Université de Saint-Quentin-en-Yvelines
Master 1 : Calcul Haut Performance et Simulation

Rapport de projet de la programmation numérique

Réseau de Neurones à Convolution (reconaissance d'images)

Préparé par :

M^r.Khadimou Rassoul DIOP
M^r.Hery ANDRIANANTENAINA
M^r.Said TADJER
M^{elle}.Baya ABBACI

M^r. Mohammed Salah IBNAMAR Encadreur

Année 2019-2020

Table des matières

Introduction

Le cancer du sein représente l'un des en jeux majeurs de la santé publique,en raison du fait qu'il est le cancer le plus fréquent chez la femme et la première cause de mortalité en France et dans le monde. L'objectif de ce projet est de classifier des images de cancer en utilisant un réseau de Neurones à convolution.

Le code que nous allons implémenter aura deux résultats possibles : Cancer detected or No cancer detected. Le diagnostique sera le résultat avec la plus grande probabilité.

1

Définitions et Généralités

1.1 Réseau de Neurones convolutionnels

Convolutional Neural Network (CNN) (réseaux de neurones convolutifs) des réseau de neurones spécialisés pour traiter des données ayant une topologie qui ressemble à une grille. Les entrée comprennent des données de type série temporelle, qui peuvent être considérées comme une grille 1D ,les échantillons à des intervalles de temps réguliers et les données de type image peuvent être considérées comme une grille 2D de pixels. Le nom « réseau de neurones convolutif » indique que le réseau emploie une opération mathématique appelée convolution. La convolution est une opération linéaire spéciale. Les réseaux convolutifs sont simplement des réseaux de neurones qui utilisent la convolution à la place de la multiplication matricielle dans au moins une de leurs couches.
Ils ont de larges applications dans la reconnaissance de l'image et de la vidéo, les systèmes de recom- mandations et le traitement du langage naturel .

1.1.1 L'opération de convolution

La convolution est une opération sur deux fonctions d' argument réel. Supposons que nous suivons l'emplacement d'un objet avec un capteur. Notre capteur fournit une sortie x(t) qui est la position du l'objet au moment t. x et t sont des réelles, donc on peut obtenir une lecture différente du capteur à tout moment.

Supposons maintenant que notre capteur soit un peu bruité. Pour obtenir une estimation moins bruitée de la position du notre objet, nous aimerions combiner plusieurs mesures. Donc nous voulons que ces mesures soient une moyenne pondérée et donner plus de poids aux mesures récentes. Nous pouvons le faire avec une fonction de pondération w(a).

On appliqe cette opération de moyenne pondérée à chaque instant et nous obtenons une nouvelle fonction qui fournit une estimation e de la position du l'objet :

$$s(t) = \int_a^b x(a)w(t-a)da \tag{1.1}$$

Cette équation qu'on obtient est l'equation de la convolution. L'opération de convolution est généralement présentée de cette maniére :

$$s(t) = (x * w)t \tag{1.2}$$

Dans les réseaux convolutionnels, le premier argument (dans notre exemple, la fonction x) de la convolution est appelé 'INPUT' (entrée)et le second argument (dans cet exemple, la fonction w) comme 'KERNEL' (noyau).
La sortie est appelée 'OUTPUT' (sortie) et parfois 'FEATURE MAP'.

1.1.2 Couche convolutif

La couche de convolution est le bloc de construction de base d'un CNN. Trois paramètres permettent de dimensionner le volume de la couche de convolution : la profondeur, le pas et la marge.

1. **Profondeur de la couche :**
   Nombre de noyaux de convolution (ou nombre de neurones associés).
2. **Le pas :**
   Contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
3. **La marge (à 0) ou zero padding :**
   La taille de ce 'zero-padding' est le troisième hyper paramètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée.

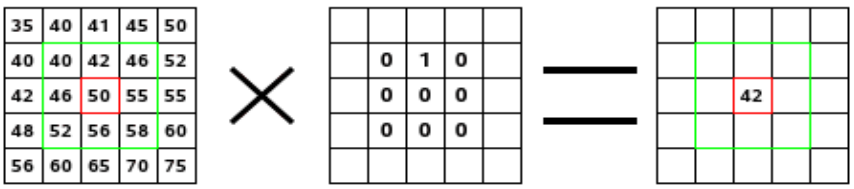

FIGURE 1.1 – Exemple d'une convolution 2D

1.1.3 Couche de pooling (POOL) :

Le pooling est un autre concept trés important des CNNs, ce qui est une forme de sous-échantillonnage de l'image. L'image d'entrée est découpée en une série de rectangles de n pixels de côté ne se chevauchant pas (pooling).

Le pooling réduit la taille spatiale d'une image intermédiaire, réduisant ainsi la quantité de paramètres et de calcul dans le réseau.

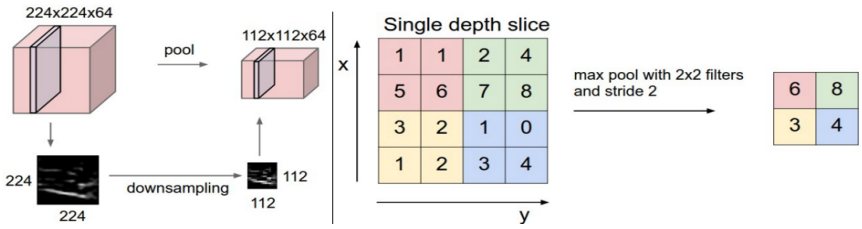

FIGURE 1.2 – Exemple d'une couche de pooling

1.1.4 Couches de correction (ReLU)

Pour améliorer l'efficacité du traitement on intercale entre les couches de traitement une couche qui va opérer une fonction mReLU sur les signaux de sortie :

$$F(x) = max(0, x) \tag{1.3}$$

Cette fonction force les neurones à retourner des valeurs positives.

1.2 Exemple d'un modèle de CNN

Un CNN se trouvre sous plusieurs forme mais La forme standard et commune d'une architecture CNN empile quelques couches Conv-ReLU, les suit avec des couches Pool, et répète ce schéma jusqu'à ce que l'entrée soit réduite à une taille suffisamment petite.
À un moment, il est fréquent de placer des couches entièrement connectées (FC)qui est reliée directement vers la sortie. Ici on a quelques architectures CNN communes :

- ⋆ INPUT -> CONV -> RELU -> FC
- ⋆ INPUT -> [CONV -> RELU -> POOL] * 2 -> FC -> RELU -> FC
  Ici, il y a une couche de CONV unique entre chaque couche POOL
- ⋆ INPUT -> [CONV -> RELU -> CONV -> RELU -> POOL] * 3 -> [FC -> RELU] * 2 -> FC
  Ici, il y a deux couches CONV empilées avant chaque couche POOL.

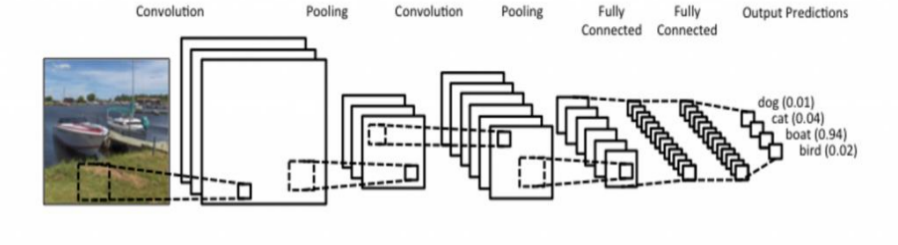

FIGURE 1.3 – Exemple d'un modèle de CNN

2

Pré-processing des images d'entrée

Dans cette partie, on exposera les différentes étapes sur les traitements des images avant de les utiliser pour entraîner notre réseau de Neurone. Ce traitement sera une grande aide dans nos travaux car ce la facilitera la compréhension pour le réseau de Neurone.

2.1 Conversion des images

Une image numérique est définie en pixel composant l'image en hauteur et en largeur. En général, on peut distinguer les images en deux sortes de catégorie. Les images matricielles qui sont composées d'un tableau de points à plus dimensions et les images vectorielles dont le principe est de représenter les données de l'image par des formules géométriques.

2.1.1 Conversion du format des images

Ils existent plusieurs formes d'extensions d'image comme jpeg, gif, png ... a l'origine, l'extension des images du cancer pour entraîner notre réseau de neurone sont tous en png.
Pour faire des tests sur notre réseau de neurone on a besoin des images en extension pgm. Pour convertir les image en pgm, nous avons exécuter des commandes sur le terminal.

2.1.2 Conversion des images en Grayscale

La conversion des images en Grayscale permettra de bien appliquer les filtres aux images. Les images sont changes en noir et blanc grâce à la fonction rgbengrayscale que l'on a implémenté.
Cette fonction reçoit une image rgb. Elle parcourt tous les pixels de l'image et modifie les pixels en niveau de gris. Dans notre code on prend le niveau de gris en blue qui correspond au b de rgb pour avoir un niveau de gris plus sombre. Les deux traitements ci-dessus, nous permettront de détecter les bords des images pour pouvoir entraîner notre réseau de neurone.

2.2 Détection de bord

Dans une image en niveaux de gris, un changement brutal de la valeur car- actérise un contour. Le but de l'opération est de transformer cette image en une autre de mêmes dimensions dans laquelle les contours apparaissent par convention en blanc sur fond noir. Les contours sont les lieux où on trouve les variations significatives de l'information. Pour la détection des bords, nous avons procédé à l'implémentation de fonctions de filtre tel que celui de Sobel, Kirsch et Prewitt.
Ces filtres utilisés ici se caractérisent par une grande rapidité et un faible coût, du fait de leur aspect local. Dans notre cas on remarque que le filtre de Prewitt est beaucoup plus adapté pour la détection de bords sur les images avec cancer. Vous pouvez remarquer par vous même en observant les résultats obtenus après l'application de chaque filtre. Ceci est sans doute dû au fait que les tumeurs ont des formes plus ou moins arrondies.

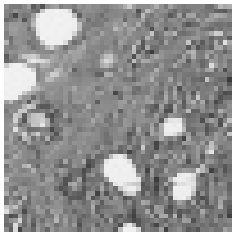

FIGURE 2.1 – Image avant l'application du filtre

2.2.1 Filtre de Sobel

Il s'agit d'un des opérateurs les plus simples qui donne toutefois des résultats corrects. Son implémentation est simple par conséquent. Pour faire simple, l'opérateur calcule le gradient de l'intensité de chaque pixel. Ceci indique la direction de la plus forte variation du clair au sombre, ainsi que le taux de changement dans cette direction.
On connaît alors les points de changement soudain de luminosité, correspondant probablement à des bords, ainsi que l'orientation de ces bords. Le filtre de Sobel calcule une approximation assez inexacte du gradient d'intensité, mais cela suffit en pratique dans beaucoup de cas. En effet, il n'utilise qu'un voisinage (généralement de taille 33) autour de chaque point pour calculer le gradient, et les poids utilisés pour le calcul du gradient sont entiers.

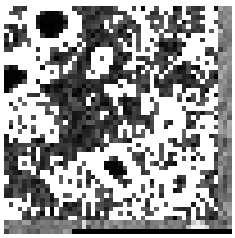

FIGURE 2.2 – Image après l'application du filtre de Sobel

2.2.2 Filtre de Kirsch

Le filtre de kirsch est un détecteur de bord non linéaire qui trouve la force de bord maximale dans quelques directions prédéterminées. Contrairement à l'implémentation de Sobel et Kirsch, ce filtre a un paramètre en moins qui est le threshold c'est à dire La valeur du seuil est choisie empiriquement pour obtenir le meilleur compromis entre la suppression de bruit et la conservation des contours. Ces défauts peuvent être compensés par des filtrages ultérieurs relativement simples mais dont l'enchaînement est souvent délicat.

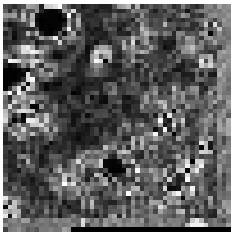

FIGURE 2.3 – Image après l'application du filtre de Kirsch

2.2.3 Filtre de Prewitt

Il s'apparente à l'implémentation du filtre de Prewitt à la différence près que le filtre de Prewitt utilise un lissage rectangulaire alors que Sobel utilise un lissage triangulaire. En terme simple, le filtre calcule le gradient d'intensité lumineuse de l'image à chaque point, donnant la direction et le taux de la plus grande décroissance.
Le résultat nous indique les changements abrupts de luminosité de l'image et donc exhibe les contours probables de celle-ci. En pratique cette technique est plus fiable et facile à mettre en oeuvre qu'un algorithme plus direct.

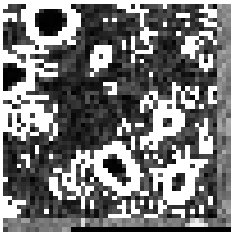

FIGURE 2.4 – Image après l'application du filtre de Prewitt

3

Entrainement du réseau de Neurones

3.1 Poids et fonction d'activation

La phase d'entraînement est la phase la plus importante du réseau de neurones à convolution. C'est la phase qui permet au réseau de neurones d'apprendre et de reconnaître les résultats qu'on lui présente. Dans le cadre de notre projet on cherche à détecter la présence d'un cancer du sein dans nos images. Pour ce faire on commence par donner à notre réseau de neurones des images filtrées obtenues grâce à la détection des bords appliquée pendant le pré-traitement. Cela permet au réseau de neurones de faire la détection plus facilement.
Pour l'implémentation de la fonction d'entraînement on a choisi pour les poids des couches d'entrée 2500 car on a des images 50x50 pixels et 100 pour les poids des hidden layers ou couches cachées. Ces hidden layers forment la boîte noire du réseau de neurones.
C'est grâce au calcul de convolution entre ces poids qu'on obtient les pourcentages affectés à l'output pendant la phase de test. Les pourcentages que l'on obtient sont des grands nombres que l'on va essayer de traduire en probabilités. D'où le besoin d'implémenter des fonctions d'activation telles que sigmoïd et sigmoïd derivative. La fonction sigmoïd nous permet de transformer les pourcentages en probabilités entre 0 et 1. Si on a une valeur négative le pourcentage se rapproche de 0 et si on a une valeur positive le pourcentage se rapproche de 1.

3.2 Gradient de descente

Notre réseau de neurones nous renvoie beaucoup d'erreurs aux premiers tests. Ainsi on a besoin de lui appliquer un algorithme pour réduire le taux d'erreurs. D'où l'implémentation de l'algorithme du gradient de descente. Cet algorithme nous permet de minimiser l'erreur afin d'avoir des résultats plus ou moins corrects. Pour ce faire on utilise la dérivée de la fonction sigmoïd vue précédemment afin de casser plus l'erreur et d'aller la retirer dans les poids entre les diffférents neurones. La phase d'entraînement permet ainsi d'affiner les poids entre les couches en appliquant le back-propagating sur l'erreur de classification. Ainsi on entraîne notre réseau de neurones jusqu'à ce qu'on ait une erreur acceptable (<0.1).

4

Test du réseau de Neurones

4.1 Classification d'images

La phase de test est la phase finale de l'implémentation du réseau de neu- rones. Elle permet de vérifier la justesse de notre output. Ayant déjà classifié nous même les images selon qu'elles sont cancéreuses ou pas.
Le test de notre réseau de neurones se fait avec des images sans présence de cancer car il n'est pas recommandé de tester le réseau de neurones avec les images utilisées pour l'entraînement pour une question de précision et nos outputs sont "Cancer not detected" et "Cancer detected". Ainsi si on a une probabilité proche de 1 cela nous révèle que le résultat est correct qu'on a pas détecté de cancer et si la probabilité est proche de 0 cela révèle la présence de cancer.

4.2 Evaluation des mauvaises classifications

Il s'agit de compter le nombre de fois où le réseau de neurones nous donne des résultats incorrects. Ainsi on stocke ce nombre d'erreurs obtenu après plusieurs tests. Puis on le traduit en pourcentage. Plus le réseau de neurones nous donne des erreurs moins il est efficace et plus on a besoin de l'entraîner. Et plus on l'entraîne moins il fait des erreurs. A la fin on a un réseau 10de neurones à convolution qui sait classifier des images selon qu'elles sont tumorales ou pas.

Perspectives et conclusion

Nous avons réussi à coder un réseau de neurones qui marche c'est à dire qu'il nous donne des résultats. Cependant on peut encore l'améliorer pour avoir par exemple le nombre de fois où on a des mauvaises classifications ce qu'on a pas réussi à faire à cause de l'architecture du code de test.

Ainsi c'est pas faute d'essayer mais on a pas réussi à le faire et il nous restait peu de temps pour demander l'intervention de l'encadreur. Par ailleurs dans la fonction d'entraînement le code devient extrêmement lent dès qu'on dépasse les 100 itérations sur la boucle qui minimise l'erreur ce que l'on pourrait améliorer.

En conclusion on peut dire que l'objectif est atteint à savoir avoir un réseau de neurones qui fait la classication des images de cancer du sein mais on pourrait encore l'améliorer.

Organisation du travail

Le travail s'est fait essentiellement ensemble et de manière progressive.

Tout d'abord, notre encadreur M^r. Mohammed Salah IBNAMAR nous a expliqué notre projet et ses objectifs et surtout son importance dans la vie active.

On se rencontrait des fois pour voir l'avancement du travail et s'aider entre nous, envoyer notre travail a notre encadreur afin de l'améliorer en corrigeant nos erreurs. Et c'est dans ce cadre, que nous avons généré les différents codes :

- ⋆ Le code pour convertir les images en noir et blanc.
- ⋆ Le code de la détection des bords des images.
- ⋆ Le code de réseau de Neurones.

Vers la fin, il a fallu se partager le travail afin d'être dans le temps. Certains se sont concentrés sur la recherche de la documentations, l'utilisation de Github, sur la programmation en C et d'autre sur Latex pour la rédaction du rapport.

Les outils utilisés :

La programmation des différents codes pour la détection du cancer de notre projet est faite en langage C.

Pour la rédaction du rapport on s'est servit du Latex, qui est l'un des meilleur langage et un système de composition de documents.

On a également utiliser Github, la ou on a créé notre organisation qu'on a nommée : Artificial-Breast-Cancer-Detection, a partir de laquelle on a fait nos branche afin que chacun travaille de son coté et puisse avancer sans toucher a la branche principale qui est par défaut la branche master.

Voir le lien : https ://github.com/Artificial-Breast-Cancer-Detection/CNN_ABCD

Références bibliographique

[1] https ://www.kaggle.com/paultimothymooney/breast-histopathology-images

[2] https ://www.sciencedirect.com/science/article/pii/S2405959518304934

[3] https ://fr.wikipedia.org/wiki/D