.8 Exemples d'Applications du Deep Learning

Les applications du Deep Learning sont vastes et transforment de nombreux secteurs [Tahiri, 2024] :

— Super-résolution et restauration d'images.
— Colorisation d'images noir et blanc.
— Intégration d'effets sonores et de bandes-son à des films muets.
— Traduction automatique.

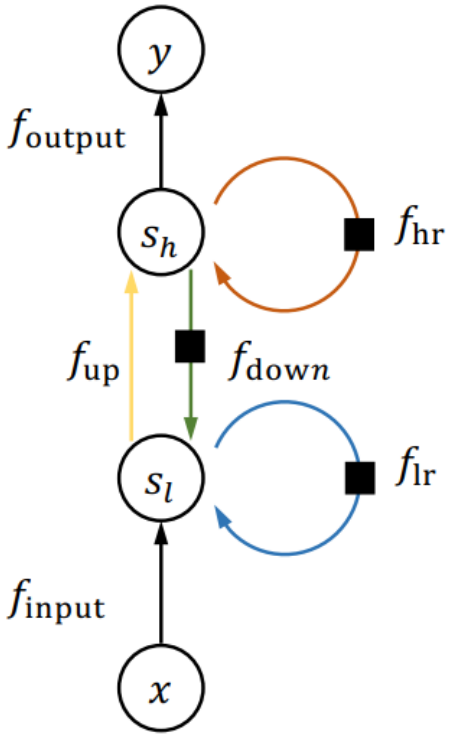

FIGURE 1.8 – Architecture DSRN [LeewayHertz, 2025].

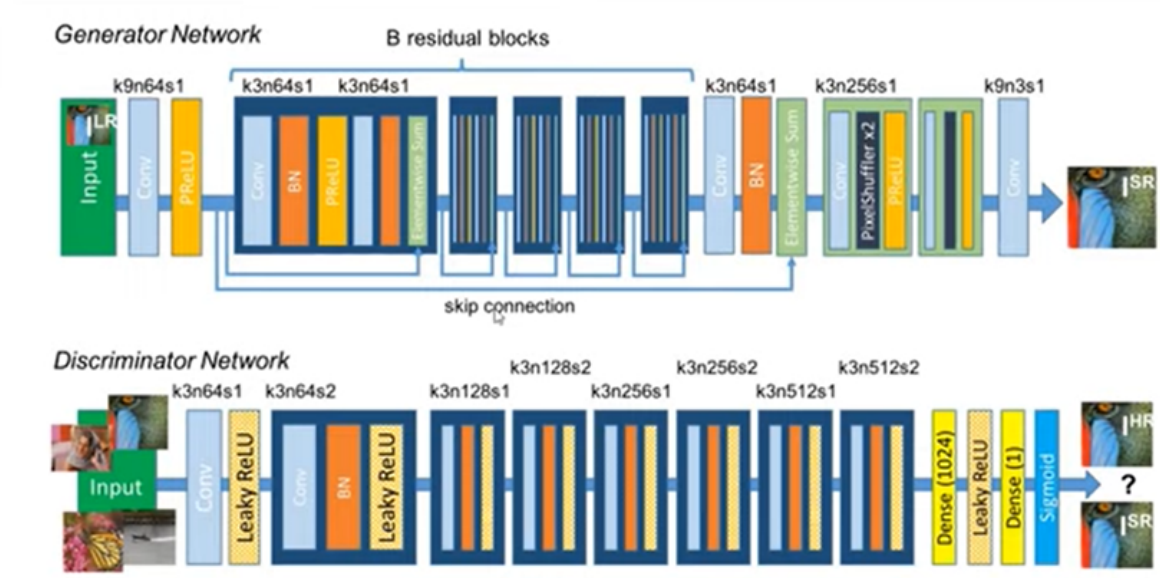

FIGURE 1.9 – Architecture du réseau générateur et du réseau discriminateur, avec la taille des noyaux, le nombre de cartes de caractéristiques (n) et le pas (s) indiqués pour chaque couche convolutionnelle [Shi et al., 2021]

.

— Classification d'objets dans des photographies.
— Génération d'écriture automatique et de légendes pour les images.

— Jeu automatisé dans des environnements complexes.

1.3 Travaux Connexes

Le domaine de la Super-Résolution d'Image Unique (SISR) a connu une progression significative ces dernières années, notamment grâce à l'avènement des méthodes basées sur l'apprentissage profond. Ces approches visent à reconstruire une image de haute résolution à partir d'une unique image de basse résolution, un défi fondamental en vision par ordinateur. La littérature foisonne de techniques innovantes, que nous synthétisons et comparons dans la Table 1.2 afin de contextualiser notre propre contribution.

La Table 1.2 ci-dessous offre une synthèse comparative des principales méthodes de Super-Résolution d'Image (SISR), détaillant leurs techniques, avantages, limites et résultats notables. Ce panorama met en lumière l'évolution du domaine, des approches basées sur les GAN (telles que SRGAN et ESRGAN) qui génèrent des textures réalistes malgré des défis de stabilité, aux modèles de diffusion (comme SRDiff et SR3) qui offrent une qualité supérieure mais à un coût computationnel élevé. Le modèle, ResDiff, se distingue en proposant une synergie hybride pour des détails plus riches et une convergence accélérée, cherchant ainsi à surmonter les compromis actuels.

Méthode	**Techniques Principales**	**Avantages**	**Limites**	**Résultats Notables / Caractéristiques Clés**
SRGAN [Ledig et al., 2017]	GAN, Perceptual Loss, Adversarial Loss	Génère des détails visuellement réalistes ; capable de capturer des détails complexes.	Difficulté de généralisation ; risque de mode collapse ; artefacts visuels.	Bonne qualité visuelle, mais avec une convergence lente.
ESRGAN [Wang et al., 2018]	GAN, Residual-in-Residual Dense Block (RRDB), Perceptual Loss	Génère des bords plus nets et des textures plus réalistes.	Sujet aux artefacts perceptuels.	Amélioration significative par rapport à SRGAN.
SRFlow [Kawar et al., 2022]	Normalizing Flow, Conditional Invertible Neural Network (cINN)	Offre une bonne diversité d'échantillons générés.	Coût computationnel élevé.	Bonne diversité des résultats, mais coûteux en ressources.
PULSE [Menon et al., 2020]	GAN, Latent Space Optimization, Progressive Upscaling	Optimisé pour la reconstruction de visages.	Moins fidèle aux détails exacts de l'image originale.	Produit des textures faciales de bonne qualité.
SRDiff [Li et al., 2022]	Diffusion Probabilistic Model (DPM), U-Net, Conditional Guidance	Génère des détails fins.	Lenteur inhérente au processus de diffusion.	Bonne restauration des informations de haute fréquence.
SR3 [Lugmayr et al., 2020]	Diffusion Probabilistic Model (DPM), U-Net, Noise Conditioning	Produit des images très réalistes.	Coût computationnel élevé.	Génération d'images de haute qualité.
ResDiff [Shang et al., 2023]	CNN pré-entraîné, Diffusion Model, Residual Learning, Frequency-Domain Guidance (FFT & DWT), Cross-Attention	Génère des détails plus riches ; convergence plus rapide.	Complexité de calcul potentielle due au traitement dans le domaine de fréquence.	Meilleure convergence et amélioration significative des détails fins par rapport aux approches existantes.

Table 1.2 – Panorama comparatif des techniques de SISR basées sur l'Apprentissage Profond

Historiquement, les premières avancées majeures en SISR basée sur le Deep Learning ont été marquées par les réseaux génératifs antagonistes (GANs). Des méthodes comme SRGAN [Ledig et al., 2017] ont révolutionné la génération d'images en introduisant une

perceptual loss et une adversarial loss, permettant de produire des détails visuellement très réalistes, bien que confrontées à des défis de généralisation, de mode collapse et à la présence d'artefacts visuels. ESRGAN [Wang et al., 2018] a ensuite amélioré cette approche en intégrant des Residual-in-Residual Dense Blocks (RRDB) et une perceptual loss avancée, conduisant à des bords plus nets et des textures encore plus réalistes, bien qu'il reste sujet à certains artefacts perceptuels. Des travaux plus spécialisés, tels que PULSE [Menon et al., 2020], ont exploité l'optimisation dans l'espace latent des GANs pour des tâches spécifiques comme la reconstruction de visages, produisant des textures de haute qualité au détriment parfois de la fidélité exacte à l'original.

Plus récemment, les modèles basés sur les Normalizing Flow et les modèles de diffusion ont émergé comme des paradigmes puissants pour la super-résolution. SRFlow [Kawar et al., 2022] est un exemple notable d'architecture basée sur Normalizing Flow, capable d'apprendre la distribution des images haute résolution conditionnées par les images basse résolution, offrant ainsi une bonne diversité dans les échantillons générés, malgré un coût computationnel élevé. Les modèles de diffusion probabiliste (DPM), tels que SRDiff [Li et al., 2022] et SR3 [Lugmayr et al., 2020], ont démontré une capacité exceptionnelle à générer des détails très fins et à produire des images d'un réalisme frappant grâce à un processus de débruitage itératif. Leurs principaux avantages résident dans leur qualité de restauration des informations de haute fréquence et leur réalisme, mais leur nature séquentielle entraîne une lenteur inhérente et un coût computationnel généralement élevé.

Notre contribution, ResDiff [Shang et al., 2023], s'inscrit dans cette lignée de recherche en proposant une synergie hybride innovante. Comme le détaille la Table 1.2, ResDiff combine un réseau de neurones convolutif (CNN) pré-entraîné pour la reconstruction des basses fréquences avec un modèle de diffusion pour la prédiction des résidus haute fréquence. En intégrant des techniques comme le Residual Learning et le Frequency-Domain Guidance (via FFT et DWT), ainsi que le Cross-Attention, ResDiff vise à surmonter les compromis actuels des méthodes existantes. Cette approche permet de générer des détails plus riches et d'atteindre une convergence plus rapide, offrant une amélioration significative de la qualité des détails fins par rapport aux modèles de diffusion purs, tout en gérant la complexité de calcul potentielle liée au traitement dans le domaine fréquentiel.

En somme, cette section des travaux connexes a mis en évidence l'évolution constante des méthodes de SISR, des architectures basées sur les GANs axées sur le réalisme aux modèles de diffusion privilégiant la qualité et la diversité, en passant par les approches basées sur les Flows. Notre travail, ResDiff, s'appuie sur ces avancées en proposant une solution novatrice qui cherche à optimiser à la fois la qualité des détails et l'efficacité du processus de super-résolution.

1.4 Conclusion

Ce chapitre a posé les fondations nécessaires à la compréhension de la super-résolution d'image. Nous avons commencé par définir ce qu'est la super-résolution, avant d'explorer les principales approches algorithmiques existantes (telles que l'interpolation, les GANs, les modèles basés sur les Flows et les modèles de diffusion), ainsi que les métriques d'évaluation couramment employées dans ce domaine. Nous avons ensuite détaillé les principes de l'apprentissage en profondeur (Deep Learning), en soulignant ses distinctions et avantages par rapport aux méthodes classiques d'apprentissage automatique. Enfin, nous avons

passé en revue les architectures fondamentales des réseaux de neurones profonds, à savoir les réseaux convolutifs, les réseaux récurrents et les modèles génératifs, ces derniers représentant un axe de recherche particulièrement dynamique.

Le chapitre suivant s'appuiera sur ces concepts en approfondissant l'étude des modèles de réseaux de neurones spécifiquement adaptés et utilisés dans le cadre de notre approche.

Chapitre 2

Méthodologie et Conception du Modèle ResDiff pour la SISR

2.1 Introduction

Le modèle ResDiff introduit une **approche hybride novatrice** pour la super-résolution d'image, combinant les atouts reconnus des **Efficient Sub-Pixel Convolutional Neural Networks (ESPCN)** avec la capacité des **modèles de diffusion**. Cette synergie vise à tirer parti de la rapidité et de l'approximation robuste des ESPCN pour l'upsampling initial, tout en exploitant la puissance des modèles de diffusion pour générer des détails fins et une diversité de textures, surpassant ainsi les compromis inhérents aux approches monolithiques.

2.2 Solution proposée "Architecture Hybride de ResDiff"

2.2.1 L'Efficient Sub-Pixel Convolutional Neural Network (ESPCN)

L'Efficient Sub-Pixel Convolutional Neural Network (ESPCN) est une architecture pionnière en super-résolution d'image, proposée par Wenzhe Shi et al. en 2016 lors de la conférence CVPR [Shi et al., 2016]. Ce réseau de neurones convolutifs a été spécifiquement conçu pour effectuer une suréchantillonage (upsampling) des images basse résolution de manière exceptionnellement rapide et efficace. L'ESPCN a marqué un tournant en étant le premier réseau convolutif capable de traiter des vidéos HD (1080p) en temps réel sur un unique GPU, établissant ainsi une nouvelle référence en termes de célérité et d'efficience computationnelle pour les tâches de super-résolution d'images et de vidéos [Shi et al., 2016].

2.2.1.1 Présentation et Architecture de l'ESPCN

L'Efficient Sub-Pixel Convolutional Neural Network (ESPCN) est un modèle de super-résolution d'image proposé par Wenzhe Shi et al. en 2016 lors de la conférence CVPR (Conference on Computer Vision and Pattern Recognition). Ce réseau de neurones convolutifs a été spécialement conçu pour produire des images haute résolution à partir d'entrées basse résolution, de manière plus rapide et plus efficace que les approches précédentes.

L'ESPCN a marqué un tournant en étant le premier réseau convolutif capable de traiter des vidéos HD (1080p) en temps réel sur un unique GPU, établissant une nouvelle référence en termes de rapidité et d'efficacité pour la super-résolution d'images et de vidéos [Shi et al., 2016].
L'architecture de l'ESPCN (illustrée par la Figure 2.1 permet d'augmenter efficacement la taille de l'image tout en réduisant la complexité computationnelle, puisque l'ensemble du traitement est effectué dans l'espace basse résolution (LR) jusqu'à la toute dernière étape de convolution sub-pixel et de "shuffling". Contrairement aux réseaux convolutifs classiques comme le SRCNN, qui interpolent d'abord l'image LR à la taille HR avant traitement, l'ESPCN adopte une stratégie inverse et plus efficiente. Le SRCNN applique des convolutions sur l'image HR interpolée, augmentant inutilement la dimension des données et engendrant une complexité computationnelle plus élevée ainsi qu'une propagation d'erreurs due à l'interpolation.

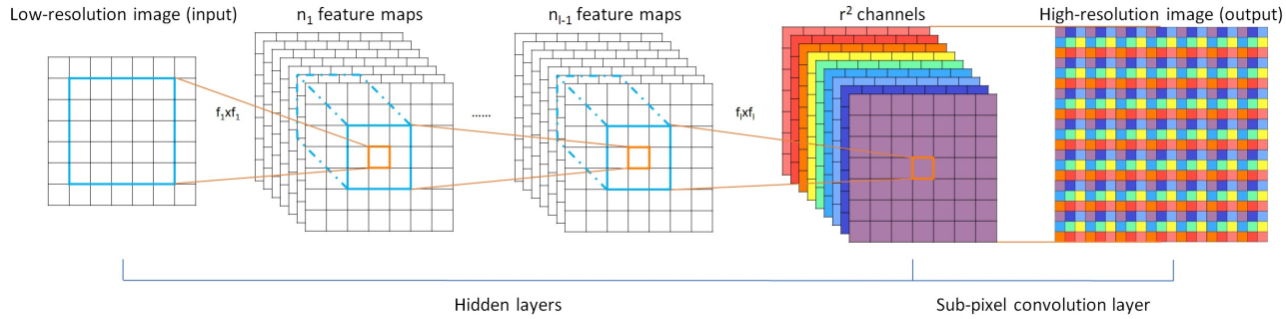

FIGURE 2.1 – Le réseau de neurones convolutif efficace à sous-pixels (ESPCN) proposé, comprend deux couches de convolution pour l'extraction des cartes de caractéristiques, suivies d'une couche de convolution sub-pixel. Cette dernière agrège les caractéristiques de l'espace basse résolution (LR) et reconstruit l'image super-résolue (SR) en une seule étape

En comparaison, l'ESPCN effectue toutes ses opérations dans l'espace basse résolution, réduisant considérablement le volume de calculs. Son architecture comprend trois couches :

— Une première convolution $5 * 5$ avec 64 filtres pour extraire les caractéristiques locales.

— Suivie d'une convolution $3 * 3$ avec 32 filtres pour affiner ces caractéristiques.

— Puis la dernière couche $3 * 3$ avec r^2 filtres, , où r est le facteur d'agrandissement souhaité, qui prépare les cartes de caractéristiques à être réorganisées via la couche sub-pixel.
De plus, ESPCN utilise une activation **tanh** en sortie afin de mieux capturer les détails fins de l'image reconstruite.

2.2.1.2 Couche Sub-Pixel Convolution

La couche sub-pixel de l'ESPCN est cruciale : plutôt que de générer directement l'image haute résolution (HR), elle produit un ensemble de r^2 cartes de caractéristiques en basse résolution. Ces cartes sont ensuite réorganisées par une opération appelée "periodic shuffling", qui consiste à redistribuer les pixels de ces cartes à leurs positions spatiales correctes dans l'image finale. Ainsi, chaque pixel de l'image HR est reconstruit à partir d'une valeur différente extraite des r^2 cartes produites par cette couche ; r^2 cartes produites par la couche sub-pixel en appliquant `tf.nn.depth_to_space`. (voir Figure 2.2)

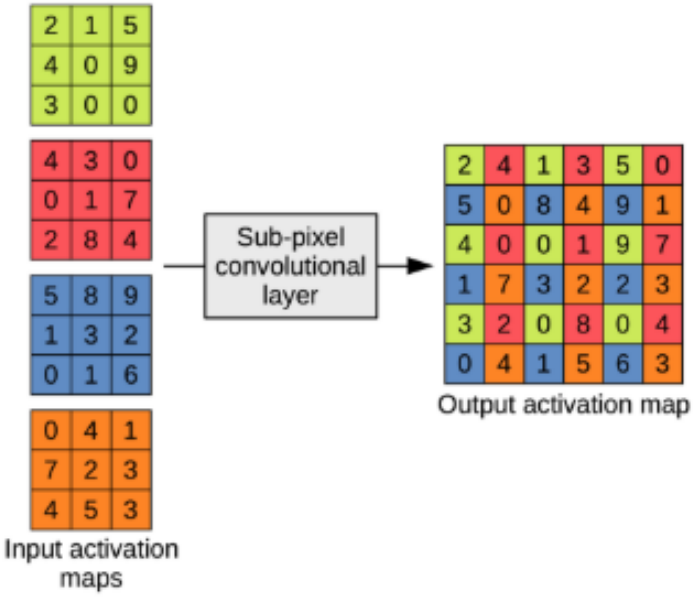

FIGURE 2.2 – Couche sub-pixel

2.2.1.3 Fonction d'Activation de Sortie

L'ESPCN utilise la fonction d'activation Tanh en sortie. Expérimentalement, Tanh donne de meilleurs résultats que ReLU en super-résolution car elle transforme les valeurs pour qu'elles soient entre -1 et 1, permettant au réseau de mieux capturer les petits détails *"fins"* des images. Contrairement à **ReLU** qui supprime toutes les valeurs négatives, **Tanh** évite la perte d'informations fines.

2.2.1.4 Fonction de Perte et Optimisation

Pour l'entraînement, l'**ESPCN** utilise la fonction de perte L1, également appelée erreur absolue moyenne (MAE). C'est une fonction de perte couramment utilisée en régression qui mesure la moyenne des écarts absolus entre les prédictions du modèle et les valeurs réelles. Elle permet d'évaluer la précision globale tout en étant moins sensible aux valeurs extrêmes que d'autres fonctions comme la MSE. L'optimiseur Adam *(Adaptive Moment Estimation)* est employé. Cet algorithme ajuste automatiquement le taux d'apprentissage de chaque paramètre en tenant compte des gradients passés, combinant les premier et deuxième moments pour stabiliser la direction de mise à jour et la dispersion des gradients. Cette approche favorise une convergence plus rapide et plus efficace vers une solution optimale.

2.2.1.5 Rôle de l'ESPCN au sein de ResDiff

Au sein de l'architecture ResDiff, l'ESPCN joue un rôle fondamental en agissant comme un bloc de prédiction initiale. Sa fonction principale est de générer une première version, rapide et cohérente, de l'image à haute résolution à partir de l'entrée basse résolution. L'ESPCN assure la tâche d'`upsampling` efficace et de reconstruction des basses fréquences de l'image. Il travaille ainsi en étroite synergie avec le modèle de diffusion, ce dernier se chargeant d'affiner cette première prédiction en y ajoutant les détails haute fréquence manquants et en garantissant une qualité perceptuelle supérieure. Cette complémentarité

permet à ResDiff d'optimiser conjointement la qualité visuelle, la rapidité d'inférence et la précision de la super-résolution d'image.

2.2.2 Le Modèle de Diffusion dans ResDiff

2.2.2.1 Introduction aux Modèles de Diffusion

Un modèle de diffusion est un générateur d'images qui apprend à transformer progressivement du bruit en une image haute résolution. Dans le contexte de ResDiff, le modèle de diffusion est spécialement conçu pour **reconstruire uniquement les détails fins**, en travaillant sur le résidu entre l'image prédite par l'ESPCN et l'image cible réelle [Shang et al., 2023]. Il fonctionne en deux phases principales :

— **Diffusion directe (forward process)** : L'image haute résolution est progressivement bruitée jusqu'à devenir un bruit gaussien pur.

— **Processus inverse (reverse process)** : Le modèle apprend à inverser ce processus en supprimant progressivement le bruit pour reconstruire une image réaliste.
Les modèles de diffusion se déclinent en plusieurs variantes, incluant les modèles probabilistes de débruitage par diffusion (DDPMs), les modèles génératifs basés sur le score (SGMs), et les équations différentielles stochastiques basées sur le score (Score SDEs).

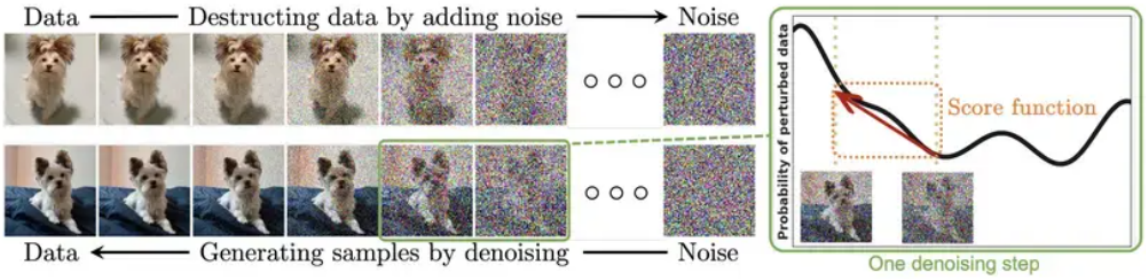

FIGURE 2.3 – Les modèles de diffusion perturbent progressivement les données en ajoutant du bruit, puis inversent ce processus pour générer de nouvelles données à partir du bruit. Chaque étape de débruitage dans le processus inverse nécessite généralement d'estimer la fonction de score (voir la figure illustrative à droite), laquelle correspond à un gradient pointant dans la direction des données ayant une plus grande probabilité et moins de bruit [Superannotate, 2025]

2.2.2.2 Processus de Diffusion de l'Algorithme (Spécifique à ResDiff)

L'algorithme de diffusion au sein de ResDiff repose sur un modèle Conditional U-Net intégré à un processus de diffusion probabiliste. Son rôle est de prédire le bruit ajouté à une image haute résolution au fil du temps, puis de le corriger progressivement afin de reconstruire une image finale nette et réaliste.
Les paramètres principaux de ce processus de diffusion sont :

— $T = 1000$: le nombre total d'étapes de diffusion.

— β_t (*beta schedule*) : le coefficient de variance du bruit ajouté à chaque étape t. β_t varie linéairement entre 10^{-4} et 0.02.

— $\alpha_t = 1 - \beta_t$: α_t représente la proportion conservée de l'image originale à chaque étape ; plus β_t est élevé, plus α_t est petit.

— $\hat{\alpha}_t = \prod_{k=1}^{t} \alpha_k$: un produit cumulatif qui indique la proportion de l'image initiale restante à l'étape t.

La fonction de bruitage, `q_sample`(x_0, t), permet de générer une version bruitée x_t de l'image HR selon la formule :

$$x_t = \sqrt{\hat{\alpha}_t} \cdot x_0 + \sqrt{1 - \hat{\alpha}_t} \cdot \epsilon \tag{2.1}$$

Où : x_0 est l'image HR propre, ϵ est un bruit aléatoire normal $\mathcal{N}(0, I)$, et x_t est l'image bruitée simulée à l'étape t.
Ce processus implique l'ajout progressif de bruit gaussien à chaque étape, de x_0 à x_T. Cette injection répétée altère graduellement l'image d'origine, la rendant de plus en plus dégradée ou pixellisée à mesure que l'on avance vers x_T (Figure 2.3). Chaque étape de débruitage dans le processus inverse nécessite l'estimation d'une fonction de score, correspondant à un gradient indiquant la direction des données ayant une plus grande probabilité et moins de bruit (Figure 2.3).

2.2.2.3 Le Modèle U-Net et son Adaptation Conditionnelle

Le **U-Net** est un modèle de réseau de neurones convolutif conçu spécifiquement pour les tâches de vision par ordinateur, notamment la segmentation sémantique, qui consiste à attribuer une étiquette de classe à chaque pixel d'une image pour une prédiction *dense* [Chen et al., 2023]. Son architecture en forme de "U" est particulièrement adaptée grâce à deux parties symétriques (Figure 2.4) [Khattab et al., 2023], [Khaledyan et al., 2020] :

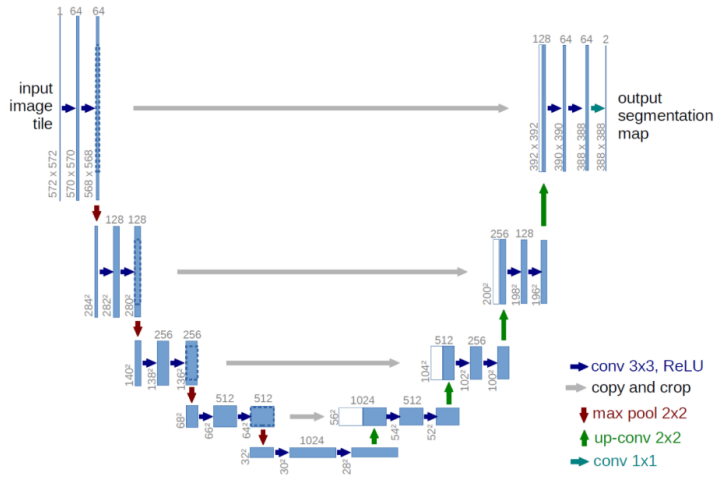

FIGURE 2.4 – Architecture modèle U-Net [Chen et al., 2023]

Encodeur (chemin de contraction) : Extrait le contexte de l'image en réduisant progressivement ses dimensions et en capturant les caractéristiques pertinentes via des couches de convolution et de `max pooling`.

Décodeur (chemin d'expansion) : Reconstruit l'image à sa résolution d'origine, permettant une localisation précise grâce à des *up-convolutions* (convolutions transposées). Des *connexions de saut* (*skip connections*) sont ajoutées après chaque étape de convolution et ReLU, reliant l'encodeur au décodeur pour préserver les détails fins de l'image.

2.2.2.4 Intégration dans ResDiff

Au sein de `ResDiff`, un **U-Net conditionnel** (*Conditional U-Net*) est employé, prolongeant l'architecture U-Net classique en intégrant une information supplémentaire, dite *condition*. Ce U-Net conditionnel prédit le bruit $\hat{\epsilon}$ contenu dans x_t, tout en étant conditionné par l'image basse résolution (LR) (*Voir Eq (2.2)*) :

$$\hat{\epsilon} = f_\theta(x_t, t, \text{LR}) \tag{2.2}$$

où :

- f_θ représente le modèle U-Net conditionnel
- x_t est l'image bruitée à l'étape t
- LR est l'image basse résolution

2.2.2.5 Fonctionnement du U-Net conditionnel dans ResDiff

L'architecture s'articule autour des blocs suivants :

Embedding temporel : Transforme le pas de temps t en un vecteur numérique de dimension 64, permettant au modèle de comprendre l'étape de diffusion.

- Utilise la fonction d'activation `SiLU` (Sigmoid Linear Unit ou Swish)
- Fonction lisse et non monotone, conservant plus d'information que ReLU
- Adaptée aux variables semi-continues comme le pas de temps
- Affinée par une couche linéaire supplémentaire

Chemin encodeur (Encoder) :

- Concatène l'image basse résolution (LR) et l'image à corriger (x_t)
- Réduit la taille spatiale et augmente le nombre de canaux
- Utilise des couches de convolution (avec ReLU) et `max pooling`
- Capture des caractéristiques visuelles complexes

Bloc Goulot (Bottleneck) :

- Couche la plus profonde avec la plus petite taille spatiale
- Résume toute l'information de l'image
- Fusionne l'embedding temporel avec les cartes de caractéristiques
- Guide le réseau en fonction de l'étape de bruitage t

Chemin décodeur (Decoder) :

- Reconstruit progressivement l'image à la taille originale
- Utilise des *connexions de saut* (skip connections)
- Combine les caractéristiques compressées avec les informations de l'encodeur
- Implémente des couches de convolution + ReLU
- Produit une prédiction du bruit $\hat{\epsilon}_\theta$voir (à chaque pixel (pas une image classique)

Notations clés :

$$\hat{x}_0 = \frac{1}{\sqrt{\hat{\alpha}_t}} \left(x_t - \sqrt{1 - \hat{\alpha}_t} \hat{\varepsilon}_\theta(x_t, t) \right.$$

(2.3)

x_t : Image bruitée à l'étape t
$\hat{\epsilon}_\theta$: Bruit prédit par le Conditional U-Net
x_0 : Image débruitée (cible)

2.3 Conclusion

Dans ce chapitre, nous avons présenté en détail l'architecture ResDiff, qui associe la simplicité du traitement basse résolution d'ESPCN à la puissance des modèles de diffusion conditionnels. Cette architecture commence par extraire les caractéristiques dans l'espace LR afin de réduire la complexité computationnelle, puis reconstruit l'image haute résolution à l'aide d'un processus de diffusion guidé. Ce processus repose sur un U-Net conditionnel, capable de prédire le bruit à chaque étape de débruitage, en intégrant explicitement l'image LR agrandie, le bruit injecté et le pas de temps via un encodage temporel. Nous avons également mis en œuvre une stratégie d'apprentissage stable, basée sur la perte MAE, et clarifié les concepts clés de ResDiff de manière accessible.

Chapitre 3

Implémentation et résultats expérimentaux

3.1 Introduction

Ce chapitre présente les étapes d'implémentation des approches développées pour la super-résolution d'images. Nous décrivons d'abord l'environnement technique (outils, langage de programmation et plateforme de développement), puis détaillons la mise en œuvre du modèle proposé. Le chapitre s'achève par l'analyse des tests d'évaluation des performances.

La structure s'articule autour de deux axes principaux :

— **Implémentation du système** : Description technique de l'architecture et des composants

— **Analyse des résultats expérimentaux** : Évaluation quantitative et qualitative des performances

3.2 Environnement de développement

3.2.1 Kaggle

Kaggle est une plateforme en ligne gratuite dédiée à la science des données et à l'intelligence artificielle. Elle permet aux utilisateurs de participer à des concours organisés par des entreprises et institutions, souvent avec des récompenses financières, mais aussi de partager des jeux de données, des scripts (notebooks), et d'échanger au sein d'une communauté active de data scientists.

Fondée en 2010 par Anthony Goldbloom et Ben Hamner, Kaggle a été rachetée par Google en 2017. La plateforme propose également des cours interactifs gratuits avec certification, accessibles à tous, débutants comme experts [DataCamp, 2025].

Kaggle offre un environnement de développement intégré dans le navigateur, avec la possibilité d'activer des GPU (jusqu'à 30 heures par semaine) pour l'entraînement de modèles, ce qui facilite le travail sans nécessiter de configuration locale.

Très utilisée à la fois par les étudiants et les professionnels, Kaggle constitue un excellent espace d'apprentissage, de pratique, de visibilité et de collaboration dans le domaine de la data science [DataCamp, 2025].

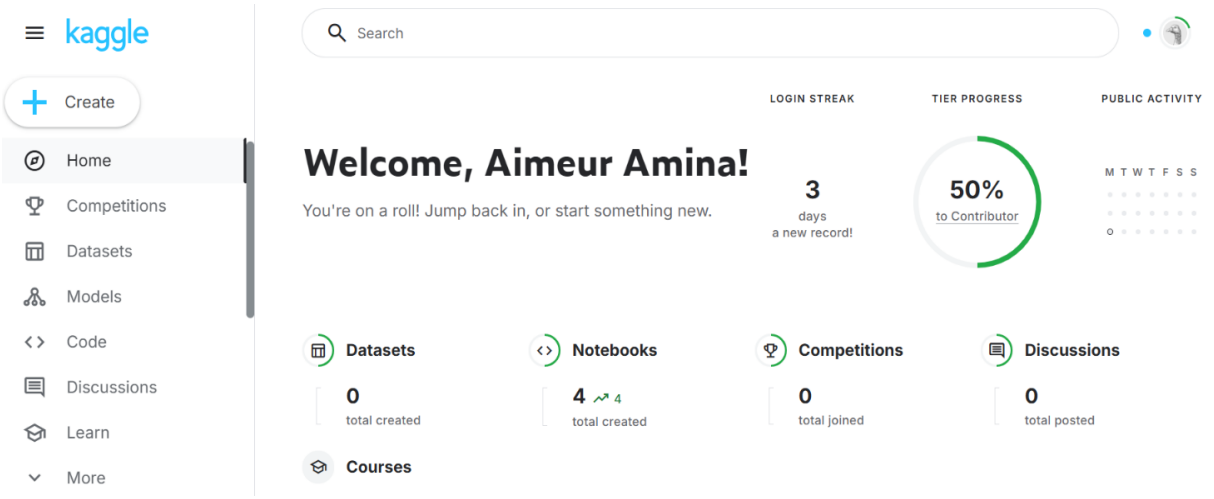

FIGURE 3.1 – Environnement de kaggle

3.2.2 CUDA

CUDA (*Compute Unified Device Architecture*) est une plateforme de calcul parallèle et un modèle de programmation développé par NVIDIA. Il permet d'exploiter la puissance de traitement massivement parallèle des GPU NVIDIA pour des tâches de calcul général (GPGPU), bien au-delà du simple rendu graphique. Grâce à son architecture composée d'un grand nombre de cœurs de calcul, CUDA est spécialement conçu pour exécuter efficacement des opérations parallélisables à grande échelle. Cette technologie est cruciale pour programmer et accélérer les calculs sur GPU dans de nombreux domaines, notamment l'intelligence artificielle et le deep learning, le traitement d'images, le calcul scientifique et les jeux vidéo [NVIDIA, 2025a].

3.2.3 Pourquoi les GPU ?

Les GPU (Graphics Processing Unit) sont des processeurs hautement spécialisés, capables d'exécuter simultanément un très grand nombre d'opérations. Cette architecture intrinsèquement parallèle les rend particulièrement adaptés aux tâches computationnelles intensives, telles que celles rencontrées en apprentissage automatique. Sur des plateformes comme Kaggle, les GPU sont utilisés pour accélérer significativement l'entraînement des modèles de deep learning. Leur puissance de calcul surpasse largement celle des CPU pour le traitement de vastes quantités de données et l'exécution d'opérations matricielles complexes, qui sont au cœur des réseaux neuronaux. Contrairement aux CPU, optimisés pour les tâches séquentielles et les calculs polyvalents, les GPU excellent dans l'exécution parallèle massive, ce qui est indispensable à la performance des algorithmes d'apprentissage profond.

Kaggle met à disposition de ses utilisateurs des GPU performants, tels que les NVIDIA Tesla T4 (avec 16 Go de mémoire) et les NVIDIA Tesla P100 (offrant entre 16 Go et 64 Go selon la configuration). Pour tirer pleinement parti de ces accélérations matérielles, il est impératif d'utiliser des bibliothèques logicielles optimisées comme TensorFlow ou PyTorch. Il convient toutefois de noter les limitations d'usage imposées par Kaggle, notamment un temps d'exécution restreint pour les notebooks (environ 9 heures par session) et une limite hebdomadaire d'utilisation des GPU (jusqu'à 30 heures). L'optimisation du code est donc

cruciale pour maximiser l'exploitation des ressources disponibles tout en respectant ces contraintes [DataCamp, 2025].

3.2.4 Python

Python est un langage de programmation de haut niveau, polyvalent et très largement adopté pour le développement d'applications de toutes envergures. Conçu par Guido van Rossum et lancé en 1991, il se distingue par sa syntaxe claire et lisible, ce qui favorise l'écriture d'un code propre, maintenable et facile à comprendre. Python prend en charge de multiples paradigmes de programmation, incluant l'orienté objet, le fonctionnel, le procédural et l'impératif.

Grâce à son système de typage dynamique et à sa gestion automatique de la mémoire, Python simplifie considérablement le processus de développement sans compromettre la puissance du langage. Il bénéficie en outre d'une vaste bibliothèque standard et d'une large compatibilité avec la plupart des systèmes d'exploitation. Ces caractéristiques en font un choix privilégié dans de nombreux domaines contemporains, allant du développement web à la science des données et à l'intelligence artificielle [Python Software Foundation, 2025].

3.3 Bibliothèques utilisées

3.3.0.1 OpenCV (Open Source Computer Vision Library)

OpenCV est une bibliothèque open source exhaustive dédiée à la vision par ordinateur et au traitement d'images, offrant un vaste éventail de plus de 2500 algorithmes. Polyvalente, elle est compatible avec de nombreux langages de programmation (notamment C, C++ et Python) et fonctionne sur diverses plateformes (Windows, Linux, macOS, Android). Distribuée sous licence BSD, elle peut être utilisée librement, aussi bien dans des projets personnels que professionnels [OpenCV, 2025]. Dans le cadre de notre travail, cette bibliothèque a été essentielle pour le traitement et la manipulation des images, incluant des opérations telles que le redimensionnement, le chargement et la sauvegarde.

3.3.0.2 NumPy

NumPy est la bibliothèque fondamentale de Python pour le calcul numérique et scientifique. Elle fournit des objets tableau (N-dimensionnels ou ndarrays) hautement performants, ainsi qu'un ensemble riche de fonctions mathématiques avancées, particulièrement adaptées au calcul vectoriel et matriciel. NumPy permet également une intégration aisée de code écrit dans des langages compilés comme C, C++ ou Fortran [NumPy, 2025]. Sa capacité à gérer efficacement de grandes structures de données et à optimiser les opérations numériques en fait un pilier indispensable pour la préparation des données et les manipulations de tenseurs au sein des modèles d'apprentissage profond, y compris ResDiff.

3.3.0.3 Pathlib

Pathlib est une bibliothèque standard de Python qui propose une approche orientée objet pour la manipulation des chemins de fichiers et de répertoires. Elle facilite une gestion robuste et moderne des systèmes de fichiers, simplifiant des tâches telles que la récupération structurée et l'organisation des fichiers image dans les différents dossiers de données du projet [GeeksforGeeks, 2025a].

3.3.0.4 Math

La bibliothèque Math est un module standard intégré à Python, spécifiquement conçu pour simplifier les tâches mathématiques fondamentales. Elle fournit un accès à diverses constantes mathématiques (telles que) et à une multitude de fonctions pour des calculs élémentaires (par exemple, racines carrées, exponentielles, logarithmes et fonctions trigonométriques) [GeeksforGeeks, 2025b]. Dans notre projet, son utilisation a été ciblée pour des opérations mathématiques spécifiques requises, notamment lors de l'implémentation des formules et des calculs précis au sein du processus de diffusion.

3.3.0.5 OS (Miscellaneous operating system interfaces)

Ce module offre une interface portable pour accéder aux fonctionnalités spécifiques du système d'exploitation. Il permet, par exemple, d'interagir avec le système de fichiers ou d'exécuter des commandes système. Pour des opérations plus courantes comme la lecture ou l'écriture de fichiers, il est recommandé d'utiliser la fonction open(). Pour la gestion des chemins de fichiers, le module os est plus approprié [OpenCV, 2025].

3.3.0.6 Matplotlib

Matplotlib est une bibliothèque open source de visualisation de données en Python, largement utilisée pour créer des graphiques statiques, animés ou interactifs. Inspirée deMATLAB et initialement développée par John Hunter, elle permet de générer des visualisations dans de nombreux formats (PDF, PNG, SVG, etc.). Matplotlib s'intègre parfaitement avec NumPy et peut être utilisée aussi bien dans des scripts, des notebooks, que dans des applications interactives. Elle fonctionne via une interface orientée objet ou grâce à son module simplifié pyplot, qui facilite la création rapide de graphiques à la manière de MATLAB [Intelligence Artificielle School, 2025]

3.3.0.7 glob

Le module glob en Python permet de rechercher des fichiers ou des répertoires en utilisant des motifs de type Unix, comme les jokers * où ? Il simplifie la localisation de chemins correspondant à un modèle donné, en parcourant automatiquement les dossiers selon les règles du shell Unix, et retourne les résultats sous forme de liste, sans ordre garanti [GeeksforGeeks, 2025a].

3.3.0.8 Time

Le module time de Python permet de mesurer et manipuler le temps d'exécution d'un programme. En apprentissage automatique, il est principalement utilisé pour chronométrer l'entraînement des modèles et évaluer la durée des epochs, afin d'optimiser les performances et l'utilisation des ressources [Python Software Foundation, 2025].

3.3.0.9 Pil

PIL, qui signifie Python Image Library, est une bibliothèque dédiée au traitement d'images en Python. Toutefois, comme elle n'est plus maintenue depuis 2011, sa version dérivée appelée Pillow est désormais largement privilégiée. Cette bibliothèque prend en charge de nombreux formats d'image courants, notamment JPEG et PNG [OpenCV, 2025].

3.3.0.10 scikit-image

scikit-image (ou skimage) est une bibliothèque Python open-source dédiée au traitement d'images. Facile à utiliser et bien intégrée avec d'autres outils scientifiques comme NumPy, SciPy ou Matplotlib, elle permet de lire, transformer, analyser et évaluer la qualité des images.]Elle offre notamment des fonctions pour appliquer des filtres, effectuer des transformations géométriques (comme le redimensionnement ou la rotation), extraire des caractéristiques (bords, textures...), et calculer des métriques de comparaison telles que le PSNR (Peak Signal-to-Noise Ratio) et le SSIM (Structural Similarity Index) [scikit image, 2025]

3.3.0.11 Torchvision

torchvision est une extension officielle de **PyTorch**, spécialement conçue pour les applications de vision par ordinateur. Elle fournit des outils pratiques pour manipuler des images et des vidéos, ainsi que des modèles pré-entraînés sur des tâches courantes [PyTorch, 2025].

Parmi ses composants, torchvision.transforms et sa version améliorée transforms.v2 jouent un rôle clé [PyTorch, 2025]. Ils permettent d'appliquer facilement des transformations standards (redimensionnement, normalisation, augmentation de données, etc.) indispensables à la préparation et à l'enrichissement des données avant l'entraînement ou l'inférence.

Bien que `torchvision.transforms` ne fasse pas partie du module central torch, il s'inscrit pleinement dans l'écosystème officiel de PyTorch et constitue un outil essentiel pour des tâches comme la classification d'images, la détection d'objets ou la segmentation sémantique [PyTorch, 2025]

3.3.0.12 PyTorch

PyTorch est un framework open-source de deep learning, largement utilisé dans la recherche et le milieu académique pour la conception, l'entraînement et l'évaluation de réseaux de neurones. Il repose sur la bibliothèque Torch, tout en offrant une API Python moderne, intuitive et flexible. PyTorch se compose de plusieurs sous-modules essentiels [IBM, 2025] :

— torch constitue le cœur du framework et fournit les structures de données fondamentales (comme les tenseurs) ainsi que les fonctions de calcul.

— `torch.nn` permet de construire facilement des modèles en combinant des couches prédéfinies (comme Conv2d, ReLU, Linear, etc.).

— torch.nn.functional offre des versions fonctionnelles de ces opérations, utiles pour écrire des architectures personnalisées avec plus de contrôle.

3.3.0.13 Torch-Fidelity

Est une bibliothèque PyTorch dédiée à l'évaluation des modèles génératifs. Elle propose des implémentations précises, efficaces et extensibles des principales métriques de performance telles que : Inception Score (IS), Fréchet Inception Distance (FID), Kernel Inception Distance (KID), Perceptual Path Length (PPL), ainsi que la précision et le rappel (PRC) [IBM, 2025].

3.4 Base de données

Dans le cadre du projet ResDiff, plusieurs jeux de données de haute qualité sont couramment utilisés pour l'entraînement et l'évaluation des modèles de super-résolution. Parmi eux, DIV2K (DIVerse 2K resolution) est l'un des plus populaires. Proposé par Agustsson Timofte en 2017, ce dataset contient 800 images haute résolution pour l'entraînement, 100 images pour la validation (souvent utilisées comme test dans les études), et 100 images supplémentaires pour le test, bien que ces dernières ne soient pas toutes accessibles en haute résolution dans le cadre public.
DIV2K est particulièrement apprécié pour sa diversité et sa complexité visuelle, ce qui en fait un excellent choix pour entraîner des modèles de super-résolution. Par ailleurs, d'autres datasets comme Urban100, FFHQ, ou CelebA sont également utilisés dans les recherches autour de ResDiff. Urban100, par exemple, se distingue par ses nombreuses structures géométriques fines, ce qui le rend plus difficile à restaurer, tandis que FFHQ et CelebA sont centrés sur les visages humains.

Il est important de noter que le modèle ResDiff n'est pas entraîné de manière conjointe sur l'ensemble de ces jeux de données. Au contraire, chaque version du modèle est entraînée indépendamment sur un dataset spécifique : une version est entraînée uniquement sur DIV2K, une autre sur Urban100, une autre sur FFHQ, etc.. [NVIDIA, 2025b].

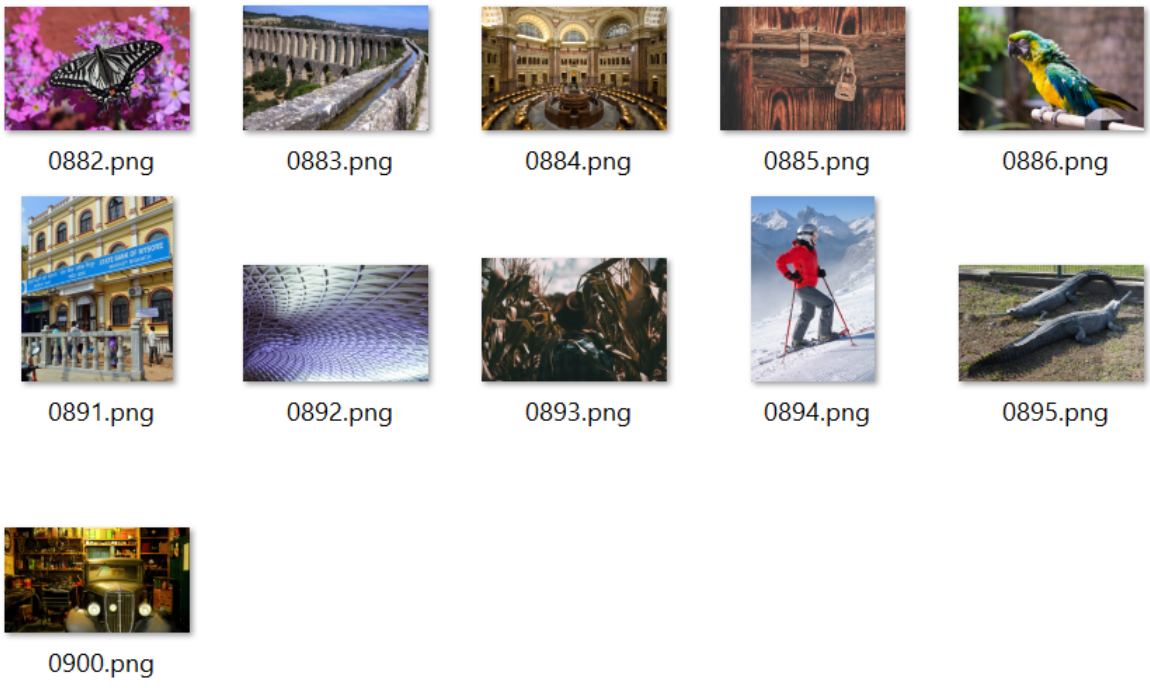

Figure 3.2 – Échantillon d'images BDD de Div2K

Dans notre projet, nous avons choisi le dataset DIV2K comme base d'entraînement et d'évaluation. Ce choix s'explique par la richesse des images qu'il propose, leur qualité haute résolution, et le fait qu'il constitue une référence largement adoptée dans la littérature pour les tâches de super-résolution.Avec :

— 800 images haute résolution redimensionnée 40x40 pour entrainement.

— 100 images haute résolution redimensionnée 40x40 pour teste.

La Figure 3.3 présente un échantillon du jeu de données DIV2K, composé d'images haute résolution pouvant atteindre, par exemple, 1024 pixels. Ces images, importées depuis la plateforme Kaggle, ont ensuite été redimensionnées à 160 pixels, comme illustré par la Figure 3.2.

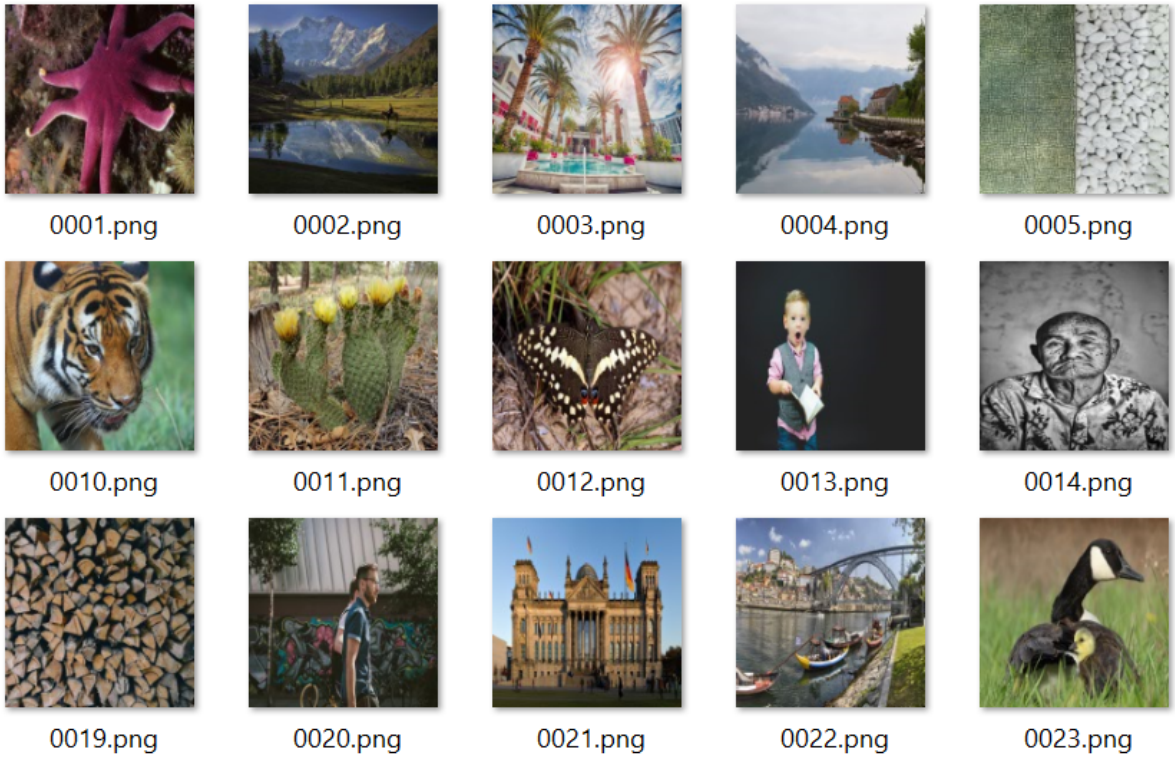

FIGURE 3.3 – Échantillon d'images BDD de Div2K 160x160

3.5 Mise en Œuvre et Résultats Expérimentaux

Cette section est dédiée à la description du protocole expérimental mis en œuvre et à la présentation des résultats obtenus par notre système de super-résolution d'image

3.5.1 Première démarche

3.5.1.1 ETAPE 1 : Super-résolution par ESPCN

Préparation des Données Pour l'entraînement du modèle ESPCN *(Pré-entraîné sur BDD-images Div2K)*, les données ont été préparées selon le protocole suivant :

— **Chargement des images :** Toutes les images du jeu de données d'entraînement ont été chargées.

— **Conversion en niveau de gris :** Chaque image couleur a été convertie en son équivalent en niveaux de gris, en extrayant le canal de luminance (canal Y), une

pratique courante en super-résolution pour simplifier le traitement et se concentrer sur la structure.

- **Création des paires Haute Résolution (HR) / Basse Résolution (LR) :**
  - L'image de référence en Haute Résolution (HR) est obtenue en rognant et en normalisant l'image originale pour assurer une taille et une plage de valeurs cohérentes.
  - La version Basse Résolution (LR) correspondante est générée à partir de l'image HR par une opération de **redimensionnement bicubique**, une méthode d'interpolation standard pour simuler les dégradations réelles.
- **Mise en place du pipeline de données :** Un pipeline de données a été configuré, en utilisant les fonctionnalités de PyTorch, pour charger efficacement les paires (LR, HR) par lots (*batches*) durant la phase d'entraînement.

Construction du Modèle ESPCN L'architecture du modèle ESPCN, qui opère principalement dans l'espace de basse résolution pour optimiser l'efficacité computationnelle, est construite comme suit :

- **Couche d'entrée :** Le modèle reçoit en entrée des images basse résolution (niveaux de gris) sous la forme d'un tenseur de dimensions `[batch, hauteur_LR, largeur_LR, 1]`.
- **Couches convolutives :**
  - **Première convolution :** Un filtre 5×5 avec 64 canaux de sortie, suivi d'une fonction d'activation ReLU, pour l'extraction des caractéristiques initiales.
  - **Deuxième convolution :** Un filtre 3×3 avec 32 canaux de sortie, également suivi d'une fonction d'activation ReLU, pour l'affinage des caractéristiques.
  - **Troisième convolution :** Un filtre 3×3 avec S^2 canaux de sortie (où S est le facteur de mise à l'échelle), **sans fonction d'activation** finale, préparant les cartes de caractéristiques pour la reconstruction sub-pixel.
- **Couche de Reconstruction (Pixel Shuffle) :** L'opération `Pixel Shuffle` (aussi connue sous le nom de `depth_to_space`) est appliquée à la sortie de la dernière couche convolutive. Cette technique réorganise les S^2 canaux de sortie en une image de taille $S \times S$ fois plus grande, reconstruisant ainsi directement l'image haute résolution.
- **Couche de sortie :** Le modèle produit une image haute résolution prédite sous la forme `[batch, hauteur_LR * S, largeur_LR * S, 1]`.

Phase d'Entraînement L'entraînement du modèle ESPCN a été réalisé en utilisant le framework PyTorch, selon le protocole itératif suivant :

- **Initialisation :** Les poids du réseau sont initialisés.
- **Boucle d'entraînement par époque :** Pour chaque époque, un nouvel itérateur de données est réinitialisé afin de parcourir l'intégralité du jeu de données d'entraînement.
- **Traitement par lot (*batch*) :** Pour chaque lot de données :
  - L'image basse résolution est passée à travers le réseau ESPCN pour obtenir une prédiction d'image haute résolution.
  - La **fonction de perte MAE** (Mean Absolute Error) est calculée entre l'image haute résolution prédite par le modèle et l'image haute résolution réelle (cible).

- Les poids du modèle sont optimisés en utilisant l'algorithme d'optimisation **Adam Optimizer**, afin de minimiser la perte calculée.
- Des métriques de performance, telles que le **PSNR** (Peak Signal-to-Noise Ratio), sont suivies pour évaluer la progression de l'entraînement.

— **Sauvegarde du modèle :** Le modèle entraîné est sauvegardé après chaque époque, permettant de conserver les meilleures performances et de reprendre l'entraînement si nécessaire.

Dans le cadre de cette expérimentation, notre démarche a consisté à évaluer les performances des composants clés de notre architecture ainsi que celles de l'approche hybride globale. Plus précisément, nous avons d'abord mené des évaluations sur l'algorithme ESPCN, servant de base pour l'estimation initiale de l'image haute résolution. Ensuite, un second ensemble d'expériences a été réalisé avec le modèle de diffusion, spécifiquement conçu pour affiner les détails et améliorer la qualité perceptuelle. Ces évaluations ont été conduites en s'appuyant sur le jeu de données DIV2K, constitué d'images haute résolution préalablement redimensionnées à 160x160 pixels, comme décrit précédemment.

3.5.1.2 ETAPE 2 : Super-résolution par Diffusion

Cette seconde étape de la méthode ResDiff intègre un modèle de diffusion pour raffiner l'image haute résolution, en se concentrant sur la génération de détails fins et la qualité perceptuelle.

Initialisation Les éléments suivants sont initialisés pour le processus de diffusion :

— **Données :** Images de basse résolution (`LR`) et de haute résolution (`HR`).

— **Réseau :** Un Conditional U-Net résiduel, complété par des embeddings temporels pour intégrer l'information de l'étape de diffusion.

— **Paramètres de diffusion :**
  - T : Nombre total d'étapes de diffusion.
  - β_t (`betas`) : Schedule des variances du bruit.
  - α_t (`alphas`) : Coefficient de conservation du signal.
  - $\hat{\alpha}_t$ (`alpha_hat`) : Produit cumulatif des α_t.

— **Optimiseur :** Adam.

Définition de la fonction de bruitage `q_sample` La fonction `q_sample(x_0, t, alpha_hat)` est définie pour bruiter l'image haute résolution `x_0` à une étape de diffusion t donnée avec du bruit gaussien. Elle retourne l'image bruitée `x_t` où ϵ représente le bruit aléatoire gaussien échantillonné (Voir Eq (2.1).

Phase d'entraînement L'entraînement du modèle de diffusion est effectué pour un nombre défini d'époques, en suivant les étapes suivantes pour chaque lot de données :

— **Pour chaque batch (`lr, hr`) dans le `train_loader` :**
  - Normalisation des images `lr` et `hr` pour qu'elles se situent dans la plage $[-1, 1]$.
  - Échantillonnage aléatoire d'un pas de temps t compris entre 0 et T.
  - Génération de l'image bruitée `x_t` et du bruit réel ϵ associé à l'aide de la fonction `q_sample(hr, t)`.

— Prédiction du bruit par le modèle : `pred_noise` = modèle(`lr`, `x_t`, `t`). Le modèle (Conditional U-Net) apprend à prédire le bruit ϵ à partir de l'image basse résolution conditionnante (`lr`), de l'image bruitée (`x_t`) et du pas de temps (t).

— Calcul de la fonction de perte **MAE** (Mean Absolute Error) entre le bruit prédit (`pred_noise`) et le bruit réel (ϵ).

— Application de la rétropropagation et mise à jour des poids du modèle via l'optimiseur Adam.

— **Clipping des gradients :** Les gradients sont écrêtés pour stabiliser l'apprentissage et prévenir les explosions de gradients.

Composants Clés du Module de Super-Résolution par Diffusion Le module de super-résolution basé sur la diffusion, partie intégrante de notre approche hybride, repose sur les éléments fondamentaux suivants :

— **Modèle Principal :** Un **U-Net conditionnel résiduel**, spécifiquement conçu pour les tâches de débruitage conditionnel, intégrant des blocs de résidus adaptatifs en fonction du pas de temps de diffusion.

— **Conditionnement :** L'inférence du modèle est guidée par le conditionnement sur l'image basse résolution (`LR`) et le pas de temps de diffusion (t), en complément de l'image bruitée d'entrée (x_t).

— **Objectif :** L'objectif principal du modèle est de **prédire le bruit** ajouté à l'image à chaque étape du processus inverse de diffusion, permettant ainsi la reconstruction progressive de détails fins.

Model: "ESPCN"

Layer (type)	Output Shape	Param #
input (InputLayer)	(None, 40, 40, 1)	0
conv1 (Conv2D)	(None, 40, 40, 64)	1,664
conv2 (Conv2D)	(None, 40, 40, 32)	18,464
conv3 (Conv2D)	(None, 40, 40, 4)	1,156
pixel_shuffle (Lambda)	(None, 80, 80, 1)	0
output (Activation)	(None, 80, 80, 1)	0

Total params: 21,284 (83.14 KB)
Trainable params: 21,284 (83.14 KB)
Non-trainable params: 0 (0.00 B)

FIGURE 3.4 – Résultats obtenus à partir du modèle ESPCN.

3.5.1.3 Évaluation du Modèle ESPCN et du Modèle de Diffusion

Dans cette section, nous détaillons l'architecture et les résultats d'exécution des modèles individuels : l'ESPCN et le modèle de diffusion probabiliste.

Évaluation du Modèle ESPCN Dans cette partie nous nous intéressons aux résultats d'exécution de l'approche développée. Pour une visualisation des couches :

Architecture du Modèle ESPCN Le modèle ESPCN est construit séquentiellement et opère principalement sur les images de basse résolution. Son architecture est détaillée ci-dessous :

- Couche 1 : `InputLayer`
  - Type : Couche d'entrée.
- Couche 2 : `conv1`
  - Type : `Conv2D`
  - Filtres : 64
  - Taille du noyau : (5,5)
  - Activation : ReLU
- Couche 3 : `conv2`
  - Type : `Conv2D`
  - Filtres : 32
  - Taille du noyau : (3,3)
  - Activation : ReLU
- Couche 4 : `conv3`
  - Type : `Conv2D`
  - Filtres : 4 (correspondant à S^2 pour un facteur d'agrandissement S=2)
  - Taille du noyau : (3,3)
  - Activation : Linéaire (pas d'activation non-linéaire)
- Couche 5 : `pixel_shuffle`
  - Type : `Lambda` (Implémentation de l'opération Pixel Shuffle)
  - Cette couche réorganise les caractéristiques pour la reconstruction en haute résolution.
- Couche 6 : `output`
  - Type : `Activation` Activation : Tanh (Hyperbolique Tangente)

Le facteur de suréchantillonnage (agrandissement) du modèle ESPCN est de 2x en hauteur et en largeur.

Paramètres d'Entraînement et Métriques de Suivi Les paramètres suivants sont cruciaux pour l'entraînement et l'évaluation du modèle :

- Patience : Ce paramètre est utilisé dans la technique d'arrêt précoce (early stopping), permettant d'interrompre l'entraînement si les performances sur le jeu de données de validation ne s'améliorent plus après un nombre défini d'époques.
- Époque (Epoch) : Représente un cycle complet de passage à travers l'intégralité de la base de données d'entraînement.
- Optimiseur(Optimizer) : Algorithme utilisé pour ajuster les poids du modèle afin de minimiser la fonction de perte (par exemple, Adam).

— Perte (Loss) : Une fonction de coût (par exemple, MAE) qui quantifie l'erreur entre les prédictions du modèle et les valeurs réelles. L'objectif de l'entraînement est de la minimiser.
— PSNR (Peak Signal-to-Noise Ratio) : Une métrique objective évaluant la qualité de l'image super-résolue en comparant ses pixels à ceux de l'image originale de haute résolution. Une valeur plus élevée indique une meilleure qualité.
— Historique (History) : Un enregistrement des valeurs de perte et des scores de validation à chaque époque, essentiel pour visualiser la convergence de l'entraînement et prendre des décisions concernant l'early stopping.

Évaluation du Modèle de Diffusion Probabiliste

Architecture Générale : Le modèle de diffusion probabiliste est basé sur une architecture U-Net conditionnelle, dont la structure générale et le nombre de paramètres par composant sont les suivants :

— **Time Embedding** : 4,288 paramètres
— **Initial Conv** : 3,520 paramètres
— **Encoder 1** : 82,432 paramètres
— **Encoder 2** : 312,320 paramètres
— **Encoder 3** : 1,214,464 paramètres
— **Middle Block** : 1,214,464 paramètres
— **Decoder 3** : 1,214,464 paramètres
— **Decoder 2** : 312,320 paramètres
— **Decoder 1** : 82,432 paramètres
— **Final Conv** : 1,859 paramètres

Flux de Données Le flux des données à travers le réseau est le suivant :
`Input LR` (80x80 pixels) → Suréchantillonnage bilinéaire → `Input LR_upscaled` (160x160 pixels)
`Input HR` (160x160 pixels) → Ajout de bruit (`Noisy HR`)
Concaténation des images `LR_upscaled` et `Noisy_HR` → Tenseur à 6 canaux
↓
Initial Conv : 6 canaux d'entrée → 64 canaux de sortie
↓
ENCODEUR :
Niveau 1 : 64 canaux (160x160)
Niveau 2 : 128 canaux (80x80)
Niveau 3 : 256 canaux (40x40)
↓
MIDDLE Block : 256 canaux (40x40)
↓
DECODER (avec skip connections) :
Niveau 3 : 256 canaux (40x40)
Niveau 2 : 128 canaux (80x80)
Niveau 1 : 64 canaux (160x160)
↓
Final Conv : 64 canaux d'entrée → 3 canaux de sortie (160x160)

Structure d'un Bloc Résiduel Chaque bloc résiduel de l'architecture est composé comme suit, avec un total de 82,432 paramètres par bloc :

- `GroupNorm(8, 64)`
- `Conv2d(64, 64, 3x3)`
- `Time MLP: Linear(64, 128)` (Module Perceptron Multicouche adaptatif au temps)
- `GroupNorm(8, 64)`
- `Conv2d(64, 64, 3x3)`
- `Skip connection` (connexion résiduelle ajoutant l'entrée du bloc à sa sortie)

Caractéristiques Générales du Modèle de Diffusion

- **Architecture :** U-Net conditionnel avec embedding temporel.
- **Tâche :** Super-résolution (passage de 80x80 à 160x160 pixels) par processus de diffusion.
- **Skip connections :** Oui, des connexions de saut sont utilisées pour préserver les informations de résolution.
- **Normalisation :** `GroupNorm` est employée pour la normalisation des couches.
- **Fonctions d'activation :** Les activations principales au sein du U-Net sont de type ReLU, tandis que la fonction SiLU (Sigmoid Linear Unit) est utilisée spécifiquement dans l'embedding temporel du DDPM.
- **Facteur d'amélioration :** 2x.

```
espcn, espcn_history = train_espcn(
    espcn,
    train_loader,
    valid_loader,
    criterion=espcn_criterion,
    optimizer=espcn_optimizer,
    device=device,
    epochs=200,       # Requested 200 epochs
    patience=20       # Requested 20 patience
)
```

(A)

```
unet, unet_history = train_diffusion(
    unet=unet,
    espcn=espcn,
    train_loader=train_loader,
    valid_loader=valid_loader,
    optimizer=unet_optimizer,
    criterion=unet_criterion,
    device=device,
    epochs=200,       # Requested 200 epochs
    patience=20       # Requested 20 patience
)
```

(B)

FIGURE 3.5 – (A)pour entrainement ESPCN, (B) pour entrainement modele de diffusion

(A) et (B) dans la figure 3.5 présentent les phases clés de l'entraînement de notre système de super-résolution, intégrant deux architectures distinctes et complémentaires : l'ESPCN et le modèle U-Net basé sur la diffusion probabiliste.

(a) et (b) dans la figure 3.6 montrent l'utilisation de bibliothèque python "matplotlib" pour afficher et sauvegarder des courbes d'entraînement et de validation.

La figure 3.7 présente une comparaison entre l'ESPCN (courbes bleue/verte) et le U-Net (courbes rouge/violette) est maintenue, tout en explicitant le rôle de la diffusion dans l'amélioration des performances du U-Net. Une explication est fournie ci-dessous.

```
plt.subplot(2, 2, 1)
plt.plot(espcn_history['train_loss'], 'b-', label='Train Loss')
plt.title('ESPCN Training Loss')
plt.xlabel('Epochs')
plt.legend()
```

(a)

```
    plt.subplot(2, 2, 4)
    plt.plot(unet_history['val_psnr'], 'm-', label='Validation PSN
R')
    plt.title('U-Net Validation PSNR')
    plt.xlabel('Epochs')
    plt.legend()

    plt.tight_layout()
    plt.savefig("/kaggle/working/training_curves.png")
    plt.show()
    print("✅ Training curves saved")
```

(b)

FIGURE 3.6 – (a) : Affichage Taux d'erreurs, (b) : Validation PSNR

— **Courbe bleue** : Le taux d'erreur diminue rapidement avant de se stabiliser à un niveau très bas, indiquant une convergence efficace du modèle ESPCN dès les premières époques.

— **Courbe verte** : On observe une forte augmentation initiale du PSNR, suivie d'une stabilisation entre 27 et 28 dB, confirmant que le modèle ESPCN génère des images de qualité satisfaisante.

— **Courbe rouge** : Comportement similaire à celui de l'ESPCN, démontrant que le modèle U-NET converge également de manière optimale.

— **Courbe violette** : Le U-NET atteint un PSNR supérieur à celui de l'ESPCN (proche de 28 dB), malgré de légères fluctuations. Cette performance reflète une nette amélioration de la qualité d'image, attribuable au mécanisme de diffusion.

Après traitement des images à cette étape, les modèles produisent des prédictions, puis les résultats sont visualisés via MATPLOTLIB. La figure 3.8 présente le segment de code responsable de l'affichage et de la visualisation des résultats.

Dans la figure 3.9, l'image de gauche représente l'entrée basse résolution initiale (40×40 pixels). Celle-ci est d'abord super-résolue par le modèle ESPCN pour générer la deuxième image intermédiaire (80×80 pixels). Le modèle de diffusion prédit ensuite l'image finale haute résolution (160×160 pixels, troisième image). Enfin, l'image de droite affiche la référence réelle (*ground truth*) pour comparaison.

L'analyse comparative de la Figure 3.10 révèle qu'entre les deux itérations de chaque modèle, le taux d'erreur diminue tandis que le PSNR et le SSIM augmentent. Cette évolution confirme que nos modèles ont été correctement entraînés et valide le principe fondamental des réseaux de neurones profonds : une architecture plus profonde améliore significativement les performances.

Métriques obtenues

— Perte : 0.0258

— PSNR : 28.85 dB

— SSIM : 0.9261

Comparativement aux résultats de l'implémentation ResDiff référencée (PSNR : 27.94 dB, SSIM : 0.72), notre modèle montre une amélioration notable de la qualité de reconstruction.

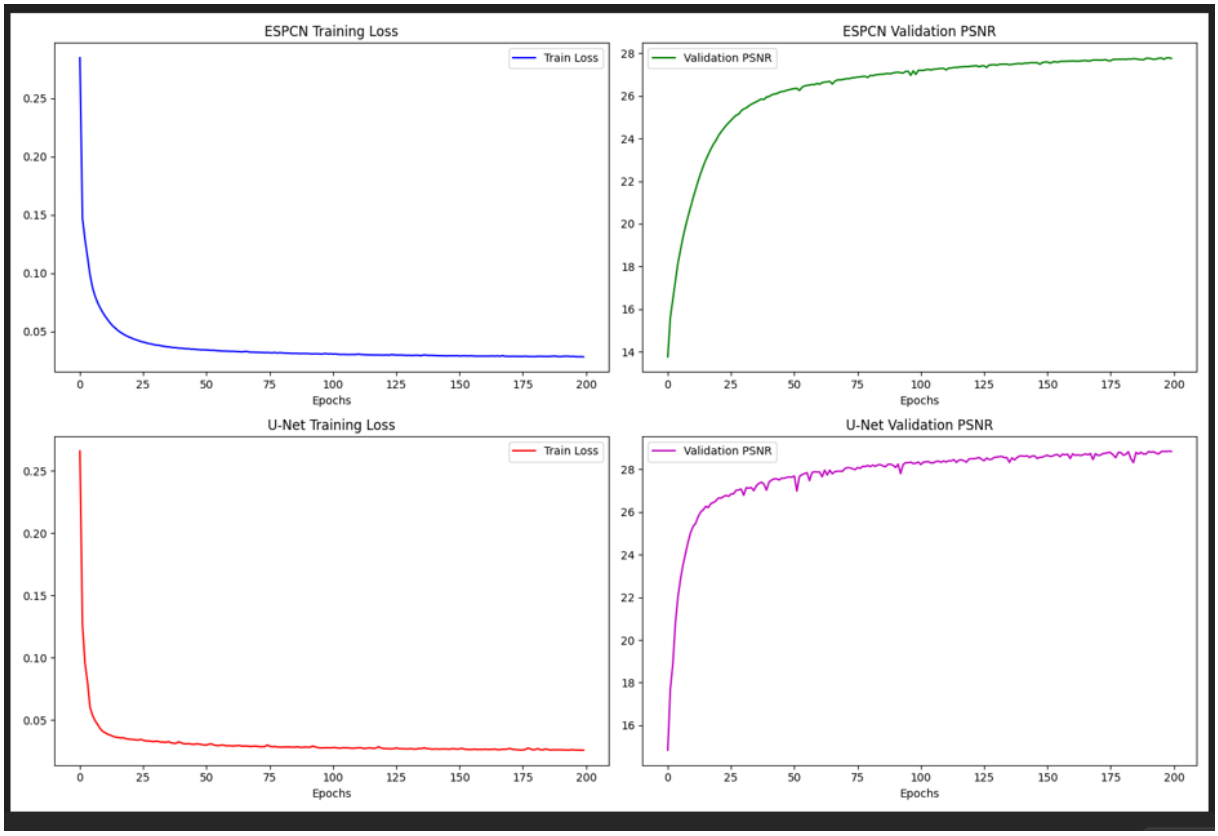

FIGURE 3.7 – (a) : Les courbes pour les taux des erreurs et validations PSNR

Cette observation nous a incités à approfondir l'analyse et à explorer des optimisations algorithmiques dans le cadre d'une seconde approche expérimentale.

Méthode	**PSNR (dB)**	**SSIM**	**Temps inférence (s)**
Interpolation bicubique [Khaledyan et al., 2020]	25.80	0.850	0.001
SRCNN [Garber et al., 2020]	26.24	0.783	0.08
ESPCN (seul) [Shi et al., 2016]	27.76	0.9049	0.05
SRDiff [Li et al., 2022]	26.87	0.690	2.50
SR3 [Lugmayr et al., 2020]	26.17	0.650	2.80
L'approche proposée	**28.85**	**0.9262**	1.20

TABLE 3.1 – Performance en Qualité d'Image (PSNR et SSIM)

L'analyse des résultats présentés dans la table 3.1 met en lumière une performance très prometteuse de notre approche en matière de super-résolution. Côté qualité d'image, notre méthode se positionne clairement en tête. Le PSNR de 28.85 dB, le plus élevé du tableau, indique une fidélité exceptionnelle de l'image reconstruite par rapport à l'originale, surpassant significativement les méthodes établies comme l'ESPCN, pourtant déjà performante. De même, le SSIM de 0.9262, également le meilleur score, confirme que notre approche préserve remarquablement bien les détails structurels et les textures, offrant ainsi un rendu visuellement plus agréable et plus proche de ce que l'œil humain perçoit comme une haute qualité. Ces chiffres suggèrent que notre solution est idéale pour des applications où la clarté et la richesse des détails sont primordiales.

```
def visualize_results(espcn, unet, dataloader, num_examples=3):
    espcn.eval()
    unet.eval()
    fig, axs = plt.subplots(num_examples, 4, figsize=(16, 4*num_exam
ples))

    with torch.no_grad():
        for i in range(num_examples):
            lr, hr = next(iter(dataloader))
            lr, hr = lr.to(device), hr.to(device)
```

FIGURE 3.8 – (a) : L'extrait de code responsable à la visualisation des résultats.

FIGURE 3.9 – (a) : Résultat de reconstruction sur un exemple d'image.

Cependant, cette qualité supérieure s'accompagne d'un compromis en termes de temps d'inférence. Avec 1.20 seconde, notre méthode est certes plus lente que les approches traditionnelles ou les réseaux convolutifs plus anciens, ultra-rapides et optimisés pour la vitesse comme l'ESPCN (0.05 s). Pour des applications exigeant une réactivité quasi-instantanée, cela pourrait être un facteur limitant. Néanmoins, il est crucial de noter que notre approche est nettement plus rapide que les modèles de diffusion récents comme SRDiff (2.50 s) et SR3 (2.80 s), qui sont connus pour leur gourmandise computationnelle. Cela positionne notre méthode comme un excellent équilibre : elle offre une qualité d'image de pointe sans la latence excessive associée à certaines des techniques les plus complexes du moment. En somme, si la qualité est la priorité et qu'une latence d'une seconde est acceptable, notre approche représente un choix très pertinent.

3.5.2 Seconde démarche

Cette deuxième démarche du processus vise à générer une estimation initiale de l'image haute résolution en utilisant l'Efficient Sub-Pixel Convolutional Neural Network (ESPCN). Notre méthodologie expérimentale s'appuie sur des travaux référents pour optimiser les modèles. L'article fondateur de Shi et al. (2016) [Shi et al., 2016] démontre que l'architecture ESPCN obtient des résultats exceptionnels pour des facteurs d'agrandissement $\times 2$, $\times 3$ et $\times 4$. L'analyse comparative révèle que le facteur $\times 3$ constitue le compromis optimal entre :

— Qualité de reconstruction (PSNR)
— Richesse des détails
— Temps d'exécution

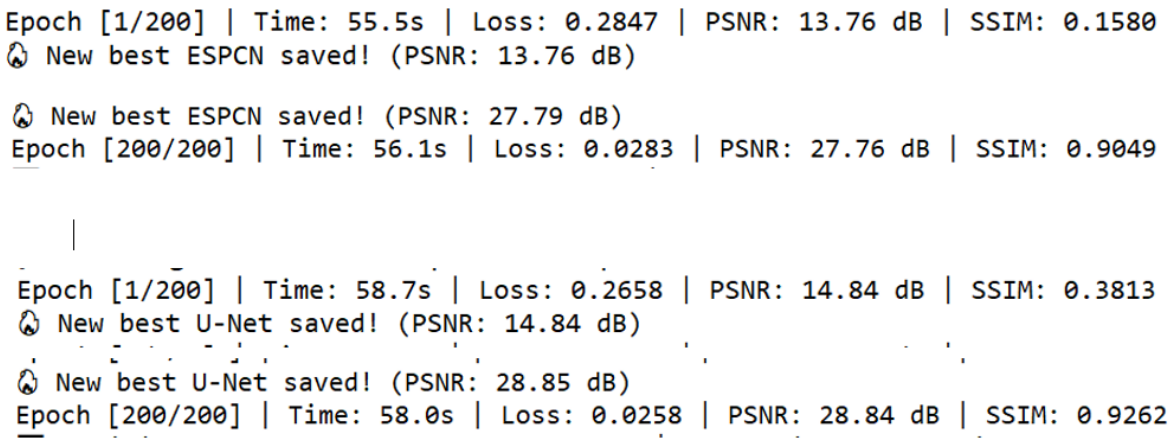

FIGURE 3.10 – (a) : Résultats d'entraînement : modèles ESPCN et diffusion.

Cette balance en fait un choix particulièrement pertinent pour les applications pratiques de super-résolution.

3.5.2.1 Innovations architecturales pour le U-Net

Le cadre théorique de cette recherche intègre deux améliorations clés visant à renforcer l'efficacité du U-Net dans la phase finale du pipeline [Shang et al., 2023] :

1. **FD Info Splitter (module d'entrée)** Le module FD Info Splitter enrichit les représentations d'entrée du U-Net par :
   — Exploitation des composantes fréquentielles
   — Suppression adaptative du bruit

   Agissant comme couche de prétraitement spectral et attentionnel, il est particulièrement adapté aux tâches de super-résolution guidée et de restauration d'image conditionnée.
2. **Architecture d'entrée** Le module reçoit deux entrées complémentaires :
   — Image x_{ESPCN} : Approximation basse résolution du contenu original
   — Image x_t : Échantillon bruité à l'instant t du processus de diffusion
3. **Extraction des caractéristiques fréquentielles** L'image d'entrée (6 canaux) est divisée en deux blocs de 3 canaux, permettant l'extraction de trois types d'informations :
   a) xHF (Haute Fréquence) :
      — Transformée de Fourier → Filtrage passe-haut gaussien → TF inverse
      — Capture des détails fins et textures locales
   b) xLF (Basse Fréquence) :
      — Mécanisme d'attention guidé par les HF
      — Renforce les structures globales
   c) x'_t (Version débruitée) :
      — Modulation attentionnelle via embedding temporel
      — Implémentation par bloc ResSE

4. **Fusion des caractéristiques** Les trois sorties sont concaténées avec les entrées initiales (x_t et x_{ESPCN}), formant une carte de caractéristiques enrichie de 15 canaux intégrant simultanément :
   — Informations haute fréquence (détails, textures)
   — Structures globales basse fréquence
   — Composante débruitée

3.5.2.2 Concepts fondamentaux

Transformée de Fourier Outil mathématique fondamental convertissant les signaux dans le domaine fréquentiel, facilitant la résolution d'équations différentielles et de convolution par transformation en expressions algébriques.

Filtre passe-haut gaussien Filtre numérique atténuant les basses fréquences pour accentuer :
— Détails fins et contours
— Variations locales de luminosité
Caractérisé par son noyau gaussien à valeurs négatives périphériques, il assure des transitions douces limitant les artefacts visuels.

3.5.2.3 HF-Guided Cross Attention (module interne U-Net)

La technique HF-Guided Cross Attention (HF_guided_CA) exploite les informations haute fréquence issues de la transformation en ondelettes (DWT) pour guider dynamiquement les mécanismes d'attention lors de l'encodage. Son objectif principal : préserver et amplifier les détails fins (textures, contours) souvent dégradés dans les processus de compression ou super-résolution conventionnels.

Implémentation À chaque réduction de résolution dans l'encodeur, le module reçoit :
— Carte de caractéristiques du réseau (x)
— Signal HF issu de la DWT (requête externe)

Mécanisme d'attention croisée
(a) Clés/valeurs dérivées des caractéristiques internes (post-normalisation)
(b) Requête (*query*) projetée depuis le signal DWT
(c) Produit scalaire query $\times$ clé $\rightarrow$ carte d'attention
(d) Pondération des valeurs par la carte d'attention
(e) Combinaison résiduelle avec l'entrée initiale

Avantages Cette architecture permet :
— Intégration ciblée d'informations fréquentielles externes
— Fusion multi-échelle efficace entre caractéristiques apprises et détails analytiques
— Amélioration mesurable de la fidélité texturale
Le HF-Guided Cross Attention constitue ainsi un levier essentiel pour les reconstructions exigeant une haute précision.

3.5.2.4 Préparer son environnement de codage

En raison de la complexité des techniques employées et de l'ampleur du code de notre seconde expérience, l'environnement open-source **Kaggle** s'est avéré insuffisant. Nous avons donc choisi de migrer vers une solution plus robuste : **Google Colab Pro**. Ce nouvel environnement nous a permis d'entraîner notre modèle plus efficacement.

Google Colab Pro **Google Colab Pro** est la version payante de Google Colaboratory, un environnement de développement cloud idéal pour l'exécution de code Python, notamment dans les domaines de la science des données et de l'apprentissage automatique. Colab Pro offre des avantages significatifs par rapport à la version gratuite :

- Accès à des **GPU plus rapides** (tels que les NVIDIA P100 ou T4).
- **Sessions prolongées** (jusqu'à 24 heures).
- Moins d'interruptions durant l'exécution.
- Accès à un terminal Unix.
- Plus de mémoire **RAM** (jusqu'à 32 Go pour Colab Pro et 52 Go pour Colab Pro+).

Malgré ces atouts, Colab Pro présente certaines limites. L'accès est restreint à certains pays, les ressources GPU ne sont pas toujours garanties, et la durée des sessions reste plafonnée.

3.5.2.5 Stratégies d'implémentation et Observations

Pour améliorer la qualité des résultats, on a initialement envisagé de configurer l'**ESPCN** avec un **facteur d'agrandissement** $\times 3$. Cela aurait permis d'atteindre une résolution intermédiaire de 120×120 pixels, supposée offrir de meilleures performances que le facteur $\times 2$ (80×80). Théoriquement, cette approche aurait permis un **meilleur enrichissement fréquentiel** avant le passage au modèle de diffusion.
Cependant, cette stratégie a révélé une **difficulté pratique** : un redimensionnement ultérieur avec un facteur $\times 1$ par **interpolation bilinéaire** (pour correspondre à la taille de l'image bruitée attendue, soit 160×160) n'a, en réalité, aucun effet sur les dimensions de l'image. Cela entraîne une **incompatibilité** au moment de la concaténation avec l'entrée du modèle **U-Net**, qui nécessite une résolution précise et un nombre de canaux bien défini.
Compte tenu de la **contrainte de temps** du projet et pour assurer la **robustesse du pipeline**, j'ai finalement opté pour une approche plus sûre : maintenir un **facteur d'agrandissement** $\times 2$ dans l'ESPCN (produisant des images en 80×80), puis appliquer un **redimensionnement supplémentaire** ($\times 2$) avant l'entrée dans le U-Net de diffusion. Cette solution garantit la **compatibilité des dimensions** tout en respectant la structure du modèle.
La figure 3.11 montre la modification du nombre d'époques. Comme nous avons intégré des techniques guidées au modèle U-Net du processus de diffusion probabiliste, la complexité des calculs s'est accrue. Après plusieurs essais d'exécution, nous avons constaté, à travers les métriques d'évaluation, qu'à certaines époques, les performances du modèle diminuaient temporairement. Pour y remédier et laisser au modèle davantage de temps pour converger, nous avons décidé d'augmenter le nombre d'époques d'entraînement. Le paramètre `checkpoint_path` permet de sauvegarder régulièrement l'état du modèle au cours de l'entraînement, afin de pouvoir reprendre le

```
# ✅ Start training
unet, history = train_cond_unet(
    unet, espcn, train_loader, valid_loader,
    unet_optimizer, unet_loss, device,
    epochs=400,
    start_epoch=start_epoch,
    history=history,
    checkpoint_path=checkpoint_path
)
```

FIGURE 3.11 – Schéma d'entraînement du modèle

processus là où il s'est arrêté en cas d'interruption. Cela s'avère particulièrement utile dans des environnements instables comme Google Colab Pro, où les sessions peuvent expirer automatiquement, ou encore où la mémoire GPU peut se révéler insuffisante face à la complexité des calculs, ce qui est précisément le cas dans notre projet. Ce mécanisme de sauvegarde est donc essentiel pour éviter toute perte de progression et assurer la continuité de l'entraînement.

```
📂 Checking for existing checkpoint...
🔁 Resuming from epoch 344

🔁 Epoch 345/400
/usr/local/lib/python3.11/dist-packages/torch/functional.py:539:
UserWarning: torch.meshgrid: in an upcoming release, it will be
pass the indexing argument. (Triggered internally at
/pytorch/aten/src/ATen/native/TensorShape.cpp:3637.)
  return _VF.meshgrid(tensors, **kwargs)  # type: ignore[attr-de
  Batch 10/100 | Loss: 0.0009 | Time: 9.2s
  Batch 20/100 | Loss: 0.0009 | Time: 5.8s
  Batch 30/100 | Loss: 0.0012 | Time: 6.8s
  Batch 40/100 | Loss: 0.0011 | Time: 6.2s
```

FIGURE 3.12 – Reprise de l'entraînement après interruption

La figure 3.12 illustre la reprise de l'entraînement du modèle U-Net, conditionné par ESPCN dans un pipeline de diffusion probabiliste. Cette reprise fait suite à une **interruption** causée par une coupure de session dans **Google Colab Pro**. Le script a détecté un **fichier de sauvegarde (*checkpoint*)** contenant l'état du modèle arrêté à l'**époque 344**. L'entraînement a donc pu reprendre directement à partir de l'**époque 345**, sans repartir de zéro. À chaque époque, le modèle s'entraîne sur **100 lots (*batches*)** d'images.

Comme le montre la **Figure 3.13**, le modèle apprend efficacement, avec une perte très faible de **0,0010** et un **PSNR élevé de 29,49** à la fin de l'entraînement. On observe également que la progression est remarquablement stable, ce qui témoigne d'une bonne robustesse malgré la complexité inhérente du modèle conditionné par diffusion. Ces résultats confirment que l'augmentation à **400 époques** a permis une

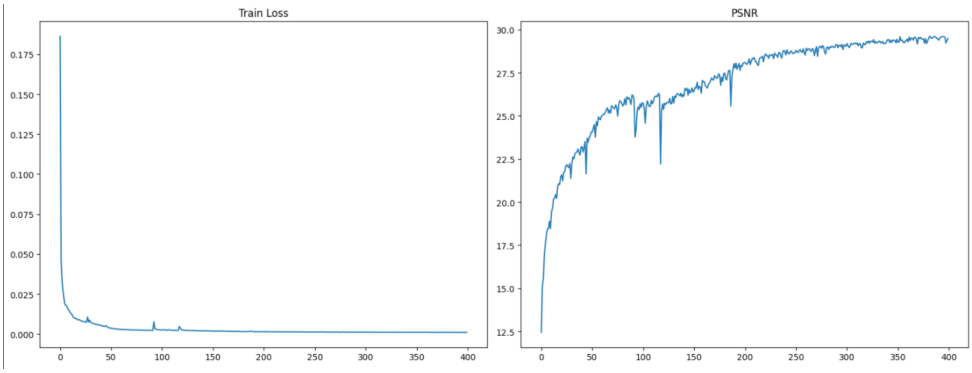

FIGURE 3.13 – Courbes d'Évaluation : Taux d'Erreur et PSNR

convergence plus fine et stable.

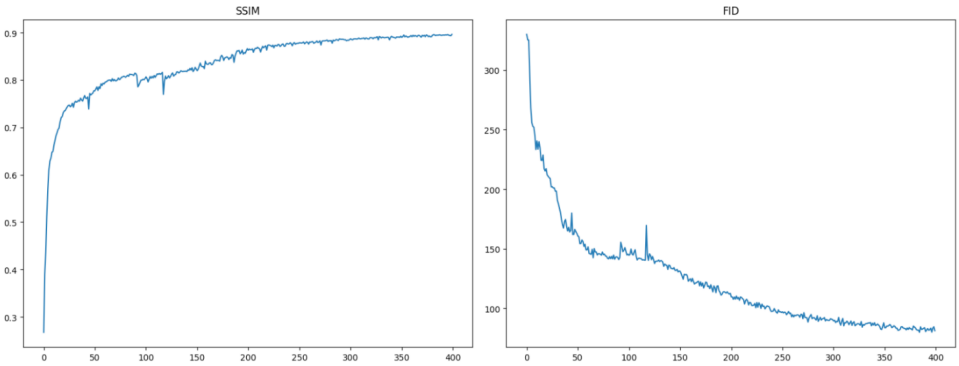

FIGURE 3.14 – Courbes d'Évaluation : SSIM et FID

Comme l'illustre la Figure 3.14, le modèle s'améliore progressivement, produisant des images **visuellement plus fidèles** avec un **FID bas de 81,40** et **structurellement plus proches** des vraies images avec un **SSIM élevé de 0,8962**. L'évolution des deux courbes valide la **qualité globale de la reconstruction**.

Voici une analyse des résultats visuels présentés dans la figure 3.15

- **Première colonne (à gauche)** : Montre l'**image d'entrée en basse résolution** (ex : test1.jpg, 6 et 7).
- **Deuxième colonne** : Présente la sortie du modèle **ESPCN** avec ses valeurs de **PSNR** et **SSIM**.
- **Troisième colonne** : Affiche la sortie du modèle **Cond U-Net**, dont les valeurs de **PSNR** et **SSIM** sont systématiquement supérieures à celles du modèle ESPCN.
- **Quatrième colonne** : Contient les **images de référence en haute résolution (*ground truth*)**, redimensionnées à la même taille pour faciliter la comparaison.

Le modèle **Cond U-Net** fournit des résultats visuellement plus nets et plus fidèles aux images d'origine.

La table 3.2 met en valeur la méthode **ResDif** dans ses versions successives et montre une amélioration constante des performances, particulièrement au niveau du PSNR

FIGURE 3.15 – Résultats Visuels de Quelques Images

Méthode	PSNR	SSIM	FID
SR3 [Lugmayr et al., 2020]	26.17	0.65	111.45
SRDiff [Li et al., 2022]	26.87	0.69	110.32
ResDif [Li et al., 2022]	27.94	0.72	106.71
ResDif (Démarche 1)	28.85	0.92	–
ResDif (Démarche 2)	29.49	0.89	81.40

TABLE 3.2 – Comparatif de certains travaux existants avec les nouvelles approches proposées

et du SSIM.
Le **FID** chute de manière significative dans la deuxième approche de **ResDif (de 111,45 à 81,40**), indiquant une meilleure qualité perceptuelle des images générées.
ResDif surpasse clairement **SR3** et **SRDiff** sur les trois métriques.
Ce qui suggère que les nouveaux modèles **ResDif** proposés sont *plus performants que les approches existantes*, à la fois du point de vue objectif **(PSNR, SSIM)** et perceptuel **(FID)**.

3.6 Conclusion

Dans ce chapitre, nous avons détaillé l'implémentation de notre approche de super-résolution d'images basée sur le modèle ResDiff. Spécifiquement conçue pour traiter des entrées de basse résolution (e.g., 40×40 pixels), cette méthode démontre une adéquation particulière avec la complexité du dataset Div2K. Contrairement aux ensembles de données centrés sur les visages (tels que FFHQ ou CelebA), les jeux DIV2K et Urban100 présentent des défis de restauration plus exigeants :

— **DIV2K** : contient des scènes naturelles à la richesse et variété significatives ;
— **Urban100** : intègre de nombreuses structures géométriques fines, augmentant considérablement la difficulté de reconstruction.

L'architecture ResDiff exploite une projection des images dans un espace latent de dimension réduite, où s'effectue le processus de super-résolution. Cette stratégie permet une reconstruction plus efficace et ciblée des détails complexes caractéristiques de ces jeux de données exigeants.

Conclusion générale

Ce travail de recherche nous a permis d'explorer l'ensemble des étapes, depuis la conceptualisation de la super-résolution d'image jusqu'à l'implémentation concrète de la méthode proposée. Il a illustré la puissance de l'informatique en tant qu'outil d'expérimentation à grande échelle, notamment dans le domaine de l'apprentissage profond.

L'approche développée, nommée **ResDiff**, est une architecture hybride innovante combinant les atouts de deux algorithmes complémentaires : l'Efficient Sub-Pixel Convolutional Neural Network (ESPCN) pour une première estimation rapide, et les modèles de diffusion probabilistes pour un raffinement des détails et une génération de textures fines. Évalué sur le jeu de données de référence DIV2K, le modèle obtenu a démontré des performances très encourageantes, les résultats ayant pu être comparés favorablement à ceux de travaux existants dans le domaine.

Il est crucial de souligner que les avancées significatives observées en super-résolution ne découlent pas uniquement d'une amélioration matérielle ou de l'augmentation de la complexité des jeux de données. Elles sont avant tout le fruit de l'émergence d'idées novatrices, du développement d'algorithmes plus robustes et de la conception d'architectures de réseaux de neurones de plus en plus efficaces.

Ce projet a également constitué une excellente opportunité de mettre en pratique et d'approfondir nos connaissances en réseaux de neurones avancés, tout en favorisant l'acquisition de nouvelles compétences. Le temps conséquent consacré à l'étude d'articles scientifiques a représenté une introduction précieuse à la démarche de recherche fondamentale et appliquée.

Dans la continuité de ce travail, une perspective d'amélioration prometteuse consisterait à intégrer un troisième algorithme dans le pipeline de notre modèle hybride : l'algorithme évolutionnaire **Differential Evolution (DE)**. Ce module serait positionné après l'étape ESPCN et avant ou en synergie avec le modèle de diffusion probabiliste.

Cette intégration viserait à renforcer les performances du modèle à plusieurs niveaux. D'une part, DE permettrait une optimisation automatique et adaptative des hyperparamètres, contribuant ainsi à une amélioration significative et autonome de la qua-

lité des reconstructions, sans nécessiter de réglages manuels intensifs. D'autre part, étant un algorithme ne reposant pas sur le calcul de gradients, DE offre une compatibilité naturelle avec des modules non différentiables, facilitant son incorporation dans des architectures hybrides complexes. De plus, sa capacité intrinsèque à s'adapter dynamiquement image par image permettrait une meilleure gestion de la diversité des contenus à traiter, particulièrement dans les cas complexes ou à haute fréquence.

Enfin, la robustesse de Differential Evolution face aux minima locaux et sa stratégie d'exploration globale de l'espace de recherche favorisent une optimisation plus stable et efficace, même dans des paysages de fonctions de coût fortement non-convexes. Une autre capacité notable de DE est la possibilité d'optimiser simultanément plusieurs métriques (telles que le PSNR, le SSIM et le LPIPS) en définissant une fonction de coût multi-objectifs. Ceci permettrait d'atteindre un équilibre optimal entre la fidélité objective et la qualité perceptuelle des images reconstruites, ce qui est souvent un défi dans les tâches de super-résolution.

Bibliographie

[Alty and Took, 2024] Alty, S. R. and Took, C. C. (2024). Widrow-hoff lms adaline demonstrator for schools and colleges. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 12807–12810. IEEE.

[An et al., 2022] An, T., Zhang, X., Huo, C., Xue, B., Wang, L., and Pan, C. (2022). Tr-misr : Multiimage superresolution based on feature fusion with transformers. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 15 :1373–1388.

[Bengio et al., 2013] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning : A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8) :1798–1828.

[Chen et al., 2023] Chen, Z., Chen, S., and Hu, F. (2023). Cta-unet : Cnn-transformer architecture unet for dental cbct images segmentation. *Physics in Medicine & Biology*, 68(17) :175042.

[DataCamp, 2025] DataCamp (2025). Introduction à kaggle. Consulté le 23 mai 2025.

[Deekshith Shetty et al., 2020] Deekshith Shetty, H., Varma, M. J., Navi, S., and Ahmed, M. R. (2020). Diving deep into deep learning : history, evolution, types and applications. *International Journal of Innovative Technology and Exploring Engineering*, 9(3) :2278–3075.

[Dong et al., 2014] Dong, C., Loy, C. C., He, K., and Tang, X. (2014). Learning a deep convolutional network for image super-resolution. In *Computer Vision - ECCV 2014*, volume 8692 of *Lecture Notes in Computer Science*, pages 184–199. Springer.

[Elsken et al., 2019] Elsken, T., Metzen, J. H., and Hutter, F. (2019). Neural architecture search : A survey. *Journal of Machine Learning Research*, 20(55) :1–21.

[Garber et al., 2020] Garber, B., Grossman, A., and Johnson-Yu, S. (2020). Image super-resolution via a convolutional neural network. *Stanford University*.

[GeeksforGeeks, 2025a] GeeksforGeeks (2025a). Pathlib Module in Python. https://www.geeksforgeeks.org/pathlib-module-in-python/. Consultée le 25 mai 2025.

[GeeksforGeeks, 2025b] GeeksforGeeks (2025b). Python Math Module. https://www.geeksforgeeks.org/python-math-module/. Consultée le 25 mai 2025.

[Hebb, 2005] Hebb, D. O. (2005). *The organization of behavior : A neuropsychological theory*. Psychology press.

[Hopfield, 1982] Hopfield, J. J. (1982). Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8) :2554–2558.

[IBM, 2025] IBM (2025). Introduction à pytorch. Consulté le 26 mai 2025.

[ImageNet, 2025] ImageNet (2025). Base de données d'images pour la recherche en vision par ordinateur. Consulté le 16 avril 2025.

[Intelligence Artificielle School, 2025] Intelligence Artificielle School (2025). Matplotlib. https://www.intelligence-artificielle-school.com/ecole/technologies/matplotlib/. Consultée le 26 mai 2025.

[Kawar et al., 2022] Kawar, B., Elad, M., Ermon, S., and Song, J. (2022). Denoising diffusion restoration models. *ICLR Workshop on Deep Generative Models for Highly Structured Data*.

[Khaledyan et al., 2020] Khaledyan, D., Amirany, A., Jafari, K., Moaiyeri, M. H., Khuzani, A. Z., and Mashhadi, N. (2020). Low-cost implementation of bilinear and bicubic image interpolation for real-time image super-resolution. In *2020 IEEE Global Humanitarian Technology Conference (GHTC)*, pages 1–5. IEEE.

[Khattab et al., 2023] Khattab, M. M., Zeki, A. M., Alwan, A. A., Bouallegue, B., Matter, S. S., and Ahmed, A. M. (2023). A hybrid regularization-based multiframe super-resolution using bayesian framework. *Computer Systems Science & Engineering*, 44(1).

[Kim et al., 2016] Kim, J., Lee, J. K., and Lee, K. M. (2016). Deeply-recursive convolutional network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1637–1645.

[Kohonen, 2012] Kohonen, T. (2012). *Associative memory : A system-theoretical approach*, volume 17. Springer Science & Business Media.

[Le Bolzer et al., 2020] Le Bolzer, F., Lambert, A., and Schnitzler, F. (2020). Vers des réseaux de neurones récurrents flexibles. In *Conférence sur l'Apprentissage automatique*.

[Ledig et al., 2017] Ledig, C., Theis, L., Huszár, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A. P., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 105–114.

[LeewayHertz, 2025] LeewayHertz (2025). Key concepts behind diffusion models. Consulté le 13 mai 2025.

[Li et al., 2022] Li, H., Yang, Y., Chang, M., Chen, S., Feng, H., Xu, Z., Li, Q., and Chen, Y. (2022). Srdiff : Single image super-resolution with diffusion probabilistic models. *Neurocomputing*, 479 :47–59.

[Lugmayr et al., 2020] Lugmayr, A., Danelljan, M., Gool, L. V., and Timofte, R. (2020). Srflow : Learning the super-resolution space with normalizing flow. In *Computer Vision - ECCV 2020*, volume 12350 of *Lecture Notes in Computer Science*, pages 715–732. Springer.

[McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5 :115–133.

[Menon et al., 2020] Menon, S., Damian, A., Hu, S., Ravi, N., and Rudin, C. (2020). Pulse : Self-supervised photo upsampling via latent space exploration of generative models. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, pages 2437–2445.

[Molini et al., 2019] Molini, A., Valsesia, D., Fracastoro, G., and Magli, E. (2019). Deepsum : Deep neural network for super-resolution of unregistered multitemporal images. *IEEE Transactions on Geoscience and Remote Sensing*, 58(5) :3644–3656.

[NumPy, 2025] NumPy (2025). Framework de calcul scientifique. Consulté le 25 mai 2025.

[NVIDIA, 2025a] NVIDIA (2025a). Cuda toolkit documentation. Consulté le 3 juin 2025.

[NVIDIA, 2025b] NVIDIA (2025b). Tensorflow technical overview. Consulté le 25 mai 2025.

[OpenCV, 2025] OpenCV (2025). Open source computer vision library. Consulté le 25 mai 2025.

[Python Software Foundation, 2025] Python Software Foundation (2025). Documentation officielle python. Consulté le 25 mai 2025.

[PyTorch, 2025] PyTorch (2025). Documentation de torchvision.transforms. Consulté le 26 mai 2025.

[Reshma and Thomas, 2023] Reshma, R. S. and Thomas, J. (2023). Review on video super resolution : Methods and metrics. In *2023 International Conference on Control, Communication and Computing (ICCC)*, pages 1–6. IEEE.

[Rombach et al., 2022] Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. (2022). High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10674–10685.

[Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron : a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6) :386.

[Saharia et al., 2022] Saharia, C., Ho, J., Chan, W., Salimans, T., Fleet, D. J., and Norouzi, M. (2022). Image super-resolution via iterative refinement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1–14.

[Salvetti et al., 2020] Salvetti, F., Mazzia, V., Khaliq, A., and Chiaberge, M. (2020). Multi-image super resolution of remotely sensed images using residual attention deep neural networks. *Remote Sensing*, 12(14) :2207.

[Schmidhuber, 2022] Schmidhuber, J. (2022). Annotated history of modern ai and deep learning. *arXiv preprint arXiv :2212.11279*.

[scikit image, 2025] scikit image (2025). Implémentation du ssim. Consulté le 3 juin 2025.

[Shang et al., 2023] Shang, S., Shan, Z., Liu, G., and Zhang, J. (2023). Resdiff : Combining cnn and diffusion model for image super-resolution. *arXiv preprint arXiv :2303.08714*, v2.

[Shi et al., 2016] Shi, W., Caballero, J., Huszár, F., Totz, J., Aitken, A. P., Bishop, R., Rueckert, D., and Wang, Z. (2016). Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pages 1874–1883.

[Shi et al., 2021] Shi, W., Ledig, C., Wang, Z., Theis, L., and Huszar, F. (2021). Super resolution using a generative adversarial network. US Patent 11,024,009.

[Superannotate, 2025] Superannotate (2025). Diffusion models explained. Consulté le 1 mai 2025.

[Tahiri, 2024] Tahiri, F. (2024). Artificial intelligence and deep learning : The concept and the applications. *REVUE DE L'ENTREPRENEURIAT ET DE L'INNOVATION*, 6(23).

[Wang et al., 2018] Wang, X., Yu, K., Wu, S., Gu, J., Liu, Y., Dong, C., Qiao, Y., and Loy, C. C. (2018). ESRGAN : Enhanced super-resolution generative adversarial networks. In *Computer Vision - ECCV 2018 Workshops*, volume 11133 of *Lecture Notes in Computer Science*, pages 63–79. Springer.