

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche
Scientifique
Université Tahri Mohamed de Béchar
Faculté des Sciences Exactes
Département des Mathématiques et Informatique



Projet de fin d'études

En vue de l'obtention du diplôme de Licence en Informatique

Option:Systèmes d'Informations

THEME

**AMELIORATION D'IMAGES PAR LES RESEAUX
NEURONNAUX GRAPHIQUES**

Présenté par:

Chaoufi Brahim

BenouahdiAbdelouahab

Encadré par: S. Himri

Année universitaire:2024/2025

REMERCIEMENTS

**EN PRÉAMBULE À CE MÉMOIRE, NOUS TENONS À
REMERCIER LE BON DIEU TOUT PUISSANT DE
NOUS AVOIR OFFERT L'OPPORTUNITÉ DE
FRANCHIR CE STADE DE SAVOIR, ET DE NOUS
AVOIR DONNÉ LE COURAGE ET LA PATIENCE DE
RÉALISER CE MODESTE TRAVAIL.**

**NOUS REMERCIONS NOTRE ENCADREUSE DR
HIMRI SALIHA, D'AVOIR ACCEPTÉ DE DIRIGER
ET DE RÉALISER NOTRE TRAVAIL.**

**NOTRE RECONNAISSANCE VA ÉGALEMENT
À TOUT LE PERSONNEL ET LES
ENSEIGNANTS DE LA FACULTÉ DE SCIENCE
EXACTE.**

DÉDICACES

NOUS TENONS À DÉDIER CE MÉMOIRE DE FIN D'ÉTUDE

A

NOS TRÈS CHERS PARENTS QUE DIEU LES PROTÈGE.

A

NOTRE PROF **HIMRISALIHA**

A

À NOS CHERS AMIS ABDELJALIL, HACHEMI, HOUCINE
ABDELSAMAD , HAYDA, DAOUDI, FELLOUS

A

TOUTE NOTRE FAMILLE ET NOTRE BELLE-FAMILLE.

LISTE DES FIGURES

FIGURE	PAGE
FIG.I.2—IMAGEBRUITEE	19
FIG.I.2—SCHEMA D'AMELIORATION D'IMAGE	20
FIG.I.8—TRANSFORMATION LOCALE	21
FIG.I.9— NOYAUX D'UN FILTRE	21
FIG.I.10— FILTREMOYENNEUR	22
FIG.I.11— FILTRE GAUSSIEN	22
FIG.I.12— FILTREMEDIAN	23
FIG.I.12— FILTRE NAGAO FENETRES 5X5	24
FIG.I.13— REPRESENTATION FREQUENTIELLE D'UNE IMAGE	25
FIG.I.14— INTERPRETATION DE LA TRANSFORMEE DE FOURIER D'IMAGE	26
FIG.I.15— IMAGE FILTREE AVEC LE FILTREDE GABOR (135°)	27
FIG.I.16—AMELIORATION D'IMAGE PAR CNN	28
FIG.II.1— GRAPHE AVEC TROIS SOMMETS ET QUATRE ARCS	36
FIG.II.2— EXEMPLE D'UNE ARCHITECTURE TYPIQUE D'UN CNN	42
FIG.II.3— DE LA MATRICE D'UNE IMAGE VERS UN GRAPHE	43
FIG.II.4— DU CNN AU GNN	45
FIG.II.5— IMAGE ET SA REPRESENTATION GRAPHIQUE.[SCA 05]	46
FIG.II.6— QUELQUES INFORMATIONS REPRESENTES PAR GRAPHE	46
FIG.II.7— GRAPHE CORRESPONDANT AUX MULTI-RESOLUTION	47
FIG.II.8— REPRESENTATION D'UN CNN PAR GRAPHE	47
FIG.II.9— SEMI-SUPERVISED LEARNING VIA GRAPHE	48
FIG.III.1— IMAGE DEBRUITEE PAR GNN	61
FIG.III.2— IMAGE ET GRAPHE	62
FIG.III.3— IMAGE ORIGINALE ET IMAGE BRUITEE	73
FIG.III.4— DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN	73
FIG.III.5—IMAGE ORIGINALE ET IMAGE BRUITEE	74
FIG.III.6— DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN	74
FIG.III.7— IMAGE ORIGINALE ET IMAGE BRUITEE	75
FIG.III.8— DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN	75
FIG.III.9— IMAGE ORIGINALE ET IMAGE BRUITEE	76
FIG.III.10— DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN	77

TABLE DES MATIERES

CHAPITRE I : LES TECHNIQUES D'AMELIORATION D'UNE IMAGE

I.1 INTRODUCTION	5
I.2 POURQUOI EFFECTUE-T-ON UN TRAITEMENT D'IMAGE ?	5
I.2.1 POURQUOI AMELIORER UNE IMAGE ?	6
I.2.2 PRINCIPE DE L'AMELIORATION D'IMAGE	6
I.2.3 EXEMPLES D'UTILISATION	7
I.2.4 BRUIT	8
II.3 TECHNIQUES CLASSIQUES D'AMELIORATION	9
I.3.1 TECHNIQUES D'AMELIORATION LOCALE	9
I.3.1.1 FILTRE MOYENNEUR	10
I.3.1.2 FILTRE GAUSSIEN	11
I.3.1.3 FILTRE MEDIAN	11
I.3.1.4 FILTRE NAGAO	12
I.3.2 TECHNIQUES D'AMELIORATION PAR FILTRAGE FREQUENTIEL.....	12
I.3.2.1 TRANSFORMEE DE FOURIER	13
I.3.2.2 FILTRE DE GABOR.....	13
I.4 TECHNIQUES D'AMELIORATION PAR LES CNN	15
I.5 CONCLUSION.....	16

CHAPITRE II : FONDEMENTS DES GRAPHES NEURAL NETWORKS

II.1 INTRODUCTION.....	21
II.2 INTRODUCTION AUX GRAPHES.....	21
II.2.1 QU'EST CE QU'UN GRAPHE ?	21
II.2.1.1 CLASSIFICATION DES SOMMETS	22
II.2.1.2 RELATIONS ENTRE SOMMETS	23
II.2.2 CATEGORIES DE GRAPHES	23
II.3 REPRESENTATION DES GRAPHES	24
II.3.1 MATRICE D'ADJACENCE	25
II.3.2 LISTE D'ADJACENCE	25

II.3.3 MATRICE D'INCIDENCE	25
II.4 DE LA CONVOLUTION CLASSIQUE (CNN) AU GRAPHIQUE (GNN).....	26
II.4.1 Qu'est-ce qu'un CNN ?	26
II.4.2 ARCHITECTURE TYPIQUE D'UN CNN	26
II.4.3 DOMAINE D'APPLICATIONS	27
II.4.4 DE LA CONVOLUTION EUCLIDIENNE AU GRAPHES	28
II.4.5 FONDEMENTS MATHÉMATIQUES : DU CNN AUX GNNs	29
II.5 NOTIONS DE BASE DES GRAPH NEURAL NETWORKS	30
II.5.1 Historique	30
II.5.2 PRINCIPE DE FONCTIONNEMENT DES GNNs	33
II.5.3 CYCLE DE PROPAGATION	34
II.5.4 PRINCIPAUX TYPES DE GNN.....	34
II. 6 DEBRUITAGE AVEC GNN	35
II. 8 DOMAINE D'APPLICATIONS	37
I.8 CONCLUSION	38

CHAPITRE III : TESTS ET RESULTATS

III.1 INTRODUCTION	42
III.1.1 UTILISATION DES GNN DANS LE TRAITEMENT D'IMAGES.....	43
III.1.2 POURQUOI UTILISER UN GNN POUR DEBRUITER UNE IMAGE ?	43
III.3 FONDEMENTS DES GNNs A PARTIR D'UNE IMAGE	44
III. 4 ENVIRONNEMENT DE PROGRAMMATION " PYTHON "	46
III. 4.1 INTRODUCTION	46
III. 4.2 CARACTÉRISTIQUES TECHNIQUES DE PYTHON	46
III. 4.3 DOMAINES D'UTILISATION DE PYTHON	47
III. 4.4 LES AVANTAGES DE PYTHON.....	48
III.5 DESCRIPTION DU PROGRAMME DEBRUITAGE D'UNE IMAGE VIA GNN	49
III.6 TESTS ET RESULTATS.....	53
CONCLUSION GENERALE	57

INTRODUCTION GÉNÉRALE

Introduction Générale

Introduction Générale



pour les scientifiques, la notion d'image est une source de préoccupation de plus en plus grande et importante depuis quelques années. De ce fait, le traitement d'images est un domaine très vaste et prometteur qui a connu, et qui connaît encore, un développement important depuis des années.. Le traitement d'image peut être défini comme l'ensemble des méthodes et techniques opérant sur l'image afin d'extraire les informations les plus pertinentes ou tout simplement pour fournir une image plus perceptible à l'œil humain ou pour extraire de l'information, par exemple identifier une séquence de texte ou un chromosome, éviter un obstacle, ou détecter des zones soumises à l'érosion. Les techniques les plus connus utilisées pour le traitement d'images sont : extraction des caractéristiques de l'image, la segmentation d'image, le rehaussement d'image (amélioration de la qualité d'image) et le recalage d'image.

En effet, parmi les nombreuses tâches du traitement d'images, l'amélioration d'image bruitée joue un rôle crucial, tant pour des applications médicales, industrielles, que pour la photographie numérique. Elle consiste à restaurer la qualité d'une image dégradée par du bruit, en révélant les structures originales tout en supprimant les perturbations indésirables.

Traditionnellement, cette tâche repose sur des méthodes comme les filtres linéaires (gaussien, médian) ou des techniques plus avancées comme les ondelettes, ou encore les autoencodeurs convolutifs. Cependant, ces approches présentent des limites : elles peinent à adapter leur comportement aux structures complexes de l'image ou à capturer des relations contextuelles riches entre les pixels.

Introduction Générale

C'est dans ce contexte que les Graph Neural Networks (GNN) émergent comme une solution innovante et prometteuse. Contrairement aux réseaux de neurones convolutifs (CNN), qui exploitent la structure régulière des images, les GNN permettent de modéliser les pixels ou régions d'une image sous forme de graphes, où les relations ne sont plus contraintes par une grille fixe mais définies de manière flexible : similarité d'intensité, proximité spatiale, ou même sémantique.

Dans un GNN, chaque nœud (souvent un pixel) interagit avec ses voisins à travers un processus itératif appelé propagation ou message passing, permettant de diffuser l'information utile tout en atténuant l'effet du bruit. Ce mécanisme permet aux GNN de capturer les dépendances locales et globales dans l'image, même dans des zones fortement dégradées. Ainsi, ils offrent un cadre naturel pour le débruitage adaptatif, s'appuyant sur la structure interne de chaque image.

Notre objectif est de présenter de manière claire et progressive le rôle des GNN dans l'amélioration d'images bruitées et de tester sur plusieurs images 2D et d'observer les résultats pour chacun des cas étudiés pour cela notre plan de travail serait structuré en trois chapitres de la manière suivante :

Dans le premier chapitre nous allons aborder en générale les caractéristiques de l'image numériques à savoir les relations entre pixels, les outils de traitement d'images, etc. Ensuite les différentes techniques d'amélioration d'images.

Dans le deuxième chapitre nous avons présenté les fondements des graph neural networks (GNN) et les notions de base des graph neural networks.

Dans le chapitre trois nous présentons notre application suivie de quelques tests expérimentaux.

Nous terminons notre mémoire par une Conclusion générale.

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE

CHAPITRE I

LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

SOMMAIRE

I.1 INTRODUCTION	5
I.2 POURQUOI EFFECTUE-T-ON UN TRAITEMENT D'IMAGE ?	5
I.2.1 POURQUOI AMELIORER UNE IMAGE ?	6
I.2.2 PRINCIPE DE L'AMELIORATION D'IMAGE.....	7
I.2.3 EXEMPLES D'UTILISATION	8
I.2.4 BRUIT	8
II.3 TECHNIQUES CLASSIQUES D'AMELIORATION	9
I.3.1 TECHNIQUES D'AMELIORATION LOCALE.....	10
I.3.1.1 FILTRE MOYENNEUR	10
I.3.1.2 FILTRE GAUSSIEN	11
I.3.1.3 FILTRE MEDIAN	12
I.3.1.4 FILTRE NAGAO	12
I.3.2 TECHNIQUES D'AMELIORATION PAR FILTRAGE FREQUENTIEL.....	13
I.3.2.1 TRANSFORMEE DE FOURIER	13
I.3.2.2 FILTRE DE GABOR	14
I.4 TECHNIQUES D'AMELIORATION PAR LES CNN.....	15
I.5 CONCLUSION.....	17

I.1 INTRODUCTION

Le traitement d'images constitue un champ de recherche et d'application particulièrement vaste, en plein essor depuis plusieurs décennies. Il englobe l'ensemble des méthodes et techniques algorithmiques visant à transformer, analyser ou interpréter des images numériques. L'objectif peut être l'amélioration de la qualité visuelle, l'extraction d'informations pertinentes ou encore la facilitation d'une étape de traitement ultérieure.

Le traitement d'images joue ainsi un rôle fondamental dans de nombreux domaines, allant de la vision par ordinateur à l'imagerie médicale, en passant par la surveillance, la robotique, etc.



FIG.I.1—AMELIORATION D'IMAGE

I.2 POURQUOI EFFECTUE-T-ON UN TRAITEMENT D'IMAGE ?

Le traitement d'images est une discipline située à l'interface de l'informatique, des mathématiques appliquées et de l'intelligence artificielle. Elle se consacre à l'étude, la manipulation et l'analyse d'images numériques à travers l'application d'algorithmes dédiés.

L'objectif principal du traitement d'image est d'agir sur les données visuelles pour en modifier le contenu de manière utile ou exploitable. Les principales finalités du traitement d'images sont les suivantes :

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

- Amélioration visuelle : il s'agit d'optimiser la qualité perçue de l'image, par exemple en corrigeant le bruit, en renforçant les contrastes ou en révélant des détails peu visibles.
- Extraction d'informations : le traitement permet d'isoler et d'identifier des structures ou des motifs pertinents dans l'image, facilitant ainsi des tâches telles que la reconnaissance de formes, la segmentation, la détection d'objets ou l'analyse de textures.

I.2.1 POURQUOI AMELIORER UNE IMAGE ?

L'amélioration d'image, ou image enhancement, est un domaine de recherche fondamental en traitement d'images, largement exploré par la communauté scientifique.

Elle vise à transformer une image numérique afin d'en optimiser les caractéristiques visuelles facilitant ainsi son interprétation par l'œil humain ou par des systèmes automatisés [Blo 08][Pil 15].

Les objectifs principaux de l'amélioration d'image sont les suivants :

- OPTIMISER LA QUALITE VISUELLE : en ajustant des paramètres tels que la luminosité, le contraste, la netteté ou la saturation, on rend l'image plus attrayante et plus lisible.
- FACILITER L'INTERPRETATION HUMAINE : en mettant en évidence des détails ou des structures spécifiques, l'image devient plus compréhensible pour l'observateur, ce qui est crucial dans des domaines comme l'imagerie médicale ou la surveillance.
- RESTAURER L'IMAGE : en éliminant les défauts et les bruits, et en renforçant les informations utiles qu'elle contient, on améliore la qualité de l'image pour une analyse plus précise.

- ⊕ METTRE EN EVIDENCE DES REGIONS SPECIFIQUES : en accentuant certaines zones d'intérêt, on facilite la détection et l'analyse de ces régions, ce qui est essentiel dans des applications telles que la reconnaissance d'objets ou la segmentation d'images.

I.2.2 PRINCIPE DE L'AMELIORATION D'IMAGE

L'amélioration d'image repose sur l'analyse et la correction des imperfections introduites lors des étapes d'acquisition et de numérisation. La qualité initiale d'une image dépend de plusieurs facteurs techniques et environnementaux qui influencent directement les opérations nécessaires pour son amélioration. Les principaux paramètres à considérer sont [Tup 16][Sun 12][BER 11][Zha 14] :

- ⊕ RESOLUTION D'ACQUISITION ET MODE DE CODAGE : La résolution détermine le niveau de détail capturé, tandis que le codage (par exemple, la profondeur de bits) affecte la précision des valeurs de pixel. Une faible résolution ou un codage inadéquat peut entraîner une perte d'informations essentielles.
- ⊕ REGLAGES OPTIQUES : La qualité des lentilles, la mise au point et l'ouverture influencent la netteté et la clarté de l'image. Des réglages optiques inappropriés peuvent introduire des distorsions ou des flous.
- ⊕ CONDITIONS D'ECLAIRAGE : L'éclairage lors de la capture d'image joue un rôle crucial. Un éclairage insuffisant ou inégal peut provoquer des ombres indésirables, un faible contraste ou une mauvaise reproduction des couleurs.
- ⊕ BRUIT DE LA CHAINE DE TRANSMISSION : Le bruit peut être introduit à différentes étapes, notamment par les capteurs, les circuits électroniques ou lors de la transmission des données. Ce bruit se manifeste souvent par des variations aléatoires de l'intensité des pixels, réduisant la qualité de l'image.

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

I.2.3 EXEMPLES D'UTILISATION

L'amélioration d'image contribue dans plusieurs domaines comme par exemple [VIN 19] [Vil 95]:

- Reconnaissance de document
- Images aériennes ou issues des satellites
- Images météorologiques
- Imagerie médicale
- Biométrie
- Surveillance vidéo

I.2.4 BRUIT

Le bruit est une information parasite qui s'ajoute aléatoirement à une image numérique causant la perte de la netteté dans les détails. Ce bruit peut être d'origines diverses [Vil 95][Tho 18]:

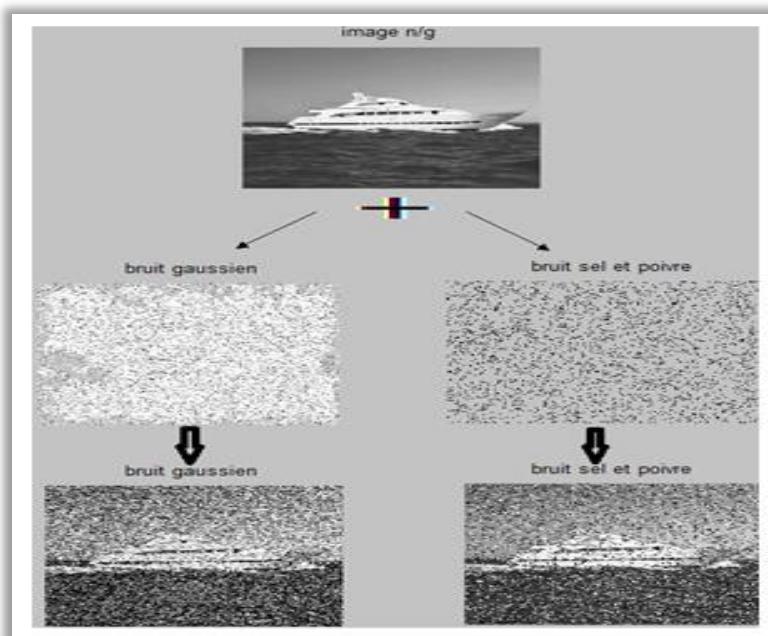


FIG.I.2 – IMAGE BRUITEE

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

- ❖ Bruits D'acquisition : flou, poussière, bougé.
- ❖ Bruits Lie Au Systeme d'acquisition (capteur) : mauvaise mise au point, bruit thermique, bruit de grenaille.
- ❖ Bruit Gaussien (Additive) : est obtenu en ajoutant à chaque pixel une valeur aléatoire .Ce bruit modélise bien un grand nombre de bruits de capteurs visuels. Pour avoir une image nette il faut faire appel à des techniques permettant de «supprimer» le bruit (opération de lissage) et/ou de mettre en évidence et détecter les points frontières.
- ❖ Bruit Sel Et Poivre Ou Bruit (Impulsionnel) : Ce bruit apparaît sous la formes de points noirs et blancs réparties arbitrairement sur une image. Il peut être causé par un capteur comportant des pixels défectueux, ou lors de la transmission de l'image ou la conversion analogique numérique [NOA 10].

II.3 TECHNIQUES CLASSIQUES D'AMELIORATION

Rendre les images plus aptes à l'interprétation humaine ou à celle de la machine fait appel à des prétraitements d'images. En effet les prétraitements d'une image visent à améliorer les caractéristiques d'une image c'est-à-dire améliorer sa qualité visuelle, la restaurer en éliminant les défauts et les bruits et en renforçant l'information utile qu'elle contienne[CHI 11][Bar 12][Xav 07].



FIG.I.2—SCHEMA D'AMELIORATION D'IMAGE

I.3.1 TECHNIQUES D'AMELIORATION LOCALE

A partir des valeurs des composantes d'une image numérique, on calcule de nouvelles valeurs définissant les composantes de l'image transformée. Les composantes du nouveau pixel sont calculées à partir de celles des pixels d'un voisinage du pixel initial [MAS 19].

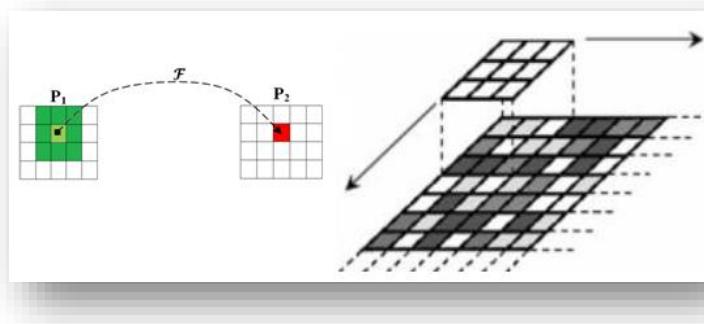


FIG.I.8–TRANSFORMATION LOCALE

Un filtre est un masque "fenêtre" carré, généralement de dimension impaire qui est appliqué à une image en se déplaçant horizontalement et verticalement sur chaque pixel de l'image.

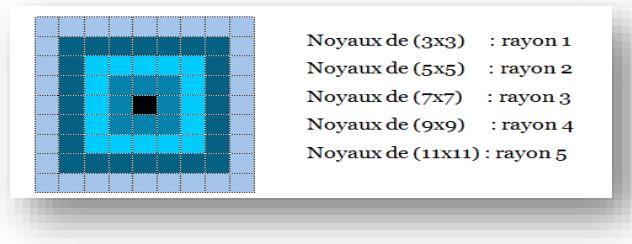


FIG.I.9– NOYAUX D'UN FILTRE

I.3.1.1 FILTRE MOYENNEUR

Le principe de filtre moyen que la valeur d'un pixel est relativement similaire à son voisinage [NAO 10]. Il fait donc remplace la valeur du pixel central par la moyenne de niveaux de gris pondérée de ses voisins (N-1).

$$g(x, y) = \frac{1}{N} \sum_{(i,j) \in N} f(x, y) \quad (\text{II.3})$$

Ce filtre est basé sur le masque de convolution, voir la figure suivante :

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

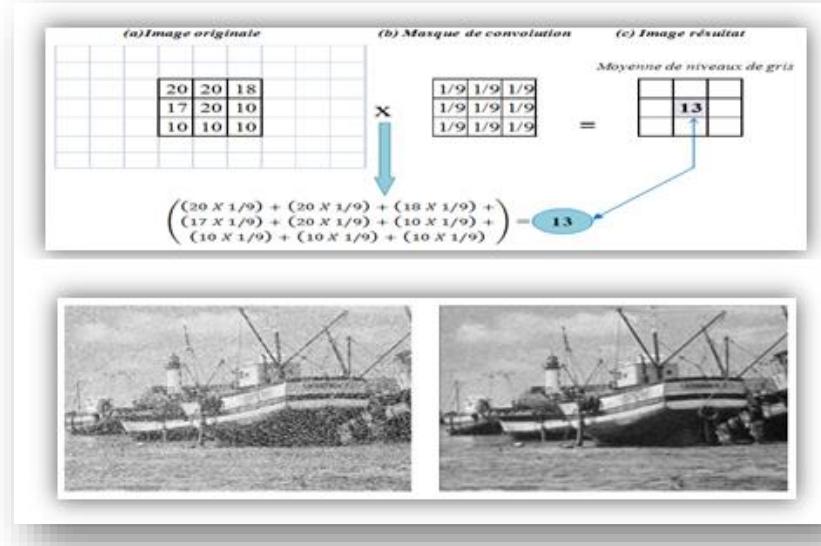


FIG.I.10– FILTRE MOYENNEUR

I.3.1.2 FILTRE GAUSSIEN

Le principe de filtre gaussien a une logique analogue au filtre moyenne, mais il utilise une fenêtre différent qui représente la forme d'une gaussienne, le calcul de masque de convolution est basés à fonction gaussien comme montre la figure II.11 suivante:

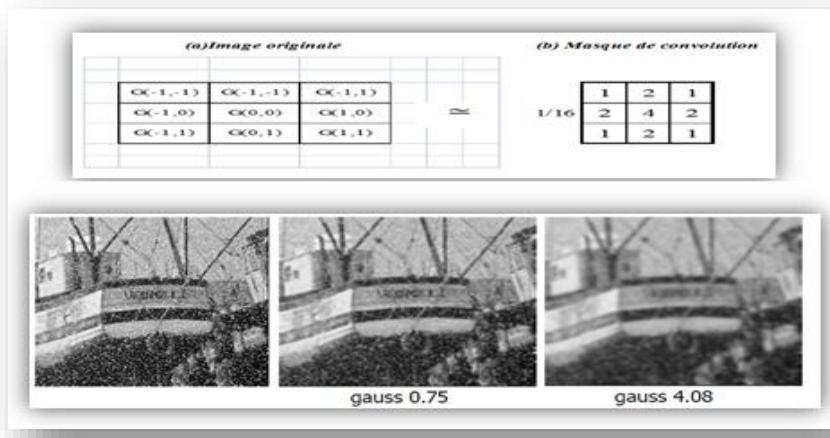


FIG.I.11– FILTRE GAUSSIEN

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

I.3.1.3 FILTRE MEDIAN

Le principe du filtre Médian est de procéder tout d'abord par un tri des valeurs de niveau de gris du voisinage suivi d'une sélection de l'élément milieu du tri comme montre la figure suivante[BON 16] :

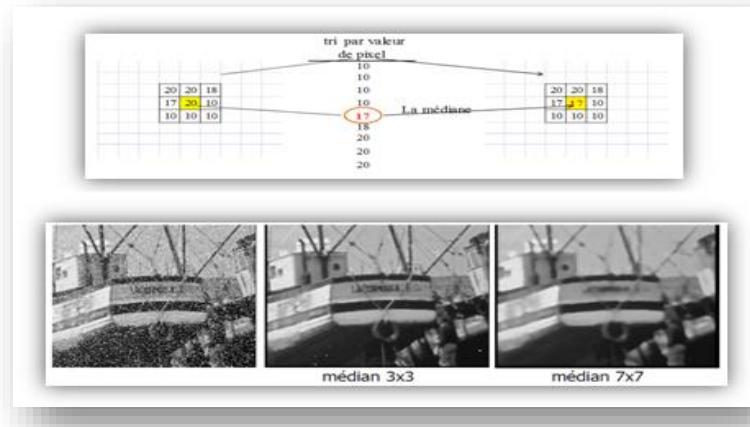


FIG.I.12– FILTRE MEDIAN

I.3.1.4 FILTRE NAGAO

Le principe: Découper d'une fenêtre 5×5 centrée sur le pixel en 9 sous-fenêtre de 7 pixels mesure sur chacune de ces fenêtres d'une valeur de l'homogénéité.

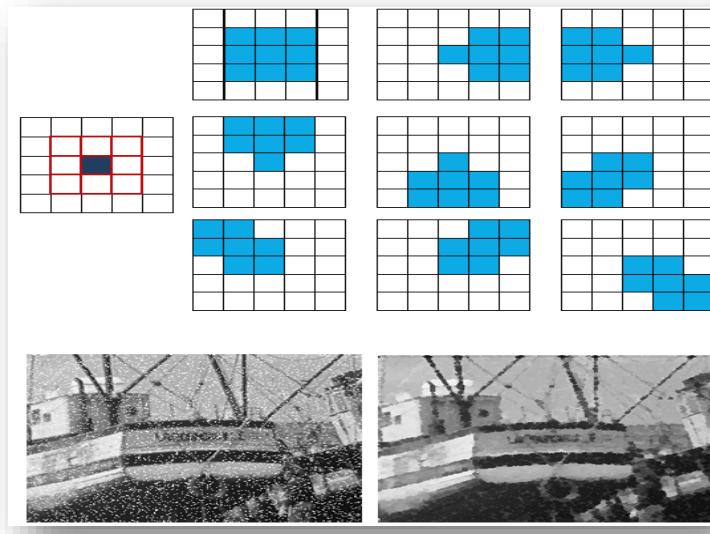


FIG.I.12– FILTRE NAGAO FENETRES 5X5

I.3.2 TECHNIQUES D'AMELIORATION PAR FILTRAGE FREQUENTIEL

La représentation fréquentielle d'une image se base sur des techniques mathématiques qui permettent de transformer une image en l'ensemble de ses fréquences spatiales. En représentation fréquentielle, la fréquence dans une image représente la variation de l'intensité des pixels de l'image, et qui se divise en deux fréquences :

- ✚ Les basses fréquences (correspondent à des changements d'intensité lents) représentent les régions homogènes et floues.
- ✚ Les hautes fréquences (correspondent à des changements d'intensité rapides) représentent les contours, les bruits « les changements brusques d'intensité ».

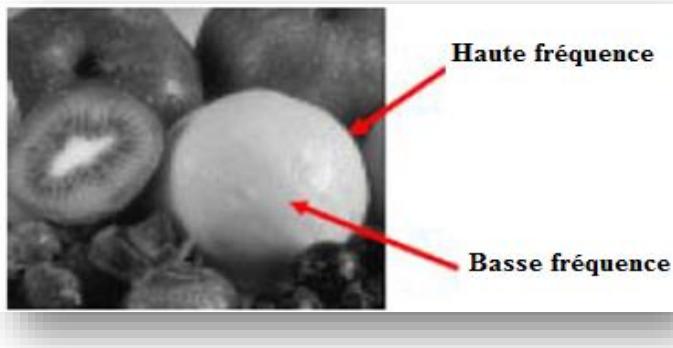


FIG.I.13– REPRESENTATION FREQUENTIELLE D'UNE IMAGE

I.3.2.1 TRANSFORMEE DE FOURIER

La transformée de Fourier (TF), ou plus généralement l'analyse fréquentielle ou spectrale, est un outil fondamental pour la compréhension et la mise en œuvre de nombreuses techniques numériques de traitement des signaux et des images. La transformée de Fourier permet de passer d'une représentation de l'image dans le domaine spatial à sa représentation dans le domaine fréquentiel [Men 11] [Tho 15].

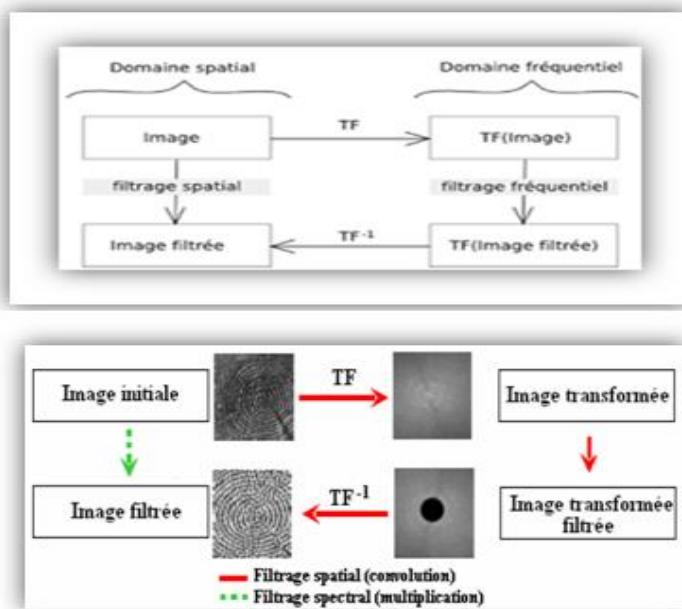


FIG.I.14– INTERPRETATION DE LA TRANSFORMEE DE FOURIER D'UNE IMAGE

I.3.2.2 FILTRE DE GABOR

Un filtre de Gabor est une fonction sinusoïdale à laquelle on a rajouté une enveloppe gaussienne. Dans le plan fréquentiel, cette fonction se transforme en gaussienne. La fonction sinusoïdale est caractérisée par sa fréquence et par son orientation

Ces filtres constituent une classe particulière des filtres linéaires; ce sont des filtres orientés et ils ont une réponse impulsionnelle de la forme:

$$h(x, y) = g(x', y') e^{2\pi j(Ux+Vy)} \quad (\text{II.7})$$

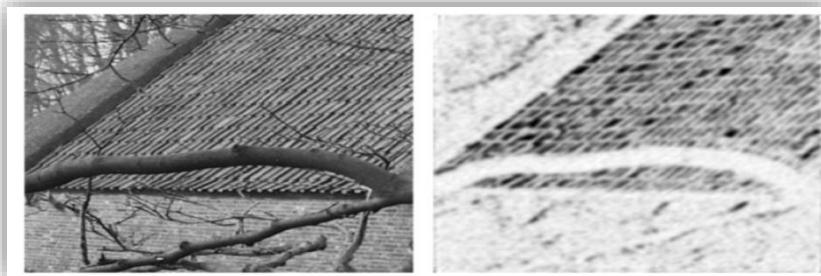


FIG.I.15– IMAGE FILTREE AVEC LE FILTRE DE GABOR (135°)

I.3.3 LIMITES DES TECHNIQUES CLASSIQUES

En général ces techniques des limites importantes :

- Plus le filtre grossit plus le lissage devient important et plus le flou s'accentue
- Détruit les coins et mélange de structures voisines.
- Supprime les détails fins qui ne sont pas du bruit.
- Généralement certains filtres sont dédiés pour l'extraction des caractéristiques.

Malgré leur simplicité et leur rapidité, les méthodes classiques de débruitage reposent sur des hypothèses statiques et des filtrages locaux qui atteignent vite leurs limites face à la diversité des images et des types de bruit. Elles traitent chaque pixel selon des règles fixes, sans réellement comprendre le contenu de l'image.

Donc ces techniques sont insuffisantes pour des images complexes et très bruitées, seulement, avec l'arrivée du deep learning, une nouvelle génération de méthodes s'est imposée, capable d'apprendre automatiquement à retirer le bruit à partir de milliers d'exemples d'images bruitées et propres.

I.4 TECHNIQUES D'AMELIORATION PAR LES CNN

Les réseaux de neurones convolutifs (CNN) sont largement utilisés dans le traitement d'images, notamment pour la suppression du bruit. Leur efficacité repose sur leur aptitude à détecter et exploiter des motifs locaux. En apprenant des filtres adaptatifs, ces réseaux parviennent à isoler et éliminer le bruit présent dans les images d'entrée.

En s'entraînant sur un grand nombre d'images originales et bruitées, les CNN apprennent à construire des filtres appelés filtres convolutifs. Ces filtres sont ajustés automatiquement pendant l'entraînement pour reconnaître les motifs caractéristiques du bruit.

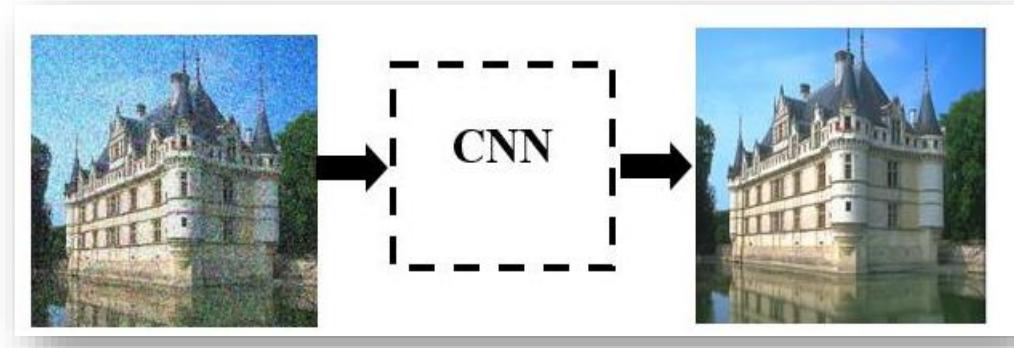


FIG.I.16—AMELIORATION D'IMAGE PAR CNN

I.4.1 LIMITES DES CNN

Les Convolutional Neural Networks (CNN) ont marqué un tournant majeur: ils ne se contentent plus d'appliquer des règles fixes mais apprennent à filtrer de manière adaptative, en tenant compte du contexte visuel local. Toutefois, malgré leur puissance les CNN restent limités à des voisinages fixes (fenêtres locales), aussi ils peinent à exploiter des relations globales ou structurelles entre régions d'image.

I.5 CONCLUSION

Le CNN est excellent pour exploiter le local, mais aveugle au global. Pour surmonter cette rigidité spatiale, on se tourne vers les Graph Neural Networks (GNN), capables de modéliser des relations personnalisées entre pixels ou régions, au-delà de la grille d'image "matrice". Afin de bien comprendre le principe fonctionnement des GNN, le chapitre suivant sera dédié pour les GNN, leurs origines, architectures et leurs applications.

Référence

- [Bar 12] : A.Baroudi, N. Larabi , "Comparaison entre les différents filtres d'images" Mémoire de fin d'études , Licence en Informatique Abou BakrBelkaid Tlemcen 2012 .
- [BER 11] : M.O. Berger, Modèles mathématiques pour le traitement et l'analyse d'images INRIA Nancy Grand Est, Colloque Les mathématiques dans la société, janvier 2011.
- [Blo 08] : I. Bloch et F. Tupin, "Traitement et reconnaissance d'images", Cours Master 2 IAD, 2008. [MAS 19] : F. Cabestaing, Traitement d'images : transformations ponctuelles, 2019. https://www.google.com/url?sa=t&source=web&rct=j&url=http://master-ivi.univ-lille1.fr/fichiers/Cours/ti-cours-transformations-ponctuelles_20/3/2019.
- [Bon16] : P. Bonnet " Traitement d'Image : Cours de" USTL , 2016.
- [Chi 11] : M-TChikh, " Amélioration des images par un modèle de réseau de neurones (Comparaison avec les filtres de base)", Master en Informatique Université Abou BakrBelkaid- Tlemcen , 2011.
- [LAM 17] : D. Lamraoui et H. Slimani, Filtrage Des Images Par Differentes Approches Master en Génie Electrique Universite M'hamed Bougara-Boumerdes, Juin 2017.
- [Nao 10] : M.Naouai " Filtrage d'image : cour7 de" , Université de tunis el-manarFaculte de sciences 2010 .
- [Pil 15] : Jean- PILLOU " Traitement d'images " Septembre 2015 Francois
- [Sun 12]: He, K., Sun, J., & Tang, X. (2012). Guided image filtering.IEEE transactions on pattern analysis and machine intelligence, 35(6), 1397-1409.

CHAPITRE I: LES TECHNIQUES D'AMELIORATION D'UNE IMAGE BRUITEE

- [Tho 15] : N. Thome, Bases du traitement des images "Filtrage d'images : filtrage linéaire et filtrage non linéaire", support de cours 2015.
- [Tho 18] : N. Thome, S.Dubuisson, D. Béréziat, Support De Cours : Bases du traitement des images - Filtrage d'images, octobre 2018.
- [Tup 16] : F. Tupin , " Introduction au traitement d'images : Cours OASIS " , 2016.
- [Vil 95] : J.L. Vila. Filtrage d'ordre directionnel adaptatif: application aux autoradiographies de séquences d'A. D. N.. Traitement du signal et de l'image. Université de Savoie, 1995. Français.
- [VIN 19] : N.Vincent, Traitement des Images Numériques, Support De Cours, Paris des cartes LIPADE, 2019.
- [Xav 07] : Xavier Philippeau, Cours : Les Filtres Usuels En Traitement D'images, 2007
[:https://xphilipp.developpez.com/articles/filtres/?page=page_3](https://xphilipp.developpez.com/articles/filtres/?page=page_3)
- [Xav 07] : Xavier Philippeau, Cours : Les Filtres Usuels En Traitement D'images, 2007
[:https://xphilipp.developpez.com/articles/filtres/?page=page_3](https://xphilipp.developpez.com/articles/filtres/?page=page_3)
- [Zha 14]: Zhang, Q., Shen, X., Xu, L., &Jia, J. (2014).Rolling guidance filter. In Computer Vision-ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part III 13 (pp. 815-830). Springer International Publishing.

CHAPITRE II:

FONDEMENTS DES

GRAPHES NEURAL

NETWORKS

CHAPITRE II

FONDEMENTS DES GRAPH NEURAL NETWORKS

SOMMAIRE

II.1 INTRODUCTION	21
II.2 INTRODUCTION AUX GRAPHES	21
II.2.1 QU'EST CE QU'UN GRAPHE ?	21
II.2.1.1 CLASSIFICATION DES SOMMETS.....	22
II.2.1.2 RELATIONS ENTRE SOMMETS.....	23
II.2.2 CATEGORIES DE GRAPHES	23
II.3 REPRESENTATION DES GRAPHES	24
II.3.1 MATRICE D'ADJACENCE	25
II.3.2 LISTE D'ADJACENCE	25
II.3.3 MATRICE D'INCIDENCE	25
II.4 DE LA CONVOLUTION CLASSIQUE (CNN) AU GRAPHIQUE (GNN)	26
II.4.1 Qu'est-ce qu'un CNN ?	26
II.4.2 ARCHITECTURE TYPIQUE D'UN CNN.....	26
II.4.3 DOMAINE D'APPLICATIONS	27
II.4.4 DE LA CONVOLUTION EUCLIDIENNE AU GRAPHES	28
II.4.5 FONDEMENTS MATHEMATIQUES : DU CNN AUX GNNS	29
II.5 NOTIONS DE BASE DES GRAPH NEURAL NETWORKS	30
II.5.1 Historique	30
II.5.2 PRINCIPE DE FONCTIONNEMENT DES GNNS.....	33
II.5.3 CYCLE DE PROPAGATION	34
II.5.4 PRINCIPAUX TYPES DE GNN	34
II. 6 DEBRUITAGE AVEC GNN	35
II. 8 DOMAINE D'APPLICATIONS	37
I.8 CONCLUSION	38

II.1 INTRODUCTION

Les graphes sont une représentation puissante et générale des données, permettant de modéliser des structures variées comme les images, le texte, etc. En effet, ils sont des structures fondamentales permettant de représenter des objets et leurs relations.

Si les graphes sont intuitifs pour les humains (via des schémas ou visualisations), les ordinateurs, eux, comprennent et traitent des données numériques. Les matrices, structures mathématiques bien établies, offrent un cadre rigoureux et efficace pour représenter les connexions entre les noeuds (adjacence), capturer les propriétés locales (degrés), formuler des opérations algébriques sur les graphes (produits, inverses, spectres), utiliser des modèles d'apprentissage automatique, notamment les réseaux de neurones sur graphes « Graph Neural Networks (GNN) », etc.

II.2 INTRODUCTION AUX GRAPHES

II.2.1 QU'EST CE QU'UN GRAPHE ?

Un graphe G est défini par un ensemble de sommets fini $V=\{v_1, v_2, \dots, v_n\}$ et un ensemble d'arêtes fini $E=\{e_1, e_2, \dots, e_m\}$, noté $\mathbf{G}=(V, E)$. Et donc par définition [Gys 90][Dor 92]:

- Sommet (nœud) : Élément fondamental du graphe représentant un objet, une entité ou un point.
- Arête : Lien non orienté connectant deux sommets.
- Arc : Lien orienté reliant un sommet à un autre, indiquant une direction spécifique.
- Degré d'un sommet: Nombre total d'arêtes connectées à un sommet. Autrement dit, le nombre d'arêtes incidentes à ce sommet.
- Degré entrant (dans un graphe orienté) : Nombre d'arcs arrivant "entrant" vers le sommet.

- Degré sortant (dans un graphe orienté) : Nombre d'arcs partant "sortant" du sommet.
- Degré total est la somme du degré entrant et du degré sortant.

Dans un graphe non orienté, la somme des degrés de tous les sommets est égale à deux fois le nombre d'arêtes :

$$\sum_{v \in V} \deg(v) = 2|E|$$

Dans un graphe orienté, la somme des degrés entrants est égale à la somme des degrés sortants :

$$\sum_{v \in V} \deg \text{in}(v) = \sum_{v \in V} \deg \text{out}(v) = |E|$$

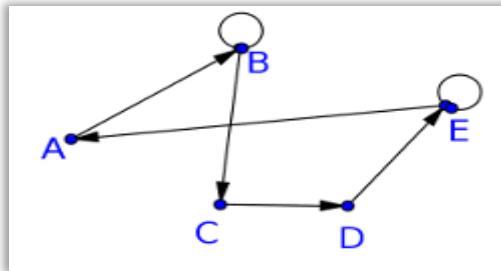


FIG.II.1– GRAPHE AVEC TROIS SOMMETS ET QUATRE ARCS

II.2.1.1 CLASSIFICATION DES SOMMETS

Les sommets peuvent être classés selon leur rôle et leur position dans le graphe [Dor 92][Ang 08]:

- Sommet isolé : Nœud sans aucune connexion (degré nul).
- Sommets adjacents : Deux sommets directement reliés par une arête.
- Sommet de degré pair/impair : Selon le nombre d'arêtes incidentes (pair ou impair).
- Sommet terminal (feuille) : Sommet ayant exactement une arête incidente.

II.2.1.2 RELATIONS ENTRE SOMMETS

Plusieurs notions décrivent la relation entre les sommets d'un graphe :

- Voisinage : Ensemble des sommets directement connectés à un sommet donné.
 - Voisin sortant : sommet u tel que $(v, u) \in E$
 - Voisin entant : sommet u tel que $(u, v) \in E$
- Distance : Nombre minimal d'arêtes séparant deux sommets.
- Boucle est une arête ou un arc qui relie un sommet à lui-même.
- Chemin : le chemin est une suite finie de sommets (v_1, v_2, \dots, v_k) tels que chaque paire consécutive de sommets (v_i, v_{i+1}) est reliée par une arête (dans le cas d'un graphe non orienté) ou par un arc orienté dans la direction $v_i \rightarrow v_{i+1}$ (dans le cas d'un graphe orienté).
- Cycle : Chemin qui commence et se termine au même sommet.

II.2.2 CATEGORIES DE GRAPHES

Les graphes peuvent être caractérisés selon la manière dont les sommets sont connectés[Cau 96][Ang 08][Har 62] :

- GRAPHE SIMPLE : Un graphe est qualifié de simple s'il est non orienté, sans boucles (arête d'un sommet vers lui-même) et sans arêtes multiples.
- GRAPHE ORIENTE (DIGRAPH) : Un graphe orienté est un graphe dont les arêtes ont une direction et chaque arête est une paire ordonnée (u, v) , signifiant $u \rightarrow v$.
- GRAPHE NON ORIENTE : Un graphe non orienté est un graphe dont les arcs n'ont pas de direction et chaque arc est une paire $\{u, v\}$.
- GRAPHE COMPLET : Un graphe complet est un graphe dans lequel chaque sommet est relié à tous les autres par une unique arête, sans

aucune boucle. Chaque sommet est alors de degré $n-1$, où n est le nombre total de sommets. Le nombre total d'arêtes dans un graphe complet est donné par la formule suivante :

$$\text{Nbre} = n(n-1)/2$$

- GRAPHE CONNEXE : Un graphe est dit connexe si, depuis n'importe quel sommet, il est possible d'atteindre tous les autres en empruntant les arêtes. Autrement dit, il existe un chemin entre chaque paire de sommets.
- GRAPHE REGULIER : est un graphe dans lequel tous les sommets ont le même degré, noté k . On parle alors de graphe régulier .
- GRAPHE BIPARTI : Les sommets sont divisés en deux groupes, chaque arête connectant uniquement des sommets de groupes différents.
- GRAPHE DYNAMIQUE : un graphe dynamique (ou temporel) est un graphe dont les éléments changent au cours du temps :
 - Apparition/disparition de nœuds ou d'arêtes
 - Évolution des attributs des entités
 - Représenté soit comme une séquence discrète de graphes soit comme un flux continu d'événements
- GRAPHE VALUE : un graphe valué est un graphe (orienté ou non orienté) où chaque arête est associée à une valeur, c'est à dire on associe au graphe une application $\mathcal{V}: V \rightarrow R$. Cette application v est appelée valuation du graphe.

II.3 REPRESENTATION DES GRAPHS

Les graphes sont des structures fondamentales permettant de représenter des objets et leurs relations. Leur simplicité conceptuelle cache une puissance expressive immense. Donc pour pouvoir analyser, et

manipuler à partir des graphes il est crucial de disposer d'une représentation mathématique exploitable par des machines : les matrices.

Représenter un graphe de manière efficace est essentiel pour le traitement algorithmique. Voici les principales structures utilisées :

II.3.1 MATRICE D'ADJACENCE

Une matrice d'adjacence A est une structure de représentation des graphes sous forme de tableau 2D « carré » où chaque cellule de la matrice à l'indice (i,j) indique l'existence d'une arête entre le sommet i et le sommet j[Sin 12].

- Si le graphe non orienté : la matrice est symétrique.

$$A[i][j] = \begin{cases} 1 & \text{si une arête relie le sommet } i \text{ et } j. \\ 0 & \text{sinon} \end{cases}$$

- Si le graphe orienté : la matrice peut être asymétrique.

$$A[i][j] = \begin{cases} 1 & \text{si une arête va de } i \text{ vers } j. \\ 0 & \text{sinon} \end{cases}$$

II.3.2 LISTE D'ADJACENCE

Une liste d'adjacence est un tableau de listes, où chaque élément $A[i]$ contient les sommets adjacents au sommet i. C'est-à-dire chaque sommet est associé à une liste de ses voisins (sommets directement connectés).

- Dans un graphe non orienté, si le sommet j est dans la liste de i, alors i sera dans la liste de j.
- Pour les graphes orientés, on utilise soit une liste de successeurs, soit une liste de prédécesseurs selon les besoins.

II.3.3 MATRICE D'INCIDENCE

La matrice d'incidence est une matrice $n \times p$, où n est le nombre de sommets du graphe et p est le nombre de liens (arêtes ou arcs).

Cette matrice est définie de deux façons différentes selon que le graphe est orienté ou non orienté[Van 76][Hes 23].

- Pour un graphe non orienté :

$$A[i][j] = \begin{cases} 1 & \text{si le sommet } i \text{ est incident à l'arête } j. \\ 0 & \text{sinon} \end{cases}$$

- Pour un graphe orienté (plus courant en théorie des graphes) :

$$A[i][j] = \begin{cases} -1 & \text{si le sommet } i \text{ est l'origine l'arête } j. \\ +1 & \text{si le sommet } i \text{ est l'extrimité l'arête } j \\ 0 & \text{sinon} \end{cases}$$

II.4 DE LA CONVOLUTION CLASSIQUE (CNN) AU GRAPHIQUE (GNN)

Les graphes sont une excellente façon de représenter des relations complexes entre éléments, mais ils posent un défi majeur : comment exploiter efficacement ces structures pour des tâches d'apprentissage automatique ? Contrairement aux images ou aux séquences, les graphes n'ont pas une forme régulière ni un ordre fixe, ce qui rend l'utilisation des réseaux neuronaux traditionnels difficile. C'est là qu'interviennent les Graph Neural Networks (GNN), une classe innovante de réseaux de neurones spécialement conçue pour apprendre directement sur les données structurées en graphes[Zha 22][Mcd 22].

II.4.1 Qu'est-ce qu'un CNN ?

Un CNN (Convolutional Neural Network) est un type de réseau de neurones artificiels conçu pour traiter des données structurées en grille, comme les images. Inspirés du fonctionnement du cortex visuel humain, les CNN sont capables d'apprendre à extraire automatiquement des caractéristiques locales (par exemple, bords, textures, formes) à partir des images. Cela les rend particulièrement efficaces pour des tâches telles que la classification d'images, la détection d'objets ou la reconnaissance faciale.

II.4.2 ARCHITECTURE TYPIQUE D'UN CNN

Un CNN est composé de plusieurs types de couches, chacune jouant un rôle spécifique :

CHAPITRE II: FONDEMENTS DES GRAPH NEURAL NETWORKS

- Couche de convolution : Applique des filtres (ou noyaux) à l'image d'entrée pour détecter des motifs locaux. Chaque filtre génère une carte d'activation (feature map) mettant en évidence la présence de certaines caractéristiques.
- Fonction d'activation (ReLU) : Introduit de la non-linéarité dans le modèle, permettant au réseau d'apprendre des relations complexes entre les données.
- Couche de pooling : Réduit la dimensionnalité des cartes d'activation, conservant les informations essentielles tout en diminuant le nombre de paramètres et la charge computationnelle.
- Couches entièrement connectées (fully connected) : Situées en fin de réseau, elles interprètent les caractéristiques extraites pour effectuer la classification ou d'autres tâches spécifiques.

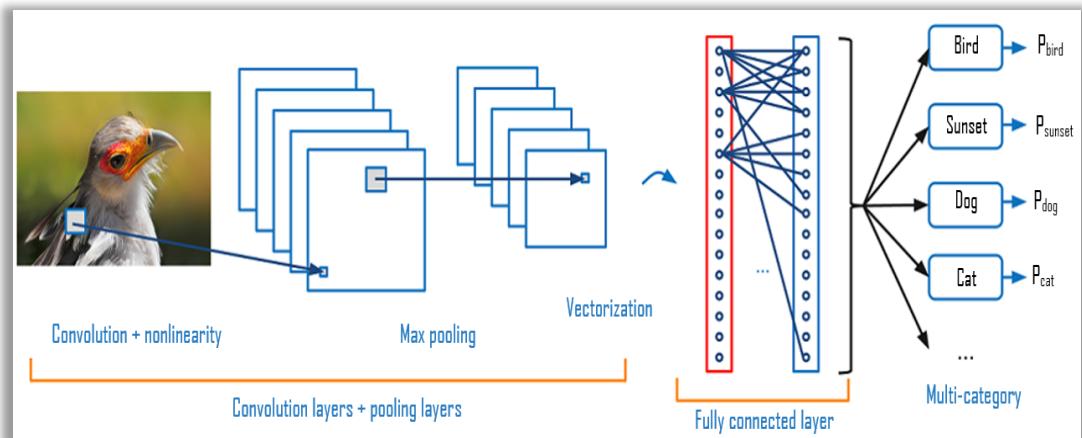


FIG.II.2 – EXEMPLE D’UNE ARCHITECTURE TYPIQUE D’UN CNN

II.4.3 DOMAINE D’APPLICATIONS

Les CNN sont largement utilisés dans divers domaines comme par exemple:

- VISION PAR ORDINATEUR : Classification d'images, détection d'objets, segmentation sémantique.

- RECONNAISSANCE VOCALE : Transcription automatique de la parole en texte.
- IMAGERIE MEDICALE : Détection de tumeurs, analyse d'images radiologiques.
- TRAITEMENT DU LANGAGE NATUREL : Analyse de sentiments, classification de textes.

II.4.4 DE LA CONVOLUTION EUCLIDIENNE AU GRAPHES

Les réseaux de neurones convolutifs (CNN) ont révolutionné le traitement d'images, grâce à leur capacité à extraire automatiquement des caractéristiques locales à partir de données structurées en grille, comme les pixels d'une image. Seulement, les graphes offrent une manière puissante et flexible de représenter des données complexes composées d'éléments en interaction, comme les pixels d'une image, les régions d'un objet ou les connexions entre structures visuelles.

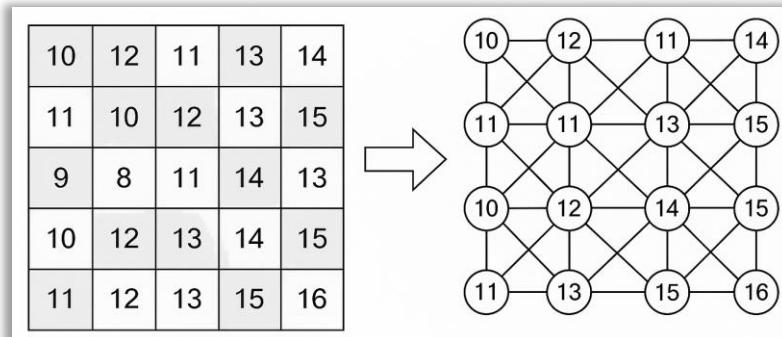


FIG.II.3 – DE LA MATRICE D’UNE IMAGE VERS UN GRAPHE

C'est ici que les Graph Neural Networks (GNN) entrent en jeu : ils généralisent l'idée de convolution aux graphes. Plutôt que d'agréger des informations à partir de pixels voisins sur une grille régulière, un GNN apprend à agréger et transformer les informations des voisins connectés dans un graphe, capturant ainsi les dépendances structurelles entre les entités.

Cette transition du CNN au GNN ouvre la porte à une nouvelle classe de modèles capables de traiter des données irrégulières, tout en conservant les avantages de la convolution à savoir, l'apprentissage de représentations locales partagées et hiérarchiques.

II.4.5 FONDEMENTS MATHEMATIQUES : DU CNN AUX GNNS

Les Convolutional Neural Networks (CNN) sont conçus pour les données euclidiennes, où la structure des voisins est régulière (par exemple, les pixels d'une image organisés en grille). Dans ce cadre, la convolution discrète peut être exprimée comme :

$$y(i, j) = \sum_{m=-k}^k \sum_{n=-k}^k W(m, n) \cdot x(i + m, j + n)$$

Ou :

- x : Image d'entrée,
- y : Noyau convolutif,
- y(i,j) : Sortie au pixel (i,j)
- k : Taille du voisinage local.

Cependant, cette opération repose sur la structure régulière de la grille. Pour d'autres données comme les graphes, où chaque nœud peut avoir un nombre variable et non ordonné de voisins, cette approche ne s'applique pas directement. Les Graph Neural Networks (GNN) généralisent cette opération comme suit :

- Un ensemble de nœuds V
- Un ensemble d'arêtes E
- Une matrice d'adjacence $A \in \{0,1\}^{N \times N}$,
- X des caractéristiques de nœuds $X \in \mathbb{R}^{N \times F}$ $N = |V|$ et F est le nombre de caractéristiques par nœud.

$$\mathbf{H}^{(l+1)} = \sigma(\tilde{\mathbf{D}}^{-\frac{1}{2}} \tilde{\mathbf{A}} \tilde{\mathbf{D}}^{-\frac{1}{2}} \mathbf{H}^{(l)} \mathbf{W}^{(l)})$$

- $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ (matrice d'adjacence avec self-loops)
- $\tilde{\mathbf{D}}$ Matrice diagonale des degrés de $\tilde{\mathbf{A}}$
- $\mathbf{H}^{(l)}$ Représentation des nœuds à la couche l (avec $\mathbf{H}^{(0)} = \mathbf{X}$)
- $\mathbf{W}^{(l)}$ Matrice de poids
- σ Fonction d'activation

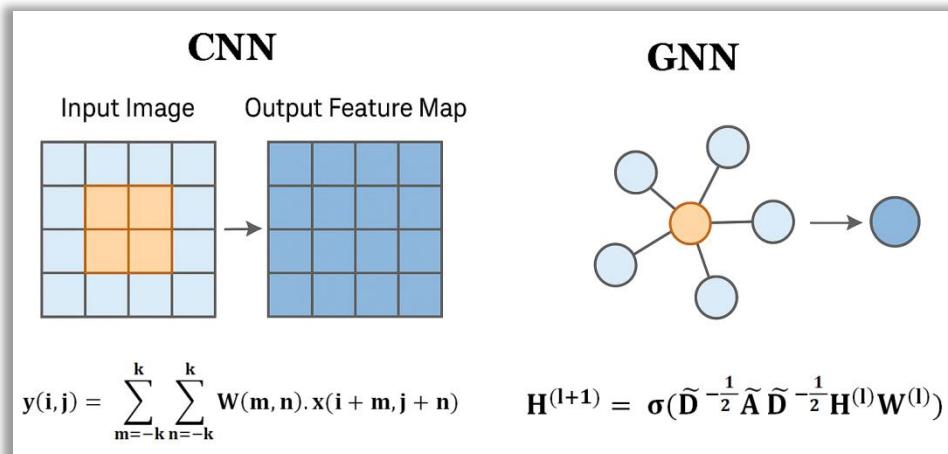


FIG.II.4 – DU CNN AU GNN

II.5 NOTIONS DE BASE DES GRAPH NEURAL NETWORKS

II.5.1 Historique

L'idée d'appliquer des réseaux de neurones sur les graphes est née de la nécessité de traiter des données complexes telles que les réseaux sociaux, les connaissances relationnelles, etc.

En 2005, Scarselli et al.[Sca 05] proposent le tout premier Graph Neural Network formel. Leur approche repose sur un processus itératif de mise à jour des représentations de nœuds jusqu'à convergence, à l'aide d'un modèle récursif. Cette méthode ouvre la voie, mais reste difficile à entraîner à grande échelle.

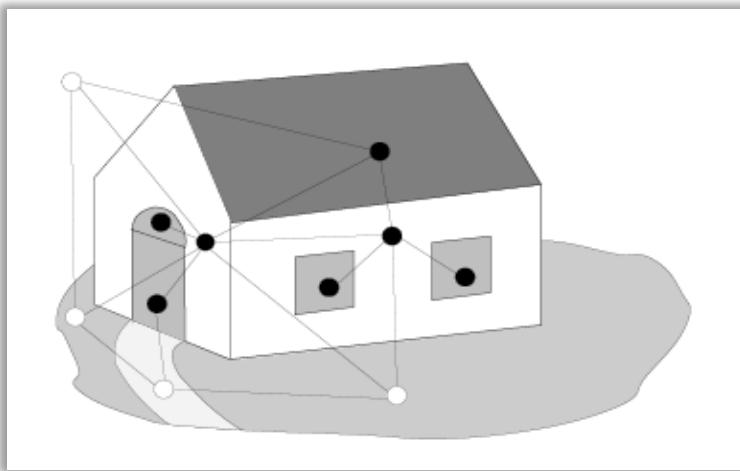


FIG.II.5– IMAGE ET SA REPRESENTATION GRAPHIQUE.[Sca 05]

À mesure que le deep learning progresse, les chercheurs redécouvrent l'intérêt des graphes. En 2009, ils introduisent le Graph Neural Network (GNN) original, fondé sur une approche de propagation de messages récurrente où les représentations des nœuds sont mises à jour de manière itérative jusqu'à convergence[Sca 09].

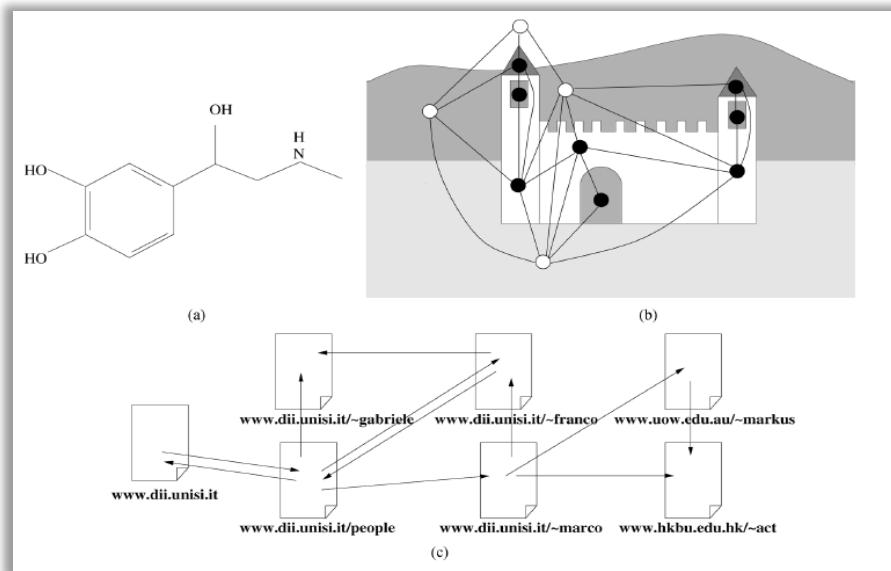


FIG.II.6– QUELQUES INFORMATIONS REPRESENTEES PAR GRAPHE

Avec l'essor du deep learning, de nouvelles formulations apparaissent, notamment par le biais de la convolution spectrale sur graphes. Ces approches utilisent la décomposition spectrale du Laplacien du graphe, introduite dans les travaux de Bruna et al. [Bru 13].

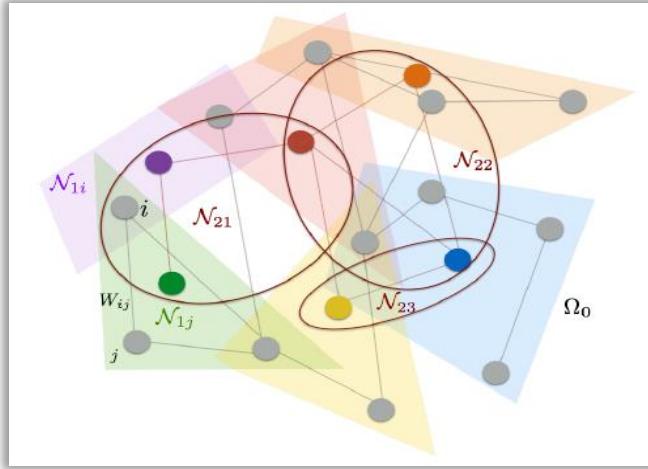


FIG.II.7 – GRAPHE CORRESPONDANT AUX MULTI-RESOLUTION

Les travaux de Defferrard et al. [Def 16] améliorent cette approche avec l'introduction des chebyshev polynomials pour un calcul plus local et plus efficace.

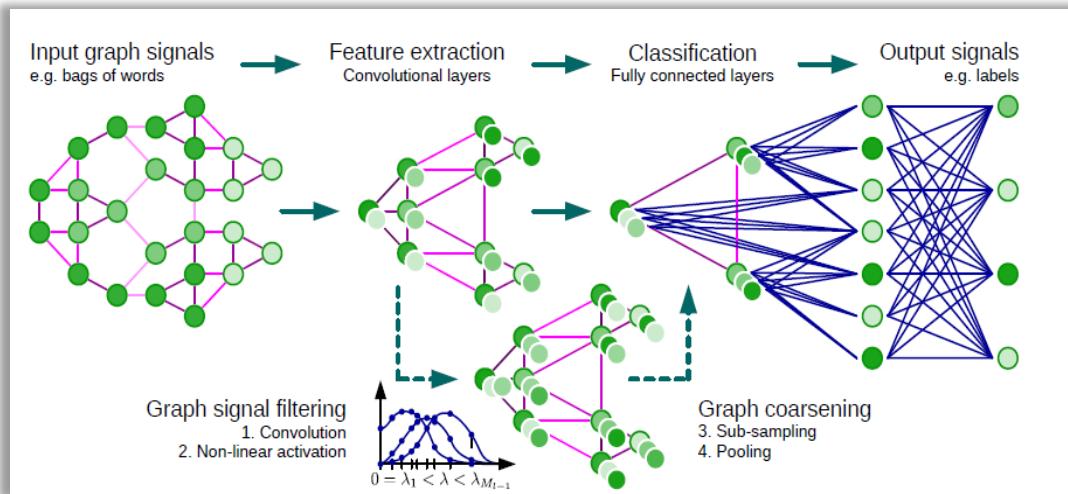


FIG.II.8 – REPRESENTATION D’UN CNN PAR GRAPHE

Le tournant décisif se produit avec Kipf & Welling [Kip 17], qui proposent le Graph Convolutional Network (GCN), une version simplifiée mais performante de la convolution spectrale, fondée sur une propagation de type message-passing avec normalisation des voisins.

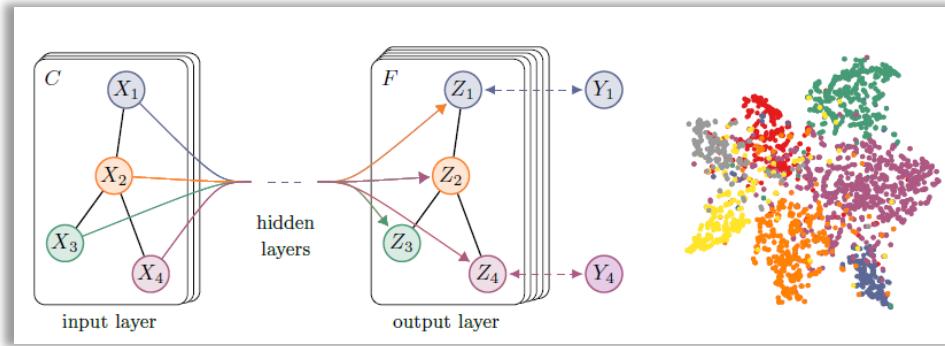


FIG.II.9– SEMI-SUPERVISED LEARNING VIA GRAPHE

Depuis, l'évolution des GNNs illustre la montée en puissance des méthodes de deep learning sur des données structurées complexes. En combinant théorie des graphes, traitement du signal et réseaux de neurones, les GNNs offrent un cadre généraliste, flexible et performant pour modéliser des systèmes relationnels dans presque tous les domaines scientifiques.

II.5.2 PRINCIPE DE FONCTIONNEMENT DES GNNs

Les Graph Neural Networks reposent sur un principe fondamental appelé propagation de messages. L'idée est simple : chaque nœud du graphe échange des informations avec ses voisins pour enrichir sa propre représentation. Concrètement, lors d'une étape d'apprentissage [Han 22] [Zho 24] :

- 1.** Chaque nœud collecte ou récupère les informations (attributs) de ses voisins, par exemple les couleurs, intensités, ou autres informations.
- 2.** Il combine ces informations avec sa propre donnée à l'aide d'une fonction mathématique (via une fonction d'agrégation).

3. Il met à jour sa propre représentation (embedding). Cette mise à jour est répétée plusieurs fois, ce qui permet à chaque nœud d'avoir une vue de plus en plus globale et riche de son environnement dans le graphe.

Autrement dit, chaque nœud apprend en regardant ce que font ses voisins. Grâce à ce mécanisme, le GNN est capable de capturer la structure locale et globale du graphe, tout en s'adaptant à des données de formes très diverses (images, textes, etc.).

C'est ce qui rend les GNN particulièrement puissants pour traiter les données en graphes, où les relations entre éléments sont aussi importantes que les éléments eux-mêmes.

II.5.3 CYCLE DE PROPAGATION

La propagation se déroule en plusieurs couches (étapes). Chaque couche comprend trois étapes clés [Wu 23][Wan 24]:

- i) Message
- ii) Agrégation
- iii) Mise à jour

II.5.4 PRINCIPAUX TYPES DE GNN

Il existe plusieurs types de Graph Neural Networks. Voici quelques variantes les plus populaires :

- ➡ GCN (GRAPH CONVOLUTIONAL NETWORK) : Étendre la notion de convolution aux graphes en exploitant la structure locale d'un nœud (ses voisins). Cette opération est similaire à une convolution sur le graphe. Chaque nœud met à jour sa représentation en agrégeant celles de ses voisins, pondérées et normalisées.

- **GRAPH SAGE** : Effectue une agrégation échantillonnée des voisins, permettant un apprentissage inductif. À chaque couche, un nœud met à jour sa représentation en combinant celles d'un sous-ensemble de ses voisins via des fonctions comme mean, max-pooling .
- **GAT (GRAPH ATTENTION NETWORK)** : Introduit un mécanisme d'attention pour pondérer dynamiquement l'importance des voisins dans l'agrégation. Capte des relations hétérogènes sans nécessiter de normalisation par degré.
- **GIN (GRAPH ISOMORPHISM NETWORK)** : Utilise un MLP pour transformer l'agrégation des représentations voisines..

II. 6 DEBRUITAGE AVEC GNN

Le débruitage avec les Graph Neural Networks (GNNs) constitue une approche avancée et prometteuse pour la restauration de données perturbées par du bruit, en particulier dans des domaines tels que la vision par ordinateur et le traitement du signal. Les GNNs offrent une méthode novatrice pour exploiter les relations complexes entre les éléments de données en utilisant une représentation sous forme de graphes, où chaque élément est relié à d'autres par des arêtes.

Voici quelques stratégies spécifiques de débruitage utilisant des GNNs :

- **GRAPHES DE PIXELS POUR LES IMAGES** : On construit un graphe où chaque pixel est un nœud et les arêtes sont établies entre les pixels voisins. Le GNN peut ainsi apprendre à restaurer l'image en exploitant les relations spatiales locales entre les pixels, tout en éliminant le bruit.

- **GRAPHES BASES SUR DES CARACTERISTIQUES DE PIXELS :** Plutôt que d'utiliser simplement la proximité spatiale des pixels, on peut définir des graphes où chaque nœud représente un vecteur de caractéristiques plus complexe (par exemple, les couleurs ou les textures). Cela permet d'utiliser des informations plus riches pour mieux discriminer le signal utile du bruit.
- **GNNs POUR LE DEBRUITAGE DE SIGNAUX NON-IMAGE :** Les GNNs peuvent également être appliqués à des signaux ou des séries temporelles, où les nœuds représentent différentes parties du signal et les arêtes reflètent des dépendances temporelles. Cela permet de restaurer le signal tout en supprimant les interférences indésirables.

II. 7 AVANTAGES DES GNNs DANS LE DEBRUITAGE

- **MODELISATION DES RELATIONS COMPLEXES :** Les GNNs sont particulièrement efficaces pour apprendre des dépendances complexes entre différentes parties des données, qu'il s'agisse d'images ou de signaux. Elles permettent ainsi une restauration plus fine et précise.
- **FLEXIBILITE ET ADAPTABILITE :** Les GNNs s'adaptent à différents types de données, qu'elles soient structurées sous forme d'images, de signaux temporels, ou d'autres types de données.
- **PROPAGATION EFFICACE DE L'INFORMATION :** La structure des GNNs permet une propagation fluide de l'information entre les nœuds, ce qui permet d'atténuer le bruit tout en préservant les informations pertinentes.

II. 8 DOMAINE D'APPLICATIONS

Les GNNs pour le débruitage sont particulièrement adaptées à des domaines tels que :

- **LA VISION PAR ORDINATEUR** : Améliorer la qualité d'images ou de vidéos en réduisant le bruit tout en conservant les détails fins.
- **LE TRAITEMENT AUDIO** : Dans des applications comme la reconnaissance vocale, le nettoyage de signaux audio ou la musique.
- **LES SERIES TEMPORELLES** : Pour nettoyer les données bruitées issues de capteurs, de mesures financières ou d'autres applications qui nécessitent l'analyse de signaux temporels.

I.8 CONCLUSION

Le débruitage via Graph Neural Networks représente une avancée significative par rapport aux méthodes classiques. Les GNNs permettent de capturer et d'exploiter des relations complexes entre les éléments des données, ce qui leur confère un grand potentiel pour séparer le bruit du signal tout en préservant l'intégrité des informations. Cette approche promet d'être un atout majeur dans divers domaines, en particulier pour les données ayant des structures non-euclidiennes.

Référence

- [Ang 08] Angles, R., & Gutierrez, C. (2008). Survey of graph database models. ACM Computing Surveys (CSUR), 40(1), 1-39.
- [Bru 14] Bruna, J., Zaremba, W., Szlam, A., & LeCun, Y. (2013). Spectral Networks and Locally Connected Networks on Graphs. In ICLR.
- [Cau 96] Caucal, D. (1996). Sur des graphes infinis réguliers (Doctoral dissertation, Habilitation thesis, Université de Rennes 1).
- [Def 16] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering. In NeurIPS.
- [Dor 92] Dor, D., & Tarsi, M. (1992). A simple algorithm to construct a consistent extension of a partially oriented graph. Technical Report R-185, Cognitive Systems Laboratory, UCLA, 45.
- [Gys 90] Gyssens, M., Paredaens, J., & Van Gucht, D. (1990, April). A graph-oriented object database model. In Proceedings of the ninth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems (pp. 417-424).
- [Han 22] Han, K., Wang, Y., Guo, J., Tang, Y., & Wu, E. (2022). Vision gnn: An image is worth graph of nodes. Advances in neural information processing systems, 35, 8291-8303.
- [Har 62] Harary, F. (1962). The determinant of the adjacency matrix of a graph. Siam Review, 4(3), 202-210.
- [Hes 23] Hessert, R., & Mallik, S. (2023). The inverse of the incidence matrix of a unicyclic graph. Linear and Multilinear Algebra, 71(4), 513-527.

CHAPITRE II: FONDEMENTS DES GRAPH NEURAL NETWORKS

- [Kip 17] Kipf, T. N., & Welling, M. (2017). Semi-Supervised Classification with Graph Convolutional Networks.
- [McD 22] McDonnell, K., Abram, F., & Howley, E. (2022). Application of a novel hybrid CNN-GNN for peptide ion encoding. *Journal of Proteome Research*, 22(2), 323-333.
- [Sca 09] Scarselli, F., Gori, M., Tsoi, A. C., Hagenbuchner, M., & Monfardini, G. (2009). The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1), 61-80
- [Sin 12] Singh, H., & Sharma, R. (2012). Role of adjacency matrix & adjacency list in graph theory. *International Journal of Computers & Technology*, 3(1), 179-183.
- [Van 76] Van Nuffelen, C. (1976). On the incidence matrix of a graph. *IEEE Transactions on Circuits and Systems*, 23(9), 572-572.
- [Wan 24] Wang, X., Guan, K., He, D., Hrovat, A., Liu, R., Zhong, Z., ... & Yu, K. (2024). Graph neural network enabled propagation graph method for channel modeling. *IEEE Transactions on Vehicular Technology*.
- [Wu 23] Wu, X., He, H., Yang, H., Tai, Y., Wang, Z., & Zhang, W. (2023). PDA-GNN: propagation-depth-aware graph neural networks for recommendation. *World Wide Web*, 26(5), 3585-3606.
- [Zha 22] Zhao, B., Guo, J., & Yang, C. (2022, April). Learning precoding policy: CNN or GNN?. In *2022 IEEE Wireless Communications and Networking Conference (WCNC)* (pp. 1027-1032). IEEE.
- [Zho 24] Zhou, Q., & Yang, S. (2024, July). Research on Graph Feature Aggregation Algorithm Based on GCN and GAT. In *2024 IEEE 6th International Conference on Power, Intelligent Computing and Systems (ICPICS)* (pp. 964-970). IEEE.

CHAPITRE III: TESTS ET RESULTATS

CHAPITRE III

TESTS ET RESULTATS

SOMMAIRE

III.1 INTRODUCTION	42
III.1.1 UTILISATION DES GNN DANS LE TRAITEMENT D'IMAGES.....	43
III.1.2 POURQUOI UTILISER UN GNN POUR DEBRUITER UNE IMAGE ?.....	43
III.3 FONDEMENTS DES GNNS A PARTIR D'UNE IMAGE.....	44
III. 4 ENVIRONNEMENT DE PROGRAMMATION " PYTHON "	46
III. 4.1 INTRODUCTION	46
III. 4.2 CARACTERISTIQUES TECHNIQUES DE PYTHON.....	46
III. 4.3 DOMAINES D'UTILISATION DE PYTHON	47
III. 4.4 LES AVANTAGES DE PYTHON	48
III.5 DESCRIPTION DU PROGRAMME DEBRUITAGE D'UNE IMAGE VIA GNN	49
III.6 TESTS ET RESULTATS	53

III.1 INTRODUCTION

Amélioration d'une image à partir d'une observation bruitée est un problème central en traitement d'images. Les approches basées sur les réseaux de neurones convolutifs (CNN) surpassent désormais les méthodes classiques en exploitant des caractéristiques plus discriminantes.

Cependant, les CNN sont limités par leur nature locale et ne peuvent pas exploiter les similarités non locales, pourtant cruciales pour la qualité de l'image. Seulement, les graphes offrent une manière puissante et flexible de représenter des données complexes composées d'éléments en interaction, comme les pixels d'une image, les régions d'un objet ou les connexions entre structures visuelles.

Plutôt que d'agrégner des informations à partir de pixels voisins sur une grille régulière, un GNN (Graph Neural Network) apprend à agrégner et transformer les informations des voisins connectés dans un graphe, capturant ainsi les dépendances structurelles entre les entités.

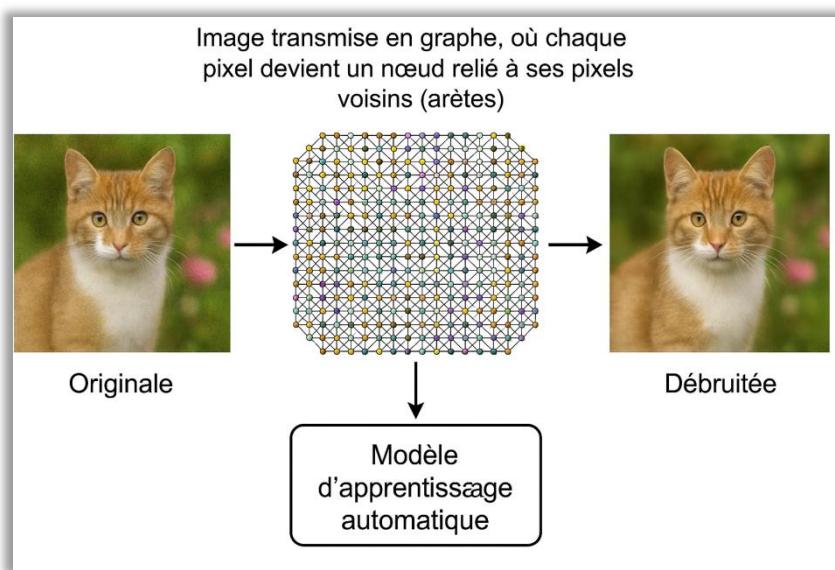


FIG.III.1– IMAGE DEBRUITEE PAR GNN

CHAPITRE III: TESTS ET RESULTATS

Un GNN est un type de réseau de neurones conçu pour travailler sur des données structurées en graphe, c'est-à-dire des données où les éléments sont liés entre eux par des relations (comme des nœuds reliés par des arêtes).

III.1.1 UTILISATION DES GNN DANS LE TRAITEMENT D'IMAGES

Les données complexes tels que les images, graphes, etc. nécessitent des méthodes adaptées pour l'extraction d'informations. Les Graph Neural Networks (GNN) sont des réseaux neuronaux spécialisés dans le traitement de données structurées en graphes, inspirés des CNN.

Les GNNs permettent de prédire des nœuds, des arêtes et des tâches basées sur des graphes, là où les CNN échouent face à des structures complexes et de taille variable.

Au lieu de voir les images comme des grilles, on peut représenter les pixels comme des nœuds connectés à leurs voisins, avec une matrice d'adjacence reflétant ces liens.

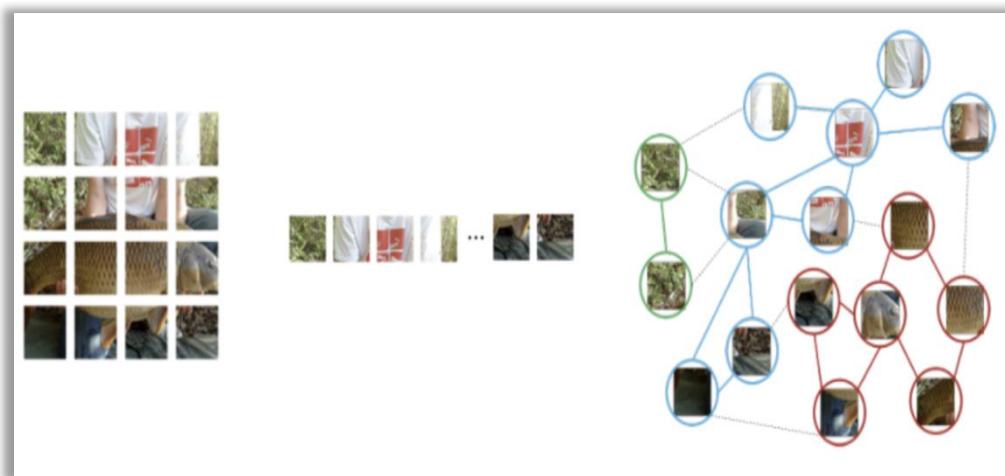


FIG.III.2 – IMAGE ET GRAPHE

III.1.2 POURQUOI UTILISER UN GNN POUR DEBRUITER UNE IMAGE ?

Contrairement aux méthodes classiques (comme le filtre médian), un GNN peut apprendre les structures visuelles complexes de l'image grâce à l'apprentissage supervisé. Il prend en compte non seulement la valeur

CHAPITRE III: TESTS ET RESULTATS

d'un pixel, mais aussi ses relations avec ses voisins ce qui donne de meilleurs résultats pour préserver les détails fins. C'est une méthode moderne et puissante, utilisée dans plusieurs domaines de vision par ordinateur et d'intelligence artificielle.

III.3 FONDEMENTS DES GNNs A PARTIR D'UNE IMAGE

Dans un graphe, Au lieu de les voir comme des grilles, on peut représenter les pixels comme des nœuds connectés à leurs voisins, avec une matrice d'adjacence reflétant ces liens. Donc, pour appliquer un GNN à une image bruitée, on doit transformer l'image en un graphe exploitable.

- NŒUDS : Chaque pixel (ou groupe de pixels) devient un nœud. Et chaque nœud est caractérisé par des attributs comme intensité de gris ou valeurs RGB, coordonnées spatiales, caractéristiques locales (gradient, texture...), etc.
- ARETES : Les arêtes peuvent relier les voisins immédiats (4 ou 8 connexités), les pixels similaires (k plus proches voisins en couleur ou texture), ou être construites dynamiquement via une heuristique de similarité.
- PONDERATION DES ARETES : Les poids peuvent refléter la similarité entre nœuds (pixels), c'est-à-dire les poids des arêtes reflètent la similarité entre pixels/régions (ex. : basée sur les niveaux de gris ou des descripteurs locaux, etc.).

$$W_{ij} = \exp\left(-\frac{\|f_i - f_j\|^2}{\sigma^2}\right)$$

Où f_i est le vecteur de caractéristiques du pixel i

Notons que le GNN apprend à estimer l'intensité (ou la couleur) débruitée de chaque pixel à partir de ses voisins dans le graphe.

CHAPITRE III: TESTS ET RESULTATS

Donc l'algorithme est structuré comme suit :

1. Prétraitement

- Ajouter du bruit (pour l'entraînement),
- Construire le graphe où chaque pixel est représenté par un nœud positionné selon sa localisation dans l'image.
- Les arêtes relient les nœuds voisins, formant une structure de maillage.
- Attribuer des poids à l'arête basée sur la similarité d'intensité entre les pixels connectés.

2. Initialisation des caractéristiques des nœuds

- Caractéristiques : Chaque nœud est initialisé avec la valeur d'intensité du pixel correspondant dans l'image bruitée Y .

3. Propagation sur le graphe (Utilisation d'un GNN)

- Chaque nœud collecte les caractéristiques de ses voisins,
- Mécanisme : À chaque couche et nœud met à jour sa représentation en agrégeant les informations de ses voisins.
- Fonction d'agrégation : Utiliser des fonctions comme la moyenne pondérée ou l'attention pour combiner les messages des voisins.
- Itérations : Répéter le processus sur plusieurs couches pour permettre une diffusion efficace de l'information à travers le graphe.

4. Prédiction :

- Chaque nœud donne en sortie la valeur estimée du pixel propre.
- Assembler les valeurs prédites des nœuds pour former l'image débruitée

5. Entraînement du modèle (fonction de Perte) :

- Comparaison avec l'image originale via MSE, ou une perte perceptuelle.

6. Évaluation et inférence

- Évaluation : Mesurer la performance du modèle sur des images de test en utilisant des métriques comme le SSIM.

CHAPITRE III: TESTS ET RESULTATS

III. 4 ENVIRONNEMENT DE PROGRAMMATION " PYTHON "

III. 4.1 INTRODUCTION

Python est un langage de programmation haut niveau, interprété et multi-paradigme, conçu pour être simple à apprendre, lisible et polyvalent. Créé par Guido van Rossum et lancé en 1991, il permet de développer des programmes dans divers domaines comme le web, l'intelligence artificielle, l'analyse de données, l'automatisation, et bien plus encore. L'une des caractéristiques distinctives de Python est sa syntaxe épurée. Contrairement à d'autres langages qui utilisent des accolades ou des mots-clés pour délimiter les blocs de code, Python repose sur l'indentation pour structurer les blocs, ce qui renforce la lisibilité et réduit les erreurs de syntaxe.

III. 4.2 CARACTERISTIQUES TECHNIQUES DE PYTHON

Parmi les caractéristiques de python nous citons les plus importantes est qui sont :

- ❖ Langage interprété : Le code est exécuté ligne par ligne par un interpréteur, sans compilation préalable.
- ❖ Syntaxe simple et lisible : Utilise l'indentation (espaces ou tabulations) au lieu des accolades {}.
- ❖ Typage dynamique : Aucune déclaration du type d'une variable nécessaire (ex: x = 5, puis x = "texte" est possible).
- ❖ Gestion automatique de la mémoire : Le garbage collector gère automatiquement l'allocation et la libération de la mémoire.
- ❖ Support de la programmation orientée objet (POO) : Permet de créer des classes, objets, méthodes, héritage, etc.
- ❖ Multi-paradigme : Supporte la programmation procédurale, fonctionnelle et orientée objet.
- ❖ Portabilité : Fonctionne sur toutes les plateformes (Windows, MacOs, Linux, Raspberry Pi...).

CHAPITRE III: TESTS ET RESULTATS

- ❖ Extensible et intégrable : Peut être combiné avec d'autres langages comme C/C++, Java ou JavaScript.
- ❖ Gros écosystème de bibliothèques standard : Livré avec une bibliothèque standard riche pour presque toutes les tâches courantes.
- ❖ 10. Gestion des erreurs intégrée : Gestion des exceptions via les mots-clés try, except, finally.

III. 4.3 DOMAINES D'UTILISATION DE PYTHON

Le domaine d'application de python est très riche nous citions par exemple :

- ✚ Développement web : Avec des frameworks comme Django, Flask, ou FastAPI, on peut créer des sites et applications web dynamiques.
- ✚ Intelligence artificielle & Machine Learning : Bibliothèques populaires : TensorFlow, PyTorch, Scikit-learn, Keras.
- ✚ Analyse et visualisation de données : Outils utilisés : Pandas (manipulation), NumPy (calcul scientifique), Matplotlib , Seaborn, Plotly (visualisation).
- ✚ Automatisation de tâches : Scripts simples pour automatiser des tâches répétitives (ex : tri de fichiers, sauvegardes...).
- ✚ Calcul scientifique et applications techniques : Utilisé dans la recherche scientifique, l'ingénierie, la physique avec des outils comme SciPy, SymPy, Astropy .
- ✚ Jeux vidéo : Développement de jeux 2D simples avec la bibliothèque Pygame .
- ✚ Applications graphiques (GUI) : Création d'interfaces graphiques avec Tkinter , PyQt , Kivy .

CHAPITRE III: TESTS ET RESULTATS

- Programmation réseau : Gestion de connexions réseau, serveurs, clients avec des modules comme socket , asyncio .
- Cybersécurité : Scripts d'analyse de vulnérabilités, tests de pénétration, scrapers sécurisés.
- Traitement du langage naturel (NLP) : Avec des bibliothèques comme NLTK , spaCy , transformers .

III. 4.4 LES AVANTAGES DE PYTHON

- 1.** Facile à apprendre et à lire : Syntaxe proche du langage humain, idéal pour les débutants.
- 2.** Langage polyvalent (multi-domaines) : Utilisé dans le web, l'IA, la science des données, l'automatisation, les jeux, etc.
- 3.** Gratuit et open source : Libre d'utilisation et constamment amélioré par une grande communauté.
- 4.** Grande communauté active : Beaucoup de tutoriels, forums, extensions et bibliothèques disponibles.
- 5.** Multiplateforme : Fonctionne sous Windows, macOS, Linux, Raspberry Pi, etc.
- 6.** Bibliothèques riches et frameworks : Pandas, NumPy, Django, Flask, TensorFlow... facilitent le développement.
- 7.** Typage dynamique : Aucun besoin de déclarer le type des variables.
- 8.** Gestion automatique de la mémoire : Le ramasse-miettes (garbage collector) libère la mémoire automatiquement.
- 9.** Supporte plusieurs paradigmes : Programmation procédurale, orientée objet et fonctionnelle.
- 10.** 10.Intégration avec d'autres langages : Compatible avec C/C++, Java, JavaScript, etc.

CHAPITRE III: TESTS ET RESULTATS

III.5 DESCRIPTION DU PROGRAMME DEBRUITAGE D'UNE IMAGE VIA GNN

Dans cette section nous allons décrire des parties du programme de débruitage d'une image via GNN :

CHARGEMENT DE L'IMAGE

```
image_path = "C:\\\\TraitementImage\\\\imag1.jpg"
image = cv2.imread(image_path)
if image is None:
    raise FileNotFoundError("Image introuvable.")
image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
h, w, c = image.shape
```

- On charge l'image à partir du chemin indiqué.
- Si l'image n'est pas trouvée → erreur bloquante.
- Conversion de BGR vers RGB (OpenCV lit en BGR).
- Récupération des dimensions : hauteur (h), largeur (w), canaux (c).

FONCTION POUR CONVERTIR L'IMAGE EN GRAPHE

```
def image_to_graph_color_with_pos(image):
    h, w, c = image.shape
    edges = []
    node_features = []
    for i in range(h):
        for j in range(w):
            pixel = image[i, j] / 255.0
            pos = [i / h, j / w]
            node_features.append(pixel.tolist() + pos)
            current_node = i * w + j
            for ni, nj in [(i-1,j),(i+1,j),(i,j-1),(i,j+1),(i-1,j-1),(i+1,j-1),(i-1,j+1),(i+1,j+1)]:
                if 0 <= ni < h and 0 <= nj < w:
                    neighbor_node = ni * w + nj
                    edges.append([current_node, neighbor_node])
    edge_index = torch.tensor(edges, dtype=torch.long).t().contiguous()
    x = torch.tensor(node_features, dtype=torch.float)
    return Data(x=x, edge_index=edge_index)
```

- Chaque pixel devient un noeud du graphe.
- Chaque noeud contient : La couleur normalisée (R, G, B), et sa position dans l'image (normalisée aussi).
- Les arêtes relient chaque pixel à ses 8 voisins
- Retourne un objet Data de PyG représentant le graphe.

CHAPITRE III: TESTS ET RESULTATS

AJOUT DE BRUIT A L'IMAGE

```
image_noisy = image / 255.0 + np.random.normal(0, 0.08, image.shape)
image_noisy = np.clip(image_noisy, 0, 1)
noisy_np = (image_noisy * 255).astype(np.uint8)
```

- On ajoute du bruit gaussien à l'image originale.
- Normalisation entre 0 et 1, ajout du bruit, puis retour à 0–255.
- np.clip() empêche les valeurs de sortir de cet intervalle.

FILTRE MEDIAN CLASSIQUE (POUR COMPARAISON)

```
image_median = cv2.medianBlur(noisy_np, 3)
```

- Applique un filtre médian pour lisser l'image bruitée.
- Cela permet de comparer plus tard avec notre méthode GNN.

CREATION DU GRAPHE A PARTIR DE L'IMAGE BRUITEE

```
data = image_to_graph_color_with_pos((image_noisy * 255).astype(np.uint8))
data.x = data.x.to(device)
data.edge_index = data.edge_index.to(device)
```

- Convertit l'image bruitée en graphe via la fonction définie plus haut.
- Envoie les données sur le GPU (si possible).

PREPARATION DE L'OBJECTIF (IMAGE ORIGINALE)

```
target = torch.tensor(image / 255.0, dtype=torch.float).view(-1, 3).to(device)
```

- L'image originale est transformée en tenseur PyTorch.
- Elle servira de cible pendant l'entraînement du modèle.

CHAPITRE III: TESTS ET RESULTATS

DEFINITION DU MODELE GNN

```
v class EnhancedColorGNN(torch.nn.Module):
v     def __init__(self, in_ch, out_ch):
v         super().__init__()
v         self.conv1 = GCNConv(in_ch, 64)
v         self.bn1 = BatchNorm1d(64)
v         self.conv2 = GCNConv(64, 64)
v         self.bn2 = BatchNorm1d(64)
v         self.conv3 = GCNConv(64, out_ch)
v         self.drop = Dropout(0.2)

v     def forward(self, x, edge_index):
v         x = torch.relu(self.bn1(self.conv1(x, edge_index)))
v         x = self.drop(x)
v         x = torch.relu(self.bn2(self.conv2(x, edge_index)))
v         x = self.drop(x)
v         x = torch.sigmoid(self.conv3(x, edge_index))
v
v         return x
```

- Classe du modèle GNN amélioré :
 - 3 couches GCN (Graph Convolutional Network),
 - Normalisation par batch,
 - Dropout pour éviter le sur-apprentissage.

INITIALISATION DU MODELE ET OPTIMISEUR

```
model = EnhancedColorGNN(in_ch=5, out_ch=3).to(device)
optimizer = torch.optim.Adam(model.parameters(), lr=0.01)
scheduler = torch.optim.lr_scheduler.StepLR(optimizer, step_size=30, gamma=0.5)
```

- On crée le modèle avec 5 entrées (R, G, B, x, y) et 3 sorties (R, G, B).
- Adam est l'optimiseur utilisé pour ajuster les poids du modèle.
- Le scheduler diminue progressivement le taux d'apprentissage.

BOUCLE D'ENTRAINEMENT DU MODELE

```
for epoch in range(100):
    model.train()
    optimizer.zero_grad()
    output = model(data.x, data.edge_index)
    output_img = output.view(h, w, 3).permute(2, 0, 1).unsqueeze(0)
    target_img = target.view(h, w, 3).permute(2, 0, 1).unsqueeze(0)
    loss = 1 - ssim(output_img, target_img, data_range=1.0, size_average=True)
    loss.backward()
    optimizer.step()
    scheduler.step()
    if (epoch + 1) % 10 == 0:
        print(f"Epoch {epoch+1}, SSIM Loss: {loss.item():.5f}")
```

CHAPITRE III: TESTS ET RESULTATS

- Pendant 100 époques, le modèle apprend à reproduire l'image originale.
- La perte (loss) est basée sur SSIM : on veut maximiser la similarité → donc minimiser $1 - \text{SSIM}$.
- À chaque itération, les gradients sont mis à jour.
- Toutes les 10 époques, on affiche la performance actuelle.

CONVERSION DE LA SORTIE EN IMAGE

```
output_np = output.view(h, w, 3).cpu().detach().numpy()
output_np = np.clip(output_np * 255, 0, 255).astype(np.uint8)
```

- On remet la sortie du modèle au format image (0–255, entier).
- np.clip() empêche les valeurs hors limites.

FONCTION POUR CALCULER SSIM

```
def compute_ssim(img1, img2):
    t1 = torch.tensor(img1 / 255.0, dtype=torch.float).permute(2, 0, 1).unsqueeze(0)
    t2 = torch.tensor(img2 / 255.0, dtype=torch.float).permute(2, 0, 1).unsqueeze(0)
```

- Prend deux images en entrée.
- Les convertit en tenseurs PyTorch.
- Calcule leur similarité structurelle (SSIM) .

CALCUL DES SSIM

```
1 ssim_noisy = compute_ssim(image, noisy_np)
2 ssim_median = compute_ssim(image, image_median)
3 ssim_gnn = compute_ssim(image, output_np)
4
5 print(f"SSIM - Bruisée : {ssim_noisy:.4f}")
6 print(f"SSIM - Médian : {ssim_median:.4f}")
7 print(f"SSIM - GNN : {ssim_gnn:.4f}")
```

Compare les 3 versions de l'image

- Originale vs image bruitée,
- Originale vs image filtrée médiane,
- Originale vs image débruitée par GNN.

CHAPITRE III: TESTS ET RESULTATS

III.6 TESTS ET RESULTATS

Nous allons présenter quelques exemples sur certaines images pour vérifier notre programme.

MESURE DE SIMILARITE SSIM

Par définition, une mesure de similarité est une fonction mathématique qui nous permet de mesurer la similarité entre images. En effet la mesure de la similarité entre deux images peut être définie de plusieurs manières, selon les critères et les algorithmes utilisés.

$$SSIM = \frac{1}{W} \sum_{i=1}^W l(i) \cdot c(i) \cdot s(i)$$

$l(i), c(i)$ et $s(i)$: Représentent respectivement les comparaisons de luminance, de contraste et de structure entre les images et w le nombre total de patchs.

Experimentation 1

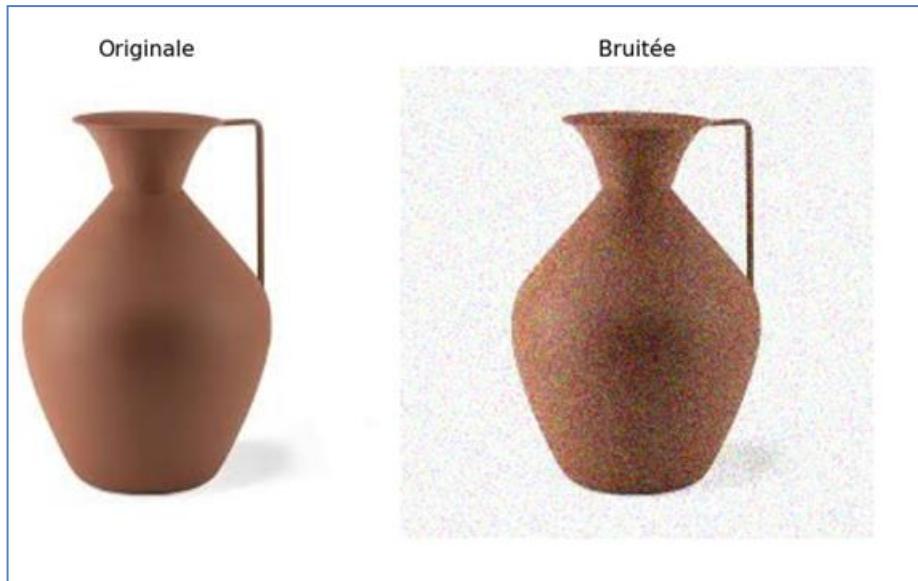


FIG.II.3 – IMAGE ORIGINALE ET IMAGE BRUITEE

CHAPITRE III: TESTS ET RESULTATS

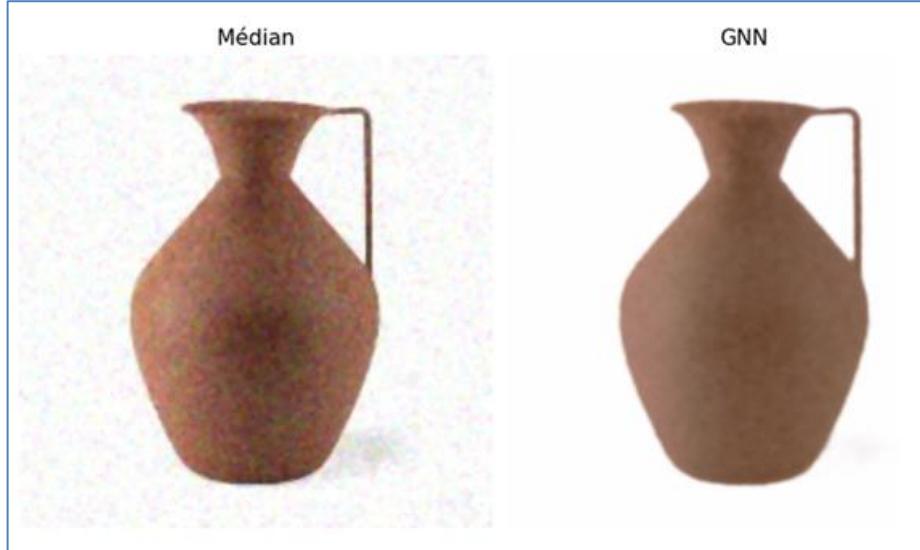


FIG.III.4 – DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN

SSIM - Bruitée : 0.3340

SSIM - Médian : 0.7264

SSIM - GNN : 0.9571

EXPERIMENTATION 2



FIG.II.5 – IMAGE ORIGINALE ET IMAGE BRUITEE

CHAPITRE III: TESTS ET RESULTATS



FIG.III.6– DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN

SSIM - Bruitée : 0.4521

SSIM - Médian : 0.9001

SSIM - GNN : 0.9596

EXPERIMENTATION 3

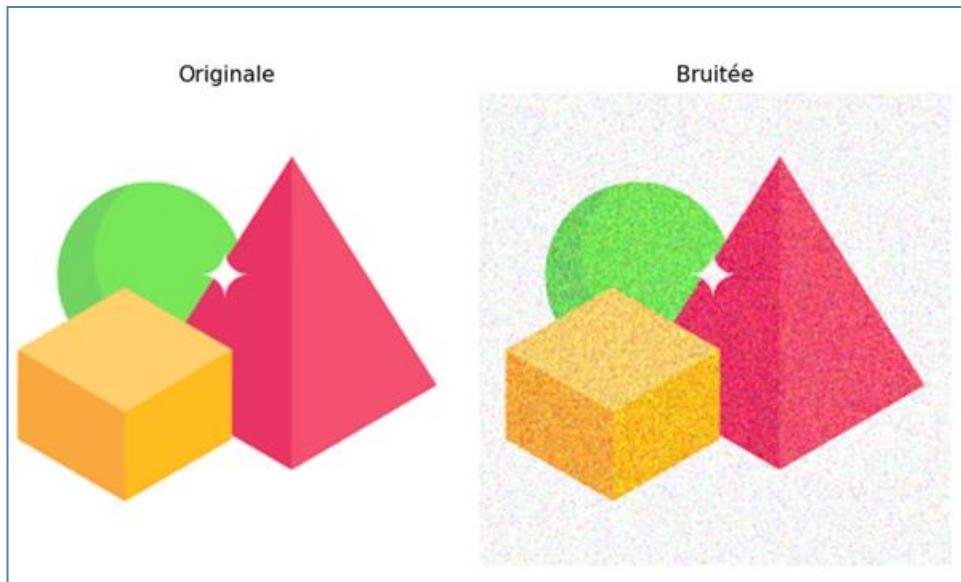


FIG.II.7– IMAGE ORIGINALE ET IMAGE BRUITEE

CHAPITRE III: TESTS ET RESULTATS

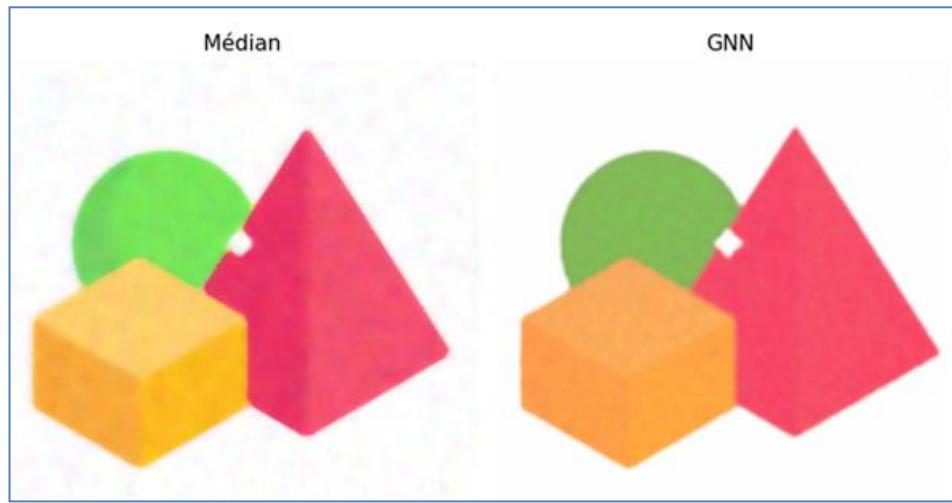


FIG.III.8 – DEBRUITAGE AVEC LE FILTRE MEDIAN ET L'APPROCHE GNN

SSIM - Bruitée : 0.3165

SSIM - Médian : 0.9223

SSIM - GNN : 0.9486

EXPERIMENTATION 4

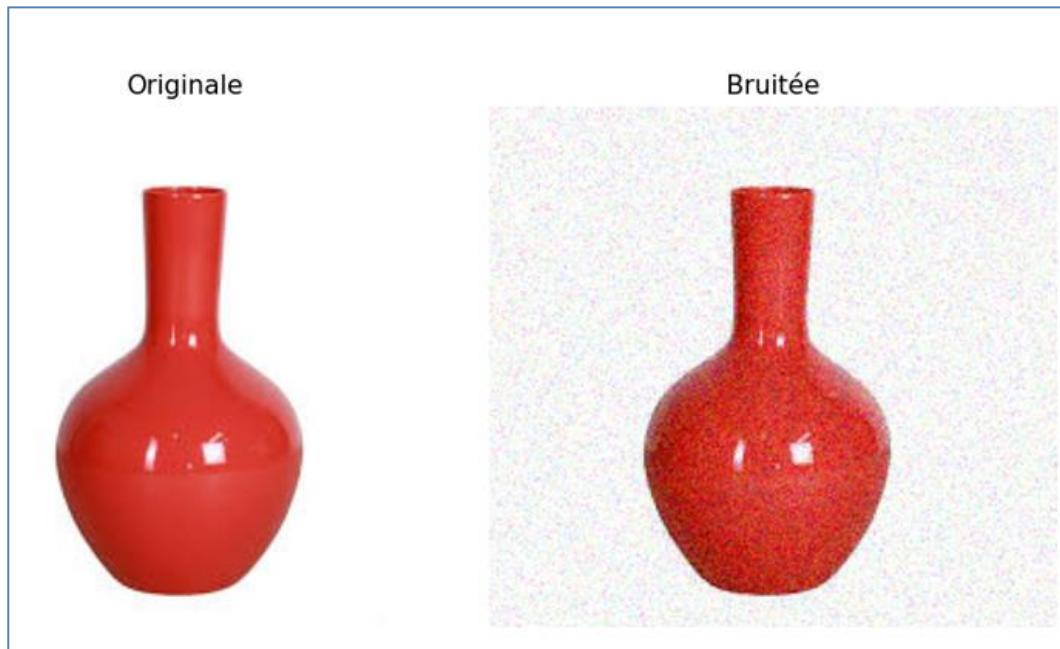


FIG.II.9 – IMAGE ORIGINALE ET IMAGE BRUITEE

CHAPITRE III: TESTS ET RESULTATS

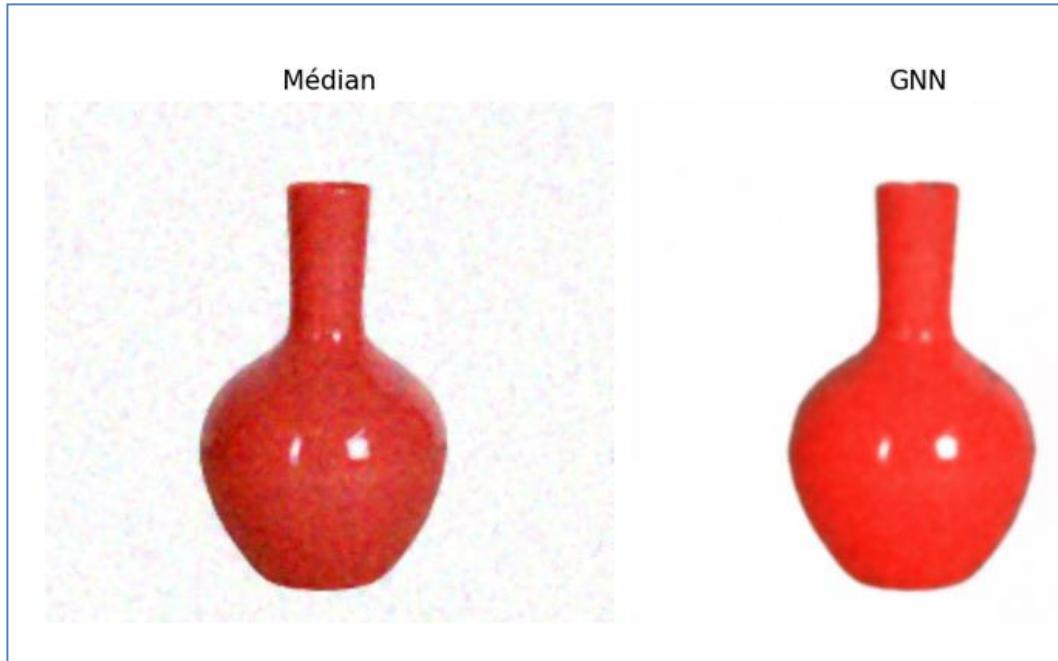


FIG.III.10 – DEBRUITAGE AVEC LE FILTRE MEDIAN ET L’APPROCHE GNN

SSIM - Bruitée : 0.3438

SSIM - Médian : 0.7428

SSIM - GNN : 0.9484

D'après les résultats (FIG.III. 4, 6, 8, et 10) nous remarquons que la méthode GNN est un outil puissant dans le domaine du traitement d'images, offrant des avantages significatifs par rapport au filtre traditionnel médian en termes de préservation des bords et de lissage de la texture. Les contours sont bien réservés.

CONCLUSION GÉNÉRALE

Conclusion Générale

Conclusion Générale

L'utilisation des Graph Neural Networks dans le traitement d'images ouvre une nouvelle perspective pour l'amélioration d'images bruitées. En s'affranchissant des contraintes de structure régulière imposées par les réseaux convolutifs, les GNN permettent une modélisation plus souple et fine des relations entre les pixels ou régions d'une image. Leur capacité à propager l'information au sein d'un graphe permet de compenser les zones dégradées en s'appuyant sur les voisinages pertinents, qu'ils soient proches spatialement ou similaires en intensité.

Nous avons vu que le cœur du fonctionnement des GNN repose sur le mécanisme de message passing, où chaque nœud apprend à affiner sa représentation en échangeant de l'information avec ses voisins. Appliqué aux images, ce principe permet de construire des modèles capables de filtrer le bruit tout en préservant les structures fines, y compris dans des contextes complexes.

Les résultats expérimentaux récents confirment l'efficacité des GNN dans cette tâche, en particulier lorsqu'ils sont associés des pertes perceptuelles basées sur la similarité structurelle (SSIM).

Enfin, le domaine est encore jeune et riche en opportunités : extensions aux vidéos, ou combinaisons avec d'autres architectures (CNN, Transformers) sont autant de pistes de recherche actives. Les GNN, en tant que modèles de traitement orientés-structure, constituent aujourd'hui une brique essentielle du traitement d'image intelligent, ouvrant la voie à des systèmes plus robustes, adaptatifs et interprétables.