REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

Université de Mohamed El-Bachir El-Ibrahimi - Bordj Bou Arreridj

Faculté *des Sciences et de la technologie*

Département d'électronique

Mémoire

Présenté pour obtenir

LE DIPLOME DE MASTER

FILIERE : **TELECOMMUNICATION**

Spécialité : Systèmes des Télécommunication

Par

- **Noui Ikram**
- **khoutri Nawal**

Intitulé

implémentation des Détecteurs de contours à Base de réseau Neurones Convolutifs CNN

Soutenu le : ……

Devant le Jury composé de :

Nom & Prénom	*Grade*	*Qualité*	*Etablissement*
M. Hassine Garbi Abde Elnour	*MCB*	*Président*	*Univ-BBA*
M. Messali Zoubeida	*MCB*	*Encadreur*	*Univ-BBA*
M.Hacini Latifa	*MCA*	*Examinateur*	*Univ-BBA*

Année Universitaire 2021/2022

Remerciements

Nous remercions dieu, le tout Puissant de nous avoir donné le courage, la sante et la volonté pour faire ce modeste travail.

Nous tenons à exprimer nos vifs remerciements à notre encadrant **Dr. Z. messali** pour avoir d'abord proposé ce projet de fin d'études et d'avoiraccepté de diriger ce travail ainsi que pour ses précieux conseils et surtout pournous avoir laissé une grande liberté dans la conception et la rédaction de ce projet.

Nous tenons à remercier les membres du jury d'avoir accepté de juger et évaluer notre travail.

Un grand merci à tous les **enseignants système télécom** du département d'électronique de la faculté des Sciences et Technologie de BBA pour tous leurs aides et services qui ne cessent de nous donner des conseils très importants en signe de reconnaissance.

Un remerciement pour toutes les personnes qui ont contribué de près ou de loin à la réalisation de ce projet de fin d'études.

Dédicace :

C’est avec une profonde gratitude et des mots sincères, que nous dédions ce travail à ceux qui nous ont aidés à faire ce modeste travail par leurs encouragements :

Nos parents

Ma seule sœur

Notre encadrant

Nos chers amis

A tous ceux qui nous ont aidés à faire ce

Travail A tout la promo ST 2022

Nous les remercions et leurs dédions ce travail

Table des matières

Liste des figures

Liste des tableaux

Liste des abréviations

LOG : Laplacian Of Gaussien.

DOG : Différance Of Gaussien

CNN : convolutional neural network

HED : Holistically-Nested Edge Detection

Introduction Générale

Introduction général :

Le but de la technologie de détection de contours est de découvrir et d'extraire des informations de frontière pour l'image .

Compte tenu des défis associés aux grands périmètres de bord, au positionnement inexact et à la faible précision de détection, les chercheurs ont proposé une gamme de techniques de détection de contours fondées sur l'apprentissage profondeur, comme l'intégration de fonctionnalités multi-échelles, le codage, la reconstruction du réseau, et ainsi de suite. Ce projet se concentre sur une analyse approfondie et une recherche spéciale sur les méthodes de détection de contours. Tout d'abord, la théorie et la méthode de chaque algorithme sont fournies en classant la structure à plusieurs niveaux des algorithmes classiques de détection de contours (méthodes traditionnelles). Deuxièmement, les problèmes techniques, les avantages des approches et la sélection du réseau central de chaque algorithme sont examinés en se concentrant sur l'algorithme de détection basé sur l'apprentissage profond. La performance de chaque algorithme est ensuite testée à l'aide d'essais sur les ensembles de données BSDS500 et NYUD.

Ce présent travail est organisé comme suit :

Dans le chapitre 1, nous examinons les types et les domaines de l'apprentissage en profondeur, en détaillant l'approche choisie pour la recherche sur CNN.

Dans le chapitre 2, nous présentons les différentes méthodes de détection des contours dérivatives premier et second ordre. Ensuite nous présentons la méthode ED basée sur l'apprentissage profond.

Dans le chapitre 3, nous présentons nos résultats de simulation. Plus précisément, nous montrerons les résultats de détection de contours des algorithmes conventionnels et l'algorithme HED. Les algorithmes conventionnels de détection de contours sont implémentés sous Matlab et l'algorithme HED à base d'apprentissage fond est implémenté sous Python.

La conclusion générale est présentée à la fin du manuscrit.

Chapitre 01 :
L'apprentissage profond (Deep Learning)

I.1 Introduction

L'apprentissage profond est un sous-ensemble de l'apprentissage automatique qui a été créé pour aider l'intelligence artificielle à atteindre son objectif premier d'imiter l'intellect humain. Les réseaux neuronaux, qui sont combinés pour former des réseaux de neurones profonds, sont les éléments fondamentaux de l'apprentissage profond.

Ces approches ont considérablement progressé dans les domaines du traitement du son et de l'image, y compris l'identification faciale, la reconnaissance vocale, la vision par ordinateur, le traitement autonome du langage et la classification du texte, avec un large éventail d'applications possibles.

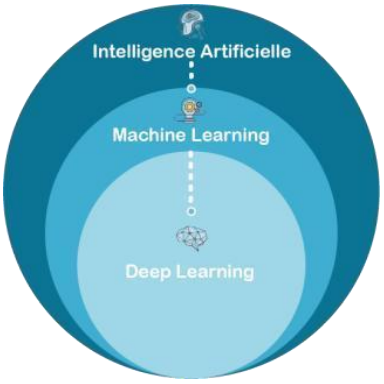

Figure 1.1 : La relation entre l'intelligence artificielle, le ML et le deep Learning(apprentissage profond)

Les réseaux neuronaux artificiels sont au cœur de l'apprentissage profond. Une cascade profonde de couches peut être utilisée pour créer une variété de structures. Ils nécessiteront des méthodes d'optimisation stochastiques intelligentes et une initialisation, ainsi qu'un choix de structure intelligent. Malgré le fait qu'il y a peu de fondements théoriques accessibles, ils fournissent des résultats assez spectaculaires.

I.2 Définition

L'apprentissage profond est une fonction d'intelligence artificielle (IA) qui traite les données et crée des modèles de prise de décision de la même façon que le cerveau humain. L'apprentissage profond est une sous-classe de l'apprentissage automatique en IA qui utilise les réseaux neuronaux pour apprendre à partir de données non structurées ou non étiquetées sans être vu. Réseaux neuronaux profonds ou apprentissage neuronal profond sont d'autres termes pour la même chose .

I.3 Types de modèles utilisant des architectures d'apprentissage en profondeur

L'apprentissage en profondeur offre une grande précision dans les ensembles de données

encombrés. Les algorithmes d'apprentissage en profondeur préfèrent les applications de prédiction, de classification et d'identification, comme le montre la figure 2. Les algorithmes d'apprentissage en profondeur et les réseaux de neurones profonds (DNN), les réseaux de neurones convolutifs en réseau (CNN) et les réseaux de neurones récurrents (RNN) et la mémoire à long terme (LSTM) sont utilisés dans ces applications.

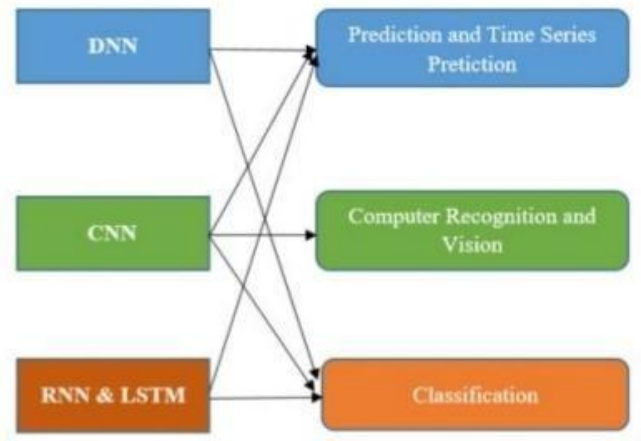

Figure 1.2. Types de modèles utilisant des architectures d'apprentissage en profondeur Types de modèles utilisant des architectures d'apprentissage en profondeur

I.4 Domaines d'application de l'apprentissage profonde :

L'apprentissage profond est une technique qui est largement utilisée dans une variété de disciplines, y compris :

I.4.1 La reconnaissance faciale :

La phase initiale sera d'alimenter l'algorithme d'un ensemble de photos, après quoi le système sera en mesure de reconnaître un visage dans une image après une certaine formation.

I.4.2 Le traitement automatique de langage naturel :

Une autre utilisation de DL est le traitement automatique du langage naturel. Son but est d'extraire le sens des mots, voire des phrases, afin de mener une analyse émotionnelle.

I.4.3 Voitures autonomes :

Les entreprises qui fournissent des services d'aide à la conduite doivent éduquer un ordinateur à apprendre certains aspects clés de la conduite à l'aide de systèmes de capteurs numériques plutôt que le cerveau humain. Les entreprises commencent normalement par utiliser une énorme quantité de données pour former des algorithmes.

I.4.4 Traduction automatique :

Il s'agit d'une tâche qui consiste à traduire mécaniquement des mots, des phrases ou des phrases d'une langue à l'autre. Bien que la traduction automatique existe depuis longtemps, DL produit les plus grands résultats dans deux domaines :

• Traduction de texte par machine

• Traduction d'images par une machine

I.4.5 Recherche en marketing :

DL pourrait être pratique en arrière-plan en plus de la recherche de nouvelles fonctionnalités qui peuvent améliorer votre application. Les modèles de régression et de classification DL peuvent aider à la segmentation du marché, à l'analyse des campagnes de marketing et bien plus encore [1].

I.5 Réseaux de neurones

Un réseau de neurones artificiels est un système dont la conception s'inspire du fonctionnement des neurones biologiques et s'est depuis rapproché des méthodes statistiques [2]. Les réseaux neuronaux artificiels sont des réseaux qui sont fortement liés par des processeurs élémentaires qui fonctionnent en parallèle. Sur la base des informations reçues, chaque processeur élémentaire (neurone artificiel) calcule une seule sortie.

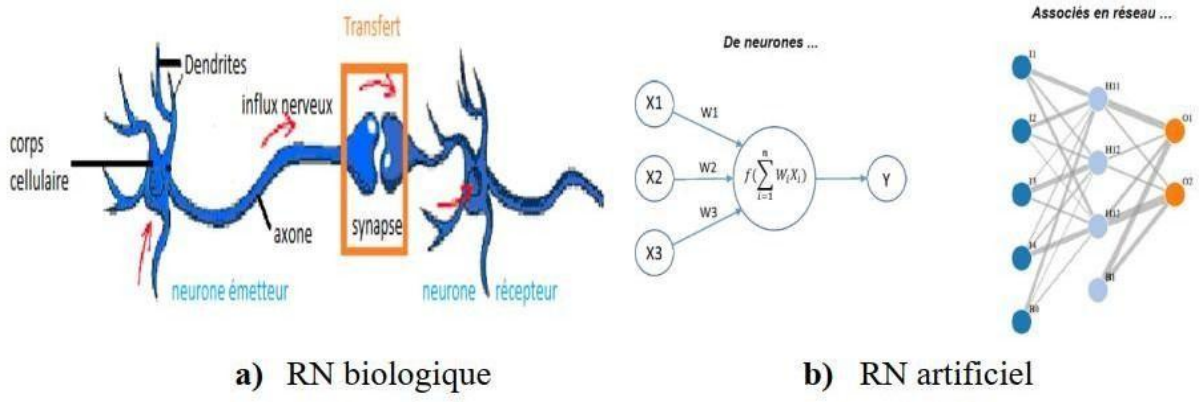

Figure 1.3 : réseau de neurones [4]

I.6 Architectures de réseaux de neurones profonds :

Le Deep Learning est un domaine en évolution rapide, avec de nouvelles architectures de variantes algorithmiques qui apparaissent chaque semaine [3]. Cette section donnera un aperçu de haut niveau des structures communes présentes dans de nombreux réseaux profonds.

I.6.1 Les réseaux de neurones convolutifs :

La vision par ordinateur évolue à un rythme rapide. L'une des raisons en est l'avancement de l'apprentissage profond. Le réseau neuronal convolutif est un terme utilisé dans la vision par ordinateur. Ils sont connus comme l'acronyme de CNN pour le réseau neuronal convolutionnel et sont divisés en deux parties :

Pour le dire autrement, la première partie d'un CNN est la partie déroutante. Elle exécute la fonction d'un extracteur de caractérisation d'image. Une image est passée à travers une série de filtres, ou noyaux de convolution, pour produire de nouvelles images connues sous le nom de cartes de convolutions. Certains filtres intermédiaires utilisent une opération locale maximale pour réduire la résolution de l'image. Enfin, les cartes convolutionnelles sont présentées et concaténées en un vecteur de caractéristiques connu sous le nom de code CNN.

La sortie du code CNN de la partie convolutive est ensuite couplée à l'entrée d'un second composant à base de couches entièrement connecté. Le travail de cette section est d'intégrer les fonctionnalités du code CNN afin de catégoriser l'image. Une couche finale avec un neurone par type est le résultat. Les valeurs d'une distribution de probabilité catégorique [3].

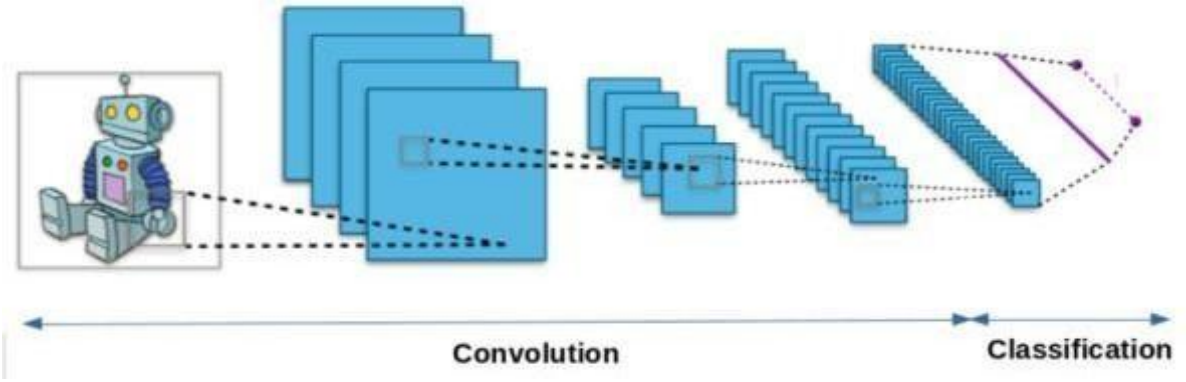

Figure 1.4 : Les réseaux de neurones convolutifs.

I.6.2 Les couche de réseaux de neurones convolutionnels :

Il y a plusieurs couches différentes dans CNN comme le montre la figure 3

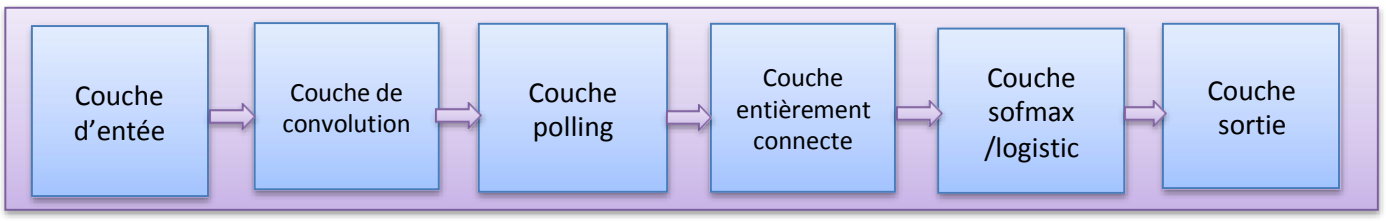

Figure 1.4 Les couches de CNN

I.6.2.1 La couche de convolution (CONV) :C'est la couche la plus importante et le cœur des éléments constitutifs du réseau convolutif, et c'est aussi elle qui effectue le plus de calculs lourd car les caractéristiques de l'image sont extraites dans cette couche.

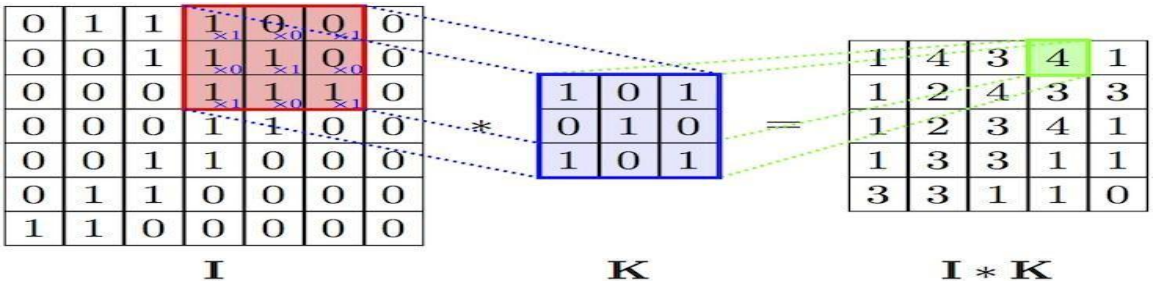

Figure 1.6 : Exemple de principe du filtre convolutionnel[6]

I.6.2.2 La couche de Pooling :

Est utilisée pour réduire le volume spatial de l'image d'entrée après la convolution. Sa fonction est de réduire progressivement la taille spatiale de la représentation pour réduire le nombre de paramètres et de calculs dans le réseau, et donc de contrôler également le overfitting .

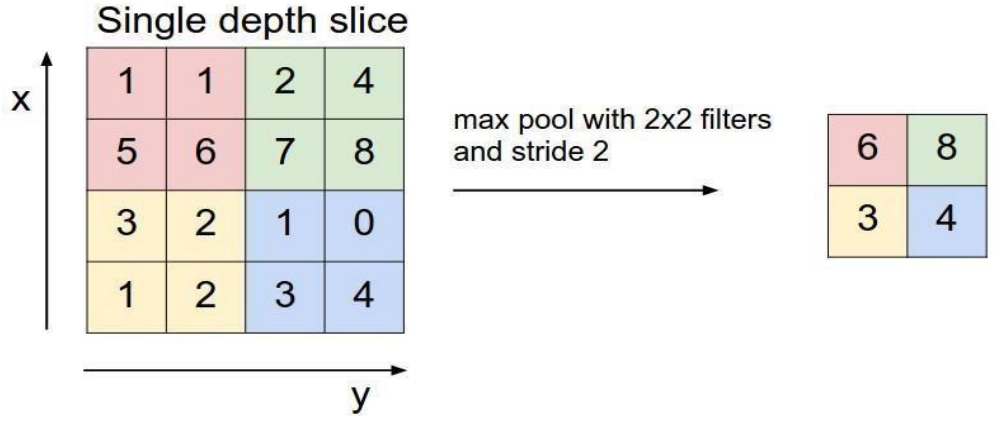

Figure 1.7 : Exemple de principe du Pooling

I.6.2.3 Couche entièrement connectée(FC) :

Après plusieurs couches de convolution et de max-pooling le raisonnement de haut niveau dans le réseau neuronal se fait via des couches entièrement connectées [4]. Il connecte les neurones d'une couche aux neurones d'une autre couche. Il est utilisé pour classer les images entre différentes catégories par formation

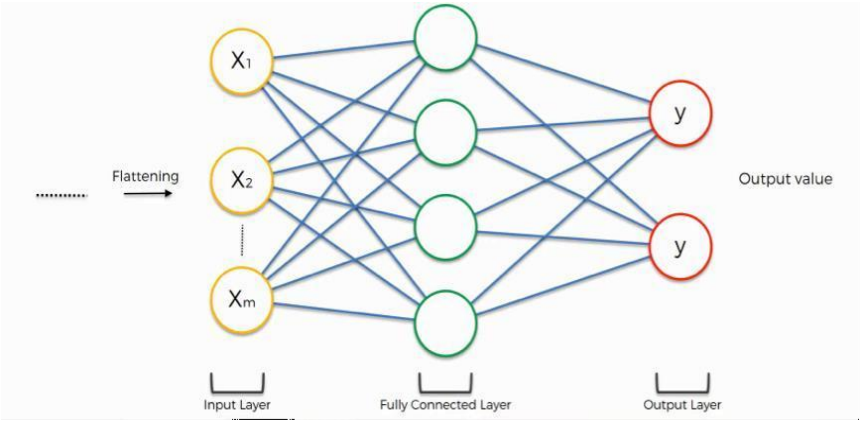

Figure 1.8 : Principe de la couche entièrement connectée (FC) [5]

I.6.2.4 Couche Logistique ou Softmax :

Est la fin de la couche entièrement connectée .La logistique est utilisée pour la classification binaire et Softmax est pour la multi-classification.

I.6.2.5 Couche de sortie (output layer) :

La couche de sortie contient l'étiquette qui est sous forme codée comme le montre la figure 6.

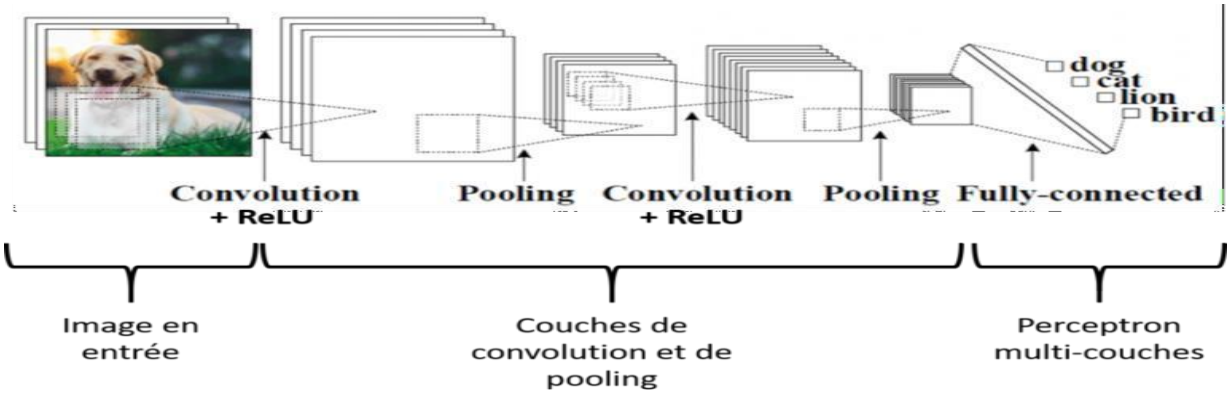

Figure 1.8 : Exemple montrant l'étiquette codée de la couche de sortie CNN

I.6.2.6 Les avantage de CNN

- l'apprentissage en profondeur utilisant devient de plus en plus populaire en raison de trois facteurs important
- CNN élimine le besoin d'extraction manuelle des fonctionnalités
- CNN apprend ces fonctionnalités directement.
- CNN produit des résultats de reconnaissance de pointe
- Les CNN peuvent être recyclés pour de nouvelles tâches de reconnaissance, ce qui vous permet de vous appuyer sur les réseaux existants [11].

- CNN offre la possibilité de calculer automatiquement des cartes d'entités, évitant ainsi aux utilisateurs d'effectuer des calculs d'entités lourds

I.6.3 : Réseau de neurones récurrents

L'idée derrière les réseaux neuronaux récurrents (RNN) est d'employer l'information séquentielle [9]. Ces réseaux neuronaux récurrents sont un type de réseau neuronal qui utilise des états cachés pour permettre aux prédictions antérieures d'être utilisées comme entrées.

Voici à quoi ressemble un RNN typique :

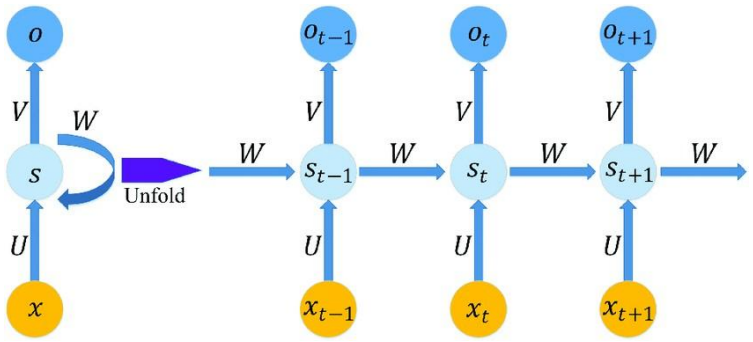

Figure 1.10 : Schéma d'un réseau de neurones récurrents à une unité reliant l'entrée et la sortie du réseau. A droite la version « dépliée » de la structure

Un RNN déroulé est vu dans le diagramme ci-dessus. Nous voulons juste dire que nous affichons le réseau pour la séquence complète quand nous disons "dérouler." Si la séquence qui nous intéresse est une phrase de cinq mots, par exemple, le réseau serait étendu en un réseau de cinq couches de neurones, une couche pour chaque mot. Voici les formules qui servent à calculer les RNN :

- X_t est l'entrée au moment t.
- U, V, W sont les paramètres que le réseau va apprendre des données de l'apprentissage.
- $S_t$est l'état caché au moment t. C'est la « mémoire » du réseau. st est calculé en fonction de l'état

  Caché précédent et de l'entrée à l'étape actuelle :

$$St = (Ux_t + Ws_{t-1}) \quad \textbf{(I. 1)}$$

I.7 L'apprentissage profond et réseaux de neurones

I.7.1 Introduction

Dans l'apprentissage profond, le processus d'apprentissage revient à la formation en réseau neuronal à l'aide d'optimisateurs itératifs, qui ne réduisent la fonction de coût à une valeur très faible. Pour apprendre, nous pouvons utiliser une variété de méthodes, mais la plus populaire est le processus itératif d'optimisation de la descente en gradient. Le processus d'apprentissage revient au problème d'optimisation, dans lequel une fonction doit être minimisée ou maximisée (x). La fonction objective est également connue sous le nom de fonction de coût, fonction de perte ou fonction d'erreur, selon les auteurs dans [7]. L'algorithme cherche à trouver le minimum global sans tomber dans le piège du minimum local pendant l'entraînement. Cette notion est illustrée dans le diagramme ci-dessous, qui a été créé par les auteurs dans [7].

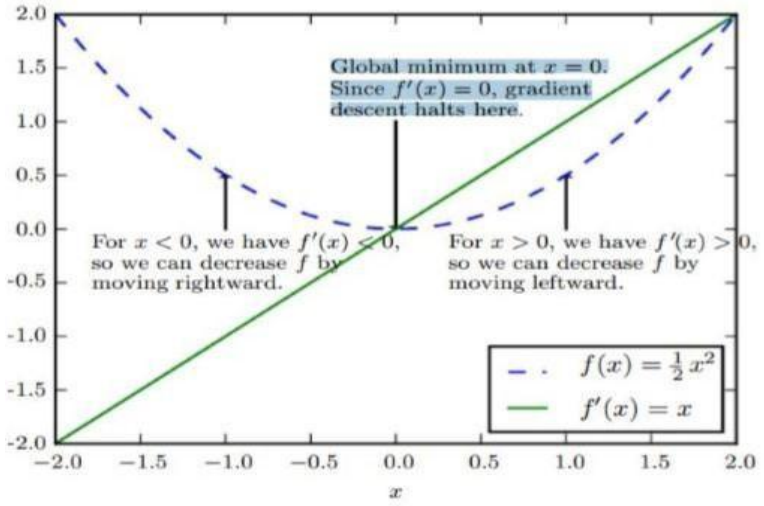

Figure 1.11 : Une illustration du processus de recherche de l'optimum.

I.7.2 Régression logistique en tant que réseau de neurones

Classement binaire

Dans un problème de classification binaire, nous avons une entrée x, disons une image, et nous devons la classer comme ayant un chat ou non. Si c'est un chat, nous lui attribuerons un 1, sinon 0. Donc ici, nous n'avons que deux sorties - soit l'image contient un chat, soit elle n'en contient pas. Ceci est un exemple de problème de classification binaire.

Nous pouvons bien sûr utiliser la technique de classification la plus populaire, la régression logistique, dans ce cas.

- **Régression logistique**

La régression logistique convient à la classification binaire. Par exemple, on peut dire un ensemble de données où y = 0 ou 1 avec 1 représente la classe par défaut. Pour illustrer on peut imaginer que l'on veuille prédire s'il pleuvra ou non. On aura 1 pour s'il pleut et 0 le cas contraire [11].

Nous avons une entrée X (image) et nous voulons connaître la probabilité que l'image appartienne à la classe 1 (c'est-à-dire un chat). Pour un vecteur X donné, la sortie sera :

$$Y=w(transposé)X + b \quad \textbf{(I.2)}$$

Ici **w** et **b** sont les paramètres. Puisque notre sortie y est une probabilité, elle devrait être comprise entre 0 et 1. Mais dans l'équation ci-dessus, elle peut prendre n'importe quelle valeur réelle, ce qui n'a pas de sens pour obtenir la probabilité. La régression logistique utilise donc également une fonction sigmoïde pour générer des probabilités :

$$\hat{y} = \sigma(w^T x + 1) \quad \textbf{(I.3)}$$

Pour toute valeur en entrée, il ne renverra que des valeurs comprises entre 0 et 1. La formule d'une fonction sigmoïde est

$$\sigma(t) = \frac{1}{1+e^{-z}} \quad \textbf{(I.4)}$$

Donc, si z est très grand, $\exp(-z)$ sera proche de 0, et donc la sortie du sigmoïde sera 1. De même, si z est très petit,$\exp(-z$) sera l'infini et donc la sortie du sigmoïde sera 0.

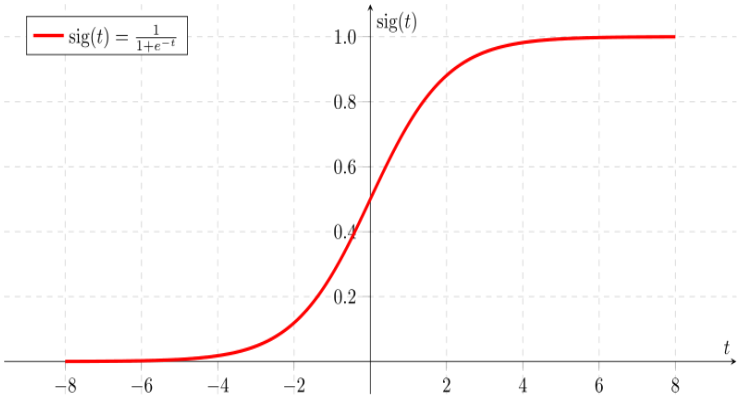

Figure 1.12 : Régression logistique.

- **Fonction de coût de régression logistique**

Pour former les paramètres w et b de la régression logistique, nous avons besoin d'une fonction de coût. Nous voulons trouver les paramètres w et b tels qu'au moins sur l'ensemble

D'apprentissage, les sorties que vous avez (y-hat) soient proches des valeurs réelles (y).On peut utiliser une fonction$y^{\wedge}$ de perte définie ci-dessous :

$$f\ (y^{\wedge}, y) = \frac{1}{2}(y^{\wedge},y)^2 \qquad \textbf{(I.5)}$$

Le problème avec cette fonction est que le problème d'optimisation devient non convexe, ce qui entraîne plusieurs optima locaux. Par conséquent, la descente de gradient ne fonctionnera pas bien avec cette fonction de perte. Ainsi, pour la régression logistique, nous définissons une fonction de perte différente qui joue un rôle similaire à celui de la fonction de perte ci-dessus et résout L'équation mise à jour pour la descente de gradient devient :

$$Log\ Loss = \sum_{(x,y)\in D} -ylog(y^{\wedge}) - (1-y)\log(1-y^{\wedge}) \qquad \textbf{(I.6)}$$

La fonction de cout est définie pour un seul exemple d'entraînement qui nous indique dans quelle mesure nous nous débrouillons sur cet exemple particulier. D'autre part, une fonction de coût concerne l'ensemble de la formation. La fonction de coût pour la régression logistique est :

$$\mathcal{J}(w,b)\ \frac{1}{M}\sum_{i=1}^{M} f\ (y^{\wedge(i)}, y^{(i)}) \qquad (\mathbf{I.7})$$

Nous voulons que notre fonction de coût soit aussi petite que possible. Pour cela, nous voulons que nos paramètres w et b soient optimisés.

- **Descente graduelle**

La descente de gradient est un algorithme d'optimisation en apprentissage automatique pour réduire la fonction de coute en général, capable de trouver des solutions optimales à un grand nombre de problèmes. L'idée générale de la descente de gradient est de corriger petit à petit les paramètres dans le but de minimiser une fonction de coût.

Il s'agit d'une technique qui permet d'apprendre les paramètres w et b de manière à minimiser la fonction de coût. La fonction de coût pour la régression logistique est de nature convexe (c'est-à-dire qu'un seul minima global) et c'est la raison du choix de cette fonction au lieu de l'erreur quadratique (peut avoir plusieurs minima locaux).

Regardons les étapes de la descente en dégradé :

1. Initialiser w et b (généralement initialisés à 0 pour la régression logistique)
2. Faites un pas dans la direction de descente la plus raide
3. Répétez l'étape 2 jusqu'à l'optimum global

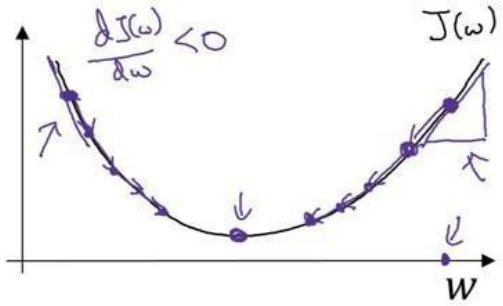

Figure 1.13 gradient graduelle d'une fonction de coût

On remarque La fonction de cout diminue en termes de poids(w).

L'équation mise à jour pour la descente de gradient devient

$$w := w - \alpha \frac{d\, I(w,b)}{d\, w} \qquad \textbf{(I.8)}$$

$$b := b - \alpha \frac{d\, I(w,b)}{d\, b} \qquad \textbf{(I.9)}$$

Ici, a est le taux d'apprentissage qui contrôle la taille d'un pas que nous devons faire après chaque itération.

Notre objectif est de fonction de cout fonction pour la trouve la valeur optimisée des poids.

I.7.3 Optimisation

I.7.3.1 Les variantes de la descente de gradient

La descente de gradient est l'approche la plus connue pour améliorer un réseau neuronal [8], dans lequel les paramètres x sont mis à jour dans la direction opposée de la direction de gradient de la fonction objective pour minimiser une fonction objective (x).

Cette approche vient en trois saveurs. Nous allons créer un compromis entre la précision de la mise à jour des paramètres et le temps d'exécution de cette mise à jour, en fonction de la quantité de données

.- Batch gradient descente.

-Descente de gradient stochastique.

-Mini-batch gradient descente.

I.7.3.2 Algorithmes d'optimisation de la descente de gradient

Pour aborder les questions de l'optimisation des réseaux neuraux avec descente en gradient nous pourrions mettre en évidence certaine technique couramment utilisées par la

communauté du deep Learning.

-Momentum.

-Nesterov accelerated gradient.

-Adagrad

-RMSprop

- Optimizer Adam

I.8 Les fonctions d'activation

La fonction d'activation est une procédure mathématique qui normalise la sortie (signaux neuronaux). Sa fonction en tant que composant de réseau neuronal est d'introduire la non-linéarité dans le réseau. L'utilisation d'une fonction d'activation permet au réseau neuronal de représenter de grandes quantités de données et de résoudre des problèmes complexes. Les fonctions d'activation viennent dans une variété de formes et de tailles.

- **Fonction Softmax**

Il est utilisé pour calculer la distribution de probabilité d'un ensemble d'entiers dans une entrée vectorielle. Le résultat d'une fonction d'activation Softmax est un vecteur dont les valeurs indiquent la probabilité qu'une classe ou un événement se produise. Les valeurs du vecteur se résument à une. [9]

- **Fonction sigmoïde**

  L'équation de la fonction est :

$$f(x) = \frac{1}{1-e^{-z}} \qquad \textbf{(I.10)}$$

  C'est une fonction non linéaire, la valeur est entre 0 et 1(**Figure 1.15**).

  La fonction sigmoïde est particulièrement utilisée pour les modèles où elle devait prédire la probabilité en tant que sortie, car la probabilité de qu'elle que soit n'existe qu'entre 0 et 1, le sigmoïde est le bon choix [9]

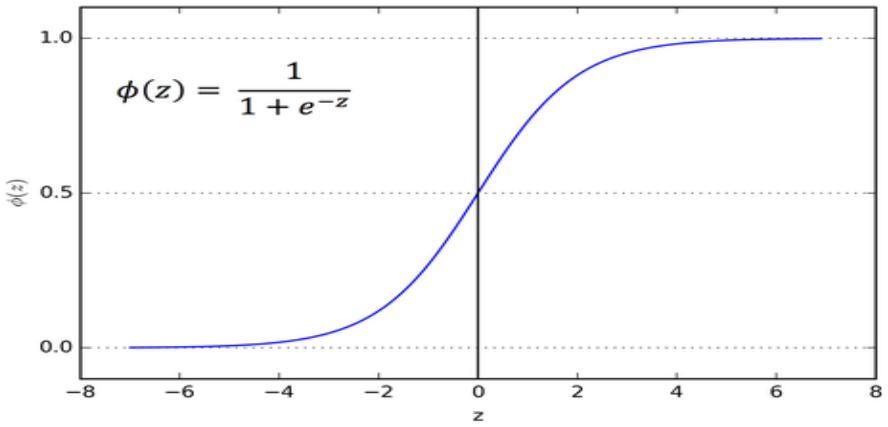

Figure 1.14 : fonction sigmoïde

- **Fonction Tanh**

L'équation de fonction est :

$$f(x = (\frac{2e^{x}}{e^{x}+e^{-x}}) \quad \textbf{(I.11)}$$

La fonction tanh est pareille avec la fonction sigmoïde logistique mais mieux. La valeur estentre 1 et -1 (**Figure 1.16**). Après la différenciation, la valeur de cette fonction devient inférieure à 1.

La relation entre tanh (x) et sigmoïde (x) est :

$$(\boldsymbol{x}) = (\boldsymbol{2x}) - \boldsymbol{1} \quad \textbf{(I.12)}$$

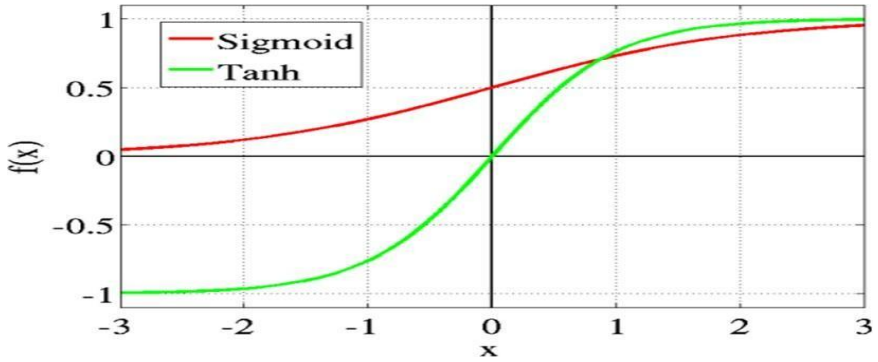

Figure 1.15 : tanh v/s Logistic Sigmoid

Dans un graphique tanh, les entrées négatives seront mappées fortement négatives, tandis que les entrées nulles seront mappées près de zéro, grâce à cette fonction. [9]

- **Fonction d'activation ReLU**

Dans l'usage contemporain, la fonction Relu (Unité linéaire rectifiée) a été mondialisée, et elle est employée dans pratiquement tous les réseaux de neurones convolutifs (couche cachée) (Figure 3.11):

$$(\boldsymbol{x}) = \textbf{max}\ (\boldsymbol{0}, \boldsymbol{x}) \quad \textbf{(I.13)}$$

La plage de valeur de cette fonction est : (0, inf.) [10].

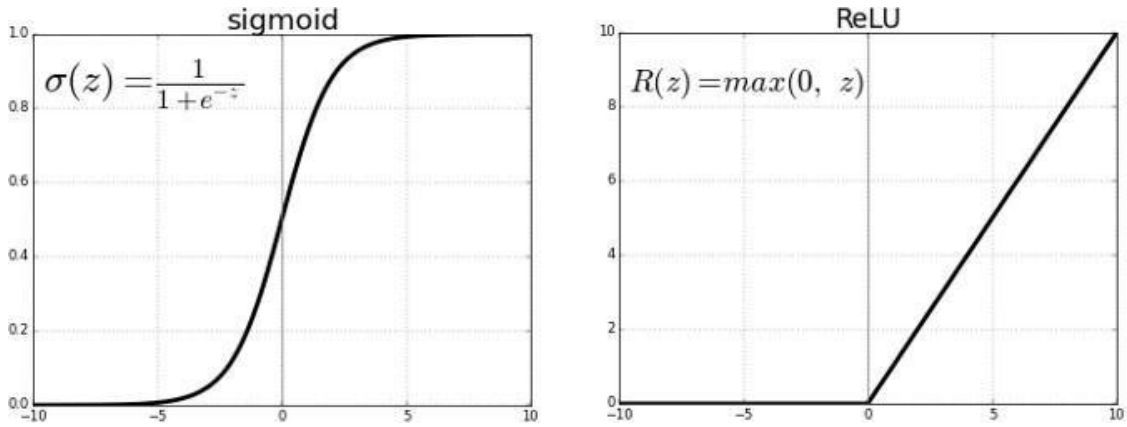

Figure 1.16 : ReLU v/s Logistic Sigmoïde [9].

Comme on observe dans la **Figure 3.11** le ReLU est à moitié rectifié (par le bas).

- R (z) égale zéro : ((z) = 0) lorsque z est inférieur à zéro.
- R (z) est égal à z : ((z) = z) lorsque z est supérieur ou égal à zéro

I.9 Conclusion

Aujourd'hui, l'apprentissage profond va au-delà des différentes méthodes d'apprentissage automatique en termes de performance et est largement utilisé pour une variété de tâches différentes. L'apprentissage approfondi améliore la précision par rapport à d'autres moyens de tâches tels que la traduction linguistique et la reconnaissance d'image. Mais cela ne se produira pas dans quelques années, il a fallu des décennies [11].

Dans ce chapitre, nous donnons un aperçu des types et des domaines de l'apprentissage profond, détaillant la méthode choisie de notre recherche, CNN.

Chapitre 02 :
Algorithmes De détection de contours d'images à base De CNN

2.1 Introduction :

Le contour de l'image est l'une des caractéristiques les plus fondamentales et significatives. La détection des contours est toujours l'un des projets classiques d'étude de la vision par ordinateur et du traitement de l'image. C'est la première étape de l'analyse et de la compréhension des images. Le but de la détection des bords est de découvrir l'information sur les formes et la réflectance ou la transmittance dans une image. C'est l'une des étapes fondamentales du traitement d'image, de l'analyse d'image, de la reconnaissance des modèles d'image et de la vision par ordinateur, ainsi que de la vision humaine. L'exactitude et la fiabilité de ses résultats affectent directement le système de machine de compréhension fait pour le monde objectif. Décrit d'abord les caractéristiques des contours, les propriétés des détecteurs traditionnels et la méthodologie de détection des bords commune, Ce chapitre présente les avantages et les inconvénients des détecteurs conventionnels. A la fin le concept de détection de contours à base de deep Learning est introduit.

2.2 Méthodes conventionnelle de détection de contours :

2.2.1 Définition d'un contour :

La mise en évidence des points représentant les contours d'objets dans une image peut servir à reconnaître des objets présents dans une scène, à différencier des zones de l'image, à faire de la segmentation d'images, à extraire une information réduite souvent pertinente pour caractériser l'image. Un contour se matérialise par une rupture d'intensité dans l'image suivant une direction donnée. Plusieurs méthodes existent pour détecter cette rupture, les unes plus ou moins complexes, les autres plus ou moins gourmandes en calculs. Dans la plupart des cas et en particulier pour ceux présentées ici, la même méthodologie est employée. Elle s'applique en deux étapes : la première permet de localiser les contours à partir d'un calcul de Gradient ou de Laplacien dans des directions privilégiées tout en quantifiant l'importance du contour (voir figure ci-après). La seconde étape va permettre d'isoler les contours du reste de l'image à partir d'un seuillage judicieux. Plusieurs méthodes permettent de déterminer le Gradient ou le Laplacien d'une image. Il en est de même des techniques de seuillage. Ces deux étapes sont indépendantes, il existe donc un grand nombre de combinaisons calcul de Gradient-opération de seuillage conduisant à la mise en évidence des contours. Le rôle de l'ingénieur est souvent

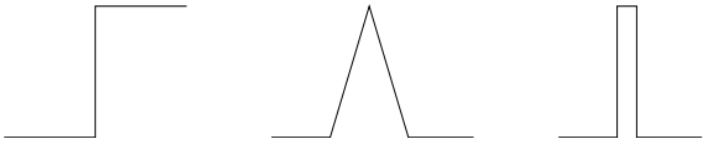

Figure.2.1 quelque profil de contours : marche d'escalier, toit et pointe

2.2.2 Méthodes dérivatives :

Il convient avant d'exposer ces méthodes de définir les contours et leurs types.

La figure (2.2) illustre deux types de contours(a) et (b) en escalier

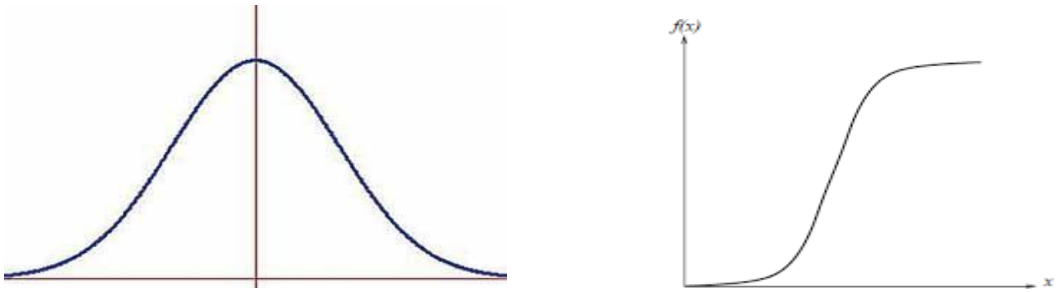

Figure 2.2 Point contour n'en cloche et point de contour en escalier

Soit une image continue $f(x, y)$ un contour apparait comme une ligne ou sont localisées les très fortes variations de (x, y) . En 1D, un contour correspond à un maximum de la dérivée première, c'est _à_ dire à un passage par zéro de la dérivée seconde, comme c'est illustré sur la figure 2.3

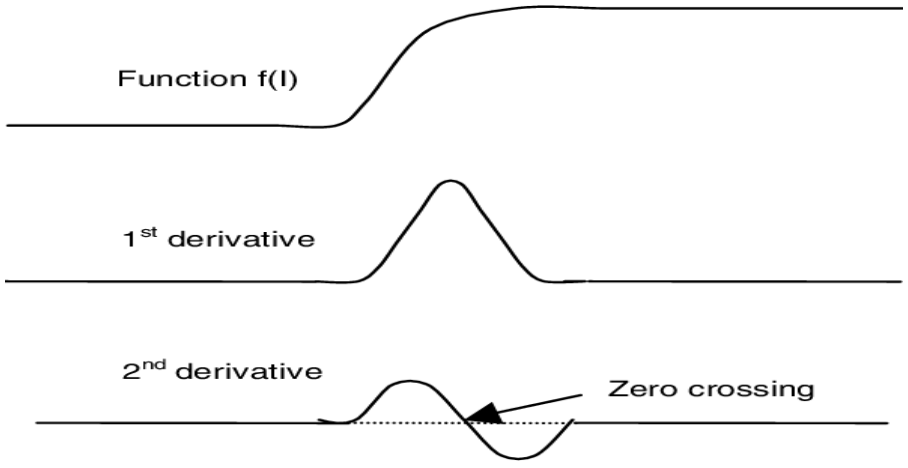

Figure.2.3 dérivée première et dérivée seconde de fonction en escalier

2.2.2.1 Méthode dérivative du premier ordre :

- **Le gradient** : Maxima du gradient dans la direction du gradient.

$$\nabla f = (\frac{\partial f}{\partial x} , \frac{\partial f}{\partial y}) \quad \textbf{(2.1)}$$

L'estimation dérivée partielle de premier ordre :

$$\nabla f = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}) = (\nabla xf, \nabla yf) = \begin{cases} \nabla xf = f * M1 \\ \nabla yf = f * M2 \end{cases} \quad \textbf{(2.2)}$$

On peut écrire :

$$\nabla f = f * (\nabla x, \nabla y) = f * (M1, M2) \quad \textbf{(2.3)}$$

M1 et M2 sont des noyaux de convolution approximation le gradient. Approximation numérique du gradient : différences finies

- $\nabla xf \approx f(x,y) - f(x-1,y)$ Et $\nabla yf \approx f(x,y) - f(x,y-1)$ **(2.4)**

Où

- $\nabla xf \approx f(x+1,y) - f(x,y)$ Et $\nabla yf \approx f(x,y+1) - f(x,y)$ **(2.5)**

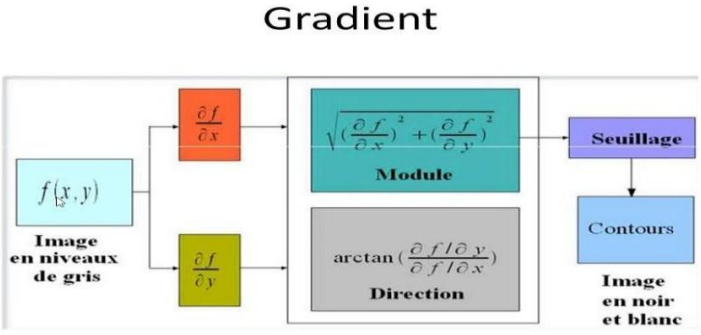

Figure.2.4 synoptique d'une détection de contour par gradient.

Pour obtenir le gradient, on convolue l'image avec des masques dans la logique des formules précédentes. Pour conclure, la méthode dérivative du premier ordre consiste en quatre étapes essentielles :

1. Calcule du gradient en chaque point de l'image
2. Création de l'image de la norme du gradient

3. Extraction des maximas locaux dans la direction gradient
4. Le seuillage

Les masques qui approximent les composantes du gradient sont essentiellement :

- **L'opérateur de sobel :**

L'algorithme de Sobel est une détection de bord basée sur des gradients pour extraire les bords d'une image en niveaux de gris en utilisant la première dérivée. En calculant h horizontal et vertical dérivés de direction d'un pixel contre l'environnement pixels, l'algorithme segmente l'image en zones où objets. Ce processus réduit la quantité de données pendant préserver les propriétés structurelles de l'image. Le Gx et les dérivés Gy représentent les composantes du gradient [12]

Vecteur donné par l'équation 1 :

$$A = (Gx, Gy)$$

L'amplitude du gradient exprime le taux de changement de l'intensité en pixels voisins et définit la force des bords. Un changement soudain de pixels contigus augmente le gradient magnitude résultant en la bordure d'un objet, donnée par Équation 2 :

Amplitude : $A=\sqrt{Gx^2 + Gy^2}$ **(2.6)**

L'équation 2 peut être approchée comme l'équation 3. Cette dernière formule offre un calcul plus rapide, tout en préservant changements relatifs d'intensité.

$A \approx |Gx|+|Gy|$

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \quad et \quad G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \qquad \textbf{(2.7)}$$

Direction :

$$\Theta=\arctan\left(\frac{Gy}{Gz}\right) \qquad \textbf{(2.8)}$$

- **L'opérateur de prewitt :**

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * A \quad et \quad G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * A \qquad (\mathbf{2.9})$$

Amplitude :

$$A=\sqrt{Gx^2 + Gy^2} \quad \textbf{(2.10)}$$

Direction :

$$\Theta=\arctan(\frac{Gy}{Gz}) \quad \textbf{(2.11)}$$

- **L'opérateur de robert :**

1	0
0	-1

G_x

0	1
-1	0

G_y

Amplitude :

$$A=\sqrt{G1^2 + G2^2} \quad \textbf{(2.12)}$$

Direction :

$$\Theta=\frac{\pi}{4} \quad \textbf{(2.13)}$$

2.2.2.2 Méthode dérivative deuxième ordre :

- **Laplacien** :

L'opérateur Laplacien d'une fonction f est en fait :

$$\nabla f = \nabla^2 f = \frac{\partial^2 f}{\partial g^2} + \frac{\partial^2 f}{\partial t^2} \quad \textbf{(2.14)}$$

- **Laplacien de gaussien :(LOG)**

Comme mentionné ci-dessus, les points de bordure détectés en trouvant le zéro croisement de la deuxième dérivée de l'intensité de l'image sont très sensibles au bruit. Par conséquent, il est souhaitable de filtrer le bruit avant l'amélioration des bords. Pour ce faire, le Laplacien de Gauss (LOG), dû à Marr et Hildreth [164], combine Filtrage Gaussien avec le Laplacien pour la détection des bords. Les caractéristiques fondamentales du Laplacien de détecteur de bord gaussien sont :

- Le filtre de lissage est gaussien.

- L'étape d'amélioration est la deuxième dérivée (Laplacien en deux dimensions).
- Le critère de détection est la présence d'un passage à zéro dans le second dérivé avec un pic correspondant dans le premier dérivé.
- L'emplacement des bords peut être estimé à l'aide de la résolution des sous-pixels à l'aide d'une interpolation.

Dans cette approche, une image doit d'abord être entourée d'un gaussien filtre. Ceci lisse une image et réduit le bruit.

Points de bruit isolés et petits les structures seront filtrées. Puisque le lissage entraînera l'étalement des bords, le détecteur de bord ne considère comme bords que les pixels la pente maximale locale. Ceci est réalisé en utilisant zéro croisement du second dérivé. Le Laplacien est utilisé comme l'approximation du second dérivé en 2-D parce que c'est un opérateur isotrope. Pour éviter la détection de bords insignifiants, seuls les croisements zéro dont le premier dérivé correspondant est au-dessus de certains seuils sont sélectionnés comme points de bord. [13]

L'opérateur LOG vise

- D'abord à lisser l'image : par un filtre Gaussien H_{Gauss}
- Détecter les contours : par le placien Δ
- On effectuera donc : $\Delta I \approx I * H_{Gauss} * \Delta = I * \Delta H_{Gauss}$ **(2.15)**
- ΔH_{Gauss} Est l'opérateur LOG

Le calcul du Laplacien d'une Gaussienne donne :

$$\Delta H_{Gauss} = \frac{4}{\sqrt{2\pi\sigma}}\left(\frac{x^2+y^2}{2\sigma^2} - 1\right) e^{\frac{x^2+y^2}{2\sigma^2}} \quad \textbf{(2.16)}$$

- **Différence de Gaussien (LOG) :**

En science de l'imagerie, la différence des Gaussiens (DoG) est une caractéristique d'algorithme améliorée qui inclut la soustraction d'une version floue d'une image originale d'une autre, version moins floue de l'original Lorsque l'on convoque des images théoriginales en niveaux de gris avec des noyaux gaussiens présentant des écarts types différents, on obtient des images floues dans le cas simple des images en niveaux de gris. Les informations relatives au patient à fréquence élevée sont supprimées uniquement lorsqu'une image est floue à l'aide d'un noyau gaussien. Les informations spatiales qui se trouvent entre les plages de fréquences

sont enregistrées lorsque l'image est soustraite de l'autre. Ainsi, la différence de Gaussien un filtre passe-bande où il y a quelques-unes des récurrences spatiales existent dans l'image en niveaux de gris d'origine mais le reste est ignoré. [14]

La différence de Gaussien est ondelette mère de somme totale nulle qui approche l'ondelette chapeau mexicain en soustrayant une gaussienne large d'une gaussienne étroite, comme défini dans cette formule dans le cas d'une dimension :

$$f(x;\mu;\sigma1;\sigma2) = \frac{1}{\sigma1\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)}{2\sigma1^2}^2\right) - \frac{1}{\sigma2\sqrt{2\pi}}\exp\left(-\frac{(x-\mu)^2}{2\sigma2^2}\right) \quad \textbf{(2.17)}$$

Et dans le cas bi-dimensionnel centré :

$$f(u,v,\sigma) = \frac{1}{2\pi\sigma^2}exp^{-(u^2+v^2)/(2\sigma^2)} - \frac{1}{2\pi K^2\sigma^2}exp^{(u^2+v^2)/(2K^2\sigma^2)} \quad \textbf{(2.18)}$$

- **Laplacien de Gaussien vs Différence de Gaussiens :**

Considérez la distribution gaussienne unidimensionnelle :$f(x,\mu,\sigma) = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right.$

avec $-\infty < x < \infty$ et $\sigma > 0$, où μ est la moyenne, σ est l'écart-type et σ^2est l'écart.

Si nous prenons la seconde dérivée de la fonction gaussienne unidimensionnelle en considérant μ = 0, nous obtenons le Ricker wavelet :

$$T(x) = \frac{1}{\sqrt{2\pi\sigma^2}}\frac{1}{\sigma^4}(x^2-\sigma^2)exp - \frac{x^2}{2\sigma^2} \quad \textbf{(2.19)}$$

Le Laplacien de LoG gaussien est une généralisation multidimensionnelle de l'ondelette de Ricker. Pour l'obtenir, il faut prendre le Laplacien bidimensionnel de la distribution gaussienne $LoG(x,y) = -\frac{1}{\pi\sigma^2}\left[1 - \frac{(x^2+y^2)}{2\sigma^2}\right] exp - \frac{x^2+y^2}{2\sigma^2}$ **(2.20)**

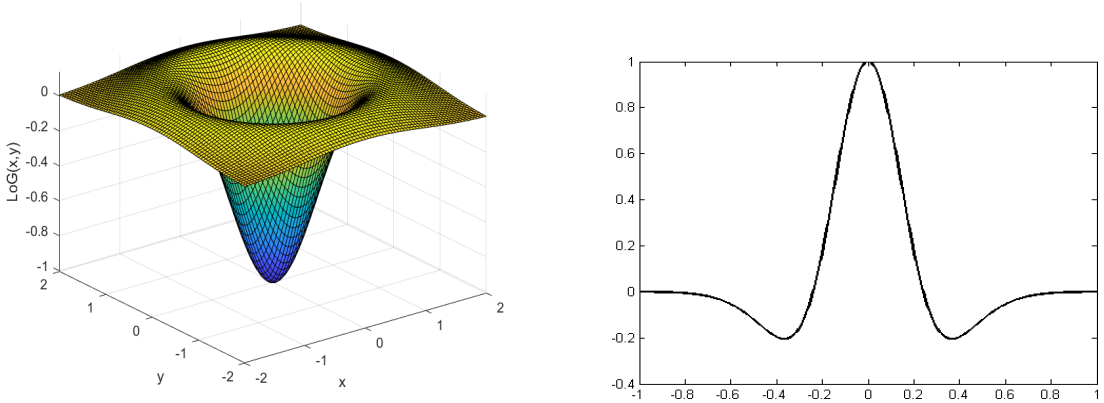

Figure 2.5 LOG de gaussien et DOG

Toutefois, dans la pratique, le Laplacien de Gauss (LoG), figure (a), est approché par le Différence de fonction gaussienne (DoG) puisque cela réduit les coûts de calcul pour deux où Plus de dimensions. Le DoG est obtenu en effectuant la soustraction de deux noyaux gaussiens où un noyau doit avoir un écart-type légèrement inférieur à celui du précédent. Figure (b) Compare la fonction LoG avec $\sigma = 2,5$ avec la fonction DoG en utilisant des noyaux avec2, 5 et $\sigma 2 = 2.15$. La convolution du filtre DoG avec l'image d'entrée génère la détection de bord Pour cette image.

2.2.3 Détecteur optimale de détection de contours (détecteur de Canny) :

Canny Edge detection utilise un filtrage linéaire avec un noyau gaussien pour lisser le bruit, puis calcule la force et la direction des bords pour chaque pixel de l'image lissée. Les pixels de bord candidats sont identifiés comme les pixels qui survivent à un processus d'amincissement appelé suppression non axiale.

Dans ce processus, la force de bord de chaque pixel de bord candidat est réglée à zéro si sa force de bord n'est pas supérieure à la force de bord des deux pixels adjacents dans la direction du gradient. Le seuillage est ensuite effectué sur l'image de magnitude de bord aminci en utilisant l'hystérésis. Dans l'hystérésis, deux seuils de résistance de bord sont utilisés. Tous les pixels de bord candidats en dessous du seuil inférieur sont étiquetés comme non-pixels et tous les pixels au-dessus du seuil bas qui peuvent être connectés à n'importe quel pixel au-dessus du seuil élevé à travers une chaîne de pixels de bord sont étiquetés comme pixels de bord.

Le détecteur Canny Edge ne requiert que l'utilisateur entre trois paramètres. Le premier est sigma, l'écart type du filtre gaussien spécifié en pixels. Le deuxième paramètre bas est le seuil bas qui est spécifié comme une fraction d'un seuil élevé calculé. Le troisième paramètre élevé est le seuil élevé à utiliser dans l'hystérésis et est spécifié comme point de pourcentage dans la distribution des valeurs de magnitude de gradient pour les pixels de bord candidats. [15]

Étapes de l'algorithme du détecteur Canny Edge :

- ✓ Réduction du bruit
- ✓ Détermination du gradient d'intensité de l'image
- ✓ Suppression non maximale
- ✓ Hystérésis Seuil

Les avantages :

- Bonne détection : faible taux d'erreurs dans la signalisation des contours
- Bonne localisation : minimisation des distances entre les contours détectés et les contours réels,
- Clarté de réponse : une seule réponse par contour et pas de faux positifs

2.2.4 Algorithme de détection de contour à base de deep learning

Nous abordons le problème de la détection des contours et les limites des objets dans les images naturelles. Ce problème est fondamental et d'une grande importance pour une variété de Domaines de la vision par ordinateur.

Dans cette section, nous décrivons en détail la formulation de système de détection de contours Nous commençons par discuter les approches fondées sur les réseaux neuronaux connexes, en particulier celles qui mettent l'accent sur l'apprentissage de fonctionnalités multi-niveaux et multi-niveaux.

La tâche de détection des limites des contours et des objets est intrinsèquement difficile. Après des décennies de recherche, ils ont émergé un certain nombre de propriétés qui sont essentielles et qui sont susceptibles de jouer un rôle dans un système réussi :

1. Soigneusement conçu
2. Fusion de réponse à plusieurs échelles
3. Incorporant l'information structurelle
4. Faire des prédictions d'images holistiques (se référant aux approches qui effectuent la prédiction en prenant le contenu global et direct de l'image) et Exploitation géométrie 3D

En raison de la nature de l'apprentissage hiérarchique en profondeur réseaux neuronaux convolutionnels, le concept de multi-échelle et l'apprentissage à plusieurs niveaux peut varier d'une situation à l'autre. Par exemple, l'apprentissage à plusieurs niveaux peut être « à l'intérieur »

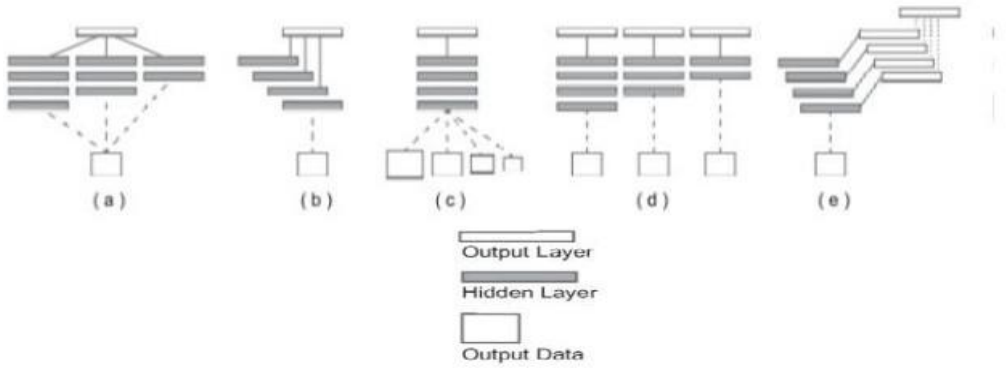

Figure2.6 Illustration de configurations d'architecture d'apprentissage profond

Réseaux intégrés de façon holistique Le réseau de détection de contours holistique que nous avons implémenté dans notre étude est une variante relativement simple qui est capable de produire des prédictions à partir de multiples échelles. L'architecture peut être interprétée comme une version « imbriquée de façon holistique » des « réseaux indépendants » L'architecture comprend un réseau à flux unique avec plusieurs sorties latérales. Cette architecture ressemble à plusieurs travaux antérieurs, en particulier l'approche du réseau profondément supervisé [16] dans laquelle les auteurs montrent que la supervision des couches cachées peut améliorer à la fois l'optimisation et la généralisation pour les tâches de classification des images. Le réseau de HED :

- Se base sur la représentation on multi échelle (multi scale)
- Se base sur la prédiction à partir de la présentation multi échelle

Les multiples sortis latérales nous donnent également la flexibilité d'ajouter une couche de fusion supplémentaire si une sortie unifiée est souhaitée

2.3 Conclusion :

Tout au long de ce présent travail, nous avons essayé de présenter et de cerner les étapes clés de détection de contours. Nous avons détaillé les méthodes dérivatives premier et second ordre ainsi que le détecteur de contours optimisé Canny. Nous avons présenté l'avantage du réseau HED de détection de contours que nous allons implémenter dans nos expériences.

Dans cet article, nous avons développé un nouveau système de détection de bord basé sur un réseau de neurones convolutionnels qui démontre une performance de pointe sur des images naturelles à une vitesse pertinente sur le plan pratique (p. ex., 0,4 seconde en utilisant GPU et 12 secondes en utilisant le processeur). Notre algorithme construit en plus des idées de réseaux neuronaux entièrement convolutifs et filets surveillés en profondeur

Chapitre 3 : Implémentation et résultat

3.1 Introduction

Dans ce Chapitre, nous allons présenter nos résultats de simulation. Plus précisément, nous exposerons les résultats de détection de contours effectués par les algorithmes conventionnels : Masque de Sobel, Laplacian of Gaussian (LoG), Difference of Gaussian (DoG), détecteur optimal de Canny et l'algorithme de détection de contours à base de CNN. Les expériences de simulation des algorithmes conventionnels sont réalisées sous Matlab et celles de l'algorithme à base de CNN sont réalisées sou Python (environnement Anaconda, navigator spyder). Le jeu de données utilisé est constitué de la bibliothèque Toolbox Matlab, images réelles, base de données BSDS500. Une étude comparative qualitative entre les différents algorithmes sera établie après chaque expérience de simulation.

3.2 Jeu de Données Utilisées

Dans notre travail, nous avons utilisée des images niveaux de gris et images couleurs RGB prises de différentes bases de données représentant un Object et/ou des objets sur un fond dans une image. Nous avons également utilisé les images CT à rayon X (Computed tomogrraphy CT). Le tableau 3.1 résume les images test utilisées dans notre étude de simulation. Dans un souci de comparaison, nous avons pris les mèmes images pour tester les algorithmes considérés dans cette étude. Nous avons effectué plusieurs expériences de simulation et vu la limitation de pages, nous avons sélectionné une partie de nos résultats. Nous avons pris des images de la base BSDS500 dataset.

Tableau 3.1 Jeu de données

images	format	Résolution (pixels) (size)	Base de données
Input	jpg	340x511	BSDS500 dataset
covid	jpg	179x282x3	Bibliothèque toobox matlab
IRM	jpg	480x480	BSDS500 dataset
Moon	jpeg	180x211	Bibliothèque toobox matlab

L'ensemble de données de la base de données BSDS500

L'ensemble de données de segmentation de Berkeley (BSDS500) a été proposé pour la segmentation des images et la détection des contours des objets par le groupe de vision par ordinateur de Berkeley. L'ensemble de données contient un total de 500 images, dont 200 images de formation et 200 images de test, et les 100 images restantes de validation. BSDS500 st maintenant le jeu de données le plus utilisé dans le domaine de la détection des contours. La Figure 3.1 illustre une partie des images test que nous avons utilisé.

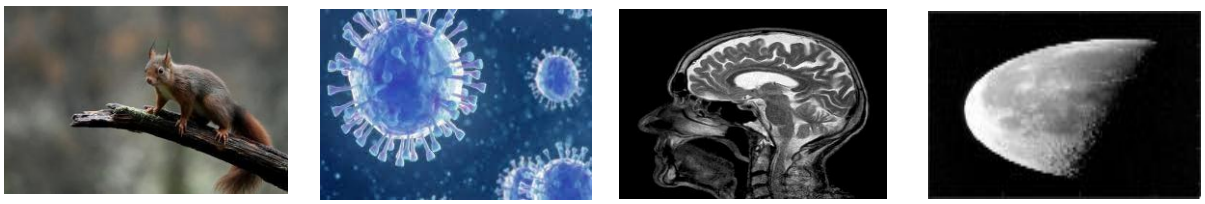

Figure 3.1 Images originales

Pour l'implémentation de l'algorithme à base de CNN, La base utilisée est également BSDS500

Le pourcentage des données d'apprentissage est 80%.

Le pourcentage des données test est 20%.

3.3 Résultats de simulation et Discussions

3.3.1 Algorithmes conventionnelles

3.3.1.1 Méthodes gradient

Expérience 1

Dans cette partie, nous avons implémenté l'algorithme de détection de contours par les masques de Sobel. La Figure 3.2 illustre les étapes d'implémentation que nous avons suivie dans nos expériences.

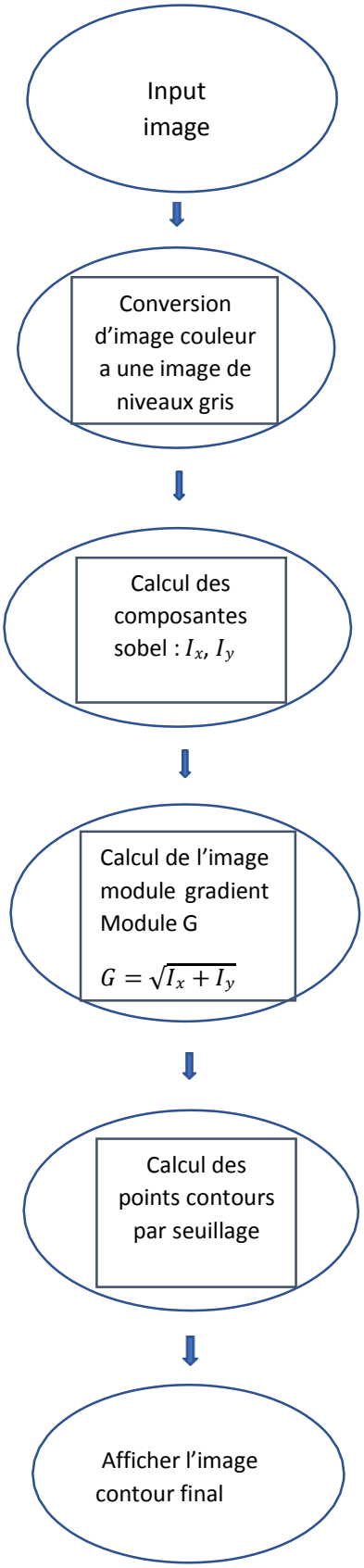

Figure 3.2 Etapes de simulation de détection de contours par Sobel

La première partie de notre étude de simulation consiste à détecter les contours de l'image par la méthode dérivative 1er ordre. Plus précisément, nous utilisons les masques de Sobel suivants,

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Nous avons appliqué l'image test Fig.3.1 input. La Figure 3.3 représente les composants gradient horizontal et vertical I_x, I_y et l'image gradient G respectivement. Remarquons que sur les composantes apparaissent les détails de l'image originale.

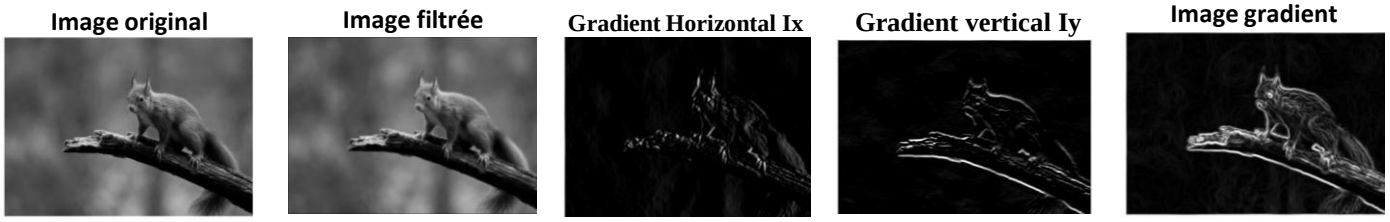

Figure 3.3 Résultats de détection de contours par Sobel ; composantes et module gradient

L'image contours est obtenue par simple seuillage. La valeur du seuil est fixée à la valeur moyenne du niveau de gris après plusieurs tests de simulation.

La Figure 3.4 représente l'image contour finale après seuillage

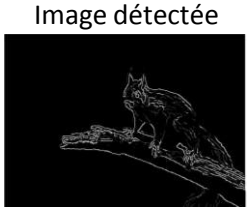

Figure 3.4 Image contour

Après chaque détection, nous avons effectué un seuillage fin pour améliorer la qualité de détection. Ce traitement consiste à éliminer les faux contours. La Figure 3.5 représente l'image contour finale après le seuillage fin.

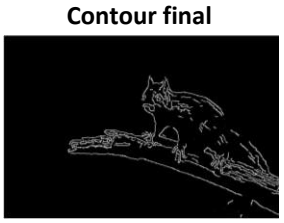

Figure 3.5 seuillage fin

Pour compléter notre étude, nous avons appliqué le masque Sobel sur d'autres images de différentes bases de données. Les Figures 3.6-3.8 illustrent les résultats de simulation. Les remarques de détection ont tendance à se reproduire pour l'ensemble d'images tests.

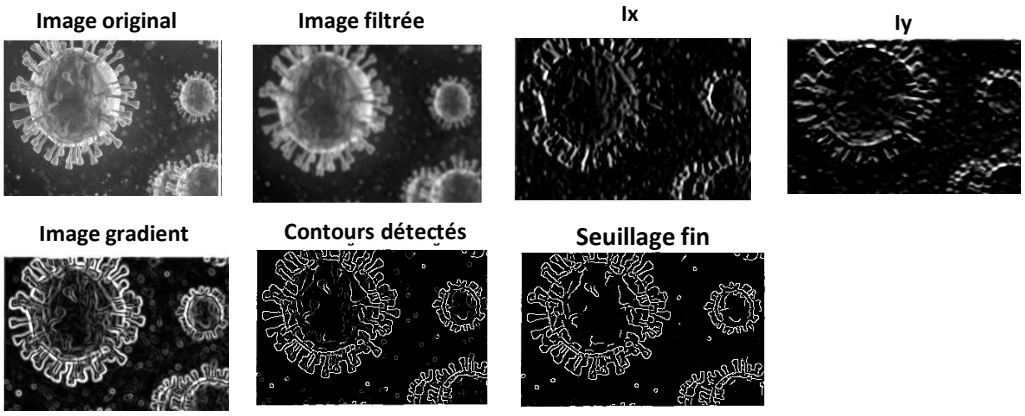

Figure 3.6 contour détecté par Sobel sur image covide

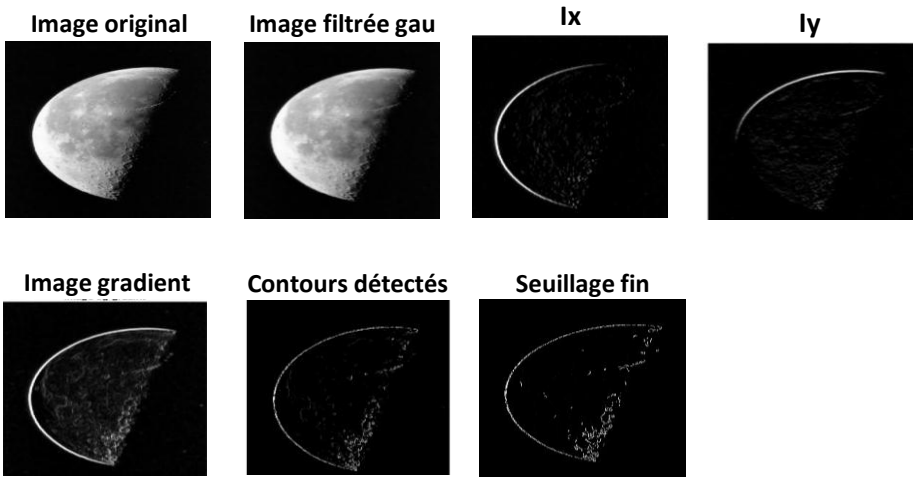

Figure 3.7 contour détecté par Sobel sur image Moon

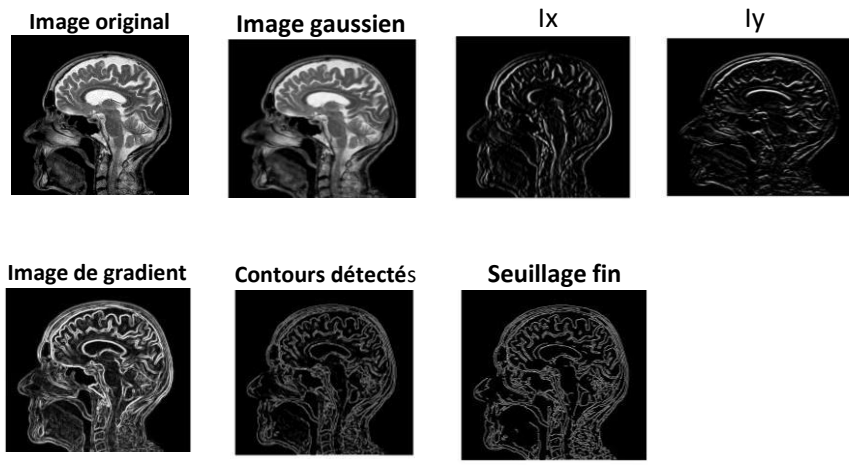

Figure 3.8 contour détecté par Sobel sur image IRM

Expérience2 (détecteur de Canny)

Nous avons appliqué le détecteur optimal Canny sur le même jeu de données. Rappelons pour mémoire que le détecteur de Canny assure la bonne détetion, la bonne localisation et la solution unique de détection. Les Figures 3.9-3.11 représentent les résultats de détection par Canny sur les images tests.

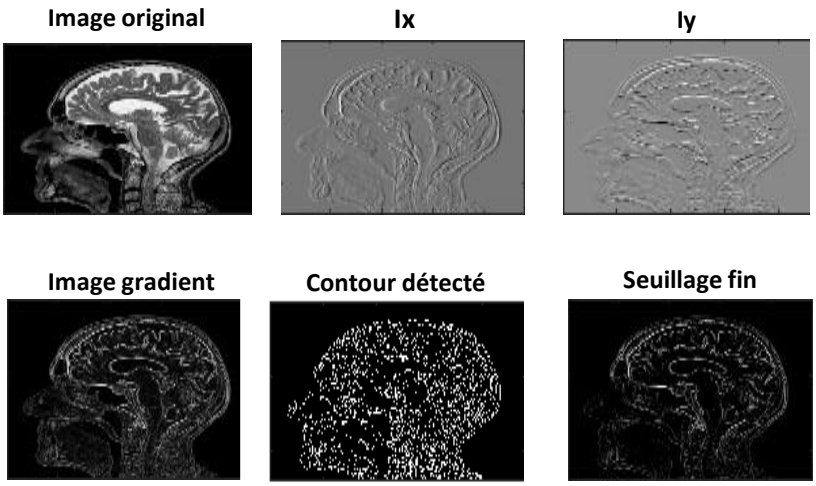

Figure 3. 9 contours détectés par le détecteur Canny sur image IRM

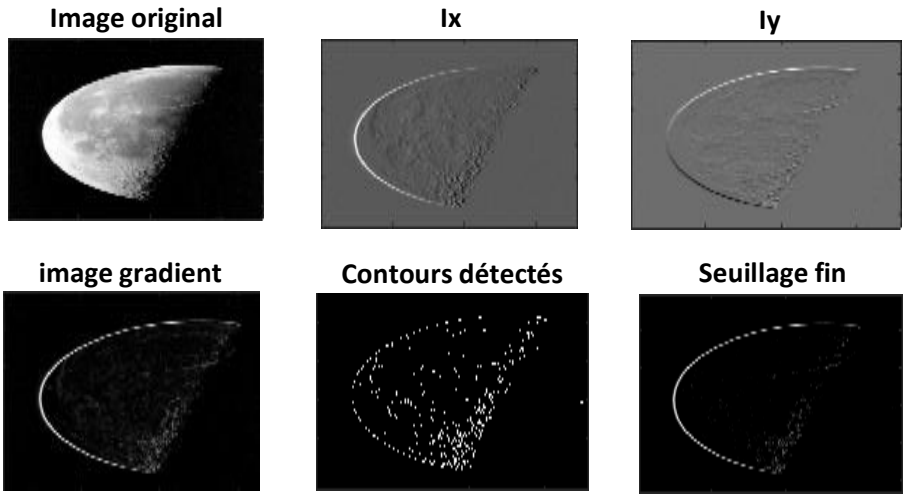

Figure 3.10 contour détecte par algorithme Canny sur image Moon

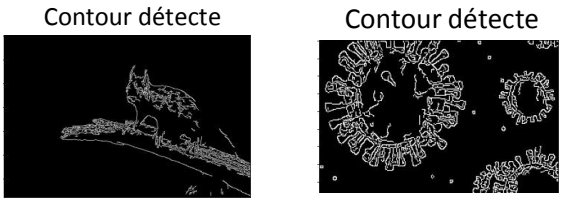

Figure 3.11 contour détecte par le détecteur Canny sur image input et covid

D'après les résultats obtenus, nous pouvons facilement constater que les contours obtenus par Canny sont bien meilleurs par rapport à ceux détectés par Sobel, en termes de fermeture et localisation de contours.

3.3.1.2 Expérience 3 (Méthode laplacienne)

Nous montrons dans cette section les résultats obtenus en utilisant les détecteurs dérivatifs deuxième ordre. Les Figures 3.12, 3.13 et 3.14 représentent les conteurs détectés par les opérateurs de Laplacian of Gaussian LoG et difference of gaussian DoG, sur les images input, covid, Moon et IRM, respectivement. Pour chaque image, nous montrons les deux composantes laplaciennes Ix2 et Iy2, le laplacien et l'image contours finale Nous constatons que ces opérateurs permettent d'avoir des contours fins et cela est leur principal avantage.

- **LoG**

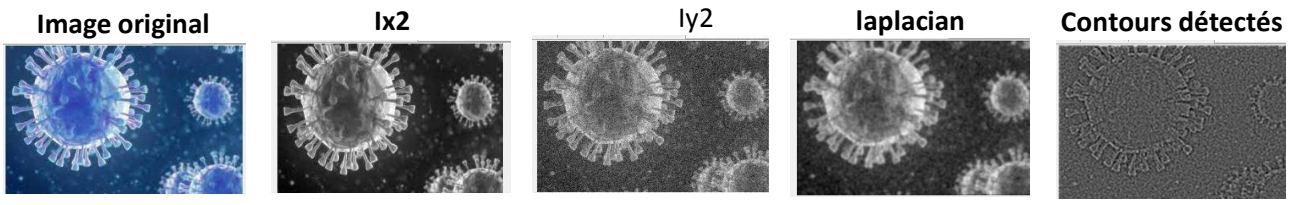

Figure 3.12 contour détecte par filter LoG sur image covide

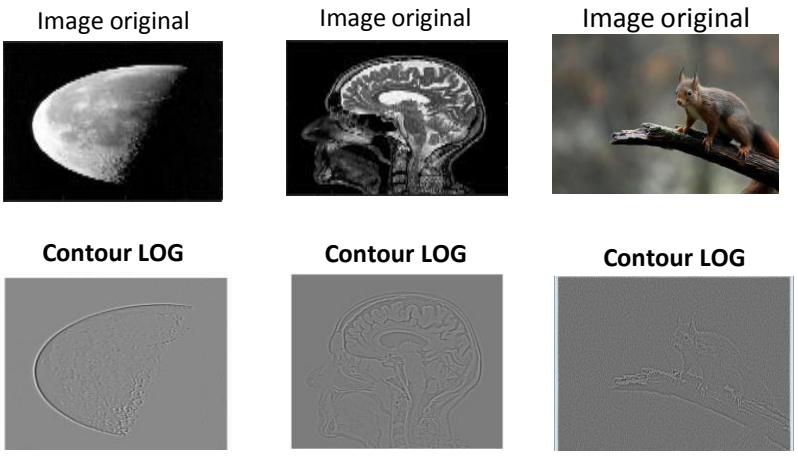

Figure 3.13 contour détecte par filtre LoG sur l'image irm, Moon

- **DoG**

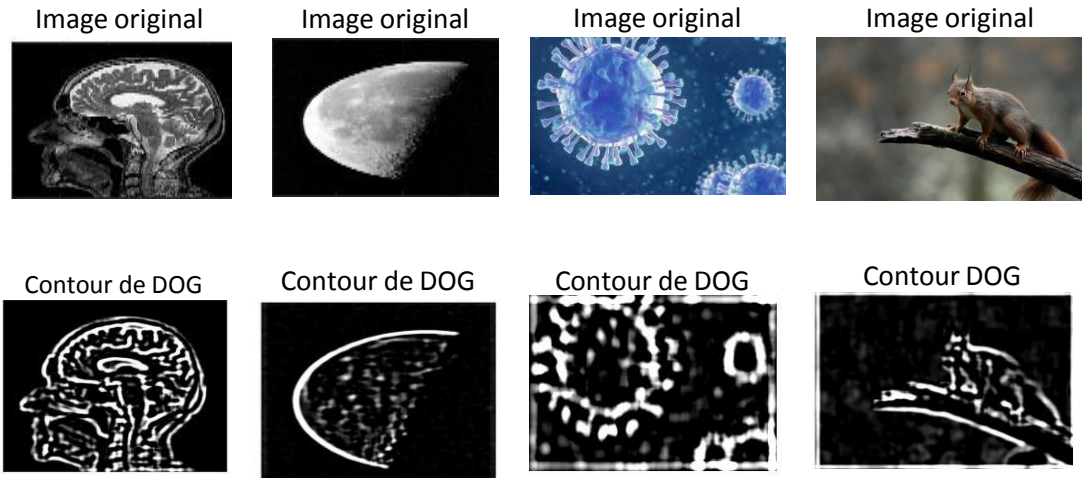

Figure 3.14 contours détectés par filtre DoG sur les images test : input, Moon, covide et IRM

3.3.2 Expérience 4 : Algorithme de détection de contours à base de CNN

Dans cette partie, nous exposons les résultats de détection de contours à base du réseau CNN. Dans un souci de comparaison, nous avons pris les mèmes images tests. Il est à noté que dans cette partie nous avons exécuté les codes sous python sous Anaconda, environnement spyder. L'exécution des codes sous google colab est aussi possible. Avant chaque exécution, nous précisons le nombre d'epochs. Le nombre d'epochs optimal est 100 epochs et ceci pour toutes les images tests que nous avons utilisé. La Figure 3.15 illustre les contours obtenus par CNN. Comparativement aux algorithmes conventionnels, la détection à base de CNN a permis d'obtenir des contours fermés et sans présence de faux contours. Nous remarquons également que les contours obtenus sont épais et ne sont pas fin par rapport aux contours obtenus par LoG et DoG.

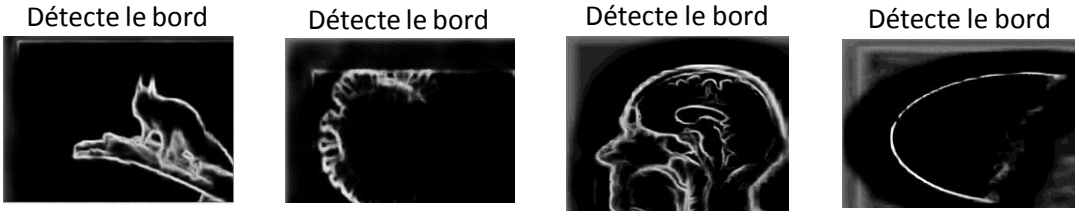

Figure 3.15 contours détectés à base de CNN sur l'image Moon, IRM, covid et input

Pour mieux voir la différence de qualité visuelle entre les différents algorithmes de détection que nous avons considérée, nous les avons exposés sur la Figure 3.16.

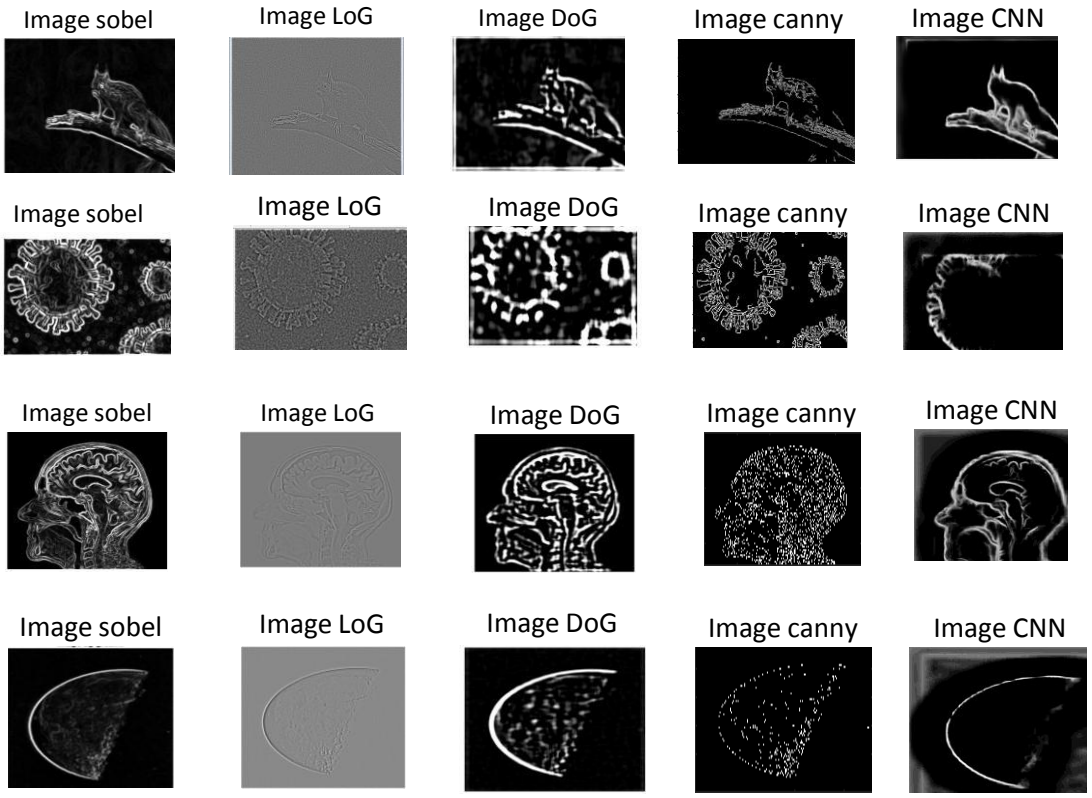

Figure 3.16 Comparaison visuelle des différents algorithmes de détection de contours

3.4 Conclusion

Dans ce chapitre, nous avons appliqué les différents opérateurs conventionnels de détection de contours d'images, à savoir : Sobel, Canny, LoG et DoG. Nous avons ensuite implémenté les détecteurs de contours à base de CNN. Les différents algorithmes sont appliqués sur le même jeu de données dans l'optique de bien juger les performances de détection. La modalité d'image IRM est aussi prise en considération. Nous avons effectué plusieurs expériences de simulation. Pour la partie des algorithmes conventionnels, les codes sont exécutés sous Matlab. Pour la partie de détection de contours à base de CNN, les codes sont exécutés sous python. D'après les résultats obtenus, nous pouvons constater :

- Absence de faux contours sur les images détectés par Sobel, Canny et à base de CNN.
- Contours détectés épais en appliquant Sobel et Canny

- Contours fins en appliquant LoG et DoG
- Présence de faux contours pour LoG et DoG
- Contours fermés pour l'algorithme de détection à base de CNN.

Pour conclure, les résultats de simulation effectuée ont bien confirmé l'étude théorique des algorithmes conventionnels de détection de contours d'images. Aussi, ils ont mis en lumière l'avantage du détecteur de Canny ainsi que la détection à base de CNN qui ont permis d'obtenir des contours bien fermés.

Conclusion Générale

Conclusion Générale

Dans ce travail, nous avons étudié la problématique de la détection des contours dans tout système de vision artificielle. Plus précisément, nous avons implémenté une méthode de détection de contours qui se base sur l'apprentissage profond. Nous avons utilisé plusieurs jeux de données pour tester les performances de la méthode. Pour mieux voir l'intérêt apporté par l'apprentissage profond, nous avons implémenté tout d'abord les méthodes conventionnelles de détection de contours, à savoir : méthodes dérivatives premier et deuxième ordre.

Pour les méthodes premier ordre, nous avons appliqué le masque de Sobel ainsi que le détecteur de Canny qui est un détecteur optimal. Ces détecteurs sont appliqués sur des images niveaux de gris.

Pour les méthodes dérivatives second ordre, nous avons appliqué le masque laplacien.

Nous avons constaté la présence de faux contours, pour les méthodes dérivatives second ordre.

Ensuite, nous avons implémenté la méthode de détection de contours HED qui est basé sur l'apprentissage. D'après les résultats de simulation, nous constatons que les contours obtenus par HED sont bien fermés ce qui traduit l'efficacité de la méthode.

Par ailleurs, plusieurs taches peuvent être complétées à ce travail, tel que les critères quantitatives qui constituent une problématique à part entière.

Références

Références

[1] : Top 15 Deep Learning applications that will rule the world en 2018 and beyond, Vartul Mittal, 3 Oct 2017.

[2] : Djerioui.M, « Contribution au Développement de Systèmes Multicapteurs Intelligents Dédiés à la Surveillance et au Contrôle de la Qualité des Eaux Propres », Thèse de Doctorat en électronique, Université de M'sila, 2019 Mr.

[3] : Mr. Mokri Mohammed Zakaria.MMZ, 2017.Classification des images avec les réseaux de neurones convolutionnels « : mémoire de projet de fin d'étude. Université Abou Bakr Belkaid Tlemcen.

[4] : A. HABBA et O. ISHAK. Année Universitaire 2018/2019. La classification des images satellitaires par l'apprentissage profonde (deep learning). Mémoire de fin d'étude. Université Ahmed Draia - Adrar

[5] : Z.Sellami. 7 juillet 2019. Optimisation du CNN par l'algorithme Génétique pour la Reconnaissance de Visage. Mémoire de fin d'étude. Université Mohamed Khider de Biskra.

[6] : Wei Bao, Jun Yue, Yulei Rao. "A deep learning framework for financial time séries using tacked autoencoders and longshort term memory". Business School, Central South. University, Changsha, China, Institute of Remote Sensing andGeographic Information System, Peking University, Beijing, China.2017.

[7] : Deep Learning. Ian Goodfellow, YoshuaBengio, Aaron Courville. MIT Press. 2016.

[8] : RiccoRakotomalala. « Principe de la descente de gradient pour l'apprentissage supervisé Application à la régression linéaire et la régression logistique ». Université Lumière Lyon 2

[9**]** SAGAR SHARMA. (Sep 6, 2017). Activation Functions in Neural Networks.

[10] : Richmond Alake. (aug 14, 2020), Implementing AlexNet CNN Architecture Using TensorFlow 2.0+ and Keras.

[11] : Z.Sellami. 7 juillet 2019. Optimisation du CNN par l'algorithme Génétique pour la Reconnaissance de Visage. Mémoire de fin d'étude. Université Mohamed Khider de Biskra.

[12]: F. M. Vallina, C. Kohn, and P. Joshi, "Zynq all programmable SoC sobel filter implementation using the vivado HLS tool," 2012. [Online]. Available: https://bit.ly/3h6egD1

[13]: Y. Zheng, "The design of sobel edge extraction system on FPGA," ITM Web Conf., vol. 11, 2017. [Online]. Available: https://doi.org/10.1051/itmconf/20171108001

[14] : S. Wang, W. Li, Y. Wang, Y. Jiang, S. Jiang, R. Zhao, An Improved Difference of Gaussian filter in face recognition,Journal of Multimedia.

[15]: C.-Y. Lee, S. Xie, P. Gallagher, Z. Zhang, and Z. Tu. Deeplysupervised nets. In AISTATS, 2015.

[16]: J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.

RESUME

La détection des contours est la première étape de nombreuses applications de vision par ordinateur. La détection des contours réduit considérablement la quantité de données, filtre les informations indésirables ou sans importance et donne des informations importantes dans l'image. Cette information est utilisée dans le traitement d'image pour détecter les objets. Il y a certains problèmes tels que la détection des bords erronés, les problèmes causés par le bruit, la perte des limites de faible contraste, etc. Nous allons donc comparer différents détecteurs de bord pour déterminer quel détecteur de bord mène à de meilleurs résultats. Le logiciel a été développé en utilisant MATLAB, python, google colab,.. L'opérateur de réfraction modifié a prouvé qu'il donnait un meilleur résultat que les autres détecteurs de bord.

Notre recherche est basée sur des systèmes de détection de bord utilisant d'anciennes méthodes (Method Classique) ou utilisant l'apprentissage profond basé sur les réseaux neuronaux CNN.

Abstract

Edge detection is the first step in many computer vision applications. Edge detection significantly reduces the amount of data, filters out unwanted or unimportant information and gives important information in the image. This information is used in image processing to detect objects. There are some problems such as erroneous edge detection, problems caused by noise, loss of low contrast limits, etc. So we will compare different edge detectors to determine which edge detector leads to better results. The software was developed using MATLAB, python, google colab,.. The modified refraction operator has proven to give a better result compared to other edge detectors.

Our research is based on edge detection systems using old methods (Method Classic) or using deep learning based on CNN neural networks.

ملخص:

اكتشاف الحافة هو الخطوة الأولى في العديد من تطبيقات الرؤية الحاسوبية. يقلل اكتشاف الحافة بشكل كبير من كمية البيانات، ويصفي المعلومات غير المرغوب فيها أو غير المهمة ويعطي معلومات مهمة في الصورة. تستخدم هذه المعلومات في معالجة الصور للكشف عن الأشياء. هناك بعض المشاكل مثل الكشف الخاطئ عن الحافة، والمشاكل الناجمة عن الضوضاء، وفقدان حدود التباين المنخفضة، إلخ. لذلك سنقارن أجهزة الكشف عن الحافة المختلفة لتحديد كاشف الحافة الذي يؤدي الى نتائج أفضل. تم تطوير البرامج باستخدام، Matlab، python، Google colab، ...

اثبت مشغل الانكسار المعدل أنه يعطي نتيجة أفضل مقارنة بأجهزة كشف الحافة الأخرى.

يعتمد بحثنا على أنظمة الكشف عن الحافة باستخدام الأساليب القديمة (ميثود كلاسيك) أو باستخدام التعلم العميق بناءً على الشبكات العصبية التلفافية CNN لذالك سنتعرف على وظائف ومزايا وعيوب كل نظام.