Master Calcul Haut Performance et Simulation

Rapport de projet

Thème :

Réseau de Neurones à Convolution (reconnaissance d'images)

Préparé par :

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

M^{r}. Mohammed Salah IBNAMAR Encadreur

Année 2019-2020

Table des matières

Introduction

Le cancer du sein est l'une des maladies les plus fréquentes chez les femmes et elle est considérée la première cause de mortalité dans le monde. Et donc elle représente l'un des majeurs enjeux de la santé publique.

Ce qui rend les études qui concernent le cancer du sein occupent une place de plus en plus importante. Et conduit les chercheurs et les spécialistes du domaine à se pencher sur de nouvelles technologies autres qu'humaines dans le but de remédier à cette maladie.

L'objectif de ce projet et d'implémenter un code capable de classifier des images de cancer en utilisant un réseau de neurones à convolution. Ce code aura deux résultats possible : Cancer detected ou Cancer not detected.

On va commencer notre travail avec un petit chapitre qui contiendera quelques définitions et notions de base sur les réseaux de neurones.

Dans le deuxième chapitre on expliquera les différentes étapes de l'implémentation.

Et dans le but d'améliorer les performances du code on va utiliser des outils de parallélisation, et les expliquer dans le troisième chapitre.

Dans le quatrième chapitre, on va tester le code avec des machines différentes et on fera une études globale sur les performances.

A la fin on achèvera le travail avec une conclusion générale et quelques perspectives.

1 Définitions et Généralités

Dans ce chapitre on va voir quelques définitions et généralités des réseaux de neurones. On va donc introduire les notions de base de ces réseaux, et pour cela on va commencer par les définir, et on présetera leurs structure par la suite.

1.1 Réseau de neurones convolutionnels

1.1.1 Définitions

Définition 1.1 ***Qu'est-ce qu'un neurone ?***

Un neurone est une cellule qui permet d'effectuer la communication dans un système en assurant la transmission des signaux. [1]

Définition 1.2 ***Qu'est-ce qu'un réseau de neurone artificiel ?***

Les réseaux de neurones artificiels sont un moyen de modélisation des mécanismes d'apprentissage et de traitement des informations que le cerveau humain établie. [2]

Définition 1.3 ***Qu'est-ce qu'un réseau de neurone convolutif ?***

Les réseaux de neurones convolutifs sont l'un des types des réseaux de neurones artificiels, qui ont de larges applications dans la reconnaissance de l'image et de la vidéo, en utilisant la convolution. [3]

Définition 1.4 ***C'est quoi la convolution ?***

La convolution est un outil mathématique, plus exactement une opération linéaire qui réalise les opérations d'addition et de multiplication. Et elle représente le coeur du réseau de neurones a convolution car elle effectue de la retouche d'image. [4]

1.1.2 La structure des réseaux de neurones

Les réseaux de neurones sont construits d'une succession de couches, pour les réseaux de neurones a convolution on a des couches convolutifs. [5]

Définition 1.5 ***Couches convolutifs :***

Les couches de convolution représentent le bloc de construction de base d'un réseau de neurones a convolution, et le volume de ces couches de convolution est dimensionné en se basant sur 3 critères :

- *La profondeur de la couche, qui indique le nombre de noyaux de convolution.*
- *Le pas qui contrôle le chevauchement des récepteurs, et de plus il sera petit plus le volume de sortie sera grand.*
- *La marge qui permet de contrôler la dimension spatiale du volume de sortie, et ça serait mieux qu'elle soit égale a celle du volume d'entrée.*

On peut représenter les éléments d'une couche d'un réseau de neurones a convolution par la figure si dessous. [5]

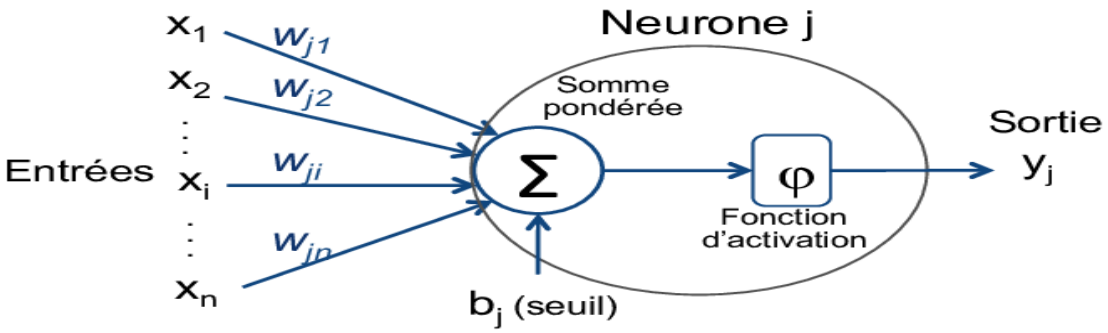

FIGURE 1.1 – Structure d'un neurone artificiel

Comme la figure nous le montre, une couche est constituée de 4 éléments principaux :

- **Les entrées** x_i **:** qui proviennent soit des couches précédentes du réseau de neurones
- **Les poids** w_i **:** qui détermine l'influence de chaque entrée.
- **La fonction** b_j **:** représente la fonction qui combine les entrées avec leurs poids.
- **LA fonction** y_j **:** c'est la fonction de transfert qui calcule la sortie.

2

Implémentation d'un réseau de neurones

Dans ce chapitre on va essayer d'implémenter un code d'un réseau de neurones qui sera capable de détecter le cancer du sein. Ce code aura deux résultats possibles :

- Cancer detected
- No cancer detected

Le diagnostique sera le résultat avec la plus grande probabilité obtenue.

2.1 Pré-processing des images d'entrée

Dans cette partie, on exposera les différentees étapes sur les traitements des images avant de les utiliser pour entraîner notre réseau de neurone. Ce traitement facilitera la compréhension pour le réseau de neurone.

Pour avoir un réseau de neurones à convolution fonctionnel, on aura besoin de plusieurs bases de données pour l'entraîner. Les données que nous allons traiter dans notre projet sont des images de cellules, qui se divisent en deux catégories, cellules saines et cellules malades.

Une image numérique est définie en pixels composant l'image en hauteur et en largeur. En général, on peut distinguer les images en deux sortes de catégories. Les images matricielles qui sont formées d'un assemblage de points nommés pixels et les images vectorielles dont le principe est de représenter les données de l'image par des formules géométriques.

Ils existent plusieurs formes d'extensions d'image comme jpeg,gif,png ...

2.1.1 Conversion des images en Grayscale

La première couche de notre réseau de neurones consiste à stocker l'image en binaire. Les images d'entrée de notre projet sont en couleur, afin de bien détecter les contour d'une image on va les convertir en grayscale. [6]

Cette conversion transforme les images qui sont en couleurs en gris. Après la réception de l'image, elle parcourt tous les pixels de l'image et les modifie en niveau de gris. En prenant le niveau de gris en bleu qui correspond au b de rgb pour avoir un niveau de gris plus sombre.

FIGURE 2.1 – Image en couleur avec l'extension .ppm converti en grayscale [6]

2.1.2 Détection des bord

Cette partie concerne la deuxième opération dans le réseau de neurone. Dans laquelle commencent les différents calculs de convolution entre l'image d'entrée et les différentes filtres.
Pour voir les changements des valeurs sur une image on a besoin de faire une détection des contours. Car ce sont les lieux où on trouve les variations significatives de l'information. Cette opération permet de transformer l'image d'entrée à une image sur fond noir de même dimension que l'image d'entrée.
Ils existent plusieurs variantes de filtre mais nous avons procédé à l'implémentation des fonctions de trois filtres : Sobel, Kirsch et Prewitt. Ces filtres se caractérisent par une grande rapidité et un faible coût, du fait de leur aspect local.

a) Filtre de Sobel :

L'opération du filtre de Sobel peut être décrit sur l'opérateur calcul des gradients de l'intensité de chaque pixel. Ceci peut être indiqué par les fortes variations du clair au sombre. Et c'est ce changement qui indique les contours des images.
L'opération de convolution avec le filtre de Sobel s'effectue sur une matrice de taille 3*3, celui-ci nous donne les approximations des dérivées horizontales et verticales.

L'image ci-dessous montre le résultat de convolution d'image avec le filtre de Sobel. Sur laquelle on peut voir les différents contours qui donnent la meilleure caractérisation de l'image.

FIGURE 2.2 – L'image après l'application du filtre de Sobel [6]

b) Filtre de Prewitt :

Le filtre de Prewitt s'apparente à l'implémentation du filtre de Sobel à la différence près que le filtre de Prewitt utilise un lissage rectangulaire alors que Sobel utilise un lissage triangulaire.
Le filtre de Prewitt est aussi composé de deux matrices 3*3. On va utiliser cette matrice pour une convolution avec l'image d'entrée pour calculer une approximation de sa dérivée en tout point.

L'image ci-dessous montre le résultat de convolution d'image avec le filtre de Prewitt.

FIGURE 2.3 – L'image après l'application du filtre de Prewitt [6]

c) Filtre de Kirsch :

Le filtre de Kirsch est un détecteur de contours non linéaire qui trouve la force de contour maximale dans quelques directions prédéterminées. Mathématiquement, le filtre de Kirsch prend une masque à noyau unique et le fait tourner par incrément.

L'image ci-dessous montre le résultat de convolution d'image avec le filtre de Kirsch.

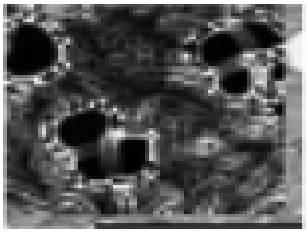

FIGURE 2.4 – L'image après l'application du filtre de Kirsch [6]

L'utilisation de ces filtres facilitent l'apprentissage du réseau de neurone ainsi que de faire les tests. Chaque filtre à ces propres qualités et les résultats obtenus dépendent de la qualité des images d'entrée.

2.2 Entraînement du réseau de neurones

Dans cette partie on va parler de l'entraînement du réseau de neurones à convolution. Qui représente la phase la plus importante, car c'est la phase qui permet au réseau de neurones d'apprendre et de reconnaître les résultats qu'on lui présente.

2.2.1 Training set

Le jeu de données d'entraînement est un dossier avec des images de cancer du sein comme le montre la figure suivante :

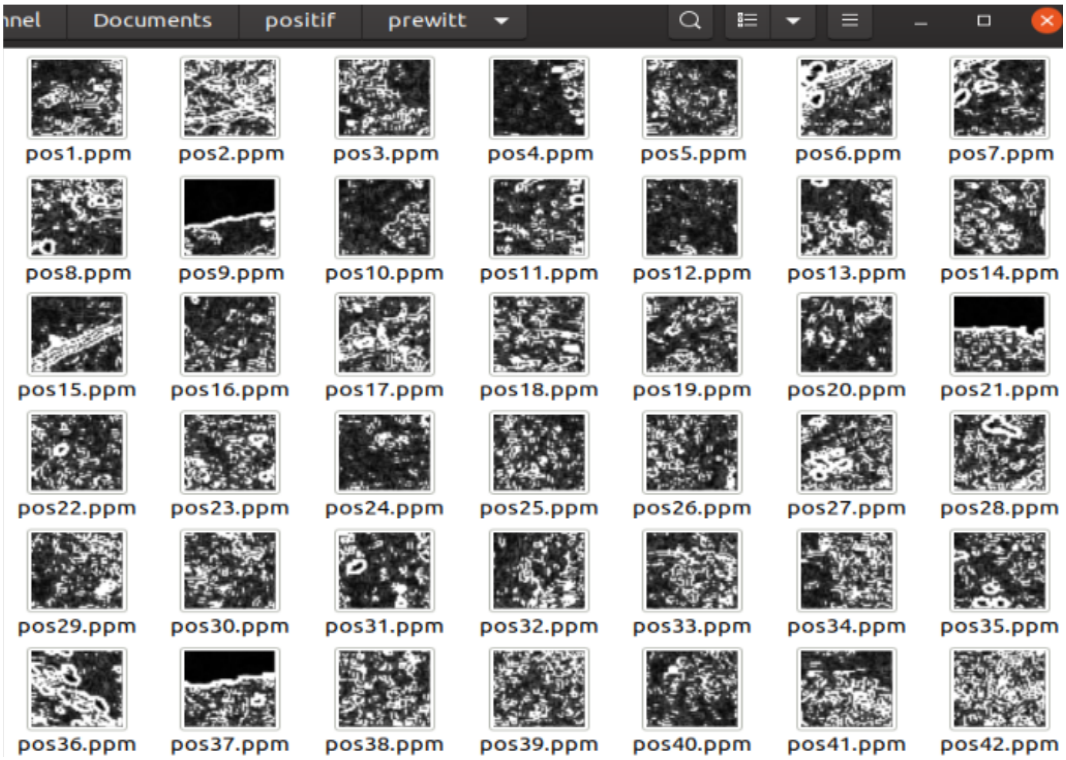

FIGURE 2.5 – Images de cancer du sein [6]

2.2.2 Choix du langage de programmation

La plupart des implémentations de réseau de neurones sont écrites dans des langages de haut niveau utilisant de puissantes bibliothèques mathématiques telles que numpy, tensorflow ... Bien que le code écrit dans ces langages peut être propre et lisible pour n'importe qui, il peut être difficile de saisir les détails derrière la back-propagation par exemple lorsque des opérations matricielles complexes sont regroupées en une seule déclaration.

C'est pour cela on s'est imposé d'écrire les boucles en C, ce qui favorise le fait que le calcul derrière les concepts devient beaucoup plus clair.

2.2.3 Poids et fonction d'activation

Après avoir passé à notre réseau de neurones des images filtrées obtenues grâce aux fonctions de détection des bords appliquées 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 les poids des neurones d'entrée et ceux des neurones dans les hidden layers qu'on obtient les valeurs affectées à l'output pendant la phase de test.

Les valeurs 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 valeurs en probabilités entre 0 et 1. Si on a une valeur négative la valeur se rapproche de 0 et si on a une valeur positive la valeur se rapproche de 1.

```
//Sigmoid derivative
float d_sigmoid(float x)
{
    return x * (1 - x);
}

//fonction sigmoid
inline float sigmoidbis(float x)
{
    return 1.0 / (1.0 + exp(-x));
}
```

FIGURE 2.6 – Capture du code des fonctions d'activation

2.2.4 Gradient de descente et back-propagation

Le 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.

En ce qui concerne l'algorithme de back-propagation, bien qu'il existe de nombreuses variantes, nous avons choisi la descente de gradient stochastique (SGD). Dans ce cas, les poids sont mis à jour sur la base d'une seule paire d'entrées/sorties attendues. Il était plus facile de conceptualiser l'algorithme avec cette approche. 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 foncion sigmoïd vue précédemment afin de casser plus l'erreur et d'aller la retirer dans les poids entre les différents neurones.

La phase d'entraînement permet ainsi d'affiner les poids entre les couches en appliquant la back-propagation sur l'erreur de classification. Ainsi on entraîne notre réseau de neurones jusqu'à ce qu'on ait une erreur acceptable(<0.1).

2.3 Test du réseau de neurones

Dans cette partie on va tester notre réseau de neurones. Cette phase de test est la phase finale de l'implémentation du réseau de neurones. Elle permet de vérifier la justesse de nos résultats.

2.3.1 Classification d'images

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.

2.3.2 Évaluation 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 et plus il est bon, et a la fin on a un réseau de neurones à convolution qui sait classifier des images selon qu'elles sont tumorales ou pas.

2.3.3 Quelques fonctions supplémentaires par rapport à la version du premier semestre

Ce qui a changé par rapport au premier semestre, c'est que nous avons implémenté une nouvelle fonction pour tester le réseau de neurones avec les poids enregistrés après l'entraînement du réseau neuronal. Cette fonction est appelée data_test().

Aussi nous avons rajouté une fonction output_test_char() qui stocke le résultat de l'output afin de le comparer avec l'output que l'on souhaitait obtenir. Ainsi par la suite on arrive à évaluer le pourcentage de mauvaises classifications donné par le réseau de neurones.

```
// test des résultats enregistrés dans data.txt
void data_test(char *fname,ppm_t *pp_images);

const char *output_test_char(const char * res,ppm_t *pp_images);

const char* get_res(ppm_t *pp_images);
```

FIGURE 2.7 – Les prototypes des fonctions implémentées

3

La parallélisation du code

4

Étude des performances

Perspectives et conclusion

Organisation du travail

Bibliographie

[1] https ://fr.wikipedia.org/wiki/Neurone.

[2] https ://fr.wikipedia.org/wiki/Réseau_de_neurones_artificiels.

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

[4] http ://penseeartificielle.fr/focus-reseau-neurones-convolutifs/.

[5] https ://fr.wikipedia.org/wiki/Réseau_de_neurones_artificiels.

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