RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Électronique

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en Électronique

---

Fusion de caractéristiques pour la classification des différents niveaux de démence de la maladie d'Alzheimer

---

NENNOUCHE Mohamed & ATCHI Abdel Malek

Sous la direction de **Dr. BOUADJENEK Nesrine** ENP

Présenté et soutenu publiquement le (04/07/2022)

Composition du jury :

Président :	Pr. BOUSBIA SALAH Hichem	ENP
Promotrice :	Dr. BOUADJNEK Nesrine	ENP
Examinatrice :	Dr. LANI Fatiha	ENP

ENP 2022

Électronique / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE
MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE SCIENTIFIQUE

ÉCOLE NATIONALE POLYTECHNIQUE

المدرسة الوطنية المتعددة التقنيات
Ecole Nationale Polytechnique

Département d'Électronique

Mémoire de projet de fin d'études

Pour l'obtention du diplôme d'ingénieur d'état en Électronique

---

Fusion de caractéristiques pour la classification des différents niveaux de démence de la maladie d'Alzheimer

---

NENNOUCHE Mohamed & ATCHI Abdel Malek

Sous la direction de **Dr. BOUADJENEK Nesrine** ENP

Présenté et soutenu publiquement le (04/07/2022)

Composition du jury :

Président :	Pr. BOUSBIA SALAH Hichem	ENP
Promotrice :	Dr. BOUADJNEK Nesrine	ENP
Examinatrice :	Dr. LANI Fatiha	ENP

ENP 2022

Électronique / ENP, 10 Rue des Frères OUDEK, El Harrach 16200 Alger Algérie.

ملخص

مرض الزَهايمر هو مرضٌ يصيبُ أكثرَ مِن 55.2 مليون شخصٍ في جميعِ أنحاء العالم. ويؤثّر هذا المرضُ على هؤلاءِ الأشخاصِ بشكلٍ رهيبٍ حيث يجعلُ مِن أبسطِ الأعمالِ اليومية عائقاً لهم. يهدف مشروعنا إلى إنشاء نموذجِ ذكاءٍ اصطناعيٍّ يرتكز على دمجِ طريقتين لاستخراج الخصائص: شبكة عُصبونية التِفافية عميقة (CNN) والرّسم البَياني لاتّجاه التّدرّج (HoG) للسّماح بالتّصنيف الثّنائي أوّلا (مريض وسليم) وثانيا تصنيف المسْتويات المختلفة لهذا المرض باستخدام أنواعٍ مختلفةٍ من المُصنِّفات مع مقارنةِ نموذجِنا مع مستوى التّقدم الجاري.

الكلمات المفتاحية :

الزهايمر - التصوير بالرنين المغناطيسي - التعلم الآلي - التعلم العميق - معالجة الصور - التصنيف الآلي - الطب الحيوي

Abstract

Alzheimer's disease is a disease affecting more than 55.2 million people worldwide, greatly handicapping these people, making every daily action more complicated. Our project aims to setup up a diagnostic aid system based on a fusion of two feature extraction methods : a deep convolutional neural network (CNN) and the Histogram of Oriented Gradients (HOG) allowing binary classification in a first step (sick and healthy) and in a second step the classification of the different levels of this disease by comparing the performances of different types of classifiers with a comparative study with the state of the art.

Keywords : Alzheimer - MRI - Machine Learning - Deep Learning - Image processing - Classification - Biomedical.

Résumé

La maladie d'Alzheimer est une maladie touchant plus de 55,2 millions de personnes dans le monde, handicapant énormément ces personnes, rendant chaque action du quotidien plus compliquée. Notre projet a pour but de mettre en place un système d'aide au diagnostique se basant sur une fusion de deux méthodes d'extraction de caractéristiques : un réseau de neurones convolutif (CNN) profond et de l'Histogramme de Gradients Orientés (HOG) permettant la classification binaire dans un premier temps (malade et sain) et dans un deuxième temps la classification des différents niveaux de cette maladie en comparant les performances de différents types de classifieurs avec une étude comparative avec l'état de l'art.

Mots clés : Alzheimer - IRM - Machine Learning - Deep Learning - Traitement d'images - Classification - Biomédical.

Remerciements

Ce mémoire marque un tournant décisif dans notre vie, la fin d'un long périple éducatif, et le début de nouvelles aventures. Il est le fruit de notre dur labeur, concrétisé grâce à notre travail mais aussi à l'aide indéniable que nous a apporté notre entourage. C'est pourquoi nous tenons à remercier toute personne ayant contribué de près ou de loin à la réussite de ce travail.

Nous tenons tout particulièrement à remercier nos parents qui nous ont accompagné depuis le début, nous assurant le meilleur cadre possible pour étudier et nous épanouir, nous aimerions aussi remercier tous nos amis et camarades de l'école nous ayant aidé et soutenu, tout particulièrement Aziz, Lynda, Nacira, Hakim, Riad, Massylia, Abdel Baki, Nazim et Yacine pour leurs supports, conseils et remarques sur notre travail.

Bien sur, nous tenons à exprimer notre plus sincère gratitude à notre encadrante Dr. BOUADJNEK Nesrine qui nous a soutenu tout au long de ce travail. Nous la remercions pour sa patience, sa motivation et son dévouement sans pareil pour faire de ce travail un travail de valeur. Elle a toujours été disponible lorsque nous avions besoin d'elle tout au long de ce travail. Nous n'aurions pas pu espérer un meilleur encadrement. Nous lui devons donc une immense et infinie gratitude.

Pour conclure nous tenons à remercier tous nos enseignants du département d'électronique qui nous ont suivi durant ces 3 années d'études et ce jusqu'à l'aboutissement de ce projet.

Mohamed et Abdel Malek.

TABLE DES MATIÈRES

Liste des tableaux

Table des figures

Liste des acronymes

Introduction générale **14**

1 Généralités sur la détection de la maladie d'Alzheimer : contexte clinique et état de l'art **16**

1.1 Introduction 16

1.2 Problématique 16

1.3 Imagerie à résonance magnétique (IRM) 17

1.3.1 Principe de l'IRM 17

1.3.2 Utilisation de l'IRM dans l'imagerie médicale 18

1.4 Maladie d'Alzheimer 19

1.4.1 La maladie d'Alzheimer en chiffres 20

1.5 Architecture des systèmes d'aide au diagnostique de la maladie d'Alzheimer . . . 21

1.5.1 Pré-traitement 21

1.5.2 Extraction des caractéristiques 22

1.5.2.1 Méthodes classiques 23

1.5.2.2 Méthodes d'apprentissage automatique 23

1.5.3 Classification 23

1.6 État de l'art 23

1.7 Conclusion 25

2 Réseaux de neurones convolutifs 26

2.1 Introduction 26

2.2 Les réseaux de neurones à travers le temps 26

2.2.1 Perceptron 27

2.2.2 Perceptron multicouches (MLP) 28

2.2.3 Défauts des MLP 29

2.3 Réseaux de neurones convolutifs 30

2.3.1 Bloc des couches d'extraction des caractéristiques 30

2.3.1.1 Couches convolutionnelles 30

2.3.1.2 Couches de Pooling 31

2.3.1.3 Couche d'aplatissement : Flatten layer 32

2.3.2 Bloc des couches de classification 32

2.3.3 Paramètres d'un CNN 33

2.3.3.1 Filtres 33

2.3.3.2 Fonctions d'activation 34

2.3.3.3 Stride 36

2.3.3.4 Zero-padding 37

2.3.3.5 Drop-out 37

2.4 Modèles pré-entraînés 37

2.4.1 Qu'est ce qu'un modèle pré-entrainé ? 37

2.4.2 Pourquoi utiliser un modèle pré-entraîné ? 38

2.4.3 Apprentissage par transfert 38

2.4.3.1 Stratégies et techniques de Transfer Learning 38

2.4.3.2 Comment utiliser le Transfer Learning ? 39

2.4.4 Modèles utilisés 40

2.4.5 AlexNet 40

2.4.5.1 Architecture du modèle 40

2.4.6 ResNet-101 41

2.4.7 Inception-ResNet-V2 44

2.4.7.1 Bloc d'Inception . . . 44
2.4.7.2 GoogLeNet . . . 44
2.4.7.3 Inception-Resnet-V2 . . . 45
2.5 Conclusion . . . 47

3 Résultats expérimentaux 49
3.1 Introduction . . . 49
3.2 Ensembles de données utilisés . . . 49
3.2.1 Kaggle . . . 49
3.2.1.1 Distribution des échantillons . . . 50
3.3 Logiciels, libraires et matériel . . . 51
3.3.1 Python . . . 51
3.3.2 Scikit-learn . . . 51
3.3.3 Keras . . . 52
3.3.4 Tensorflow . . . 52
3.3.5 Matlab . . . 52
3.3.6 Matériel (Hardware) . . . 53
3.3.6.1 Google Colaboratory . . . 53
3.3.6.2 Paperspace gradient . . . 53
3.4 Métriques utilisées . . . 54
3.4.1 AUC ROC Score . . . 55
3.4.2 F1-score . . . 56
3.4.3 Exactitude (Accuracy) . . . 57
3.4.4 Matrice de confusion . . . 57
3.4.5 Sensibilité et Spécificité . . . 58
3.5 Performances atteintes . . . 58
3.5.1 Protocole utilisé . . . 58
3.5.1.1 Augmentation des données . . . 59
3.5.1.2 Description des ensembles d'entraînement, de validation et de test . . . 59
3.5.1.3 Entraînement . . . 60
3.5.2 Modèles pré-entraînés . . . 61

3.5.2.1 Première distribution de la base de données 61
3.5.2.2 Deuxième distribution de la base de données 63
3.5.3 AlexNet en tant qu'extracteur de caractéristiques 64
3.5.3.1 Différents types de classifieurs communément utilisés 65
3.5.3.2 Première distribution de la base de données 67
3.5.3.3 Deuxième distribution de la base de données 68
3.5.3.4 Choix des paramètres des classifieurs utilisés 69
3.5.3.5 GridSearchCV 70
3.5.3.6 Validation croisée (cross validation) 70
3.5.3.7 Choix optimal des paramètres du KNN 71
3.5.4 Histogramme de Gradients Orientés 71
3.5.4.1 Première distribution de la base de données 73
3.5.4.2 Deuxième distribution de la base de données 73
3.5.5 Fusion des caractéristiques 74
3.5.5.1 Fusion des caractéristiques 74
3.5.5.2 Fusion des deux descripteurs 74
3.5.5.3 Première distribution de la base de données 75
3.5.5.4 Deuxième distribution de la base de données 77
3.6 Récapitulatif des résultats 79
3.7 Conclusion 81

Conclusion générale **82**

Bibliographie **84**

LISTE DES TABLEAUX

1.1 Performances atteintes dans l'état de l'art 25

2.1 Architectures des différents modèles ResNet 42

2.2 Nombre de paramètres des réseaux ResNet 44

2.3 Architecture de GoogLeNet 45

3.1 Distribution binaire de l'ensemble de données Kaggle 51

3.2 Distribution des classes de l'ensemble de données Kaggle 54

3.3 Combinaisons des classes avec la stratégie OvO 56

3.4 Combinaisons des classes avec la stratégie OvR 56

3.5 Première répartition des différents ensembles : entraînement, validation et test . 60

3.6 Deuxième répartition des différents ensembles : entraînement, validation et test . 60

3.7 Performances des modèles pré-entraînés pour la classification binaire avec la première configuration de l'ensemble de données 62

3.8 Performances des modèles pré-entraînés pour la classification multiclasse avec la première configuration de l'ensemble de données 62

3.9 Performances des modèles pré-entraînés pour la classification binaire avec la deuxième configuration de l'ensemble de données 63

3.10 Performances des modèles pré-entraînés pour la classification multiclasse avec la deuxième configuration de l'ensemble de données 64

3.11 Performances d'AlexNet avec différents classifieurs pour la classification binaire avec la première configuration dd l'ensemble de données 68

3.12 Performances d'AlexNet avec différents classifieurs pour la classification multiclasse avec la première configuration de l'ensemble de données 68

3.13 Performances d'AlexNet avec différents classifieurs pour la classification binaire avec la deuxième configuration de l'ensemble de données 69

3.14 Performances d'AlexNet avec différents classifieurs pour la classification multi-classe avec la deuxième configuration de l'ensemble de données 69

3.15 Performances du HoG avec différents classifieurs pour la classification binaire avec la première configuration de la base de données 73

3.16 Performances du HoG avec différents classifieurs pour la classification multiclasse avec la première configuration de la base de données 73

3.17 Performances du HoG avec différents classifieurs pour la classification binaire avec la deuxième configuration de la base de données 74

3.18 Performances du HoG avec différents classifieurs pour la classification multiclasse avec la deuxième configuration de la base de données 74

3.19 Performances de la fusion de caractéristiques pour la classification binaire avec la première configuration de la base de données 75

3.20 Performances de la fusion de caractéristiques pour la classification multiclasse avec la première configuration de la base de données 76

3.21 Performances de la fusion de caractéristiques pour la classification binaire avec la deuxième configuration de la base de données 77

3.22 Performances de la fusion de caractéristiques pour la classification multiclasse avec la deuxième configuration de la base de données 78

TABLE DES FIGURES

1.1 Exemple de scanner IRM 17
1.2 Exemples d'images IRM 18
1.3 Evolution de la maladie d'Alzheimer à travers les différents stades de démence . 19
1.4 Alzheimer en quelques chiffres 21
1.5 Architecture d'un système d'aide au diagnostique 21
1.6 IRM avant et après débruitage 22
1.7 IRM avant et après correction du champ de biais 22

2.1 Évolution des réseaux de neurones dans le temps 27
2.2 Architecture d'un Perceptron 28
2.3 Réseau de neurones à une couche cachée 29
2.4 Architecture standard d'un CNN 30
2.5 Couche de convolution 31
2.6 Opération de Pooling (avec le max- pooling et average-pooling) 32
2.7 Opération de concaténation du vecteur de caractéristiques 32
2.8 Réseau de neurones entièrement connecté 33
2.9 Cartes de caractéristiques 34
2.10 Fonction d'activation ReLU 34
2.11 Fonction d'activation sigmoïde 35
2.12 Fonction d'activation tanh 35
2.13 Fonction d'activation ELU 36
2.14 Principe du stride 36

2.15 Principe du drop-out durant l'entraînement d'un modèle 37

2.16 Comparatif de performances entre des modèles entrainés de zéro et des modèles pré-entrainés faisant appel au transfert learning 39

2.17 Comparatif entre les architectures des modèles LeNet et AlexNet 41

2.18 Bloc résiduel 42

2.19 Comparatif d'architecture à enchainement linéaire classique avec ResNet-34 . . . 43

2.20 Module d'Inception : à gauche version naïve, à droite avec réduction de dimensions 44

2.21 Architecture du Inception-Resnet-V2 46

2.22 Modules d'Inception A,B,C dans un ResNet-Inception 46

2.23 Bloc de réduction A (réduction de taille de 35x35 à 17x17) et Bloc de réduction B (réduction de taille de 17x17 à 8x8) 47

2.24 Mise à l'échelle des unités résiduelles 47

2.25 Architecture finale de l'Inception-ResNet-V2 47

3.1 Échantillons d'images IRM de l'ensemble de données Kaggle 50

3.2 Distribution des classes de l'ensemble de données Kaggle 51

3.3 Courbe ROC 55

3.4 Matrice de confusion 57

3.5 Augmentation des données sur notre ensemble de données 59

3.6 Les différentes configurations testées dans notre projet 61

3.7 Architecture de l'extracteur de caractéristique de AlexNet 65

3.8 Principe d'un SVM 66

3.9 Principe du KNN 66

3.10 Principe d'une forêt aléatoire 67

3.11 Comparaison entre la validation croisée à retenue et à k blocs 71

3.12 Exemple de l'application du HoG sur une IRM 72

3.13 Architecture du descripteur utilisé dans le projet 75

3.14 Matrice de confusion du modèle de fusion de caractéristiques pour la classification binaire suivant la première distribution de la base de données 76

3.15 Matrice de confusion du modèle de fusion de caractéristiques pour la classification multiclasse suivant la première distribution de la base de données 77

3.16 Matrice de confusion du modèle de fusion de caractéristiques pour la classification binaire suivant la deuxième distribution de la base de données 79

3.17 Matrice de confusion du modèle de fusion de caractéristiques pour la classification multiclasse suivant la deuxième distribution de la base de données 79

3.18 Comparaison des performances des différents modèles sélectionnés suivant la première distribution 80

3.19 Comparaison des performances des différents modèles sélectionnés suivant la seconde distribution 80

LISTE DES ACRONYMES

- **ANN** : Artificial Neural Network
- **API** : Application Programming Interfaces
- **ADNI** : Alzheimer's Disease Neuroimaging Initiative
- **CNN** : Convolutional Neural Network
- **CPU** : Central Processing Unit
- **DICOM** : Digital Imaging and Communications in Medicine
- **DNN** : Deep Neural Network
- **EC** : Entièrement connecté
- **GPU** : Graphic Processor Unit
- **HOG** : Histogramme de Gradients Orientés
- **IA** : Intelligence Artificielle
- **IRM** : Imagerie par Résonance Magnétique
- **KNN** : K-Nearest Neighbors
- **MLP** : Multi-layer Perceptron
- **NIfTI** : Neuroimaging Informatics Technology Initiative
- **ReLU** : Rectified Linear Unit
- **RGB** : Red Green Blue
- **RMN** : Résonance magnétique nucléaire
- **ROC** : Receiver operating characteristic
- **SVM** : Support Vector Machine
- **TEP** : Tomographie par émission de positrons

INTRODUCTION GÉNÉRALE

"**La mémoire est la sentinelle de l'esprit**"

Shakespeare, Macbeth, I, VII – 1605

Perdre la mémoire pour beaucoup est considéré comme un supplice pire que la mort, faisant perdre souvenirs, moments de joie, moments de famille, réussites et faisant tomber la personne dans une folie sans pareil, la maladie d'Alzheimer est ainsi une maladie qui touche une bonne partie de nos seniors faisant des dégâts considérables, les faisant tomber dans une démence profonde et avec un style de vie en dépendance de leurs proches.

La maladie d'Alzheimer est une maladie cérébrale irrémédiable, neuro-dégénérative et dynamique qui affecte directement la mémoire, l'apprentissage, les pensées et le comportement de l'individu. Actuellement, la maladie d'Alzheimer est la troisième cause de décès dans le monde après le cancer et les maladies cardiaques [1]. La détection précoce de cette maladie avec des outils d'aide au diagnostique permettrait donc de retarder sa progression et de prendre des mesures thérapeutiques rapidement pour un combat plus efficace.

Pour cela, plusieurs travaux se sont portés dans le but de pouvoir classer des patients atteints d'Alzheimer de patients sains ou encore de parvenir à classer les différents patients suivant leurs niveaux de démence, travaux remontant aux années 80, jusqu'à récemment avec, entre autres, les travaux de Syed et al. [2], N. Khan et al. [3], H. Nawaz et al. [4] et A. Loddo [5] atteignant respectivement 98.6%, 99.36%, 99.21% et 97.71% de précision sur les bases de données qu'ils ont traités.

Ce travail décrira alors notre apport dans le combat contre cette maladie, autant sur la classification entre les patients sains et malades (classification binaire), que sur la classification des différents niveaux de démence des patients atteints d'Alzheimer, en se basant sur la fusion de caractéristiques entre l'Histogramme de Gradients Orientés (HOG), descripteur ayant prouvé son efficacité dans diverses applications de traitement d'image et d'intelligence artificielle et des caractéristiques extraites à partir d'un réseau de neurones convolutifs (CNN) suivis d'un classifieur K-plus proches voisins (KNN) atteignant une précision dépassant les résultats retrouvés dans l'état de l'art. Le travail se reposant sur une base de données comptant 6400 IRM disponible sur Kaggle [6] en libre accès. Cette base de données classe les patients suivant le niveau de démence qu'ils présentent, on a alors 4 classes :

- Non Demented (Aucune démence).
- Very Mild Demented (Très légère démence).
- Mild Demented (Légère démence).

- Moderate Demented (Démence modérée).

Ce mémoire est structuré en 4 chapitres de la manière suivante :

Chapitre 1 : Ce chapitre présente le contexte médical en retraçant l'impact de cette maladie sur les personnes et sur la société, les étapes constituant un système de classification des niveaux de démence de la maladie d'Alzheimer ainsi que les travaux effectués.

Chapitre 2 : Ce chapitre fournit la théorie des réseaux de neurones artificiels, ainsi que les réseaux de neurones à convolution que l'on a employés pour la génération des caractéristiques et la classification.

Chapitre 3 : Dans ce dernier chapitre on parlera plus en détails du travail effectué dans notre projet à travers la description et l'utilisation des modèles pré-entraînés et des ressources utilisées, du protocole expérimental ainsi que du modèle et de l'architecture adoptés dans notre projet. On conclura par une discussion et une comparaison des résultats des tests effectués.

Enfin, nous terminerons ce mémoire par les principales conclusions et quelques perspectives.

CHAPITRE 1

GÉNÉRALITÉS SUR LA DÉTECTION DE LA MALADIE D'ALZHEIMER : CONTEXTE CLINIQUE ET ÉTAT DE L'ART

1.1 Introduction

L'imagerie médicale a fait son apparition avec la découverte des rayons X par Wilhelm Röntgen en 1896 et depuis cette date elle n'a cessé d'évoluer, passant en 1934 par la découverte de la radioactivité artificielle par Frédéric et Irène Joliot-Curie jusqu'à la mise en place de l'imagerie à résonance magnétique ou IRM qui est actuellement un des moyens les plus puissants dans le combat contre la maladie d'Alzheimer, pathologie neuro-dégénérative touchant essentiellement les personnes âgées et dont le principal symptôme est la démence qui évolue à une vitesse fulgurante attaquant les neurones des patients.

Dans ce chapitre, nous nous intéressons à la description médicale de la maladie d'Alzheimer et de ses différents niveaux de démence, au principe de l'imagerie par résonance magnétique (IRM), ainsi qu'à l'architecture d'un système d'aide au diagnostique de la maladie d'Alzheimer. À la fin, le chapitre présente un résumé de l'état de l'art donnant une idée sur les techniques précédemment utilisées.

1.2 Problématique

La maladie d'Alzheimer est une pathologie neuro-dégénérative complexe, incurable jusqu'à présent et qui provoque un dysfonctionnement des connexions entre les neurones. Cette maladie est l'une des plus répandues dans le monde en touchant essentiellement les personnes dépassant les 70 ans. Touchant une grande partie du cerveau humain, elle est très souvent mal diagnostiquée car confondue avec un certain nombre d'autres pathologiques telles que la maladie de Parkinson et d'autres maladies générant de la démence.

Par conséquent, ralentir l'évolution de la maladie ne serait-ce que d'un an pourrait réduire onze millions de cas dans le monde, atténuant ainsi considérablement son impact sur l'humanité [7]. À cet effet, la détection précoce de la maladie d'Alzheimer et le diagnostique précis des différents stades de la démence sont décisifs pour retarder la progression de la maladie et améliorer la qualité de vie du patient [8]. Pour ces raisons, nous proposons dans ce travail, différentes approches d'extraction et de fusion de caractéristiques, formant une base solide pour

diagnostiquer entre personnes non atteintes et personnes atteintes ainsi que le diagnostique du stade d'atteinte de cette maladie.

1.3 Imagerie à résonance magnétique (IRM)

L'**I**magerie par **R**ésonance **M**agnétique ou IRM est une technique d'imagerie médicale permettant d'obtenir des vues en deux ou en trois dimensions de l'intérieur du corps de façon non invasive avec une résolution en contraste relativement élevée. Le dispositif se montre habituellement de la forme suivante :

FIGURE 1.1 – Exemple de scanner IRM

1.3.1 Principe de l'IRM

L'IRM repose sur le principe de la résonance magnétique nucléaire (IRM) qui utilise les propriétés quantiques des noyaux atomiques pour la spectroscopie en analyse chimique. L'IRM nécessite un champ magnétique puissant et stable produit par un aimant supraconducteur qui crée une magnétisation des tissus par alignement des moments magnétiques de Spin. Des champs magnétiques oscillants plus faibles, dits « radiofréquence », sont alors appliqués de façon à légèrement modifier cet alignement et produire un phénomène de précession qui donne lieu à un signal électromagnétique mesurable. La spécificité de l'IRM consiste à localiser précisément dans l'espace l'origine de ce signal RMN en appliquant des champs magnétiques non uniformes, des « gradients », qui vont induire des fréquences de précession légèrement différentes en fonction de la position des atomes dans ces gradients. Sur ce principe qui a valu à ses inventeurs, Paul Lauterbur et Peter Mansfield le prix Nobel de Physiologie ou de Médecine en 2003, il est alors possible de reconstruire une image en deux dimensions puis en trois dimensions à partir de la composition chimique et donc de la nature des tissus biologiques explorés [9]. La figure 1.2 montre un exemple d'images IRM d'un cerveau.

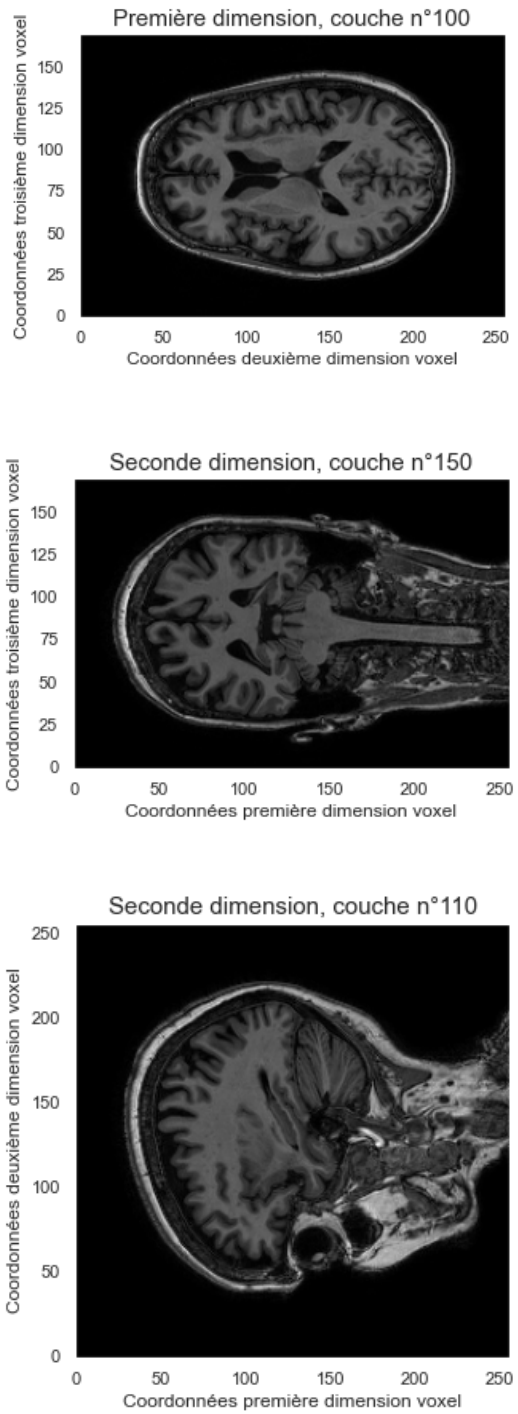


FIGURE 1.2 – Exemples d'images IRM

1.3.2 Utilisation de l'IRM dans l'imagerie médicale

En imagerie médicale, l'IRM est principalement dédiée à l'imagerie du système nerveux central (cerveau et moelle épinière), des muscles, du cœur et des tumeurs. Grâce aux différentes séquences, on peut observer les tissus mous avec des contrastes plus élevés qu'avec la tomodensitométrie ; en revanche, l'IRM ne permet pas l'étude des corticales osseuses (tissus « durs ») trop pauvres en hydrogène, ni donc la recherche fine de fractures où seul l'œdème péri-lésionnel pourra être observé.

L'examen IRM n'est pas invasif et n'irradie pas le sujet. Cela en fait donc un outil de prédilection pour la recherche impliquant la personne humaine, et notamment en neurosciences cognitives.

À partir des années 1990, la technique d'IRM fonctionnelle, qui permet de mesurer l'activité des différentes zones du cerveau, a en effet permis des progrès importants dans l'étude des fondements neurobiologiques de la pensée.

1.4 Maladie d'Alzheimer

Comme dit précédemment, la maladie d'Alzheimer est une maladie neuro-dégénérative (atteinte cérébrale progressive conduisant à la mort neuronale) caractérisée par une perte progressive de la mémoire et de certaines fonctions intellectuelles (cognitives) conduisant à des répercussions dans les activités de la vie quotidienne.La perte de ces neurones est dévastatrice et des fois de manière très rapide comme on peut le voir sur la figure 1.3 montrant les 4 stades de démence étudiés dans notre projet.

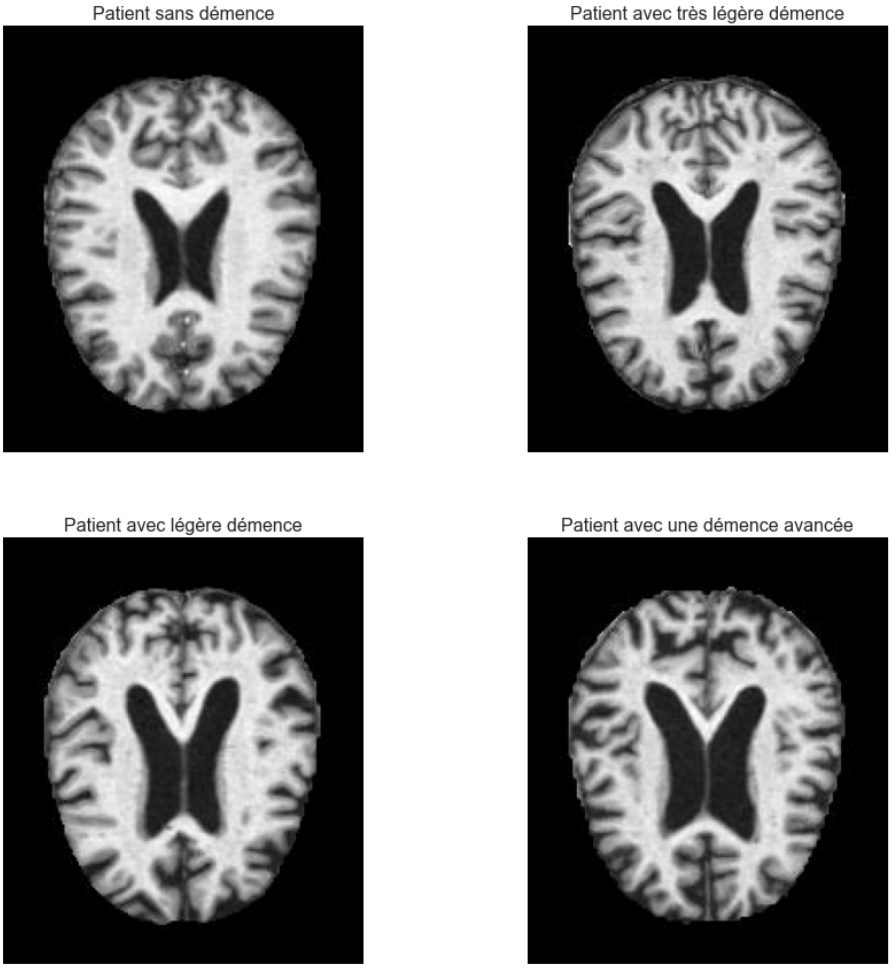


FIGURE 1.3 – Evolution de la maladie d'Alzheimer à travers les différents stades de démence

Les symptômes évoluent dans le temps de façon variable d'un individu à l'autre. Les troubles de la mémoire forment le symptôme le plus fréquent. Ils doivent être associés à un autre trouble des fonctions cognitives pour que le diagnostique de la maladie d'Alzheimer puisse être évoqué. Il peut s'agir :

- de troubles du langage (aphasie).
- de difficultés à effectuer certains gestes (apraxie).
- de la perte de la reconnaissance des objets ou des personnes (agnosie).

- de la désorientation spatiale et temporelle
- ou encore de la perte des fonctions exécutives, c'est-à-dire de la capacité à adapter son comportement à un contexte donné.

Les causes précises de la maladie ne sont pas identifiées, mais les travaux de recherche en cours sur le sujet permettent de mieux en mieux de connaître les mécanismes biologiques [10][11][12][13][14].

Sur le plan physiopathologique, la maladie d'Alzheimer est caractérisée par l'association de 2 lésions neuropathologiques cérébrales : les dépôts extracellulaires de protéine β-amyloïde et les dépôts intracellulaires de protéine τ. Ces lésions vont progresser au fil du temps de la région hippocampique vers l'ensemble du cortex cérébral expliquant la progression des troubles avec l'apparition d'une aphasie, d'une apraxie, de troubles visuo-spatiaux et de troubles des fonctions exécutives [15].

D'autres maladies sont source de signes cliniques proches :

- La démence d'origine vasculaire.
- La démence à corps de Lewy.
- La dégénérescence lobaire fronto-temporale.
- ect...

Ces maladies sont dites apparentées à la maladie d'Alzheimer.

1.4.1 La maladie d'Alzheimer en chiffres

En se basant sur le "**World Alzheimer Report 2021**" [1] ainsi que d'autres sources [16] on peut mettre en évidence les chiffres suivants (résumé figure 1.4) :

- La maladie d'Alzheimer est la plus fréquente des démences chez les patients âgés.
- Les démences sont désormais la septième cause de mortalité dans le monde.
- La prévalence des démences, toutes causes confondues, augmente exponentiellement entre 65 et 85 ans et double environ tous les 5 ans.
- La maladie touche 23% de la population après 80 ans. Après 65 ans, elle concerne environ deux fois plus de femmes que d'hommes.
- 55,2 millions de personnes sont atteintes de la maladie d'Alzheimer ou d'une maladie apparentée dans le monde selon l'OMS. Selon les estimations, ce nombre devrait atteindre les 78 millions en 2030 et 139 millions en 2050.
- Chaque année, on dénombre près de 10 millions de nouveaux cas.
- En 2019, le coût mondial de la démence était évalué à 1300 milliards de dollars américains. Il devrait s'élever à 1700 milliards de dollars américains d'ici à 2030, ou à 2800 milliards de dollars américains si l'on tient compte de la hausse du coût des soins.
- La démence est la maladie la plus coûteuse du 21e siècle pour la société.

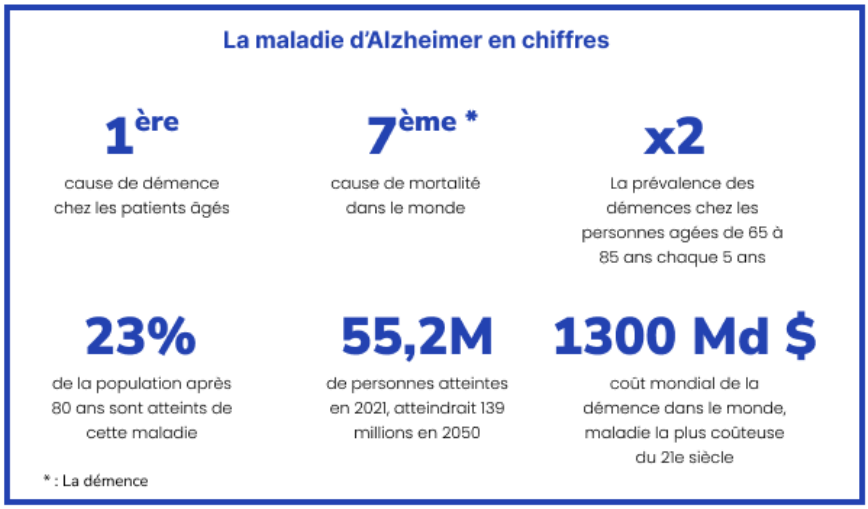


FIGURE 1.4 – Alzheimer en quelques chiffres

1.5 Architecture des systèmes d'aide au diagnostique de la maladie d'Alzheimer

L'architecture d'un système d'aide au diagnostique pour la maladie d'Alzheimer se compose principalement de 3 blocs principaux :

Bloc de pré-traitement : Ce bloc prendra en entrée les images brutes et appliquera des pré-traitements pour améliorer la qualité de l'image : débruitage, normalisation, correction du champ de biais, application de filtres...

Bloc d'extraction de caractéristiques : Ce bloc est au coeur de la qualité du système, prenant les images pré-traitées en extrayant les caractéristiques utilisées pour leurs classifications, à travers des méthodes statistiques classiques ou à base d'apprentissage profond.

Bloc de classification : Prenant en entrée le vecteur de caractéristiques que donne le bloc précédent, il s'occupe de la classification ou bien à travers des algorithmes classiques ou bien par apprentissage nous donnant ainsi en sortie la classe de l'image d'entrée.

L'architecture est décrite dans la figure 1.5 :

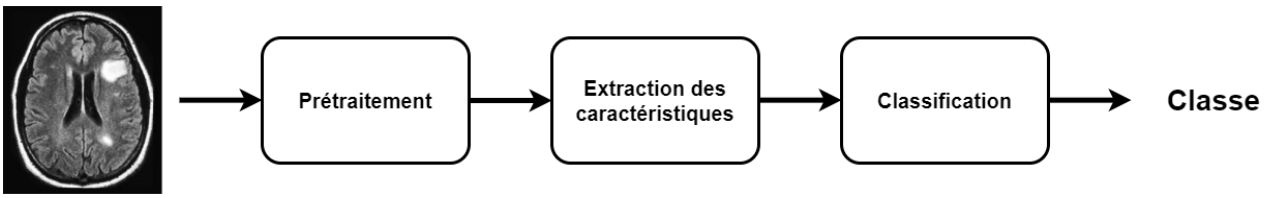


FIGURE 1.5 – Architecture d'un système d'aide au diagnostique

1.5.1 Pré-traitement

Le pré-traitement des images est l'une des parties les plus importantes dans l'imagerie médicale, la qualité des données défini la qualité du modèle final. Habituellement avec des images biomédicales on a ce genre de pré-traitement :

Débruitage des images : En utilisant différents filtres de lissages : Moyenneurs, Gaussiens, Médians pour enlever différents bruits dûs à la machine ou au mouvement du patient, cette étape est très importante pour permettre un diagnostique efficace et juste :

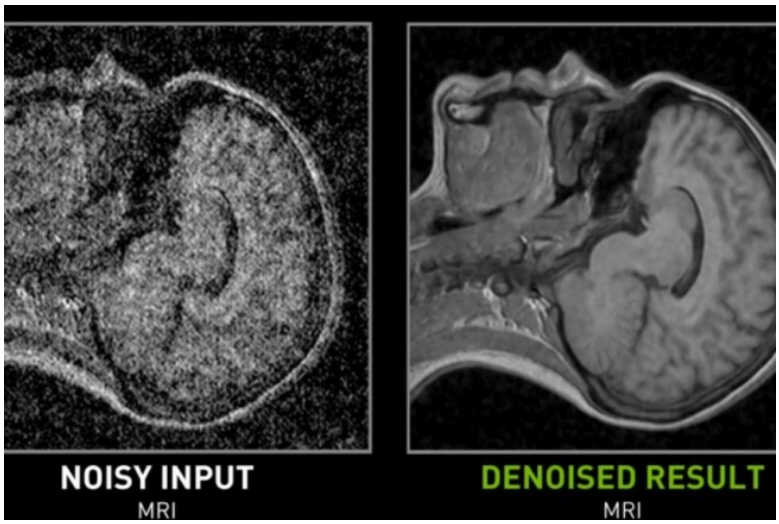


FIGURE 1.6 – IRM avant et après débruitage

Correction du champ de biais : La distorsion du "champ de biais" est une non-uniformité à basse fréquence présente dans les données IRM, qui fait varier les valeurs d'intensité IRM sur les images obtenues à partir du même scanner, du même patient et du même tissu [17], en corrigeant cela, ça nous permet de bien voir l'évolution de la pathologie et comparer différentes IRM comme l'indique la figure 1.7.

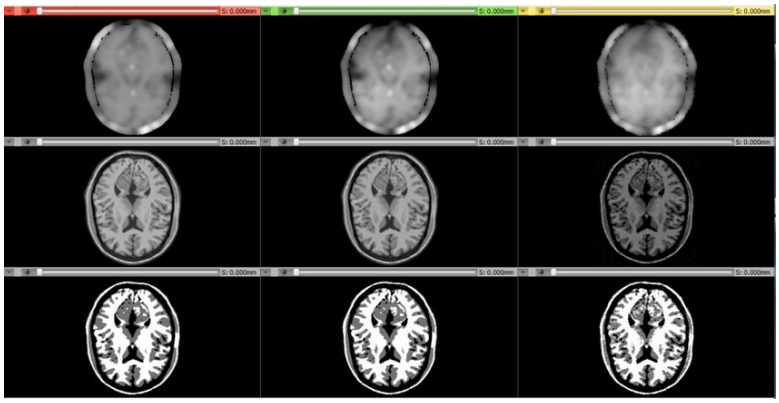


FIGURE 1.7 – IRM avant et après correction du champ de biais

Normalisation des données : On standardise nos images, la majorité des modèles d'apprentissage automatique et d'apprentissage profond sont efficient quand les valeurs d'entrées sont dans la même plage de valeurs, et plus idéalement si cette plage de valeurs est entre -1 et 1.

1.5.2 Extraction des caractéristiques

Cette partie, essentielle dans la robustesse et la qualité de la classification, a vu plusieurs méthodes se succéder depuis la mise en place de l'IRM, on peut classer ces méthodes suivants deux catégories :

- Méthodes classiques
- Méthodes basée sur l'apprentissage profond.

1.5.2.1 Méthodes classiques

Les méthodes classiques représentes toutes les techniques statistiques, mathématiques ou encore structurelles qui peuvent découler des images IRM, on a entre autre l'utilisation de la théorie des ondelettes comme dans les travaux de Y. Zhang et al. [18] et S. Basheera et al. [19] ou encore de méthodes de traitements d'images classiques à travers l'utilisation de l'histogramme de gradient orienté comme dans les travaux de H. Suresha et al. [20]. Dans plusieurs travaux, les données de textures et de lissage de l'image ont été aussi utilisées pour la classification.

1.5.2.2 Méthodes d'apprentissage automatique

Des travaux plus récents ont montré l'efficacité des réseaux de neurones convolutifs dans l'extraction des caractéristiques à partir d'images. Ces méthodes seront décrites plus en détails dans la prochaine section donnant un bref état de l'art.

1.5.3 Classification

Cette partie sert, comme dite précédemment, à la classification des images en prenant comme entrée le vecteur caractéristique qui découle de la description faite au préalable. Plusieurs travaux récents traitent de cette partie avec différents classifieurs. Souvent ils utilisent des classifieurs par apprentissage automatique tels que des SVM [21][22][23][24], le Random Forest [25][26][4], KNN [27][28] ou encore des réseaux de neurones [29][30], chaque classifieur utilisé à ses qualités et ses défauts. Un rapide état de l'art sera décrit dans la prochaine section.

1.6 État de l'art

En 1984, l'**Alzheimer's Disease and Related Disorders Association** a décidé que l'utilisation principale des techniques d'imagerie pour l'évaluation de la maladie d'Alzheimer devait être d'exclure les autres causes de démence. Cependant, certains des participants à ce groupe ont prédit la valeur potentielle de l'imagerie dans le diagnostique direct de la maladie d'Alzheimer [31]. En 1986 le scanner a été utilisé pour la première fois dans le calcul du volume du lobe temporal médian lié à la maladie d'Alzheimer [32]. Par la suite plusieurs méthodes basées sur les études métaboliques ont vu le jour comme compléments aux études des images IRM [33] [34] permettant de distinguer les patients atteints de la maladie d'Alzheimer des sujets sains avec une sensibilité de 83% et une spécificité de 98% [33].

Avec l'augmentation de la puissance de calcul des machines à la fin des années 90 et début 2000, de nouvelles méthodes basées sur de l'apprentissage automatique ont été mises au point par plusieurs équipes de recherche donnant des résultats très satisfaisants. Madhumitha et al. [35] ont calculé L'atrophie à l'aide de K-means, d'ondelettes, de bassins versants et de quelques autres algorithmes personnalisés. Les résultats de cette approche peuvent fournir une mesure diagnostique utile pour le stade précoce de la maladie d'Alzheimer.

Par la suite, plusieurs groupes ont travaillé sur la classification des différents niveaux de la maladie d'Alzheimer se basant sur des méthodes d'apprentissage automatique, entre autre, Plant et al. [36] qui ont travaillé sur une base de données constitué de 32 patients atteints de la maladie d'Alzheimer, de 24 patients atteints de troubles cognitifs légers et de 18 patients sains. Ils ont combiné les trois classificateurs SVM, les statistiques de Bayes et l'intervalle

de caractéristiques de vote (VFI), basés sur des caractéristiques extraites statistiquement par plusieurs méthodes dûment discutées par Plant et al. [36] obtenant ainsi une précision de 92% avec une sensibilité de 93.75% ainsi qu'une spécificité de 88.89% dans la classification entre les malades atteints d'Alzheimer et les autres.

Syed et al. [2], ayant quant à eux travaillé avec une base de données du site de "the Harvard Medical School" contenant 60 patients sains et 10 patients atteints d'Alzheimer, ont présenté une approche hybride pour la classification. Les caractéristiques DWT (Discrete Wavelet Transform) ont été extraites des images IRM et réduites des points non pertinents par l'Analyse en Composantes Principales. Enfin, ils ont utilisé un réseau de neurones artificiels et un KNN pour la classification en obtenant 98.6% de précision, 100% de sensibilité et 90% de spécificité avec ce dernier classifieur. H. Suresha et Al. [20] ont quant à eux travailler sur la base de données ADNI en utilisant l'histogramme de gradients orientés (HOG) comme descripteur suivi d'un réseau de neurones très profond comme classifieur ayant d'excellents résultats avec 99.5% de précision dans la classification binaire montrant la puissance d'un tel descripteur.

Plus récemment (à partir de 2017) l'utilisation des CNN a pris une place centrale dans la recherche dans le domaine, Dans [37], Muazzam Maqsood et al. ont proposé un système qui utilise des techniques d'apprentissage par transfert pour classer des images provenant de la base de données ADNI en ajustant finement le réseau AlexNet. Le CNN réentraîné a été validé à l'aide des données de test, donnant des précisions globales de 89,6 % et 96,8 % pour la classification binaire et multiclasse, respectivement. Toujours se basant sur la base de données ADNI on peut citer les travaux de D. Baska et al. [29] utilisant un ResNet-18 et obtenant 97.51% de précision ou encore les travaux de N. Khan et al. [3] utilisant quant à lui le VGG-19 et obtenant 99.36% de précision. D'autres travaux se basant quant à eux sur une autre base de données très communément utilisée, OASIS, ont aussi eu d'excellents résultats avec, par exemple, les travaux de H. Nawaz et al. [4] obtenant avec AlexNet une précision de 99.21% (en classification binaire) et d'autre part les travaux de A. Mehmood et al. [4] obtenant quant à eux 99.05% de précision avec un CNN conçu de zéro.

Plus récemment, la plus grande base de données d'images IRM labelisées dans la détection des différents niveaux d'Alzheimer a été mise à disposition sur Kaggle [6] contenant 6400 images IRM, et qui sera plus amplement décrite dans ce mémoire, a été abordée dans les travaux de A. Loddo et al. [5] ayant mis en place un modèle se basant sur une méthode d'apprentissage par ensemble en fusionnant les sorties de trois modèles pré-entraînés : AlexNet, ResNet-101, Inception-ResNet-V2 qui ont été adaptés à la problématique ayant une précision de 97.71% pour la classification multiclasse et 96.57% de précision pour la classification binaire, représentant le meilleur résultat actuellement obtenu sur cette base de données.

On résume les performances des travaux précédemment cités dans le tableau suivant :

Table 1.1 – Performances atteintes dans l'état de l'art

Travaux	Base de données utilisée	Précisions atteintes (%)
Plant et al. [36]	Personnelle	92
Syed et al. [2]	Harvard Medical School	98.6
H. Suresha et Al. [20]	ADNI	99.5
M. Maqsood et al. [37]	ADNI	89.6 (binaire) - 96.8 (multiclasse)
D. Baska et al. [29]	ADNI	97.51
N. Khan et al. [3]	ADNI	99.36
H. Nawaz et al. [4]	OASIS	99.21
A. Mehmood et al. [4]	OASIS	99.05
A. Loddo et al. [5]	Kaggle	96.57 (binaire) - 97.71 (multiclasse)

1.7 Conclusion

La maladie d'Alzheimer est une maladie neuro-dégénérative chronique qui détruit les cellules du cerveau, provoquant une dégénérescence irréversible des fonctions cognitives et une démence. A travers ce chapitre, nous avons mis en évidence le contexte clinique de cette maladie, les principes d'acquisition et de formation des images IRM, ainsi qu'un résumé de l'état de l'art mettant l'accent sur l'utilisation massive de l'apprentissage profond aujourd'hui, sans pour autant délaisser les systèmes classiques qui offrent toujours des performances satisfaisantes. Par conséquent, l'objectif de ce travail est de tirer profit de ces deux approches en proposant une fusion de caractéristiques hybrides afin de mettre en œuvre un système d'aide au diagnostique de la maladie d'Alzheimer. Le chapitre suivant présentera donc les réseaux de neurones convolutifs ainsi que les architectures pré-entrainées étudiées.

CHAPITRE 2

RÉSEAUX DE NEURONES CONVOLUTIFS

2.1 Introduction

Le concept des réseaux de neurones convolutifs a commencé à germer depuis la fin des années 60 dans la tête de plusieurs chercheurs s'inspirant des mécanismes visuels dans les organismes vivants, essentiellement des chats et des singes. Depuis, les CNN ont pris une place prépondérante dans la recherche en apprentissage profond dans la classification d'images particulièrement (essentiellement depuis 2012 avec la victoire d'AlexNet dans la compétition d'ImageNet). Leur utilisation dans l'imagerie médicale a pris de l'ampleur prouvant de plus en plus leurs efficacité et robustesse.

On développera alors dans ce chapitre plus en détails l'histoire de la création de ces réseaux de neurones, leurs composants et architectures et comment on peut les optimiser et les adapter à notre problématique, on conclura avec le concept de modèles pré-entrainés ainsi qu'une présentation de ceux qu'on a utilisé.

2.2 Les réseaux de neurones à travers le temps

L'intelligence artificielle (IA) s'est considérablement développée ces dernières années. Elle se situe au carrefour de l'informatique, des mathématiques et de la neurobiologie. L'IA vise à permettre aux machines d'imiter, voire de simuler, le comportement du cerveau humain, afin de résoudre des problèmes très complexes, tels que la classification, la reconnaissance et la détection. L'histoire des réseaux de neurones artificiels remonte à 1943, lorsque Warren McCulloch, un neurophysiologiste, et Walter Pitts, un mathématicien, ont rédigé un article [38] intitulé "A logical calculus of the ideas immanent in nervous activity" dans lequel, s'inspirant des neurones biologiques, ils modélisaient le premier neurone artificiel à l'aide de circuits électriques.

En 1949, Donald Hebb a écrit son livre [39] "The Organization of Behavior", dans lequel il a approfondi le concept de neurones en expliquant que les voies neuronales sont renforcées chaque fois qu'elles sont utilisées. Une découverte qui a apporté des connaissances essentielles sur la façon dont l'esprit humain apprend.

Dans les années 1950, alors que les ordinateurs devenaient plus perfectionnés, Nathaniel Rochester, des laboratoires de recherche d'IBM, a mené la première tentative de simulation d'un

réseau neuronal artificiel, tentative documentée par plusieurs publications [40].

En 1957, inspiré par des travaux antérieurs, Frank Rosenblatt a développé le modèle du perceptron [41], représentant le plus ancien réseau neuronal basé sur une seule couche de neurones, qui présente deux défauts principaux : il ne peut séparer qu'entre deux classes et les données doivent être linéairement séparables.

Par contre, à cette époque, les ordinateurs n'avaient pas une puissance de traitement suffisante pour gérer efficacement le temps requis par les grands réseaux neuronaux. Cette situation a fait stagner le domaine de la recherche sur les réseaux neuronaux pendant de nombreuses années, avant que l'on reconnaisse, au milieu des années 1980, lorsque les ordinateurs ont atteint une puissance de traitement supérieure, qu'un réseau neuronal à deux couches ou plus est beaucoup plus puissant en termes de traitement qu'un perceptron à une seule couche. C'est ainsi qu'est né le perceptron multicouche (en anglais dit Multi-Layer Perceptron, ou MLP). Cependant, l'utilisation du modèle MLP a été mise en attente en raison du manque d'outils conceptuels nécessaires pour analyser et prédire son comportement. Ce n'est qu'à la fin des années 1980 que les réseaux neuronaux ont réellement pris leur essor avec l'apparition de l'algorithme d'apprentissage par rétropropagation [42], suite à plusieurs travaux faits durant la décennie, essentiellement les travaux de Hopfield [43]. Plusieurs autres modèles de réseaux de neurones ont été développés pour résoudre différents problèmes de prédiction, de classification, de détection, et autres. Cependant, ces modèles nécessitent une opération préalable d'extraction de caractéristiques, obligatoire pour représenter les données. Plus récemment, l'augmentation de la puissance des infrastructures informatiques a permis à certains ordinateurs de traiter de grandes quantités de données avec l'apprentissage profond, qui désigne les réseaux de neurones artificiels (ANN) qui contiennent en plus des couches classiques, plusieurs couches dédiées à la génération de caractéristiques. Le premier modèle appelé LeNet a été proposé par LeCun et al. en 1998 [44], pour la reconnaissance de chiffres manuscrits. Par la suite, plusieurs modèles plus adaptés à d'autres applications ont été proposés. Dans les sections suivantes, nous allons présenter quelques définitions nécessaires à la compréhension du CNN utilisé dans notre travail. La figure 2.1 résume l'évolution du domaine à travers le temps :

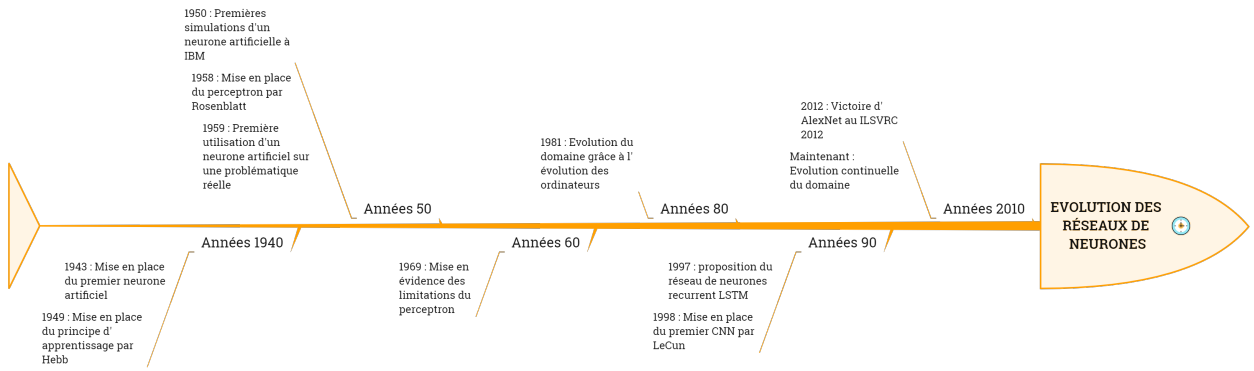


FIGURE 2.1 – Évolution des réseaux de neurones dans le temps

2.2.1 Perceptron

Le réseau neuronal existant le plus basique est le perceptron, il est composé d'un seul neurone formel. Il est considéré comme le processus d'apprentissage le plus ancien qui ait été réalisé pour la classification de deux classes linéairement séparables. Un perceptron est composé d'une ou plusieurs entrées, d'une unité de traitement et d'une sortie. Comme l'illustre la figure 2.2, étant donné un vecteur d'entrée $A = (a_0, a_1, ..., a_K) \in \mathbb{R}^{K+1}$, le neurone procède par une sommation

des éléments de A, pondérés par les éléments du vecteur de poids Ω, où Ω = (ω_0,ω_1, ...,ω_K) $\in \mathbb{R}^K$. Ensuite, la somme résultante passe par une fonction d'activation Z(.), produisant ainsi la sortie O.

Le perceptron tente de trouver une séparation linéaire entre deux classes, c'est-à-dire qu'il essaie de trouver une ligne de séparation définie par sa pente $\omega_{i,i} \neq 0$ et son biais ω_0 qui est considéré comme le poids d'une autre entrée $a_0 = 1$. Par conséquent, nous pouvons définir la fonction de sommation comme indiqué dans l'équation 2.1.

$$S = \omega_0 + \sum_{i=1}^{K} a_i \omega_i \tag{2.1}$$

La fonction d'activation qui suit, utilise un seuil d'activation pour traduire le signal d'entrée en une sortie O. Si la somme d'entrée résultante est supérieure au seuil, alors le neurone est activé et produit donc une sortie positive. Dans le cas contraire, la sortie est nulle et le neurone sera inhibé.

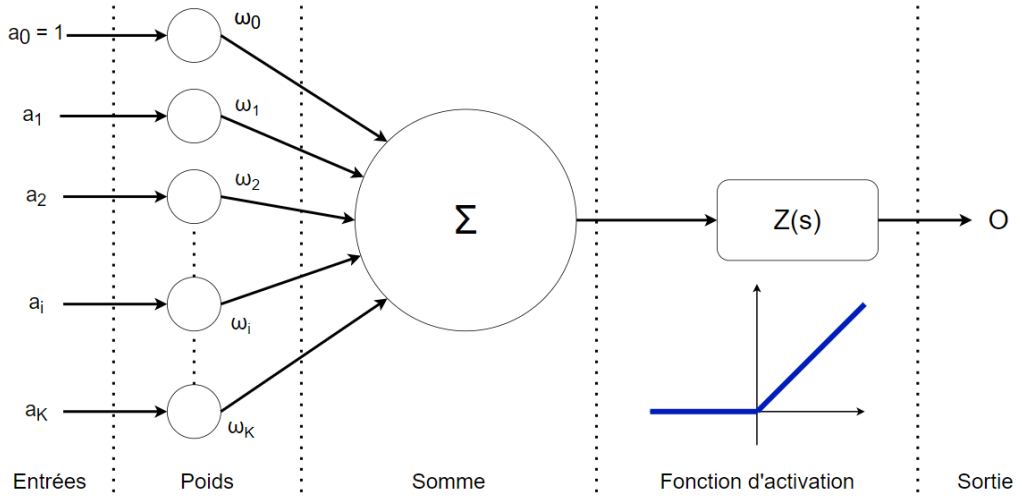


FIGURE 2.2 – Architecture d'un Perceptron

2.2.2 Perceptron multicouches (MLP)

Le perceptron multicouche est une version étendue du perceptron monocouche. Il est composé d'une couche d'entrée, de plusieurs couches cachées et d'une couche de sortie, où chaque neurone d'une couche est connecté à tous les neurones de la couche adjacente construisant ainsi un réseau entièrement connecté d'au moins trois couches (figure 2.3). Le nombre de neurones dans la couche d'entrée définit la dimension du vecteur caractéristique d'entrée, tandis que le nombre de neurones dans la couche de sortie définit le nombre de classes. Le nombre de neurones dans les couches cachées est considéré comme une question de conception. Moins il y a de neurones cachés, plus il est difficile pour le modèle d'apprendre des limites de décision complexes. D'autre part, plus il y a de couches cachées, moins le modèle est généralisé, il se spécialise.

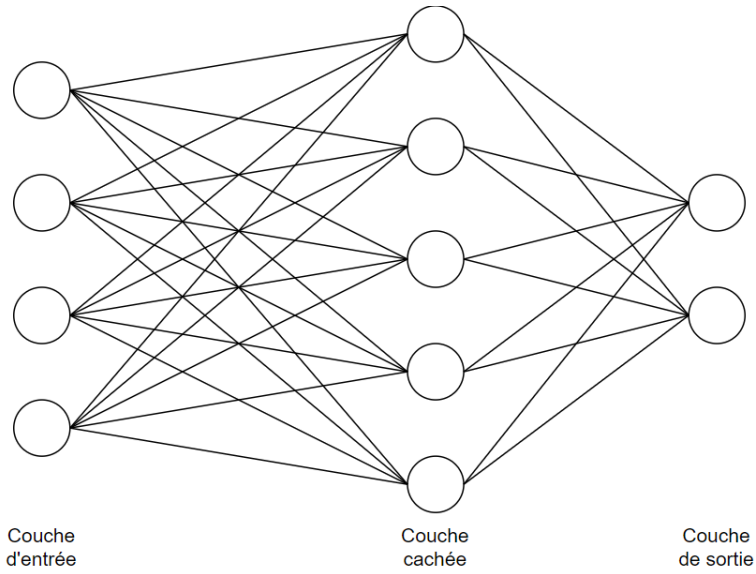


FIGURE 2.3 – Réseau de neurones à une couche cachée

Afin de déterminer les poids synaptiques ω_i, le MLP applique la rétropropagation du gradient de l'erreur lors de l'apprentissage, qui consiste à ajuster les poids du réseau en minimisant l'erreur quadratique moyenne e entre la sortie réelle et la sortie prédite $(\mu, \hat{\mu})$ respectivement.

$$e_{\hat{\mu},\mu} = \frac{1}{2}\sum_{i=1}^{m}(\hat{\mu} - \mu)^2 \tag{2.2}$$

Les poids synaptiques sont ensuite modifiés comme suit :

$$\omega(l+1) = \omega(l) + \Delta\omega(l+1) \tag{2.3}$$

où $\Delta\omega(l+1)$ est le changement de poids sur l'itération $(l+1)$, calculé par :

$$\Delta\omega(l+1) = -\tau\frac{e}{\omega} + \rho\Delta\omega(l) \tag{2.4}$$

Les deux paramètres τ et ρ, respectivement le taux d'apprentissage et le momentum, sont utilisés pour assurer la stabilité du processus de convergence en diminuant les variations des poids sur deux itérations consécutives.

2.2.3 Défauts des MLP

Les réseaux de neurones ont un grand nombre d'avantages, ayant permis de faire évoluer la science sur un bon nombre de problématiques, mais on peut en dénombrer quelques désavantages :

- Il n'existe pas de règle spécifique pour déterminer la structure des réseaux neuronaux artificiels en termes d'équilibre : nombre de neurones, nombre de couches cachées ainsi que le choix des paramètres d'apprentissage : pas d'apprentissage, momentum...
- Les MLP peuvent travailler qu'avec des informations numériques et il faut donc parvenir à décrire le problèmes sous forme de vecteur de caractéristiques.
- La qualité du vecteur influencera directement les performances du réseau.

2.3 Réseaux de neurones convolutifs

Les réseaux de neurones convolutifs (CNN ou ConvNet) sont une extension des MLP qui permet de surmonter efficacement quelques défauts de ces derniers. Ils sont conçus pour extraire automatiquement des caractéristiques depuis les données d'entrée. Ils ont été initialement inspirés par la découverte faite par Hubel et Wiesel, en 1962, où ils ont remarqué que des motifs particuliers stimulaient l'activité dans des parties spécifiques du cerveau des chats [45] [46] [47] et des singes [48]. La première utilisation de ces réseaux a été réalisée par Fukushima avec la carte auto-organisée (en anglais self organised map, ou SOM) pour l'extraction de caractéristiques en utilisant l'apprentissage non supervisé [49]. Dans le domaine de la reconnaissance des formes, un développement important a été réalisé par LeCun et al. dans [50], où ils ont proposé une architecture, dite LeNet, pour la reconnaissance des chiffres de l'écriture manuscrite. Le CNN élimine le besoin d'extraction manuelle des caractéristiques. Il extrait les caractéristiques et attribue des poids et des biais mémorisables décrivant divers aspects de l'image afin de les différencier les uns des autres. Techniquement, chaque image d'entrée passe par deux blocs de couches. Un premier bloc contenant les couches convolutionnelles, celles-ci sont responsables de la génération des caractéristiques, et un second bloc de classification qui contient des couches entièrement connectées, d'architecture similaire au MLP. La figure 2.4 illustre la disposition des couches dans un CNN.

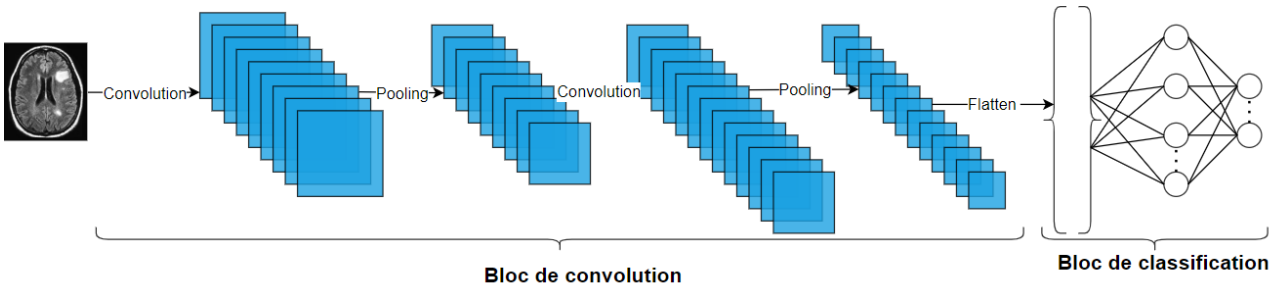


FIGURE 2.4 – Architecture standard d'un CNN

2.3.1 Bloc des couches d'extraction des caractéristiques

Ce premier bloc fait la particularité de ce réseau, puisqu'il fonctionne comme un extracteur de caractéristiques. La première couche de ce bloc filtre l'image avec plusieurs noyaux de convolution, et renvoie des caractéristiques qui sont normalisées ou redimensionnées via une fonction d'activation, ce qui donne une carte de caractéristiques. Ce processus peut être répété plusieurs fois. Les cartes de caractéristiques obtenues sont filtrées en utilisant de nouveaux noyaux à chaque fois, ce qui donne de nouvelles caractéristiques à normaliser et à filtrer à nouveau, et ainsi de suite. Enfin, les valeurs des dernières caractéristiques sont concaténées et aplaties en un vecteur colonne [51]. Ce vecteur définit la sortie du premier bloc, et l'entrée du second.

Plus précisément, il existe essentiellement deux types de couches dans cette partie du réseau. Les couches convolutionnelles et les couches de Pooling.

2.3.1.1 Couches convolutionnelles

Une couche convolutive à 2 dimensions extrait les caractéristiques d'une image d'entrée et préserve la relation entre les pixels en apprenant ces caractéristiques d'image à l'aide de filtres

de dimension carrée. Elle prend deux entrées : une matrice d'image de dimension $h \times w \times d$ (où h est la hauteur de la matrice d'image, w est sa largeur et d est le nombre de canaux) et un filtre ou noyau de dimension $f_h \times f_w \times d$ (où f_h est la hauteur du filtre, w est sa largeur et d est le nombre de canaux). Typiquement, l'entrée de la couche convolutive est une image couleur RGB, elle est donc représentée par 3 canaux. Dans le cas d'une image binaire ou en niveaux de gris, l'image d'entrée de la couche convolutionnelle est représentée par un seul canal. La matrice de sortie est calculée avec le produit scalaire séquentiel entre le filtre et une partie de la matrice d'entrée. Le filtre est glissé sur l'image de gauche à droite, de haut en bas, avec un certain nombre de pas appelés strides, pour effectuer le même calcul, comme le montre la figure 2.5.

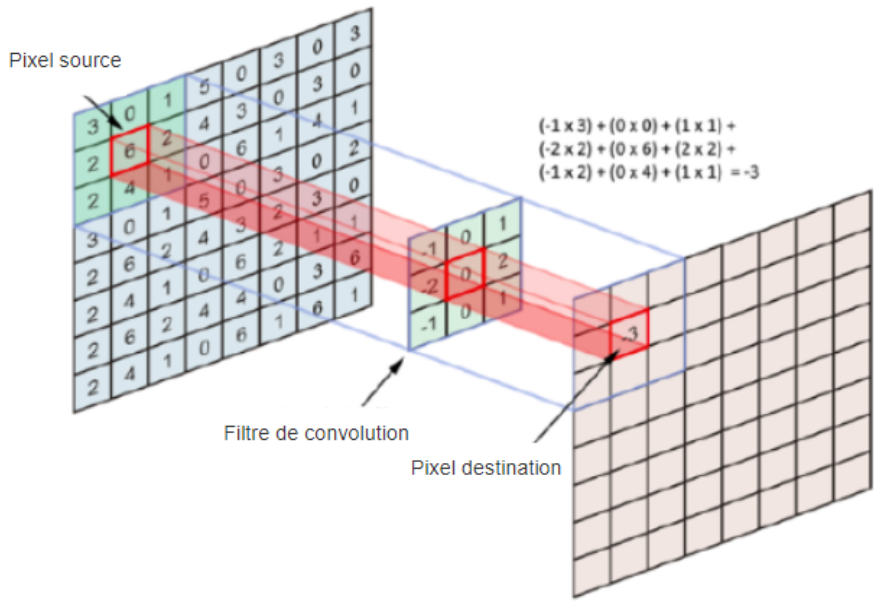


FIGURE 2.5 – Couche de convolution

2.3.1.2 Couches de Pooling

La couche de pooling effectue une opération de sous-échantillonnage de l'image en fonction des seules dimensions spatiales de la largeur et de la hauteur. L'image d'entrée est divisée en une série de carrés non superposés. Ce type de couche est fréquemment inséré entre deux couches convolutionnelles successives d'une architecture CNN afin de réduire le nombre de paramètres et la puissance de calcul requise, tout en préservant leurs caractéristiques les plus importantes. En pratique, l'image est divisée en petites cellules carrées adjacentes, espacées les unes des autres d'un pas, afin de ne pas perdre trop d'informations. On obtient le même nombre d'images de sortie que d'entrée mais chaque image a un nombre de pixels plus faible [51]. Il existe deux principaux types de Pooling : le **max-Pooling** et le **average-Pooling**. Le max-Pooling renvoie la valeur maximale de la partie de l'image couverte par le noyau. D'autre part, l'average-Pooling renvoie la moyenne de toutes les valeurs de la partie de l'image couverte par le noyau. En pratique, le max-Pooling est le plus utilisé et donne de bien meilleurs résultats que le average-Pooling. La figure 2.6 représente un exemple d'opérations de max-Pooling et de average-Pooling.

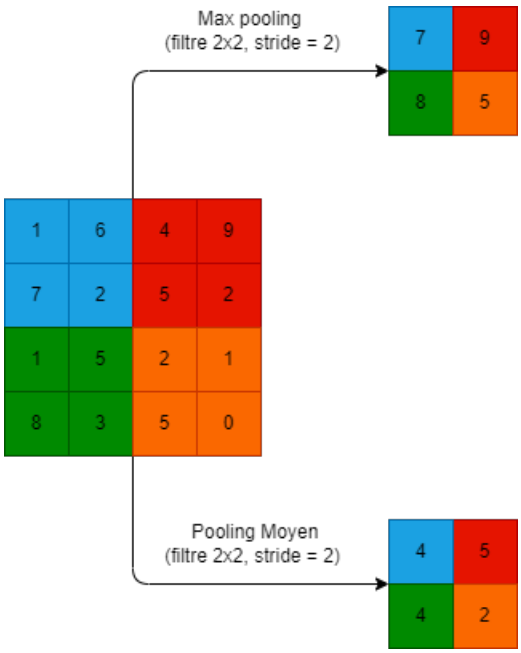


FIGURE 2.6 – Opération de Pooling (avec le max- pooling et average-pooling)

2.3.1.3 Couche d'aplatissement : Flatten layer

Il s'agit de la dernière étape du bloc d'extraction de caractéristiques. L'aplatissement consiste à concaténer les lignes des matrices en un vecteur unidimensionnel que nous pouvons connecter au bloc de prédiction (figure 2.7).

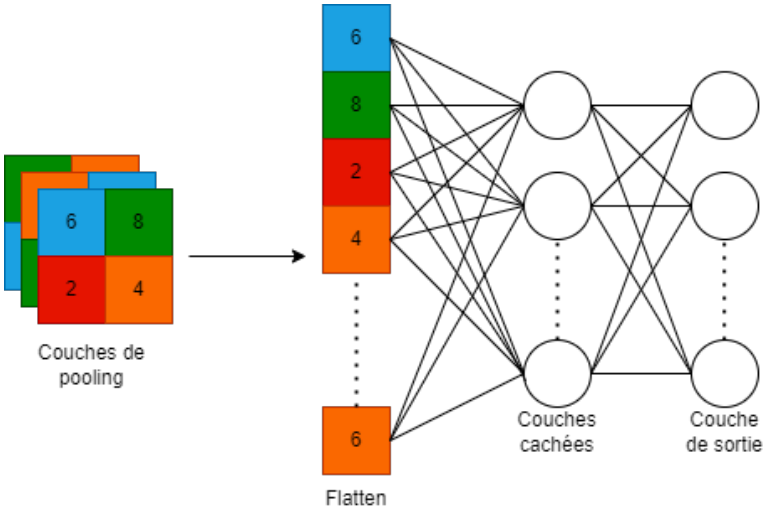


FIGURE 2.7 – Opération de concaténation du vecteur de caractéristiques

2.3.2 Bloc des couches de classification

La classification est effectuée par des couches entièrement connectées (EC), similaires au perceptron multicouche comme l'indique la figure 2.8. L'objectif des couches EC est d'utiliser le vecteur aplati en entrée pour apprendre les meilleurs paramètres permettant de mettre l'image dans une classe. En pratique, plusieurs couches sont placées l'une après l'autre pour renforcer l'apprentissage des caractéristiques et améliorer la prise de décision. La dernière couche EC

renvoie la sortie du CNN, qui correspond à un vecteur de taille égale au nombre de classes. Les neurones de cette couche utilisent la fonction d'activation **softmax** pour fournir les probabilités avec lesquelles une image d'entrée appartient à différentes classes.

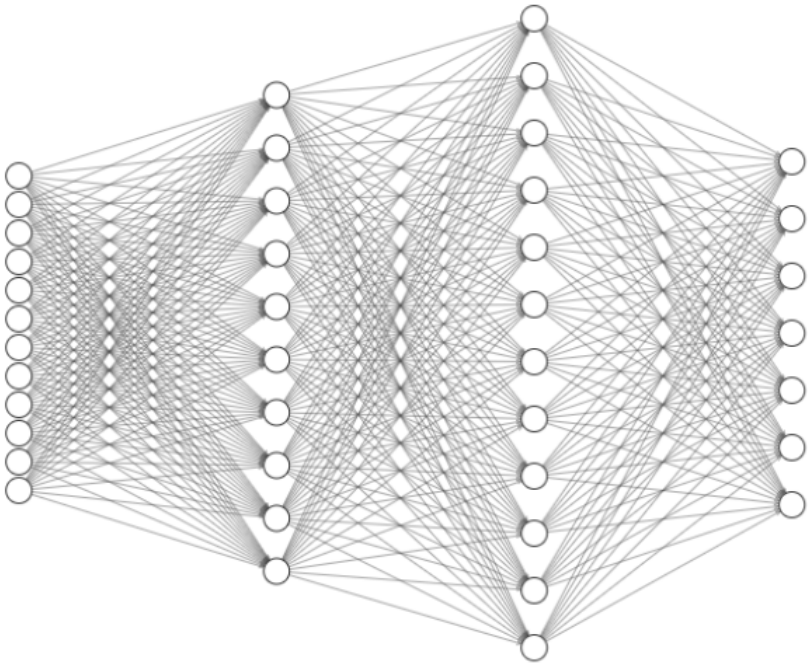


FIGURE 2.8 – Réseau de neurones entièrement connecté

2.3.3 Paramètres d'un CNN

En plus des paramètres du MLP (taux d'apprentissage, momentum, nombre d'époques, nombre de couches cachées et de neurones), qui sont utilisés dans le bloc des couches EC, plusieurs paramètres sont utilisés pour gérer le bloc de convolution. Cette section passe brièvement en revue tous ces paramètres.

2.3.3.1 Filtres

La convolution consiste à faire glisser une série de filtres ou noyaux de convolution sur l'image d'entrée. Le choix des filtres, de leur taille et de leur nombre (appelé aussi profondeur) est une tâche expérimentale. La "**profondeur de la couche convolutionnelle**" fait référence au nombre de noyaux utilisés, ce qui est à l'origine du nom "**Deep Learning**". Si nous utilisons un nombre q de filtres, l'image sera convoluée q fois. Ce processus crée q matrices filtrées appelées "**cartes de caractéristiques**". Un exemple de filtrage est illustré à la figure 2.9. Dans la plupart des architectures CNN prédéfinies, il existe toujours un choix par défaut spécifique à l'application pour laquelle le modèle a été construit [52].

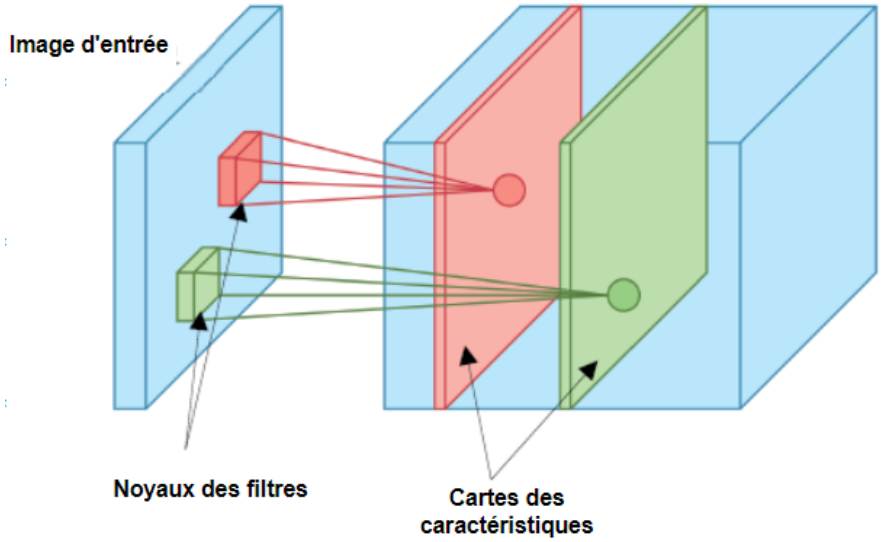


FIGURE 2.9 – Cartes de caractéristiques

2.3.3.2 Fonctions d'activation

Ce sont des fonctions mathématiques qui discernent la sortie d'un réseau neuronal. Elles déterminent si un neurone doit être activé ou non, en fonction de sa pertinence pour la prédiction du modèle. Il existe de nombreux types de fonctions d'activation, nous en citons certaines :

La fonction Rectified Linear Unit (ReLU) : Principalement appliquée dans les réseaux de neurones convolutifs après les couches de convolutions. Elle attribut toutes les valeurs négatives à 0 et garde les valeurs positives inchangées. Elle est utilisée afin d'augmenter la non-linéarité du réseau. Elle est décrite par l'équation suivante :

$$G(E) = max(0, E) = \begin{cases} E & si \quad E \geq 0 \\ 0 & sinon \end{cases} \tag{2.5}$$

Et dont l'allure est la suivante :

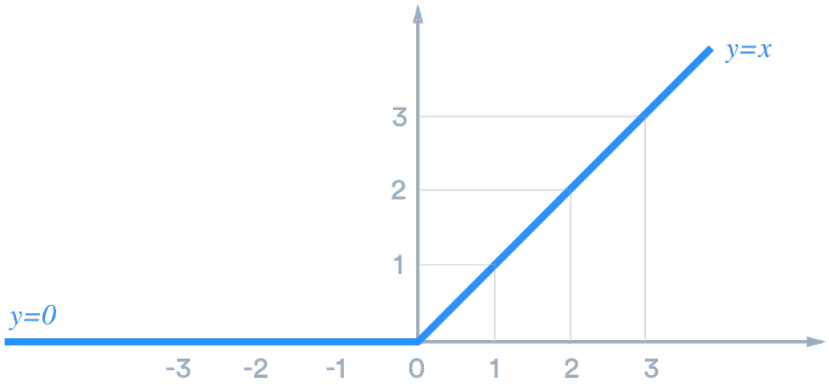


FIGURE 2.10 – Fonction d'activation ReLU

La fonction sigmoïde : Elle représente la fonction de répartition de la loi logistique. Elle est souvent utilisée dans les réseaux de neurones parce qu'elle est dérivable. La forme de la

dérivée de sa fonction inverse est extrêmement simple et facile à calculer, ce qui améliore les performances des algorithmes. Elle est décrite par la fonction suivante :

$$\alpha(x) = \frac{1}{1+e^{-x}} \quad \forall x \in \mathbb{R} \tag{2.6}$$

L'allure de la fonction est la suivante :

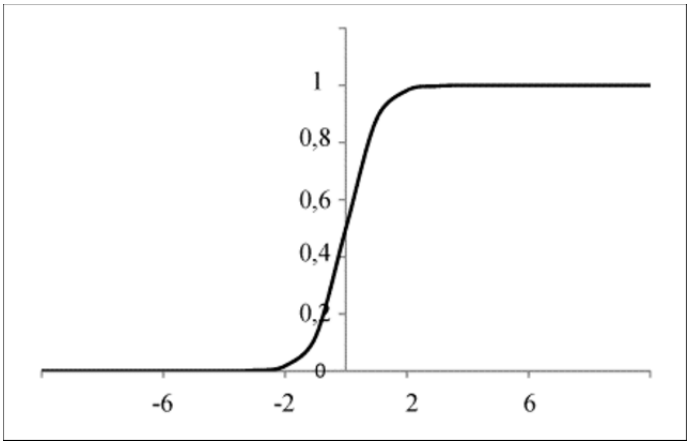


FIGURE 2.11 – Fonction d'activation sigmoïde

La fonction tangente hyperbolique (tanh) : Il s'agit en fait d'une version mathématiquement décalée de la fonction sigmoïde :

- La **sigmoïde** donne un résultat entre 0 et 1
- La fonction **tanh** donne un résultat entre -1 et 1

L'avantage de la fonction **tanh** est que les entrées négatives seront bien répertoriées comme négatives là où, avec la **sigmoïde**, les entrées négatives peuvent être confondus avec les valeurs proche de nulles. Elle a de meilleurs résultats que la fonction sigmoïde et la fonction ReLU dans un certain nombre de problèmes [53]. Elle est défini par l'équation suivante :

$$tanh(x) = \frac{sinh(x)}{cosh(x)} = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} \tag{2.7}$$

Et on a l'allure suivante :

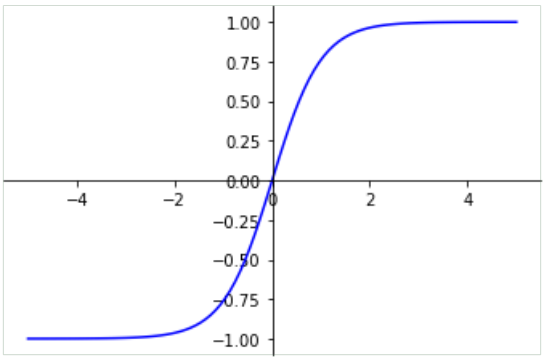


FIGURE 2.12 – Fonction d'activation tanh

La fonction Exponential Linear Unit (ELU) : Les ELU résultent de valeurs négatives et positives, permettant de pousser l'activation de l'unité moyenne vers zéro, dans une

procédure similaire à la normalisation mais avec une complexité de calcul plus faible [54], accélérant ainsi l'apprentissage du modèle. Elle est décrite par l'équation suivante :

$$G(E) = \begin{cases} E & si \quad E \geq 0 \\ \alpha(e^E - 1) & sinon \end{cases}, \quad \alpha \in \mathbb{R} \tag{2.8}$$

Qui donne l'allure suivante :

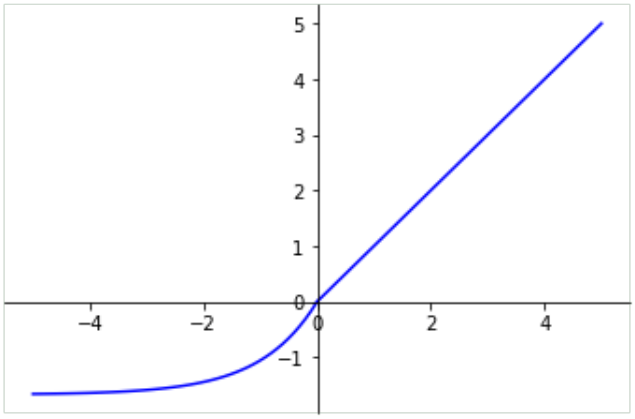


FIGURE 2.13 – Fonction d'activation ELU

La fonction softmax : Couramment utilisée dans la dernière couche du réseau, il s'agit d'une fonction de normalisation qui donne une approximation de la probabilité qu'une classe soit correcte. La valeur de la probabilité est calculée comme suit :

$$G(e_j) = \frac{e^{e_j}}{\sum_i e^{e_i}} \tag{2.9}$$

Où e_j est l'élément considéré j du vecteur d'entrée. La classe ayant la plus grande probabilité sera prise comme sortie du réseau de neurones.

2.3.3.3 Stride

Le stride est le nombre de pixels décalés sur la matrice d'entrée. Lorsque le stride est de 1, nous déplaçons les filtres d'un pixel à la fois. Lorsque le stride est de 2, nous déplaçons les filtres de 2 pixels à la fois et ainsi de suite comme l'indique la figure 2.14 :

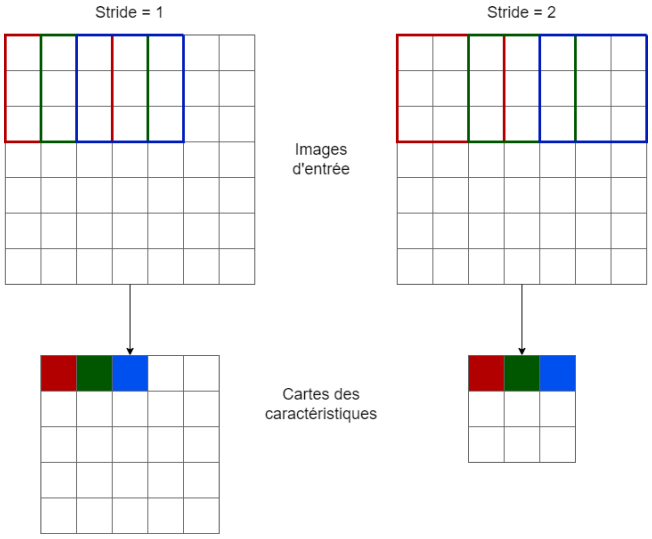


FIGURE 2.14 – Principe du stride

2.3.3.4 Zero-padding

Après chaque couche convolutive, la taille de l'image résultante est plus petite que celle de l'image d'entrée, car le filtre ne peut pas traiter les pixels de bordure. Si plusieurs couches convolutionnelles sont utilisées, l'image de sortie sera très petite par rapport à l'image d'entrée. Pour conserver la même taille que l'image d'entrée, le zero-padding ajoute des zéros aux bords de l'image, en tenant compte de la taille du filtre. De cette façon, quel que soit le nombre de couches convolutionnelles, l'image de sortie a la même taille que l'image d'entrée.

2.3.3.5 Drop-out

Il s'agit d'une technique utilisée lors de la phase d'apprentissage, destinée à réduire l'erreur de test et à éviter le problème de sur-apprentissage lié aux couches entièrement connectées. Le principe de l'utilisation du drop-out est de désactiver aléatoirement des neurones afin que le réseau soit plus réactif et puisse donc apprendre plus rapidement. La figure 2.15 présente un exemple de cette technique.

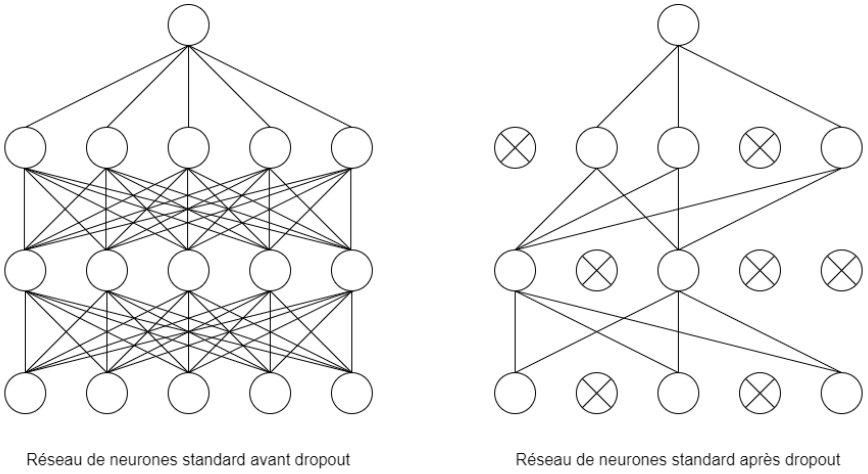


FIGURE 2.15 – Principe du drop-out durant l'entraînement d'un modèle

2.4 Modèles pré-entraînés

Très souvent dans le domaine de l'IA, on a recours à la réutilisation et l'adaptation d'un modèle fait par autrui pour notre application, ceci à travers l'utilisation de modèles pré-entraînés qui seront en partie (ou en totalité) ré-entraînés sur la base de données cible.

2.4.1 Qu'est ce qu'un modèle pré-entrainé ?

C'est un modèle de CNN qui a été entraîné sur une large base de données (par exemple ImageNet [55]) et dont les poids synaptiques ainsi que les coefficients des filtres de convolution ont été sauvegardés. Il y a des dizaines de modèles pré-entrainés disponibles en libre accès plus ou moins profond : Les modèles VGG (Visual Geometry Group) comme le VGG-16, VGG-19..., les modèles Inception (Inception-V1, Inception-V2...), les modèles ResNet (ResNet-V1, Inception-ResNet-V1...), AlexNet, LeNet, ect...

Le premier modèle pré-entraîné couramment utilisé est LeNet qui, comme dit précédemment, a été proposé par Yann LeCun en 1989 pour la reconnaissance des chiffres manuscrits.

2.4.2 Pourquoi utiliser un modèle pré-entraîné ?

L'utilisation d'un modèle pré-entrainé est privilégiée pour les raisons suivantes :

- La création d'une nouvelle architecture de modèle est une tâche coûteuse et chronophage.
- La création d'un modèle nécessite une étude théorique importante pour l'optimisation des paramètres afin d'avoir une solution efficace.
- L'utilisation de modèles pré-entraînés permet de ne pas avoir une base de données très grande ou des ressources matérielles importantes pour les entraînements répétés du modèle (vis-à-vis du nombre de paramètres à entraîner).

2.4.3 Apprentissage par transfert

L'apprentissage par transfert (Transfer Learning en anglais) est l'un des champs de recherche de l'apprentissage automatique qui vise à transférer des connaissances d'une ou plusieurs tâches sources vers une ou plusieurs tâches cibles. Il peut être vu comme la capacité d'un système à reconnaître et appliquer des connaissances et des compétences, apprises à partir de tâches antérieures, sur de nouvelles tâches ou domaines partageant des similitudes [56].

2.4.3.1 Stratégies et techniques de Transfer Learning

Le Transfer Learning repose sur une idée simple, celle de ré-exploiter les connaissances acquises dans d'autres configurations (sources) pour la résolution d'un problème particulier (cible). Dans ce contexte, on peut distinguer plusieurs approches selon ce que l'on souhaite transférer, comment réaliser le transfert et quand. Globalement, nous pouvons distinguer trois types de Transfer Learning [57] :

Apprentissage par transfert inductif : Dans la configuration de l'apprentissage par transfert inductif (ou Inductive Transfer Learning), les domaines source et cible sont les mêmes (mêmes données), mais les tâches source et cible sont différentes mais proches. L'idée consiste alors à utiliser les modèles existants pour réduire de manière avantageuse le champ d'application des modèles possibles (biais de modèle).

Apprentissage par transfert non supervisé : Comme dans le cas précédent, dans l'apprentissage par transfert non supervisé (ou Unsupervised Transfer Learning) les domaines source et cible sont similaires, mais les tâches sont différentes. Toutefois, les données des deux domaines ne sont pas labellisées.

Il est souvent plus facile d'obtenir de grandes quantités de données non labellisées, à partir de bases de données et de sources sur le Web par exemple, que des données labellisées. C'est pourquoi l'idée d'utiliser l'apprentissage non supervisé en combinaison avec le Transfer Learning suscite un grand intérêt essentiellement dans la détection d'anomalies.

A titre d'exemple, le Self-Taught Clustering est une approche qui permet de réaliser le clustering de petites collections de données cibles non labellisées, avec l'aide d'une grande quantité de données sources non labellisées. Cette approche s'avère plus performante que les approches de pointe traditionnellement utilisées, lorsque les données cibles sont labellisées de manière non pertinente.

Apprentissage par transfert transductif : Dans cette configuration, aussi dite Transductive Transfer Learning, les tâches sources et cibles sont similaires, mais les domaines correspondants sont différents soit en termes de données ou de distributions de probabilités marginales. Comme dans notre cas avec le Transfer Learning pour la classification d'images médicales.

2.4.3.2 Comment utiliser le Transfer Learning ?

On peut distinguer 2 types de stratégies :

Utilisation de modèles pré-entraînés comme extracteurs de caractéristiques : L'idée est de réutiliser un réseau pré-entraîné sans sa couche finale. Ce nouveau réseau fonctionne alors comme un extracteur de caractéristiques fixes pour la réalisation d'autres tâches, méthode adaptée quand les images sources et cibles ne constituent pas les mêmes classes.

Afin d'illustrer cette stratégie, prenons le cas où nous souhaitons créer un modèle capable d'identifier les stades de la maladie d'Alzheimer à partir de son image IRM. Il est alors possible d'utiliser les premières couches du modèle de réseau de neurones convolutif AlexNet, initialement entraîné sur la base d'images **ImageNet** [55] pour la classification d'images.

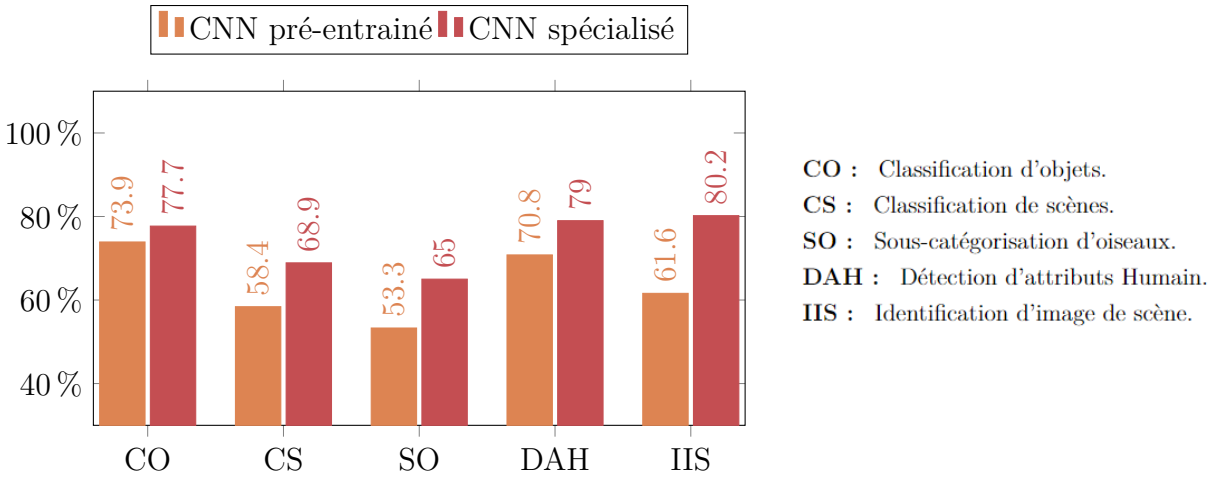


FIGURE 2.16 – Comparatif de performances entre des modèles entrainés de zéro et des modèles pré-entrainés faisant appel au transfert learning

Cette technique a très souvent prouvé son efficacité dans différentes problématique comme l'indique la figure ci-dessus.

Ajustement de modèles pré-entraînés : Il s'agit d'une technique plus complexe, dans laquelle non seulement la dernière couche est remplacée pour réaliser la classification ou la régression, mais d'autres couches sont également ré-entraîner de manière sélective. En effet, les réseaux neuronaux profonds sont des architectures hautement configurables avec divers paramètres. De plus, alors que les premières couches capturent les caractéristiques génériques, les dernières couches se concentrent davantage sur la tâche spécifique à accomplir.

Cette stratégie permet de réutiliser les connaissances en termes d'architecture globale du réseau et d'exploiter ses états comme point de départ pour l'entraînement. Elle permet donc d'obtenir de meilleures performances avec un temps d'entraînement plus court.

2.4.4 Modèles utilisés

Dans notre étude on a pu expérimenter plusieurs modèles communément utilisés dans la classification des images médicales et en se basant sur des travaux récents, essentiellement les travaux de A. Loddo et al. [5], Ainsi, nous avons sélectionné 3 architectures très populaires : AlexNet, ResNet-101 et le Inception-ResNet-V2 qui vont être décrites plus amplement dans les sections qui suivent.

2.4.5 AlexNet

AlexNet est un CNN, conçu par Alex Krizhevsky en collaboration avec Ilya Sutskever et Geoffrey Hinton. Il a été mis en place la première fois pour le concours ImageNet Large Scale Visual Recognition Challenge le 30 septembre 2012. Le réseau a obtenu une erreur de 15,3% dans le top 5, soit plus de 10,8% de moins que le second. Le résultat principal de l'article original était que la profondeur du modèle était essentielle pour sa haute performance, ce qui était coûteux en calcul, mais faisable grâce à l'utilisation de GPU pendant l'entraînement [58].

AlexNet, qui employait un CNN à 8 couches, a montré pour la première fois que les caractéristiques obtenues par apprentissage peuvent transcender les caractéristiques conçues manuellement, brisant ainsi le paradigme précédent en vision par ordinateur.

2.4.5.1 Architecture du modèle

Les architectures d'**AlexNet** et de **LeNet** sont très similaires, comme l'illustre la figure 2.17.

Les philosophies de conception d'**AlexNet** et de **LeNet** sont très similaires, mais il existe également des différences significatives. Premièrement, **AlexNet** est beaucoup plus profond que **LeNet**. **AlexNet** est composé de huit couches : cinq couches convolutives, trois couches de pooling, deux couches cachées entièrement connectées et une couche de sortie entièrement connectée. Deuxièmement, **AlexNet** utilise la ReLU au lieu de la sigmoïde comme fonction d'activation [59].

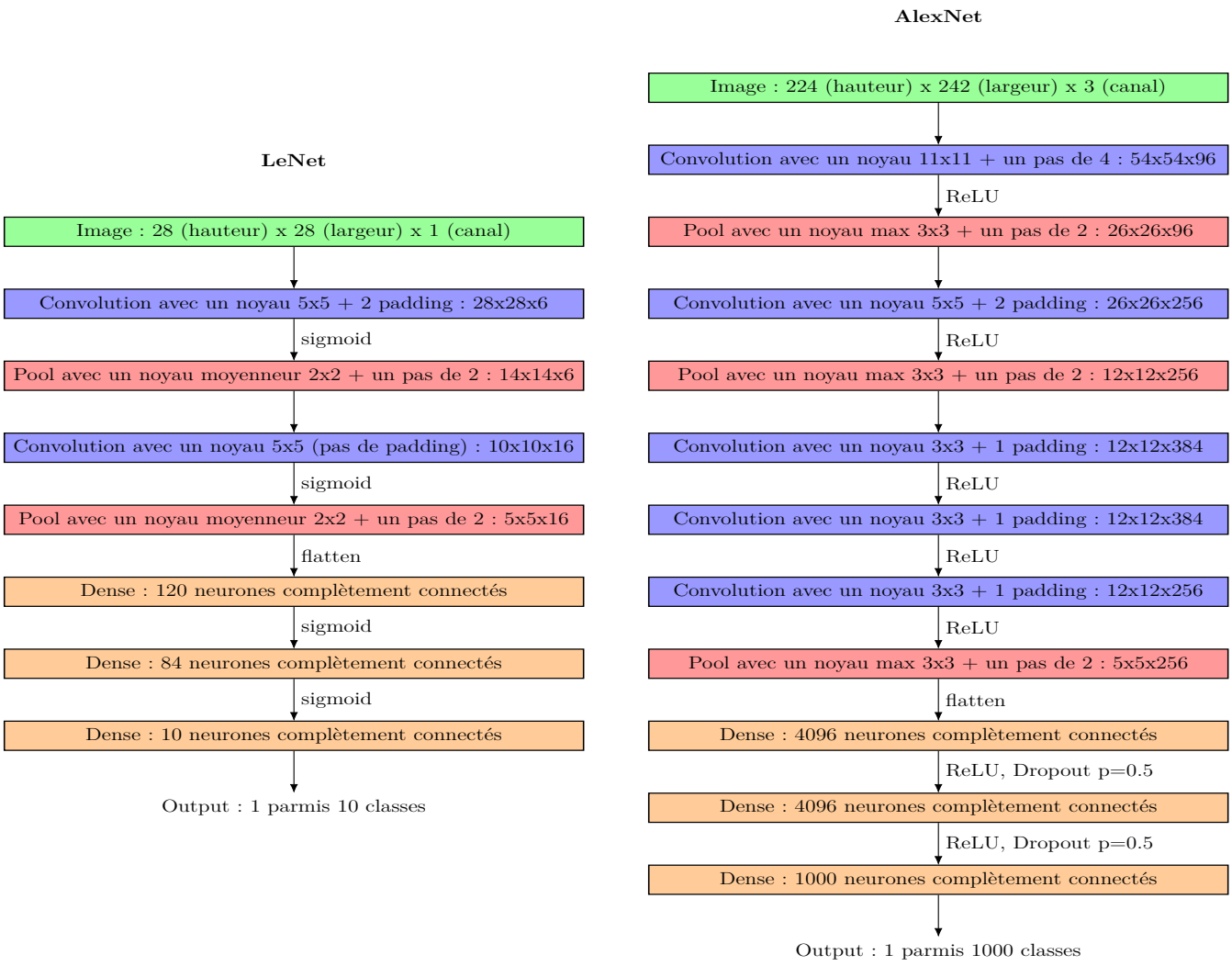


FIGURE 2.17 – Comparatif entre les architectures des modèles LeNet et AlexNet

2.4.6 ResNet-101

Lancée en 2015 par Microsoft Research Asia, l'architecture ResNet (avec ses trois réalisations ResNet-50, ResNet-101 et ResNet-152) a obtenu de très bons résultats dans les concours ImageNet et MS-COCO. Il s'avère que l'idée centrale exploitée dans ces modèles, à savoir, les connexions résiduelles, améliore considérablement le flux de gradient, permettant ainsi l'entraînement de modèles beaucoup plus profonds, avec des dizaines voire des centaines de couches [60]. Cependant, l'augmentation de la profondeur du réseau ne se fait pas simplement en empilant des couches. Les réseaux profonds sont difficiles à entraîner en raison du fameux problème du gradient évanescent : lorsque le gradient est rétropropagé vers les couches précédentes, la multiplication répétée peut rendre le gradient infiniment petit. Par conséquent, plus le réseau est profond, plus ses performances sont saturées, voire se dégradent rapidement. L'idée centrale de ResNet est d'introduire ce que l'on appelle une "connexion de raccourci d'identité" ; ou "connexion résiduelle" qui saute une ou plusieurs couches, comme le montre la figure 2.18

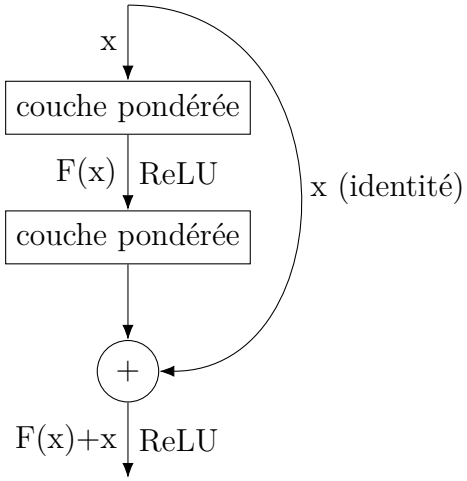


FIGURE 2.18 – Bloc résiduel

Les auteurs de l'article "**Deep residual learning for image recognition**" [61] affirment que l'empilement des couches ne devrait pas dégrader les performances du réseau, car nous pourrions simplement empiler les mappages d'identité sur le réseau actuel, et l'architecture résultante aurait les mêmes performances. Cela indique que le modèle le plus profond ne devrait pas produire une erreur d'apprentissage plus élevée que ses homologues moins profonds. Ils supposent qu'il est plus facile de laisser les couches empilées s'adapter à une cartographie résiduelle que de les laisser s'adapter directement à la cartographie sous-jacente souhaitée. C'est précisément ce que le bloc résiduel ci-dessus permet de faire.

On a alors les différents réseaux ResNet communément utilisés de 18 à 152 couches dont on peut décrire la constitution dans le tableau suivant :

TABLE 2.1 – Architectures des différents modèles ResNet

Couches	Taille de sortie	18-couches	34-couches	50-couches	101-couches	152-couches
conv1	112×112	$7\times7, 64, stride = 2$				
conv2..x	56×56	$3\times3, max - Pool, stride = 2$				
		$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, & 64 \\ 3\times3, & 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, & 64 \\ 3\times3, & 64 \\ 1\times1, & 256 \end{bmatrix}\times3$
conv3..x	28×28	11	22	33	44	55
conv4..x	14×14	1	22	3	44	5
conv5..x	7×7	11	2	33	4	55
	1×1	average-Pool, 1000-d entièrement connectées, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

La figure 2.19 dresse un comparatif entre un exemple d'architecture ResNet avec les couches résiduelles et des CNN contenant un enchaînement linéaire classique.

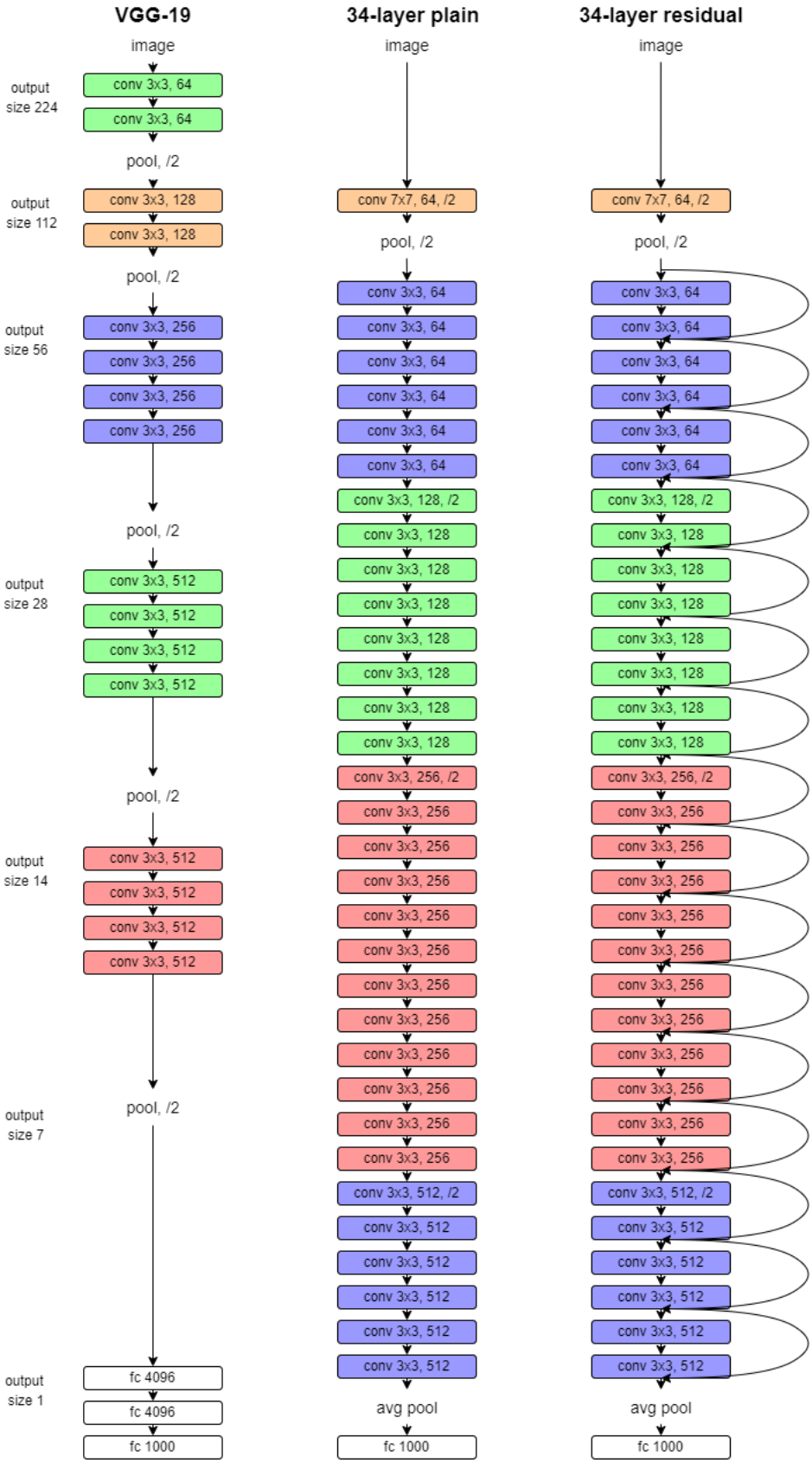


FIGURE 2.19 – Comparatif d'architecture à enchainement linéaire classique avec ResNet-34

Les réseaux convolutifs ResNet étant de plus en plus profonds, le nombre de paramètres à entraîner est extrêmement grand et peut être résumé dans le tableau 2.2

TABLE 2.2 – Nombre de paramètres des réseaux ResNet

Nombre de couches	Nombre de paramètres (millions)
Resnet 18	11.174
Resnet 34	21.282
Resnet 50	23.521
Resnet 101	42.513
Resnet 152	58.157

2.4.7 Inception-ResNet-V2

Afin de comprendre l'architecture Inception-ResNet-V2, nous présentons d'abord le bloc d'Inception ainsi que l'architecture GoogLeNet qui est à la base de l'Inception-ResNet-V2.

2.4.7.1 Bloc d'Inception

Un Module d'Inception est un bloc de réseaux de neurones qui vise à approximer une structure locale clairsemée optimale dans un CNN. En termes simples, il nous permet d'utiliser plusieurs types de filtres, au lieu d'être limités à une seule taille de filtre, dans un seul bloc d'image, que nous concaténons ensuite et transmettons à la couche suivante [62]. On le décrit dans la figure 2.20

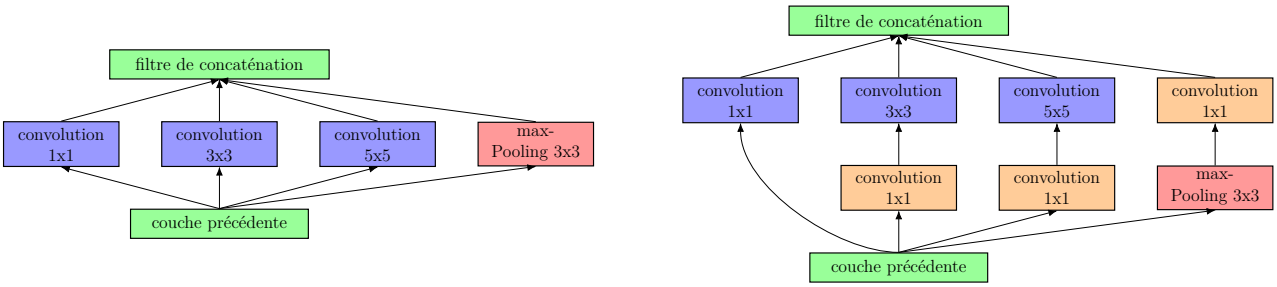


FIGURE 2.20 – Module d'Inception : à gauche version naïve, à droite avec réduction de dimensions

2.4.7.2 GoogLeNet

GoogLeNet est un réseau neuronal convolutif profond à 22 couches qui est une variante de l'Inception Network, un CNN profond développé par les chercheurs de Google.

L'architecture GoogLeNet présentée lors de l'ImageNet Large-Scale Visual Recognition Challenge 2014 (ILSVRC14) a résolu des tâches de vision par ordinateur telles que la classification d'images et la détection d'objets [63]. Le tableau ci-dessous indique la constitution de GoogLeNet :

Type	**Taille du patch/ Stride**	**Sortie Taille**	**Profondeur**	#1×1	#3×3 **réduit**	#3×3	#5×5 **réduit**	#5×5	**Pool Proj**	**params**	**ops**
Convolution	7×7/2	112×112 × 64	1							2.7K	34M
Max-pool	3×3/2	56×56 × 64	0								
convolution	3×3/1	56×56 × 192	2		64	192				112K	360M
Max-pool	3×3/2	28×28 × 192	0								
Inception (3a)		28×28 × 256	2	64	96	128	16	32	32	159K	128M
Inception (3b)		28×28 × 480	2	128	128	192	32	96	64	380K	304M
Max-pool	3×3/2	14×14 × 480	0								
Inception (4a)		14×14 × 512	2	192	96	208	16	48	64	364K	73M
Inception (4b)		14×14 × 512	2	160	112	224	24	64	64	437K	88M
Inception (4c)		14×14 × 512	2	128	112	224	24	64	64	463K	100M
Inception (4d)		14×14 × 528	2	112	144	288	32	64	64	580K	119M
Inception (4e)		14×14 × 832	2	256	160	320	32	128	128	840K	170M
Max-pool	3×3/2	7×7 × 832	0								
Inception (5a)		7×7 × 832	2	256	160	320	32	128	128	1072K	54M
Inception (5b)		7×7 × 1024	2	384	192	384	48	128	128	1388K	71M
Avg-pool	7×7/1	1×1 × 1024	0								
Dropout(40%)		1×1 × 1024	0								
Linéaire		1×1 × 1000	1							1000K	1M
Softmax		1×1 × 1000	0								

TABLE 2.3 – Architecture de GoogLeNet

À sa création, l'architecture GoogLeNet a été conçue pour être d'une efficacité de calcul accrue par rapport à certaines de ses prédécesseurs ou à des réseaux similaires créés à l'époque.

L'une des méthodes utilisées par GoogLeNet pour atteindre cette efficacité consiste à réduire l'image d'entrée tout en conservant les informations spatiales importantes [63].

L'architecture GoogLeNet se compose de neuf modules d'Inception séparés par deux couches de max-pooling qui servent à diminuer de la dimensionnalité du problème pour ajouter de l'efficacité dans les calculs.

Chose importante qu'amène le modèle GoogLeNet est la mise en place de **classifieurs auxiliaires**. Pour combattre le problème de la descente du gradient évanescent, qui touche les réseaux de neurones profonds empêchant un apprentissage de qualité des classifieurs auxiliaires sont ajoutés aux couches intermédiaires de l'architecture, à savoir la troisième (Inception (4a)) et la sixième (Inception (4d)) durant l'entraînement. Le but d'un classifieur auxiliaire est d'effectuer une classification basée sur les entrées dans la section médiane du réseau et d'ajouter la perte calculée pendant la formation à la perte totale du réseau (pondérée par un facteur).

2.4.7.3 Inception-Resnet-V2

L'architecture du modèle est décrite dans la figure 2.21 :

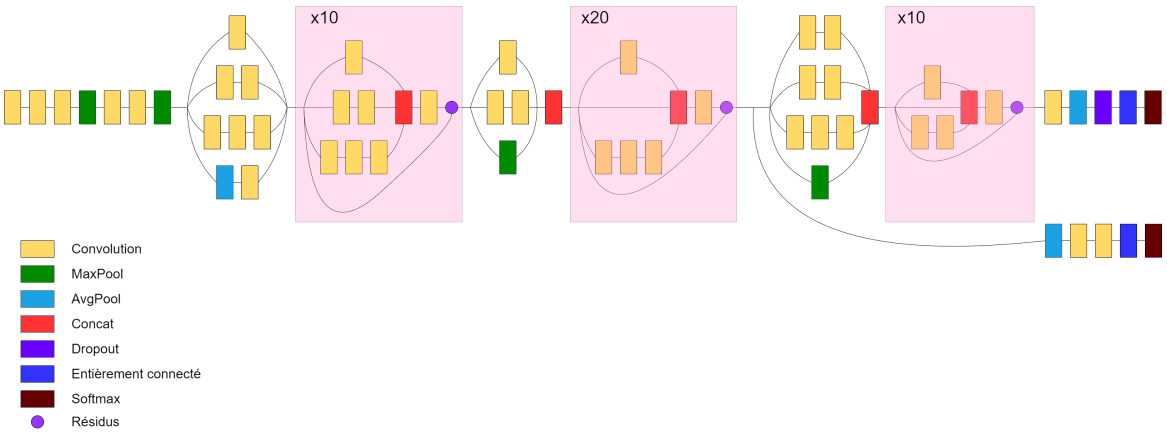


FIGURE 2.21 – Architecture du Inception-Resnet-V2

Inspiré par les performances du ResNet, un module d'Inception hybride a été proposé. Il existe deux sous-versions d'Inception ResNet, à savoir V1 et V2 [64]. Les différences entre ces deux sous-versions sont mineures, résidant principalement dans les réglages des hyperparamètres. On aura alors les ajouts suivants au niveau de ce modèle :

- Pour que l'addition résiduelle fonctionne, l'entrée et la sortie après convolution doivent avoir les mêmes dimensions. Par conséquent, nous utilisons des convolutions 1x1 après les convolutions originales, pour correspondre aux tailles de profondeur (la profondeur est augmentée après la convolution).

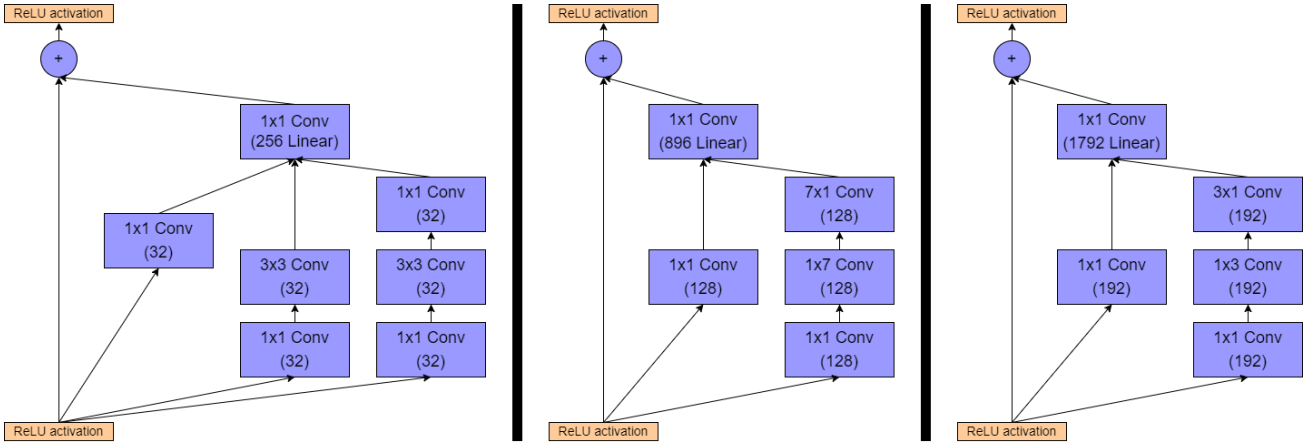


FIGURE 2.22 – Modules d'Inception A,B,C dans un ResNet-Inception

- L'opération de mise en commun à l'intérieur des modules principaux de création a été remplacée en faveur des connexions résiduelles.

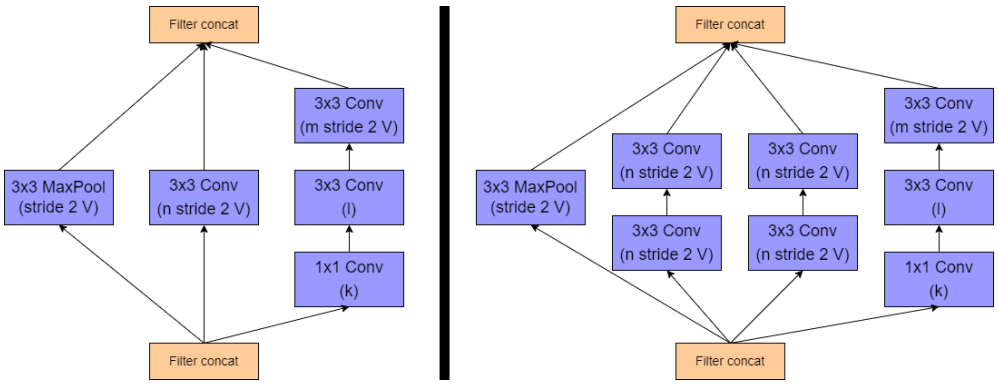


FIGURE 2.23 – Bloc de réduction A (réduction de taille de 35x35 à 17x17) et Bloc de réduction B (réduction de taille de 17x17 à 8x8)

- Les réseaux comportant des unités résiduelles plus profondément dans l'architecture provoquaient la "mort" du réseau si le nombre de filtres dépassait 1000. Par conséquent, pour augmenter la stabilité, les auteurs ont mis à l'échelle les activations résiduelles par une valeur d'environ 0,1 à 0,3.

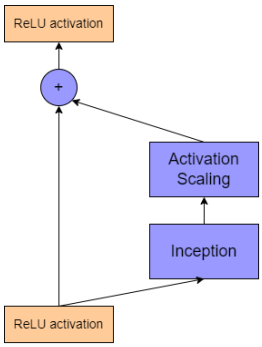


FIGURE 2.24 – Mise à l'échelle des unités résiduelles

- La configuration finale du réseau pour Inception V4 et Inception-ResNet est la suivante :

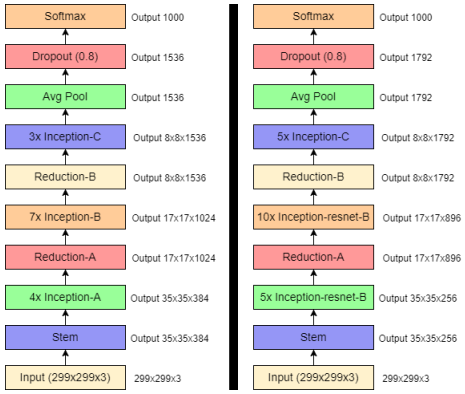


FIGURE 2.25 – Architecture finale de l'Inception-ResNet-V2

2.5 Conclusion

Dans ce chapitre, nous avons commencé par évoquer l'histoire des réseaux de neurones artificiels, puis nous avons détaillé les CNN et leurs principaux paramètres, en terminant par les architectures et principe des modèles pré-entraînés. Dans le prochain chapitre, nous présenterons

l'architecture adoptée dans ce travail pour la classification des différents niveaux de démence de la maladie d'Alzheimer, à savoir une fusion de caractéristiques générées à partir d'un descripteur nommé Histogramme des Gradients Orientés (ou Histogram of Oriented Gradients : HOG) et des caractéristiques générées par un CNN développé de zéro. Ceci en proposant différents classifieurs pour la tâche de classification.

CHAPITRE 3

RÉSULTATS EXPÉRIMENTAUX

3.1 Introduction

Ce chapitre est dédié à l'explication de la méthodologie de travail ainsi que l'analyse des performances des 3 modèles CNN pré-entrainés pour la classification des différents stades de démence de la maladie d'Alzheimer. Nos tests ont été menés sur une base de données benchmark à savoir Kaggle. Plus précisément, la première partie des expériences examine et compare les architectures pré-entraînées en se basant sur l'apprentissage par transfert et en deuxième partie, nous investiguons la performance d'un système classique composé d'un descripteur compétitif dans tous les domaines de la reconnaissance de formes associé à plusieurs classifieurs type K-plus proches voisins (KNN), Support Vector Machine (SVM), Naïve Bayes et Random Forest (RF). Enfin, nous arriverons à une fusion hybride de caractéristiques qui prêtera les meilleures performances.

L'implémentation des systèmes est effectuée sous environnement Anaconda du développement en Python. Comme l'apprentissage des CNN est une tâche très gourmande en capacités de calcul, l'exécution des programmes a été réalisée sur un ordinateur doté d'une carte graphique Nvidia RTX2070 et 16GB de mémoire, ainsi que sur des machines virtuelles dans le Cloud de Paperspace Gradient. Ce dernier est dédié aux applications Deep Learning qui requièrent l'utilisation de plusieurs bibliothèques spécifiques comme la Keras, PyTorch, et TensorFlow. La machine virtuelle est un GPU doté d'une carte graphique Nvidia RTX5000 et 16GB de mémoire.

3.2 Ensembles de données utilisés

Nous commencerons par décrire la base de données utilisée ainsi que la distribution de ses échantillons :

3.2.1 Kaggle

Kaggle est une plateforme web appartenant à Google mettant à disposition un grand nombre de bases de données et organisant des compétitions en science des données. L'entreprise a été

fondée en 2010 par Anthony Goldbloom. Cette plateforme représente la plus grande source de bases de données disponibles en libre accès.

L'ensemble de données sur lequel nous avons travaillé est alors disponible sur Kaggle [6] et est déjà subdivisé en 2 ensembles ; entraînement et test. Ainsi, il se compose d'un total de 5121 images axiales d'entraînement. Chaque image est étiquetée avec le niveau de démence correspondant : pas de démence (Non Demented), démence très légère (Very Mild Demented), démence légère (Mild Demented) et démence modérée (Moderate Demented). L'âge des patients est inconnu, et aucune autre donnée les concernant n'est fournie. L'ensemble de données comprend 2560 sujets sains et 2561 atteints de démence (1792 atteints de démence très légère, 717 de démence légère, 52 de démence modérée). Les images ont une résolution de 176 $\times$ 208 pixels. Des exemples sont présentés dans la figure 3.1. Les images sont au format jpeg et aucun détail sur le statut du patient n'a été fourni pour cet ensemble de données. En ajoutant les images de l'ensemble de test : 448 Very Mild Demented, 179 Mild Demented, 12 Moderate Demented et 640 Non Demented. Nous arrivons alors à un total de 6400 images constituant notre ensemble de données qu'on subdivisera de deux manières différentes (la subdivision originale ainsi qu'une autre subdivision plus avantageuse à l'entraînement).

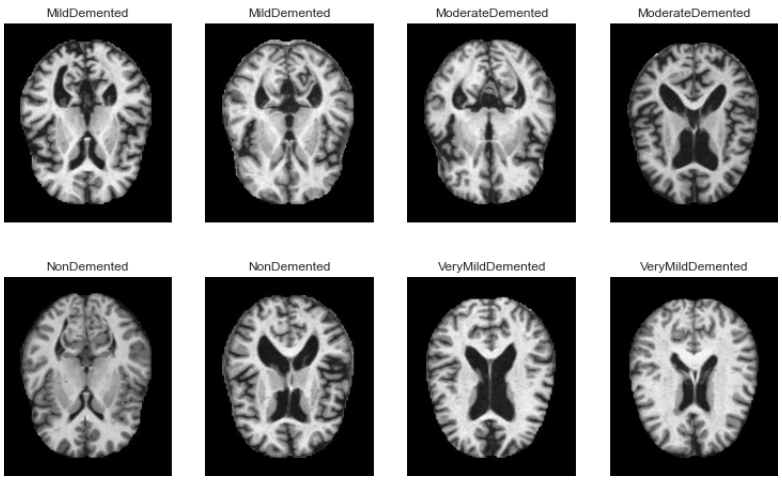


FIGURE 3.1 – Échantillons d'images IRM de l'ensemble de données Kaggle

3.2.1.1 Distribution des échantillons

On a une distribution très déséquilibrée dans cet ensemble de données comme l'indique la figure 3.2. Comme on peut le voir, l'ensemble de données est grandement déséquilibré et pour y remédier on procède à une augmentation de données, son implémentation sera plus amplement décrite plus tard dans la section "Protocole utilisé".

Cette distribution est dans le cas d'une classification multiclasse, c'est-à-dire considérer les 4 niveaux de démence. Cependant, on réalise aussi une classification entre sujet sain et sujet atteint que nous appelons classification binaire (2 classes uniquement). Cette classification binaire se caractérise par une distribution de données équilibrée telle qu'indiqué dans le tableau suivant :

TABLE 3.1 – Distribution binaire de l'ensemble de données Kaggle

Classe	Nombre d'images d'entraînement	Nombre d'images total
Sain	2560	3200
Malade	2561	3200

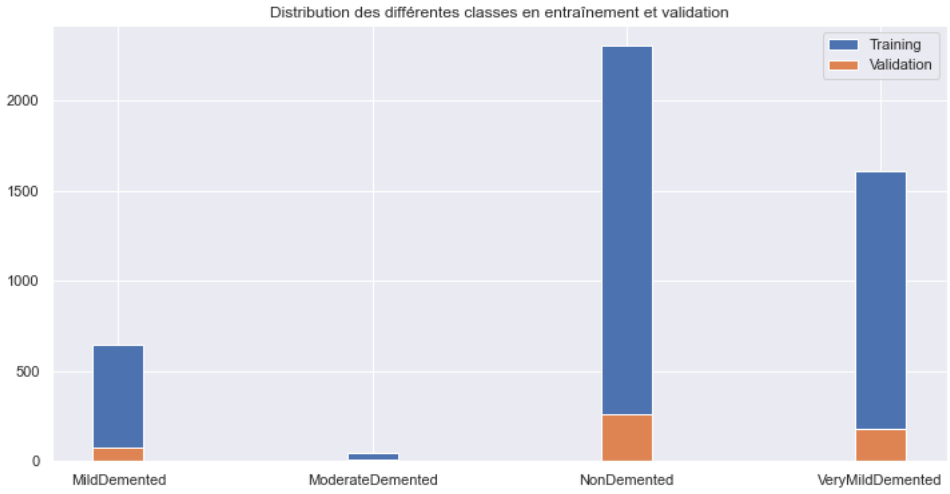


FIGURE 3.2 – Distribution des classes de l'ensemble de données Kaggle

3.3 Logiciels, libraires et matériel

3.3.1 Python

Python est un langage de programmation interprété, multi-paradigme et multiplateformes. Il favorise la programmation impérative structurée, fonctionnelle et orientée objet. Il est doté d'un typage dynamique fort, d'une gestion automatique de la mémoire par ramasse-miettes et d'un système de gestion d'exceptions.

Le langage de programmation Python a été créé en 1989 par Guido van Rossum, aux Pays-Bas. Le nom Python vient d'un hommage à la série télévisée Monty Python's Flying Circus dont G. van Rossum est fan. La première version publique de ce langage a été publiée en 1991.

3.3.2 Scikit-learn

Scikit-learn est une bibliothèque libre Python destinée à l'apprentissage automatique. Elle est développée par de nombreux contributeurs notamment dans le monde académique.

Elle propose dans son framework de nombreuses bibliothèques d'algorithmes à implémenter, clé en main. Ces bibliothèques sont à disposition notamment des data scientists.

Elle comprend notamment des fonctions pour estimer des forêts aléatoires, des régressions logistiques, des algorithmes de classification, et les machines à vecteurs de support (SVM). Elle est conçue pour s'harmoniser avec d'autres bibliothèques libres Python, notamment NumPy et Pandas.

3.3.3 Keras

Keras est une bibliothèque logicielle open-source qui fournit une interface Python pour les réseaux de neurones artificiels.

Depuis la version 2.4, seul TensorFlow est pris en charge. Conçu pour permettre l'expérimentation rapide de réseaux neuronaux profonds, il se veut convivial, modulaire et extensible. Il a été développé dans le cadre de l'effort de recherche du projet ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System) [65].

3.3.4 Tensorflow

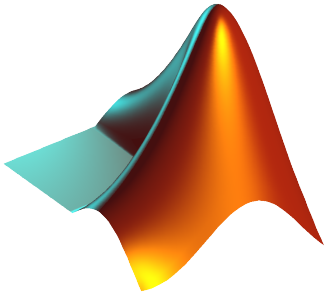
TensorFlow est un framework complet et open source permettant de créer des applications d'apprentissage automatique. Il s'agit d'une boîte à outils de mathématiques symboliques qui exécute une variété de tâches, notamment la formation et l'inférence de réseaux neuronaux profonds. Il permet aux programmeurs de construire des applications d'apprentissage automatique en utilisant une variété d'outils, de cadres et de ressources communautaires.

Il a été mis en place par Google et est aujourd'hui le package d'apprentissage profond le plus utilisé au monde. L'apprentissage automatique est utilisé par Google dans tous ses produits pour améliorer la recherche, la traduction, les légendes d'images et les recommandations.

Il a été initialement publié fin 2015, et la première version stable a suivi en 2017. Il est gratuit et open-source, grâce à la licence Apache Open Source. Sans rien payer à Google, vous pouvez l'utiliser, le modifier et redistribuer la version mise à jour moyennant un prix.

Pour notre part, à travers son API Keras, il a été au coeur de notre projet dans la conception des différents modèles de réseaux de neurones profonds pour nos classifications mais pas que, on a utilisé Tensorflow aussi pour la préparation des données et l'évaluation des différents modèles.

3.3.5 Matlab



MATLAB (abréviation de "MATrix LABoratory") est un langage de programmation multi-paradigme propriétaire et un environnement de calcul numérique développé par MathWorks. MATLAB permet la manipulation de matrices, le tracé de fonctions et de données, l'implémentation d'algorithmes, la création d'interfaces utilisateur et l'interfaçage avec des programmes écrits dans d'autres langages.

Bien que MATLAB soit principalement destiné au calcul numérique, une boîte à outils optionnelle utilise le moteur symbolique MuPAD permettant d'accéder aux capacités de calcul symbolique. Un package supplémentaire, Simulink, ajoute la simulation graphique multi-domaines et la conception à base de modèles pour les systèmes dynamiques et embarqués.

On a utilisé Matlab comme support à Python dans l'entraînement de nos modèles de deep learning, on a alors mis exactement les mêmes configurations d'entraînement sur Matlab et on a pris les modèles les plus performants à chaque fois.

3.3.6 Matériel (Hardware)

Notre problématique étant une problématique relativement complexe on a eu recours à des réseaux de neurones profonds (DNN) pour la classification, ayant pour certains des millions de paramètres à calibrer durant l'entraînement du modèle. Cette tâche demande alors une configuration matérielle respectant plusieurs critères :

- Une force de calcul assez importante (à travers des CPU ou des GPU).
- Une robustesse du matériel (pour éviter les arrêts durant les tâches)
- Un stockage suffisant.

Dans notre projet, comme dans tout projet faisant intervenir les DNN, la tâche la plus gourmande en ressource est l'entraînement des différents modèles et pour se faire on a eu recours à plusieurs solutions qu'on décrit ci-dessous :

3.3.6.1 Google Colaboratory

Colaboratory, souvent raccourci en "Colab", est un produit de Google Research. Colab permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement particulièrement adapté au machine learning, à l'analyse de données et à l'éducation. En termes plus techniques, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder sans frais à des ressources informatiques, dont des GPU.

On a principalement utilisé Google colab dans les entraînements relativement simples et court pour plusieurs raisons :

- Les ressources proposées par Google ne sont pas définies et sont fluctuantes (donc instable).
- La qualité des GPU proposés est souvent relativement faible.
- L'outil est fait pour des entraînements ne dépassant pas les 30min, le noyau s'arrête au bout de 30min d'inactivité (de l'utilisateur).

Et pour ces raisons on a opté pour les plus grands entraînements à d'autres plateformes.

3.3.6.2 Paperspace gradient

Gradient est une suite d'outils conçue pour accélérer l'IA dans le cloud et l'apprentissage automatique. C'est un outil similaire à Google colab en y ajoutant des moyens de déploiement des modèles sur le cloud, faisant de lui un outil parfait pour du développement opérationnel

(DevOps). Paperspace propose une solution gratuite de leur suite Gradient, on a opté pour la version payante pour particulier à hauteur de $ 8 comme abonnement mensuel, on a eu accès à de bien meilleurs GPU que ceux de Google Colab pour nos entraînements.

On a utilisé cette plateforme sur les entraînements plus conséquents. On a eu de bien meilleurs GPU mis à disposition (et aussi une meilleure visibilité sachant quel type de machine on peut avoir), on a utilisé pour nos entraînement essentiellement la configuration suivante :

GPU Nvidia Quadro RTX 5000 avec les caractéristiques suivantes :

Architecture : Turing

Nombre de processeurs de shaders : 3072

Fréquence de noyau : 1620 MHz

Fréquence en mode Boost : 1815 MHz

Nombre de transistors : 13.6 millions

Processus technologique de fabrication : 12 nm

Consommation d'énergie : 230 W

Vitesse de texturation : 348.5

Stockage de mémoire vive (RAM) : 32 Go

Stockage disque : 5 Go

Ce qui en fait une configuration assez suffisante pour nos entraînements.

3.4 Métriques utilisées

Notre problème est un problème de classification où les classes sont déséquilibrées, comme la majorité des problèmes dans le domaine biomédical. Comme indiqué plus haut, on a la distribution dans notre ensemble de données d'entraînement Kaggle qu'on peut résumer dans le tableau suivant :

Table 3.2 – Distribution des classes de l'ensemble de données Kaggle

Classes	Nombre d'images IRM
Non Demented	2560
Very Mild Demented	1792
Mild Demented	717
Moderate Demented	52

Et par conséquent le choix des métriques est important pour mesurer réellement la qualité du modèle. Par exemple, la métrique la plus utilisée dans le domaine de l'apprentissage automatique est la précision (accuracy) et dans ce genre de cas elle n'est pas très utile, n'indiquant pas la qualité du modèle comme on peut le lire dans les travaux de M. Galar et al. [66]. Par conséquent on va opter pour d'autres métriques plus adaptées pour juger de la qualité de notre modèle.

Comme notre projet se divise en deux parties, d'une part on a la classification multiclasse comme expliqué plus haut qui est un problème de classification très déséquilibré et donc on a fait le choix d'utiliser les métriques suivantes pour juger de la qualité de notre modèle :

- Le AUC ROC Score.
- Le F1-score.
- L'exactitude (Accuracy).
- La matrice de confusion

Vis-à-vis de la classification binaire entre patients malades et sains étant une classification binaire équilibrée comme expliqué dans le tableau 3.1 et pour se faire on a préféré utilisé les métriques suivantes :

- L'exactitude (Accuracy).
- La sensibilité.
- La spécificité.

On va décrire certaines de ces métriques plus en détails dans les prochaines sections :

3.4.1 AUC ROC Score

Les métriques de rang, comme la courbe ROC, se concentrent sur l'évaluation des classifieurs sur la base de leur efficacité à séparer les classes à travers les probabilités que chaque échantillon fasse partie d'une classe ou l'autre.

Une courbe ROC est un graphique de diagnostique permettant de résumer le comportement d'un modèle en calculant le taux de faux positifs (formule 3.2) et le taux de vrais positifs (formule 3.1) pour un ensemble de prédictions par le modèle sous différents seuils :

$$TauxVraiPositif = \frac{VP}{VP + FN} \tag{3.1}$$

$$TauxFauxPositif = \frac{FP}{FP + VN} \tag{3.2}$$

Il en résulte un graphique de courbe nous permettant de juger de la qualité de notre modèle, comme indiqué dans la figure 3.3

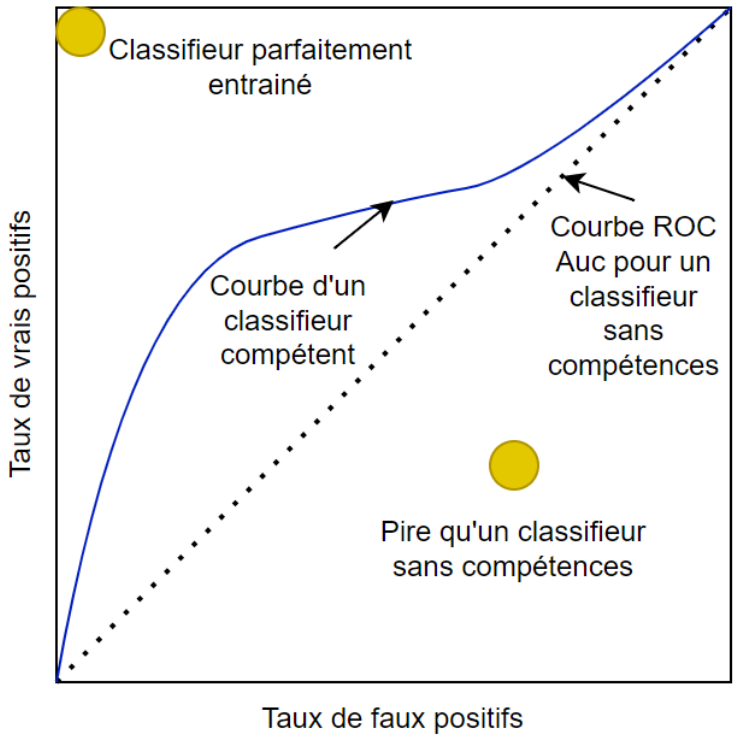


FIGURE 3.3 – Courbe ROC

La courbe ROC indique la qualité d'un modèle. Plus la surface au dessous de la courbe tend vers 1, meilleur est le modèle, donc le AUC ROC Score n'est que l'intégrale de la fonction générée ; ce qui représente tout simplement la surface au dessous de la courbe.

Cette métrique a été avant tout faite pour de la classification binaire et donc compare la qualité d'un modèle comparant deux classes distinctes, néanmoins on peut l'adapter à un problème multiclasse comme dans notre cas et cela à travers deux stratégies :

OvO (One vs One) : on aura alors toutes les possibilités entre chaque couple de classes avec chacun un score défini qui indique la qualité du modèle à différencier entre les deux classes. Dans notre cas on aura alors les 6 combinaisons suivantes :

TABLE 3.3 – Combinaisons des classes avec la stratégie OvO

Non - Very Mild	Non - Mild	Non - Moderate
Very Mild - Mild	Very Mild - Moderate	Mild - Moderate

Chaque calcul de surface fait, la moyenne est calculée et retournée et cette moyenne sera prise comme moyen de comparaison.

OvR (One vs Rest) : Pour cette stratégie on compare la qualité du modèle à reconnaitre la classe, en d'autre mots à différencier entre une classe en particulier et le reste des classes, même principe qu'avec la stratégie OvO le calcul se fera pour chaque combinaison et puis on gardera la moyenne. On aura avec cette stratégie les 4 combinaison suivantes :

TABLE 3.4 – Combinaisons des classes avec la stratégie OvR

Non Demented - Reste	Very Mild Demented - Reste
Mild Demented - Reste	Moderate Demented - Reste

On a opté dans ce travail pour la deuxième stratégie, OvR qui nous semble plus significative dans notre projet.

3.4.2 F1-score

C'est la métrique la plus utilisée dans les problèmes déséquilibrés comme dans notre cas, le score est décrit à l'équation 3.3, l'utilisation de cette métrique permet de mettre en valeur deux métriques en une seule : la précision et le rappel et donc en fait une métrique parfaite pour juger de la qualité de la classification.

$$F1 - score = \frac{2 \cdot Précision \cdot Recall}{Précision + Recall} \tag{3.3}$$

tel que la Précision et le Recall (ou Rappel) sont deux métriques complémentaires qui peuvent être décrit, respectivement, par la fraction des exemples assignés à la classe positive qui appartiennent à la classe positive, et la façon dont la classe positive a été prédite. Elles sont calculées selon les deux éauqtions suivantes :

$$Précision = \frac{VP}{VP + FP} \tag{3.4}$$

$$Recall = \frac{VP}{VP + FN} \tag{3.5}$$

3.4.3 Exactitude (Accuracy)

La métrique la plus utilisée dans l'évaluation des modèles d'intelligence artificielle, la formule de calcul est la suivante :

$$Exactitude = \frac{VP + VN}{VP + VN + FP + FN} \tag{3.6}$$

Comme dit plus haut la qualité du modèle ne peut pas être représenté uniquement en utilisant cette métrique, mais en la couplant avec les deux métriques précédentes, elle peut être significative pour la vérification de la qualité de la classification.

3.4.4 Matrice de confusion

Ce n'est pas une métrique à proprement parlé mais un moyen graphique de voir les performances de notre modèle.

La matrice de confusion est un résumé des résultats de prédictions sur un problème de classification. Les prédictions correctes et incorrectes sont mises en lumière et réparties par classe. Les résultats sont ainsi comparés avec les valeurs réelles.

Cette matrice permet de comprendre de quelle façon le modèle de classification est confus lorsqu'il effectue des prédictions. Ceci permet non seulement de savoir quelles sont les erreurs commises, mais surtout le type d'erreurs commises. Les utilisateurs peuvent les analyser pour déterminer quels résultats indiquent comment les erreurs sont commises. On aura une représentation graphique des vrais et faux positifs et négatifs sous forme d'une matrice comme indiqué dans la figure 3.4.

On aura donc les vraies classes représentées en colonnes et les classes prédites par notre modèles disposées en lignes.

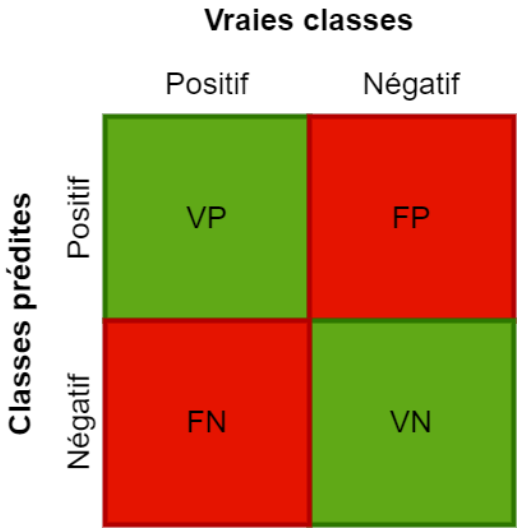


FIGURE 3.4 – Matrice de confusion

Cette matrice permet l'observation rapide et efficace de la qualité de la classification et les classes dont la distinction de la part du modèle est la plus difficile.

3.4.5 Sensibilité et Spécificité

Les métriques à seuils, comme la sensibilité et la spécificité, sont celles qui quantifient les erreurs de prédiction de la classification. C'est la catégorie de métriques la plus communément utilisées.

En d'autres termes, elles sont conçues pour résumer la fraction, le rapport ou le taux des cas où une classe prédite ne correspond pas à la classe attendue dans un ensemble de données d'attente.

Ces deux ,étriques indiquent les rapports entre les faux et vrai positifs et négatifs tels qu'on a les formules suivantes :

Sensibilité : Qui référence au taux de vrais positifs et résume la façon dont la classe positive a été prédite :

$$Sensibilité = \frac{VP}{VP + FN} \tag{3.7}$$

Spécificité : C'est le complément de la sensibilité, ou le taux de vrais négatifs, et résume la façon dont la classe négative a été prédite, elle est calculée de la même façon que le Recall :

$$Spécificité = \frac{VN}{FP + VN} \tag{3.8}$$

3.5 Performances atteintes

Dans cette section, on décrira toutes les expérimentations qui ont été réalisées ainsi qu'un comparatif des performances des différents modèles réalisés suivant les métriques décrites précédemment.

3.5.1 Protocole utilisé

Dans ce travail nous avons fait un bon nombre d'expérimentations où on a utilisé différents descripteurs, en commençant par la série de modèles pré-entrainés suivante :

- AlexNet
- ResNet-101
- Inception ResNet V2

qui ont été décrit en détails dans les chapitres précédents. On a aussi utilisé un autre descripteur qui est l'Histogramme de Gradient Orienté (HOG).

Pour les classifieurs on a comparé 5 différents classifieurs :

- Réseaux de neurones artificiels (ANN) (avec les réseaux de neurones pré-entrainés directement).
- Machine à vecteurs de support (SVM).
- K-plus proches voisins (KNN).
- Naïve Bayes.
- Random Forest.

Tout cela avec les deux types de classification : binaire et multiclasse.

On a procédé au test des performances avant et après augmentation des données. L'augmentation a été faite pour l'équilibrage des classes lors de la classification multiclasse et sera décrite dans la prochaine section.

3.5.1.1 Augmentation des données

L'augmentation des données est une technique utilisée pour accroître la quantité de données en ajoutant des copies légèrement modifiées de données existantes ou des données synthétiques nouvellement créées à partir de données existantes. Elle agit comme un régularisateur et aide à réduire le sur apprentissage lors de la formation d'un modèle d'apprentissage automatique. Elle est étroitement liée au suréchantillonnage dans l'analyse des données.

Alors dans notre cas on a procédé à une augmentation des données en appliquant des transformations sur les images comme suit :

- Rotation aléatoire entre -35° et 35°.
- Mise à l'échelle aléatoire suivant l'axe x entre 0.5 et 4.
- Mise à l'échelle aléatoire suivant l'axe y entre 0.5 et 1.

Nous permettant ainsi d'équilibrer nos classes évitant un sur apprentissage de la classe majoritaire (Non Demented). On a une prévisualisation de l'application de l'augmentation des données sur notre dataset avec la figure suivante, où chaque image a subi une ou plusieurs des transformations citées si-dessus :

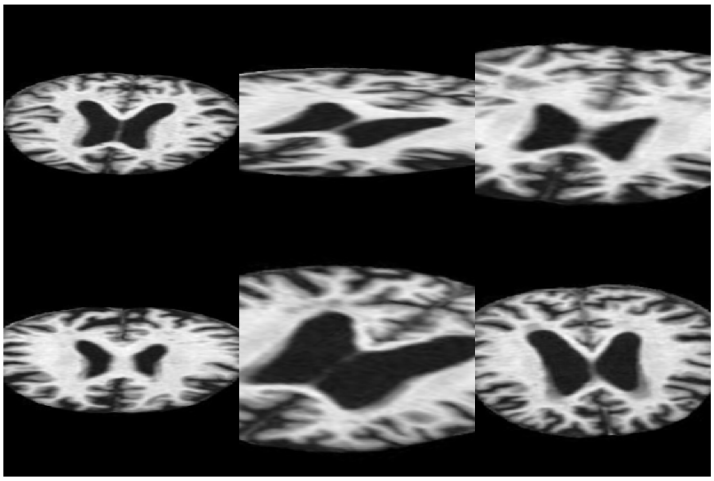


FIGURE 3.5 – Augmentation des données sur notre ensemble de données

3.5.1.2 Description des ensembles d'entraînement, de validation et de test

Durant nos entraînements et évaluations, on a procédé à deux subdivisions différentes que l'on a trouvé dans la littérature :

La première subdivision est celle utilisée dans les travaux de A. Loddo et al. [5], qui représentent actuellement le travail le plus récent sur la base de données qu'on a traitée, ainsi on a suivi dans cette subdivision à la lettre la préparation de leurs données en prenant la totalité des images, mélangeant les données d'entraînements et de test, et ensuite on applique la subdivision suivante :

Table 3.5 – Première répartition des différents ensembles : entraînement, validation et test

Ensembles	Pourcentage (%)	Nombre d'images
Entraînement	80	5120
Validation	10	640
Test	10	640

La deuxième subdivision quant à elle est la subdivision initiale de la base de données comme élaborée par l'auteur sur Kaggle [6], mettant ainsi en place deux ensembles : Entraînement et Test, on a pris l'équivalent de 10% pour la validation à partir de l'ensemble d'entraînement. Ainsi à travers cette subdivision on respecte le partage des images comme indiqué par l'auteur de la base de données.

Table 3.6 – Deuxième répartition des différents ensembles : entraînement, validation et test

Ensembles	Pourcentage (%)	Nombre d'images
Entraînement	70	4480
Validation	10	640
Test	20	1280

3.5.1.3 Entraînement

Dans notre projet on a procédé à différents entraînements, celui des différents réseaux de neurones convolutifs utilisés et aussi des classifieurs utilisés à la suite de nos descripteurs.

Entraînement du CNN : Pour ces entraînements on a procédé aux entraînements sur deux environnements différents :

1. Sous environnement Python utilisant le framework **Tensorflow** decrits plus haut.
2. Sous environnement Matlab, aussi décrit plus haut.

Dans les deux cas on a mis les mêmes conditions d'entraînements :

- Mêmes subdivisions de l'ensemble de données expliquées à la section 3.5.1.2.
- Même type d'augmentation des données expliquée à la section 3.5.1.1.
- Entraînements durant au maximum pour 200 époques.
- On a utilisé un callback d'arrêt prématuré si l'exactitude ne fait que diminuer pendant plus de 30 époques gardant ainsi le meilleur modèle.
- Le taux d'apprentissage a été défini à 10^{-4} durant la totalité de l'entraînement.

On a ensuite sélectionné le modèle ayant eu les meilleurs résultats. La même méthodologie a été suivie dans le cas de l'utilisation du CNN en tant qu'extracteur de caractéristiques en supprimant les couches de classification mise à la fin.

Entraînement des classifieurs utilisés : Les classifieurs utilisés ont été tous entraînés sous environnement Python utilisant le framework **Scikit-learn** décrit plus haut.

La figure suivante résume les nombreuses configurations testées, dues au choix des multiples critères cités plus haut :

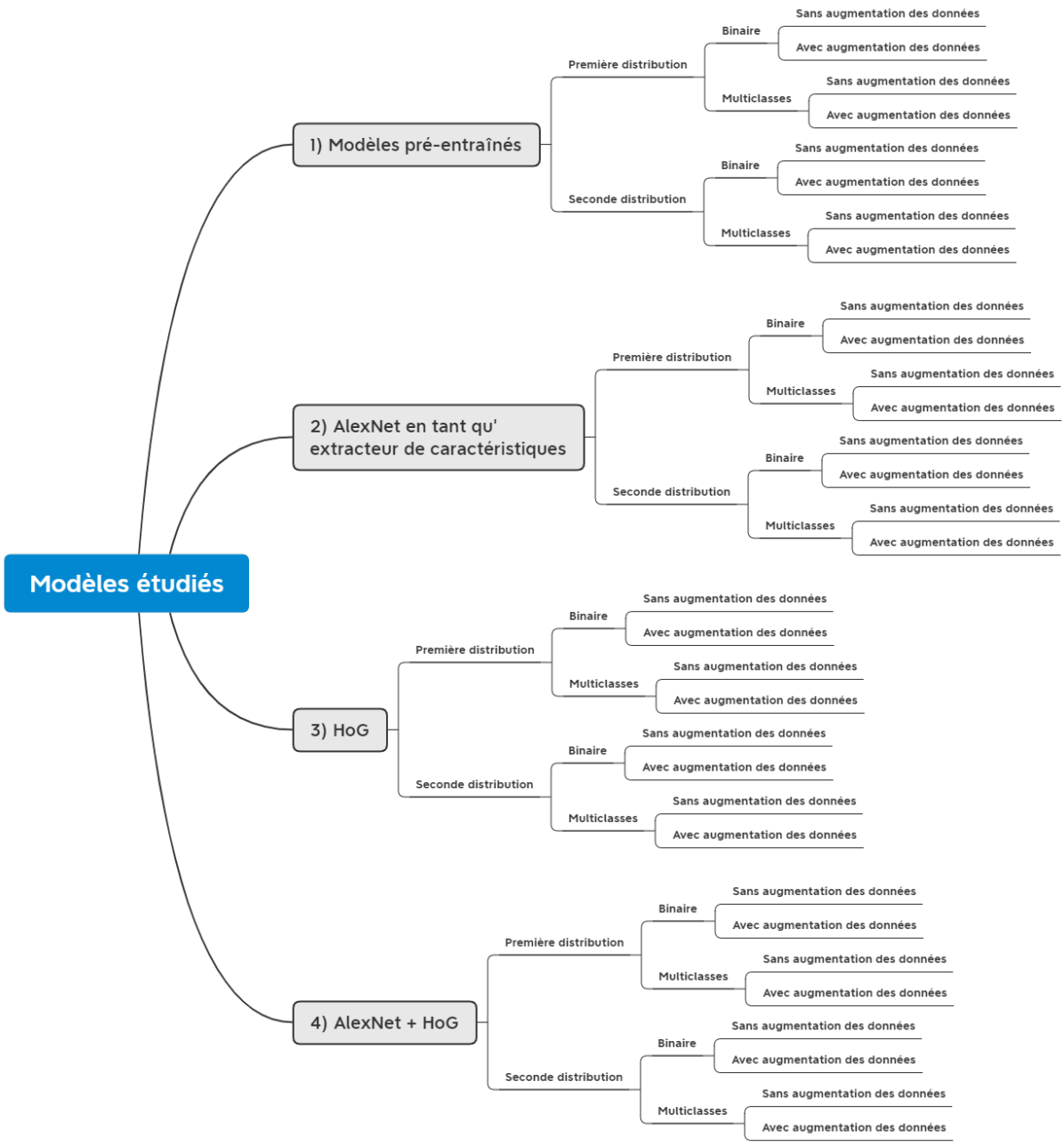


FIGURE 3.6 – Les différentes configurations testées dans notre projet

3.5.2 Modèles pré-entraînés

On a suivi la méthode d'entraînement expliquée plus haut sur les trois modèles pré-entraînés : AlexNet, ResNet-101 et Inception ResNet-V2 décrits plus haut et on a obtenu les résultats suivants :

3.5.2.1 Première distribution de la base de données

Le tableau suivant montre les performances atteintes par les différents modèles de réseaux de neurones pré-entraînés qu'on a testé suivant la distribution de l'ensemble de données et la

méthodologie suivie par A. Loddo et al. [5] pour une classification binaire.

TABLE 3.7 – Performances des modèles pré-entraînés pour la classification binaire avec la première configuration de l'ensemble de données

CNN	Exactitude (%)	Sensibilité (%)	Spécificité (%)
Avant augmentation des données			
AlexNet	67.81	41.56	94.06
ResNet-101	56.25	60.24	50.56
Inception ResNet-V2	81.25	88.75	75.41
Après augmentation des données			
AlexNet	97.81	98.75	96.88
ResNet-101	49.96	49.05	51.23
Inception ResNet-V2	77.01	80.59	70.63
AlexNet [5]	89.65	89.79	89.65
ResNet-101 [5]	96.09	96.11	96.09
Inception ResNet-V2 [5]	91.21	91.44	91.21

En commençant cette discussion on peut faire une remarque importante sur l'augmentation des données qui a bien un effet positif sur les performances de la classification pour tous les modèles utilisés ce qui indique qu'elle a été bénéfique pour la classification. On peut aussi voir que le modèle qui présente les meilleurs performances est AlexNet qui est aussi bien plus léger que les deux autres modèles pré-entraînés.

Le tableau suivant, quant à lui, montre les performances atteintes par les différents modèles pré-entraînés qu'on a testé pour une classification multiclasse cette fois.

TABLE 3.8 – Performances des modèles pré-entraînés pour la classification multiclasse avec la première configuration de l'ensemble de données

CNN	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
Avant augmentation des données			
AlexNet	90.94	92.41	94.53
ResNet-101	35.92	86.16	58.28
Inception ResNet-V2	78.41	96.54	88.59
Après augmentation des données			
Alexnet	99.51	99.37	98.61
ResNet-101	16.67	78.02	50
Inception ResNet-V2	60.36	89.94	67.03
AlexNet [5]	84.83	/	89.26
ResNet-101 [5]	97.26	/	96.48
Inception ResNet V2 [5]	87.39	/	89.65

Même constat vis-à-vis de la classification multiclasse décrite dans le tableau 3.8 sur l'impact

de la subdivision de la base de données ainsi que l'apport de l'augmentation des données sur les performances des modèles. On peut voir aussi qu'à ce niveau, AlexNet donne toujours les meilleurs résultats et ce qui nous motive à le garder par la suite aussi pour la classification multiclasse, ainsi que la classification binaire suivant cette distribution.

3.5.2.2 Deuxième distribution de la base de données

Le tableau 3.9 décrit les résultats retrouvés avec les modèles pré-entraînés dans la classification binaire suivant la deuxième distribution de notre base de données, i.e la distribution originale.

TABLE 3.9 – Performances des modèles pré-entraînés pour la classification binaire avec la deuxième configuration de l'ensemble de données

CNN	Exactitude (%)	Sensibilité (%)	Spécificité (%)
Avant augmentation des données			
AlexNet	80.37	81.25	79.50
ResNet-101	49.96	56.42	47.87
Inception ResNet-V2	73.65	78.06	70.34
Après augmentation des données			
AlexNet	83.03	92.03	74.02
ResNet-101	49.96	56.42	47.87
Inception ResNet-V2	71.23	73.41	70.03

On remarque clairement qu'avec cette distribution la classification est nettement plus complexe et les résultats sont inférieurs à ceux retrouvés avec la distribution précédente ce qui peut insinuer que les images de tests seraient corrélées et donc en utilisant une partie comme images d'entraînements lors de la subdivision comme précédemment fait pourrait impliquer une fuite de données (data leakage) donnant des résultats très bons sans forcément indiquer une vraie robustesse du modèle. Il restera donc à vérifier cette supposition selon les expériences suivantes.

On peut observer qu'on a, ici aussi, de meilleurs résultats avec AlexNet vis-à-vis des deux autres modèles avec 83.03% de précision lors des tests, ce qui nous motive à l'utiliser par la suite comme extracteur de caractéristiques en plus du fait qu'il soit, comme décris plus haut, un modèle bien plus léger que les deux autres.

Le tableau 3.10 décrit les résultats retrouvés avec les modèles pré-entraînés dans la classification multiclasse suivant la deuxième distribution.

TABLE 3.10 – Performances des modèles pré-entraînés pour la classification multiclasse avec la deuxième configuration de l'ensemble de données

CNN	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
Avant augmentation des données			
AlexNet	67.22	78.81	80.06
ResNet-101	16.68	78.06	50.04
Inception ResNet-V2	52.55	81.75	66.30
Après augmentation des données			
AlexNet	69.47	80.14	79.74
ResNet-101	16.68	78.06	50.04
Inception ResNet-V2	47.61	84.55	56.84

Le constat sur la fuite de données qu'on peut emettre avec la subdivision précédente est plus flagrante avec la classification multiclasse dont les résultats sont décrit dans le tableau 3.10 ayant des résultats bien inférieurs à ceux retrouvés précédemment (avec la première distribution). On maintient néanmoins notre choix de prendre le réseau de neurones AlexNet vis-à-vis des performances atteintes qui reste au dessus des deux autres sur cette base de données avec un F1-score de 69.47% et une précision de 79.74%.

La première méthode de subdivision prends les données en entier et les départage aléatoirement, ce départage des images influe énormément sur la qualité du modèle et ses performances en test ce qui peut expliquer la différence des résultats entre les résultats de nos modèles pré-entraînés et ceux des travaux de A. Loddo et Al. [5].

Donc en somme on retient de cette étape la sélection du modèle AlexNet comme réseau de neurones convolutifs qu'on utilisera comme extracteur de caractéristiques.

3.5.3 AlexNet en tant qu'extracteur de caractéristiques

On a choisi d'utiliser l'extracteur de caractéristiques du réseau de neurones **AlexNet**. Notre choix a été motivé comme dit durant les discussions précédentes par plusieurs facteurs :

- Le poids du modèle est le plus faible des trois.
- Il a de très bonnes performances sur notre base de données.
- Simplicité de l'architecture, nous permettant des améliorations simples et efficaces. En comparaison avec le ResNet et le Inception ResNet-V2 qui ont des architectures non linéaires compliquées à modifier.

L'architecture de l'extracteur de caractéristique est détaillée dans la figure 3.7 :

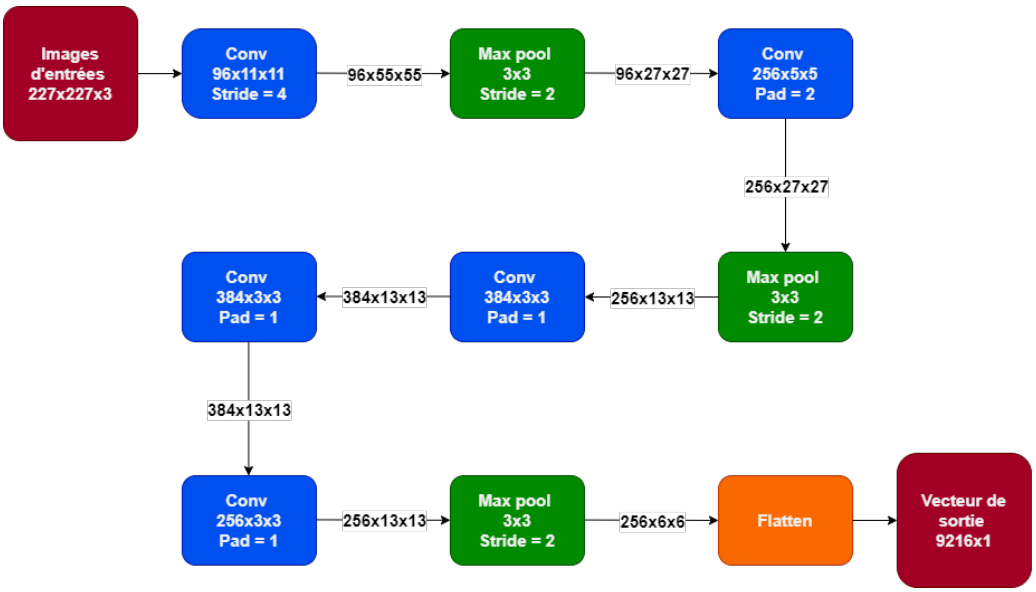


FIGURE 3.7 – Architecture de l'extracteur de caractéristique de AlexNet

Comme on peut le voir c'est une architecture relativement simple et on peut voir que l'architecture est constituée de :

- 5 couches de convolution.
- 3 couches de max-pooling.
- 1 couche flatten (ajoutée par nous pour avoir un vecteur de caractéristiques pour chaque image).

Pour l'utilisation de ce descripteur on redimensionne nos images à une taille de $227 \times 277 \times 3$, on normalise aussi les niveaux de gris faisant passer nos niveaux de gris de [0,255] à [0,1] (c'est les même conditions d'utilisation du réseau AlexNet). On a en sortie un vecteur caractéristique d'une taille de 9216 éléments.

À la suite de ce descripteur on a testé plusieurs classifieurs :
hyperlinkknnKNN, SVM, Naïve Bayes et Random Forest qu'on va décrire dans la prochaine section ainsi que la méthode pour le choix optimal des paramètres de classification basée sur la validation croisée.

3.5.3.1 Différents types de classifieurs communément utilisés

Il y a un grand nombre de classifieurs utilisés dans le domaine de l'apprentissage automatique, chacun avec des avantages et inconvéniants, on peut citer quelques uns :

Machine à vecteurs de supports (SVM) : Les machines à vecteurs de support, ou support vector machine (SVM), sont des modèles de machine learning supervisés centrés sur la résolution de problèmes de discrimination et de régression mathématiques. Elles ont été conceptualisées en 1995 par C. Cortes et al. [67]. Ce modèle a été rapidement adopté en raison de sa capacité à travailler avec des données de grandes dimensions, ses garanties théoriques et les bons résultats réalisés en pratique. Requérant un faible nombre de paramètres.

Le principe des SVM consiste à ramener un problème de classification ou de discrimination à un hyperplan (feature space) dans lequel les données sont séparées en plusieurs classes dont la frontière est la plus éloignée possible des points de données (ou "marge maximale").

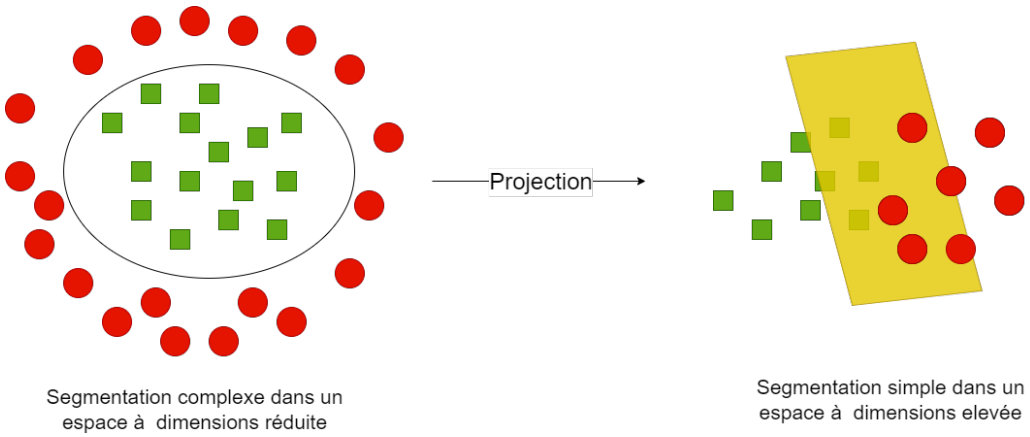


FIGURE 3.8 – Principe d'un SVM

K-plus proches voisins (KNN) : C'est un des algorithmes les plus simples à implémenter en classification et il fonctionne de la manière suivante : Dans un premier temps il prend toutes les données d'entraînement, ensuite on sélectionne le nombre K de voisins, l'algorithme procédera au calcul des distances entre la nouvelle donnée (de test) et les données d'entraînement disponibles. Ensuite, il sélectionne les K plus proches voisins et fait un vote, la classe la plus représentée parmi les K voisins sera la classe attribuée à ce nouvel élément comme l'indique la figure suivante :

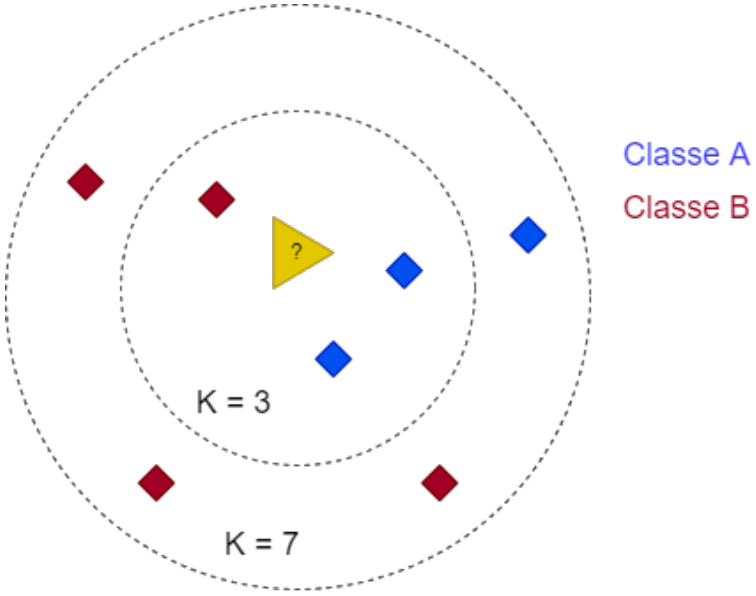


FIGURE 3.9 – Principe du KNN

Comme on peut le voir dans cet exemple, illustré dans la figure 3.9, on a un nouvel élément à classifier (en jaune) si on prend un nombre de voisins égal à 3 la classe majoritaire sera la classe A et si on prend un nombre de voisin égal à 7 la classe majoritaire sera alors la classe B, donc le choix du nombre de voisins est primordial pour la qualité de la classification.

Naive Bayesienne : La classification naïve bayésienne est un type de classification bayésienne probabiliste simple basée sur le théorème de Bayes indiqué à l'équation 3.9 avec une forte indépendance (dite naïve) des hypothèses. Elle met en œuvre un classifieur bayésien naïf, ou classifieur naïf de Bayes, appartenant à la famille des classifieurs linéaires.

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{3.9}$$

Tels que :

P(A|B) : Probabilité a posteriori de l'hypothèse étant donné que les preuves sont vraies.

P(B|A) : Probabilité de la preuve étant donné que l'hypothèse est vraie.

P(A) : Probabilité a priori de l'hypothèse.

P(B) : Probabilité a priori que la preuve soit vraie.

L'avantage du classifieur bayésien naïf est qu'il requiert relativement peu de données d'entraînement pour estimer les paramètres nécessaires à la classification, à savoir moyennes et variances des différentes variables. En effet, l'hypothèse d'indépendance des variables permet de se contenter de la variance de chacune d'entre elles pour chaque classe, sans avoir à calculer de matrice de covariance.

Forêt aléatoire (Random Forest) : Les forêts aléatoires sont une méthode d'apprentissage d'ensemble pour la classification qui fonctionne en construisant une multitude d'arbres de décision au moment de l'entraînement. La sortie de la forêt aléatoire est la classe sélectionnée par la plupart des arbres, elle est illustrée dans la figure 3.10.

Le premier algorithme de forêts de décision aléatoires a été créé en 1995 par Tin Kam Ho [68] à l'aide de la méthode du sous-espace aléatoire qui, dans la formulation de Ho, est une façon de mettre en œuvre l'approche de "discrimination stochastique" de la classification proposée par Eugene Kleinberg [69]. Algorithme qui a été développé par la suite essentiellement par L. Breinman [70] le nommant pour la première fois "Random Forest".

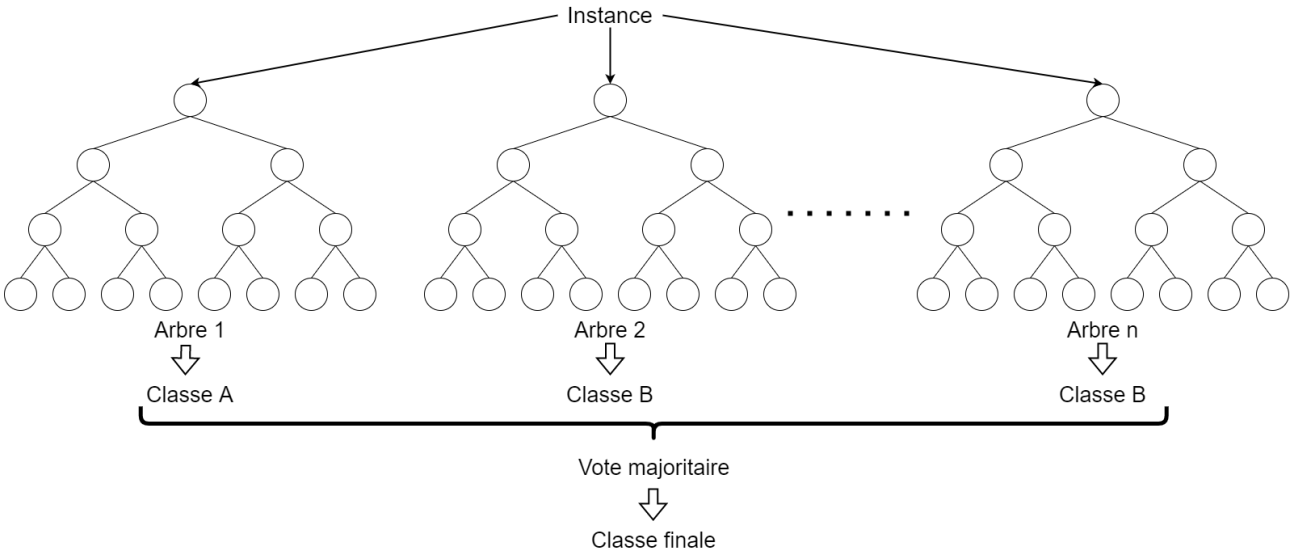


FIGURE 3.10 – Principe d'une forêt aléatoire

3.5.3.2 Première distribution de la base de données

Le tableau 3.11 montre les résultats retrouvés en utilisant les différents classifieurs décrits plus haut dans le cas d'une classification binaire avec la première distribution. On voit bien que l'augmentation des données a été bénéfique faisant grimper les performances d'un maximum de précision de 89.84% jusqu'à 99.38% de précision après l'augmentation des données et cela toujours avec le KNN qui nous donne les meilleurs performances en test lors de son optimisation se basant sur la validation croisée qui sera décrite plus bas.

TABLE 3.11 – Performances d'AlexNet avec différents classifieurs pour la classification binaire avec la première configuration dd l'ensemble de données

Descripteur	Classifieur	Exactitude (%)	Sensibilité (%)	Spécificité (%)
Avant augmentation des données				
AlexNet	KNN	89.84	91.29	88.46
AlexNet	SVM	70.94	66.41	79.17
AlexNet	Naïve Bayes	62.81	69.23	59.26
AlexNet	Random Forest	81.72	82.47	80.58
Après augmentation des données				
AlexNet	KNN	99.38	100	98.72
AlexNet	SVM	97.03	96.90	97.19
AlexNet	Naïve Bayes	75	79.76	71.55
AlexNet	Random Forest	95	98.59	91.92

On a le même constat à faire dans le cas de la classification multiclasse dont les résultats sont décrits dans le tableau 3.12, en ayant toujours le KNN comme classifieur ayant obtenu les meilleurs performances avec un F1-score de 99.82% lors des tests.

TABLE 3.12 – Performances d'AlexNet avec différents classifieurs pour la classification multiclasse avec la première configuration de l'ensemble de données

Descripteur	Classifieur	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
Avant augmentation des données				
AlexNet	KNN	99.61	99.66	99.69
AlexNet	SVM	99.32	99.43	99.22
AlexNet	Naïve Bayes	55.23	73.91	76.72
AlexNet	Random Forest	94.26	94.92	95.94
Après augmentation des données				
AlexNet	KNN	99.82	99.82	99.84
AlexNet	SVM	97.74	98.23	96.72
AlexNet	Naïve Bayes	49.34	69.45	68.28
AlexNet	Random Forest	93.93	94.54	95.31

3.5.3.3 Deuxième distribution de la base de données

Le tableau 3.13 décrit les résultats retrouvés dans le cas de la classification binaire dans le cas de la deuxième distribution de la base de données, dans ce cas là on observe que l'augmentation des données n'a pas eu d'influence positive sur les résultats ce qui indique qu'elle a été superflue dans ce cas, néanmoins avant et après augmentation des données encore une fois le classifieur le plus efficace a été le KNN avec une exactitude maximale de 85.61%.

TABLE 3.13 – Performances d'AlexNet avec différents classifieurs pour la classification binaire avec la deuxième configuration de l'ensemble de données

Descripteur	Classifieur	Exactitude (%)	Sensibilité (%)	Spécificité (%)
Avant augmentation des données				
AlexNet	KNN	85.61	86.60	84.46
AlexNet	SVM	79.20	78.43	79.59
AlexNet	Naïve Bayes	65.91	72.46	61.83
AlexNet	Random Forest	75.84	81.48	71.43
Après augmentation des données				
AlexNet	KNN	84.13	88.94	80.36
AlexNet	SVM	82.33	88.24	78.26
AlexNet	Naïve Bayes	60.44	59.13	62.35
AlexNet	Random Forest	75.45	82.05	70.49

Vis-à-vis de la classification multiclasse dont les résultats sont décrits dans le tableau 3.14, on voit que l'augmentation des données a amélioré les résultats ce qui était prévisible sachant que c'est un problème de classification déséquilibré et donc l'augmentation des données permet de compenser ce déséquilibre, ayant encore une fois les meilleures performances avec le KNN confortant son maintien comme classifieur pour la suite des expérimentations.

TABLE 3.14 – Performances d'AlexNet avec différents classifieurs pour la classification multiclasse avec la deuxième configuration de l'ensemble de données

Descripteur	Classifieur	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
Avant augmentation des données				
AlexNet	KNN	77.37	83.57	82.80
AlexNet	SVM	58.67	72.55	76.15
AlexNet	Naïve Bayes	34.98	58.88	58.25
AlexNet	Random Forest	42.59	63.86	68.65
Après augmentation des données				
AlexNet	KNN	81.55	85.79	83.50
AlexNet	SVM	72.48	78.67	77.95
AlexNet	Naïve Bayes	36.13	58.30	56.29
AlexNet	Random Forest	51.22	67.30	69.90

3.5.3.4 Choix des paramètres des classifieurs utilisés

Comme observé à travers les tests de performances qui sont décrits dans les tableaux 3.11, 3.12, 3.13 et 3.14 on a opté pour l'utilisation d'un **KNN** comme classifieur et cela en se basant sur la validation croisée à 5 blocs qui sera décrite plus bas.

Le KNN est un classifieur relativement simple dans sa manipulation n'ayant besoin que de deux paramètres pour son optimisation : le nombre de voisins K et la méthode de calcul de la

distance choisie. Pour le choix optimal des paramètres on a utilisé l'algorithme de recherche par grille (**GridSearchCV**) qui permet le choix de la meilleure combinaison de paramètres pour la classification.

3.5.3.5 GridSearchCV

Pour le choix des paramètres optimaux de chaque classifieur on a procédé à la validation croisée avec la recherche par grille. Tout d'abord, comprenons ce qu'est la recherche par grille. Il s'agit du processus d'ajustement des hyperparamètres afin de déterminer les valeurs optimales pour un modèle donné. Comme mentionné ci-dessus, la performance d'un modèle dépend de manière significative de la valeur des hyperparamètres. Il faut noter qu'il n'y a aucun moyen de connaître à l'avance les meilleures valeurs pour les hyperparamètres, donc idéalement, nous devons essayer toutes les valeurs possibles pour connaître les valeurs optimales. Faire cela manuellement peut prendre beaucoup de temps et de ressources, c'est pourquoi nous utilisons GridSearchCV pour automatiser le réglage des hyperparamètres.

En définissant les hyperparamètres à tester, comme par exemple pour le KNN le nombre de voisins :

$$k \in \{1, 3, 5, 7, 9, 11, 13, 15, 17, 19\} \tag{3.10}$$

ainsi que la méthode de calcul de la distance choisie entre la distance de Canberra, de Manhattan et Euclidienne :

$$D_{Euclidienne}(a, b) = \sqrt{(\sum_{i=1}^{n} |a_i - b_i|^2)} \tag{3.11}$$

$$D_{Manhattan}(a, b) = \sum_{i=1}^{n} |a_i - b_i| \tag{3.12}$$

$$D_{Canberra}(a, b) = \sum_{i=1}^{n} \frac{|a_i - b_i|}{|a_i + b_i|} \tag{3.13}$$

GridSearchCV essaie toutes les combinaisons des valeurs transmises et évalue le modèle pour chaque combinaison en utilisant la méthode de **validation croisée**. Ainsi, après avoir utilisé cette fonction, nous obtenons la précision/perte pour chaque combinaison d'hyperparamètres et nous pouvons choisir celle qui a la meilleure performance.

3.5.3.6 Validation croisée (cross validation)

La validation croisée est une méthode statistique utilisée pour estimer les performances (ou la précision) des modèles d'apprentissage automatique. Elle est utilisée pour se protéger contre le surapprentissage d'un modèle prédictif, en particulier dans le cas où la quantité de données peut être limitée. Dans la validation croisée, on réalise un nombre fixe de plis (ou partitions) des données, on exécute l'analyse sur chaque pli, puis on fait la moyenne de l'erreur globale estimée. Il existe plusieurs méthodes de validation croisée non exhaustives comme :

La validation croisée à retenue (Holdout) : Cette méthode permet de prendre une partie de l'ensemble de données d'entraînement et l'utiliser comme ensemble de validation pour tester les performances du classifieur. Cette méthode est essentiellement utilisée dans le domaine du Deep Learning avec les réseaux de neurones, mais a certains défauts, essentiellement qu'on ne peut pas être certain des performances du modèle puisqu'il est jugé que sur une partie des données qui peut ne pas être représentative de la distribution globale des données.

La validation croisée à k blocs (K-fold) : on divise l'échantillon original en k échantillons (ou « blocs »), puis on sélectionne un des K échantillons comme ensemble de validation pendant que les K-1 autres échantillons constituent l'ensemble d'apprentissage. Après apprentissage, on peut calculer une performance de validation. Puis on répète l'opération en sélectionnant un autre échantillon de validation parmi les blocs prédéfinis. À l'issue de la procédure nous obtenons ainsi K scores de performances, un par bloc. La moyenne et l'écart type des K scores de performances peuvent être calculés pour estimer le biais et la variance de la performance de validation. Cette méthode permet de valider le modèle à chaque fois sur un sous-ensemble de donnée différent de l'ensemble d'entraînement ce qui permet de bien voir la robustesse du modèle.

La différence est illustrée dans la figure suivante :

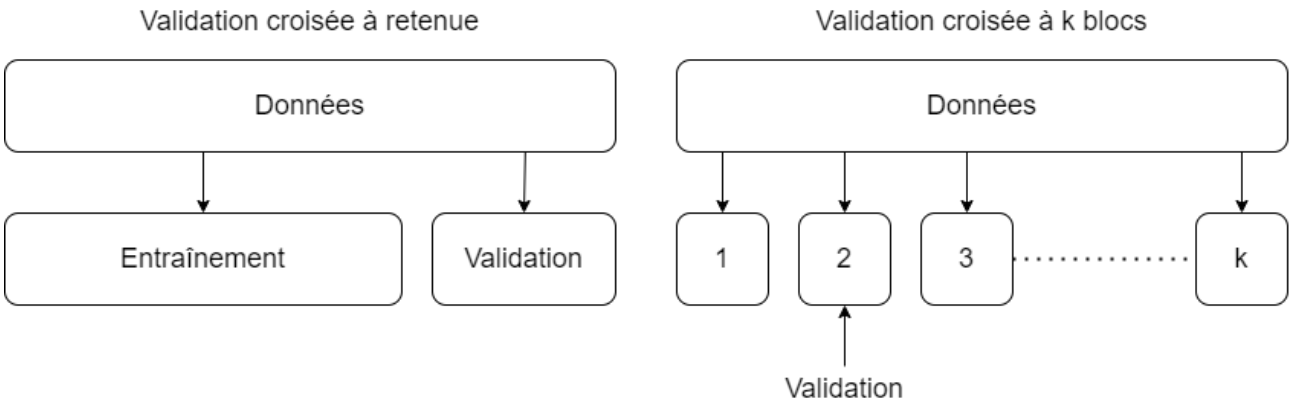


FIGURE 3.11 – Comparaison entre la validation croisée à retenue et à k blocs

3.5.3.7 Choix optimal des paramètres du KNN

Dans le choix optimal des paramètres de notre KNN on a opté pour la validation croisée à k blocs en prenant $K = 5$. On a eu les meilleurs résultats avec la configuration suivante :

- Nombre de voisin $\mathbf{k = 1}$.
- Distance choisie est la **distance de Manhattan**.

Tout cela nous conforte le choix du KNN comme classifieur, on teste maintenant un autre descripteur pour tenter d'augmenter les performances du modèle.

3.5.4 Histogramme de Gradients Orientés

Lors des expérimentations on a voulu amener une originalité en investiguant une fusion de caractéristiques pour la classification des images IRM. Donc pour se faire, on a choisi d'utiliser le HOG comme descripteur supplémentaire au CNN précédemment testé.

Proposé par N. Dalal et B. Triggs, chercheurs à l'INRIA de Grenoble, à la conférence CVPR de juin 2005 [71].

Dans les applications de traitement d'images, le HOG est un descripteur de caractéristiques efficace et très communément utilisé dans plusieurs domaines allant de la reconnaissance faciale [71] (utilisation initiale) jusqu'à son utilisation dans la classification d'images IRM dans l'étude des différents niveaux de démence d'Alzheimer [20] ce qui en fait un descripteur puissant et polyvalent.

Il est calculé à partir de l'amplitude et de l'orientation des images cérébrales. Dans le descripteur HOG, les gradients horizontaux G_h et verticaux G_v des images cérébrales sont estimés à l'aide des équations 3.14 et 3.15 :

$$G_h = I_N \times [-1, 0, 1] \tag{3.14}$$

$$G_v = I_N \times [-1, 0, 1]^T \tag{3.15}$$

Les gradients horizontaux G_h et verticaux G_v obtenus sont utilisés pour calculer l'orientation angulaire $\theta(x, y)$ et la magnitude du gradient $m(x, y)$ à l'aide des équations 3.16 et 3.17 respectivement :

$$\theta(x, y) = tan^{-1}(\frac{G_v(x, y)}{G_h(x, y)}) \tag{3.16}$$

$$m(x, y) = \sqrt{G_v^2(x, y) + G_h^2(x, y)} \tag{3.17}$$

L'orientation angulaire $\theta(x, y)$ et l'amplitude du gradient $m(x, y)$ divisent l'image normalisée du cerveau en cellules. Ensuite, un bloc (collection de cellules) est généré et les caractéristiques sont constituées en convoluant les blocs d'image. L'orientation liée aux cellules similaires est intégrée et quantifiée dans des bins d'histogramme finaux et ces bins sont combinés dans l'histogramme final. Le nombre total de caractéristiques T_{hog} est calculé comme suit :

$$T_{hog} = B_{img} \times B_s \times N_b \tag{3.18}$$

Où :

B_{img} **:** Nombre de blocs par images.

N_b **:** Nombre de bins.

B_s **:** Taille du bloc.

On a un exemple ci-dessous dans le figure 3.12 montrant l'application d'un histogramme de gradient orienté sur une IRM cerebral de notre base de données.

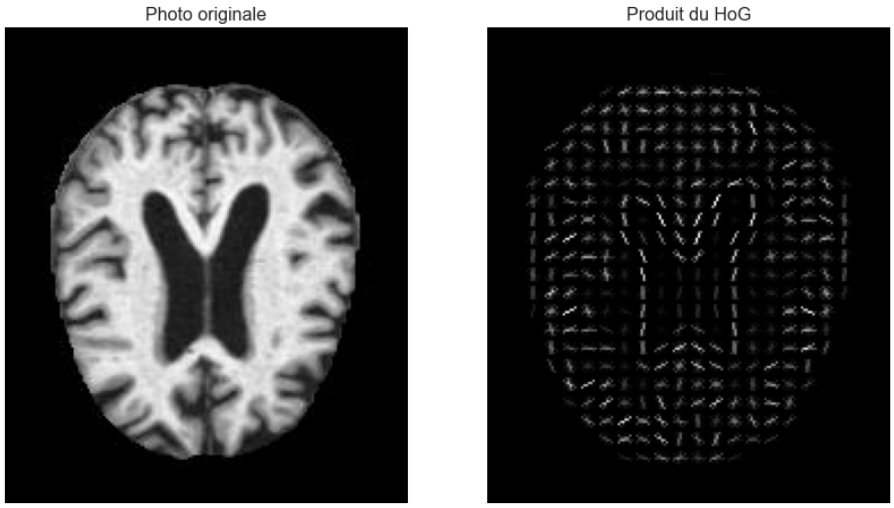


FIGURE 3.12 – Exemple de l'application du HoG sur une IRM

Dans notre cas on a des images de taille $208 \times 176 \times 3$, on a choisi les paramètres du HOG comme suit :

$B_{img} = 21 \times 25$

$N_b = 9$

$B_s = 2 \times 2$

En appliquant la relation de l'équation 3.18 on a donc un vecteur caractéristique de taille :

$$T_{hog} = B_{img} \times B_s \times N_b = 21 \times 25 \times 2 \times 9 = 18900 \tag{3.19}$$

L'application du HOG nous donne donc un vecteur caractéristique de 18900 éléments qui nous permettent de décrire l'image d'entrée. On tente dans la prochaine section la classification des images IRM avec les classifieurs précédemment testés avec le CNN et discuter des performances.

3.5.4.1 Première distribution de la base de données

Les tableaux 3.15 et 3.16 représentent respectivement les résultats de la classification binaire et multiclasse avec la première distribution en utilisant le HOG comme descripteur. On donne le même constat que précédemment : avec le HOG c'est aussi le KNN qui donne les meilleures performances avec une précision de 100% pour la classification binaire et un F1-score 99.80% pour la classification multiclasse, nous confortant dans son utilisation comme descripteur supplémentaire au CNN précédemment testé, mais aussi conforte nos soupçons sur la méthodologie suivie par A. Loddo et Al. [5].

Table 3.15 – Performances du HoG avec différents classifieurs pour la classification binaire avec la première configuration de la base de données

Descripteur	Classifieur	Exactitude (%)	Sensibilité (%)	Spécificité (%)
HoG	KNN	100	100	100
HoG	SVM	83.91	80.36	88.64
HoG	Naïve Bayes	62.66	59.85	69.12
HoG	Random Forest	68.59	100	61.35

Table 3.16 – Performances du HoG avec différents classifieurs pour la classification multiclasse avec la première configuration de la base de données

Descripteur	Classifieur	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
HoG	KNN	99.80	99.83	99.84
HoG	SVM	88.19	91.76	84.22
HoG	Naïve Bayes	30.94	60.65	40.63
HoG	Random Forest	40.35	61.83	67.34

3.5.4.2 Deuxième distribution de la base de données

Les tableaux 3.17 et 3.18 quant à eux présentent les résultats retrouvés avec le HOG comme descripteurs suivis des différents classifieurs se basant sur la deuxième distribution et on a dans ce cas aussi d'excellentes performances du KNN comme classifieur nous permettant d'avoir une précision de 84.52% lors de la classification binaire ainsi qu'un F1-score de 84.73% lors de la classification multiclasse, nous permettant de dire qu'avec la deuxième distribution aussi le HOG sera utilisé comme deuxième descripteur pour notre classification en le fusionnant avec le CNN précédemment testé.

TABLE 3.17 – Performances du HoG avec différents classifieurs pour la classification binaire avec la deuxième configuration de la base de données

Descripteur	Classifieur	Exactitude (%)	Sensibilité (%)	Spécificité (%)
HoG	KNN	84.52	81.65	87.91
HoG	SVM	65.68	64.29	68.18
HoG	Naïve Bayes	57.23	68.29	54.72
HoG	Random Forrest	50.82	100	50.51

TABLE 3.18 – Performances du HoG avec différents classifieurs pour la classification multiclasse avec la deuxième configuration de la base de données

Descripteur	Classifieur	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
HoG	KNN	84.73	88.05	82.25
HoG	SVM	43.84	61.38	57.08
HoG	Naïve Bayes	27.03	53.04	51.29
HoG	Random Forest	17.85	50.47	50.82

3.5.5 Fusion des caractéristiques

Se basant sur les résultats satisfaisant qu'on a eu avec les deux descripteurs précédemment testé on va alors en faire la fusion et donc le vecteur caractéristique utilisé est une concaténation des vecteurs caractéristiques générés par un CNN et d'un histogramme de gradient orienté HOG.

3.5.5.1 Fusion des caractéristiques

La fusion des caractéristiques est le fait de concaténer plusieurs vecteurs caractéristiques résultants de plusieurs descripteurs : plusieurs CNN, des descripteurs statistiques, structurels...etc, pour en faire un seul et même vecteur qui caractérisera notre entrée. Cette technique a fait ses preuves dans plusieurs domaines d'application de l'apprentissage automatique, entre autre dans l'imagerie médicale. Cette technique a prouvé ses performances lors de l'utilisation d'images IRM comme on peut le constater entre autre dans les travaux de M. Irfan Sharif et al. [72] ainsi que ceux de T. Saba et al. [73] qui ont travaillé sur la détection et la segmentation des tumeurs cérébrales basée sur des images IRM avec des résultats supérieurs aux travaux se basant sur un seul descripteur.

La fusion des caractéristiques proposées dans notre projet allie un vecteur produit d'un réseau de neurones convolutifs et d'un vecteur produit de l'application d'un histogramme de gradient orienté ce qui assure une diversité des caractéristiques entre les deux descripteurs décrivant bien mieux nos images.

3.5.5.2 Fusion des deux descripteurs

Comme dit précédemment, on prend alors le vecteurs de caractérsitiques des deux descripteurs précédents pour en extraire un seul et même vecteur, qui est la concaténation des deux vecteurs

préalablement calculés

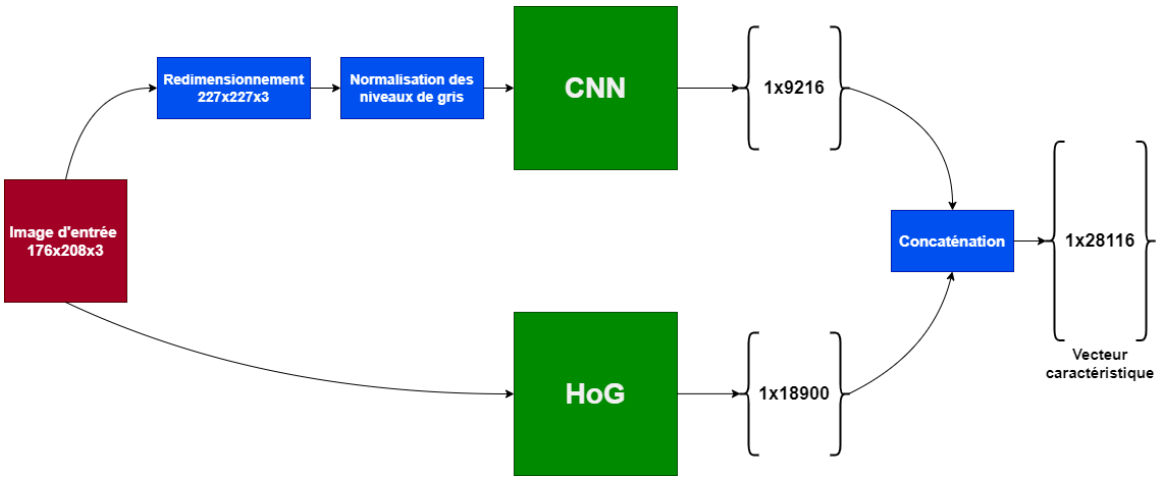


FIGURE 3.13 – Architecture du descripteur utilisé dans le projet

On a alors au bout de la fusion des deux descripteurs un vecteur caractéristiques de 28116 éléments qu'on mettra directement comme entrée de notre classifieur. Dans la prochaine section on va développer les différents classifieurs sur lesquels on pourra opter.

3.5.5.3 Première distribution de la base de données

Les tableaux 3.19 et 3.20 nous donne les performances atteintes en utilisant la fusion de caractéristiques entre le CNN de AlexNet et l'Histogramme de Gradients Orientés HOG pour la classification binaire et multiclasse, respectivement, suivant la première distribution de la base de données.

TABLE 3.19 – Performances de la fusion de caractéristiques pour la classification binaire avec la première configuration de la base de données

Descripteur	Classifieur	Exactitude (%)	Sensibilité (%)	Spécificité (%)
Avant augmentation des données				
HoG + AlexNet	KNN	100	100	100
HoG + AlexNet	SVM	71.56	67.20	78.67
HoG + AlexNet	Naïve Bayes	64.06	62.73	65.56
HoG + AlexNet	Random Forest	75.63	98.22	67.35
Après augmentation des données				
HoG + AlexNet	KNN	99.38	100	98.72
HoG + AlexNet	SVM	97.03	96.90	97.19
HoG + AlexNet	Naïve Bayes	75.31	80.72	71.79
HoG + AlexNet	Random Forest	89.22	98.40	83.19
Deep-Ensemble [5]		96.57	96.57	98.28

TABLE 3.20 – Performances de la fusion de caractéristiques pour la classification multiclasse avec la première configuration de la base de données

Descripteur	Classifieur	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
Avant augmentation des données				
HoG + AlexNet	KNN	99.80	99.83	99.84
HoG + AlexNet	SVM	99.32	99.43	99.22
HoG + AlexNet	Naïve Bayes	55.21	73.95	76.72
HoG + AlexNet	Random Forest	85.11	88.06	92.66
Après augmentation des données				
HoG + AlexNet	KNN	99.82	99.82	99.84
HoG + AlexNet	SVM	97.64	98.16	96.56
HoG + AlexNet	Naïve Bayes	49.63	69.69	68.75
HoG + AlexNet	Random Forest	79.37	83.36	86.88
Deep-Ensemble [5]		95.98	/	97.71

On confirme à travers ces tests les remarques qu'on a fait précédemment sur la puissance et la robustesse de cette fusion de caractéristiques pour la classification des images IRM suivi du KNN en ayant une classification parfaite de 100% durant les tests pour la classification binaire et un F1-score de 99.82% pour la classification multiclasse ce qui en fait un meilleur résultat que l'actuelle référence dans l'état de l'art avec les travaux de A. Loddo et Al. [5].

On peut aussi reprendre la même remarque que précédemment vis-à-vis de l'augmentation des données pour la classification binaire qui ne permet pas d'améliorer les résultats retrouvés et par conséquent l'utilisation du HOG et d'AlexNet en tant qu'extracteur de caractéristiques est amplement suffisant pour la classification des images sans même procéder à une augmentation des données ce qui en fait une solution très intéressante.

On présente dans les figures 3.14 et 3.15 les matrices de confusions simples et normalisées du meilleur modèle de classification pour la classification binaire et multiclasse respectivement, ainsi mettant en valeur la qualité de la classification.

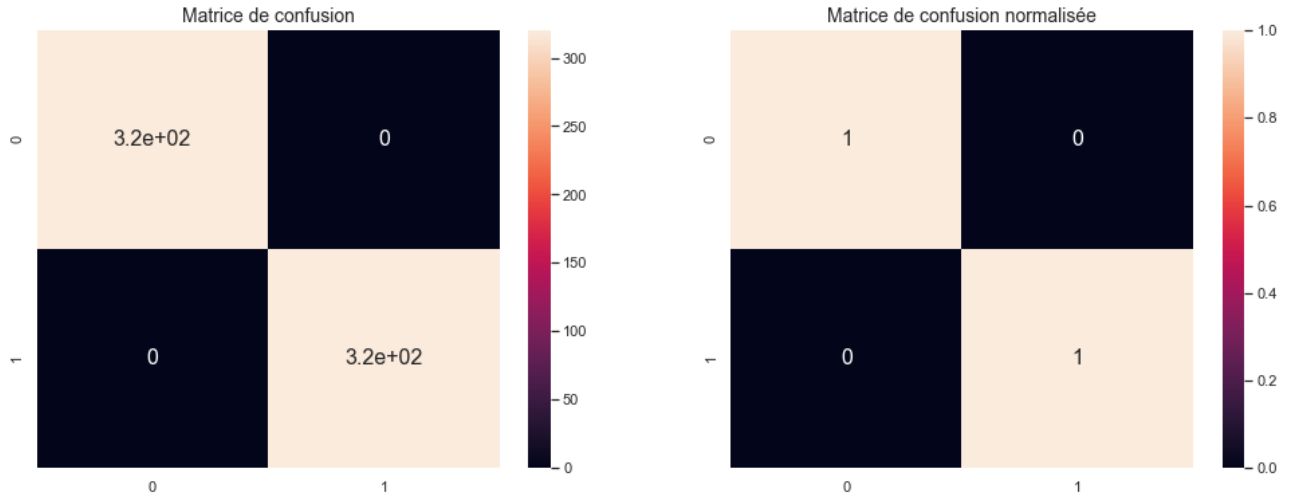


FIGURE 3.14 – Matrice de confusion du modèle de fusion de caractéristiques pour la classification binaire suivant la première distribution de la base de données

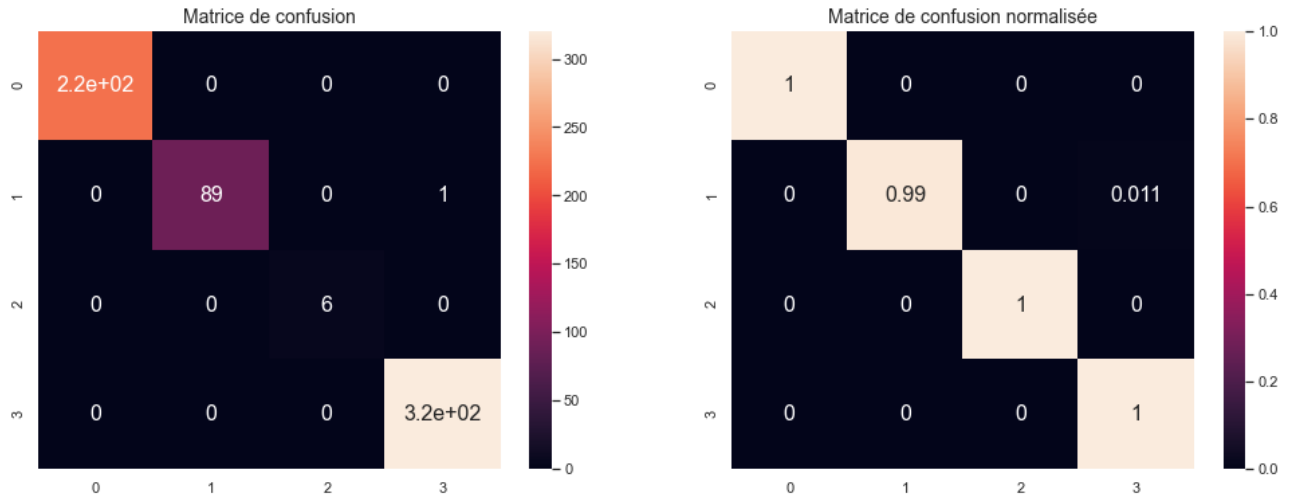


FIGURE 3.15 – Matrice de confusion du modèle de fusion de caractéristiques pour la classification multiclasse suivant la première distribution de la base de données

3.5.5.4 Deuxième distribution de la base de données

Les tableaux 3.21 et 3.22 quant à eux représentent les performances atteintes avec la fusion de caractéristiques pour la classification binaire et multiclasse respectivement se basant sur la deuxième configuration de la base de données.

TABLE 3.21 – Performances de la fusion de caractéristiques pour la classification binaire avec la deuxième configuration de la base de données

Descripteur	Classifieur	Exactitude (%)	Sensibilité (%)	Spécificité (%)
Avant augmentation des données				
HoG + AlexNet	KNN	90.23	90.54	90.10
HoG + AlexNet	SVM	79.20	78.43	79.59
HoG + AlexNet	Naïve Bayes	65.83	73.13	61.65
HoG + AlexNet	Random Forest	68.41	86.69	62
Après augmentation des données				
HoG + AlexNet	KNN	85.93	90.29	81.98
HoG + AlexNet	SVM	82.17	88.10	77.59
HoG + AlexNet	Naïve Bayes	60.44	59.29	62.07
HoG + AlexNet	Random Forest	68.57	89.36	62.09

TABLE 3.22 – Performances de la fusion de caractéristiques pour la classification multiclasse avec la deuxième configuration de la base de données

Descripteur	Classifieur	F1-score (%)	AUC ROC Score (%)	Exactitude (%)
Avant augmentation des données				
HoG + AlexNet	KNN	79.52	84.68	84.36
HoG + AlexNet	SVM	61.89	73.55	76.23
HoG + AlexNet	Naïve Bayes	58.70	34.54	58.17
HoG + AlexNet	Random Forest	32.84	58.54	62.47
Après augmentation des données				
HoG + AlexNet	KNN	85.34	88.99	85.22
HoG + AlexNet	SVM	78.63	72.43	77.87
HoG + AlexNet	Naïve Bayes	36.03	58.22	56.37
HoG + AlexNet	Random Forest	34.85	59.89	63.41

On reprend les mêmes remarques que précédemment, prouvant encore une fois, la puissance de cette architecture de fusion de caractéristiques suivi du KNN qui donne encore une fois les meilleures performances avec pour la classification binaire une précision de 90.23% au plus et un F1-score de 85.34% pour la classification multiclasse. L'apport de l'augmentation des données se ressent sur cette configuration, pour la classification binaire où les classes sont équilibrée on voit que l'augmentation des données a été néfaste passant d'une précision de 90.23% à une précision de 85.93% donc ajoutant un léger sur-apprentissage des données d'entraînement.

Par contre, pour la classification multiclasse où les classes sont clairement déséquilibrées de base, on voit bien que l'augmentation des données a amélioré les performances du modèle les faisant passer d'un F1-score de 79.52% à 85.34% après augmentation.

On saisi encore une fois l'opportunité de souligner que les résultats de la deuxième distribution sont effectivement inférieurs à ceux retrouvés avec la première distribution, et ceci pour tous les modèles traités si-dessus. Ce qui renforce notre hypothèse sur la fuite de données.

On termine cette discussion en présentant dans les figures 3.16 et 3.17 les matrices de confusions simples et normalisées du meilleur modèle de classification pour la classification binaire et multiclasse respectivement, ainsi mettant en valeur la qualité de la classification obtenues avec ces modèles.

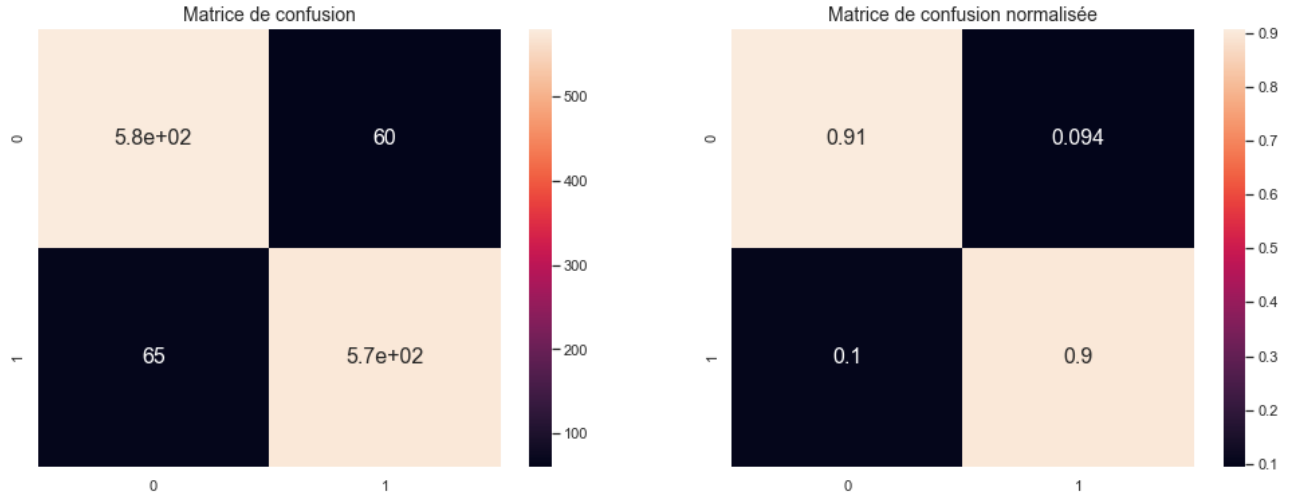


FIGURE 3.16 – Matrice de confusion du modèle de fusion de caractéristiques pour la classification binaire suivant la deuxième distribution de la base de données

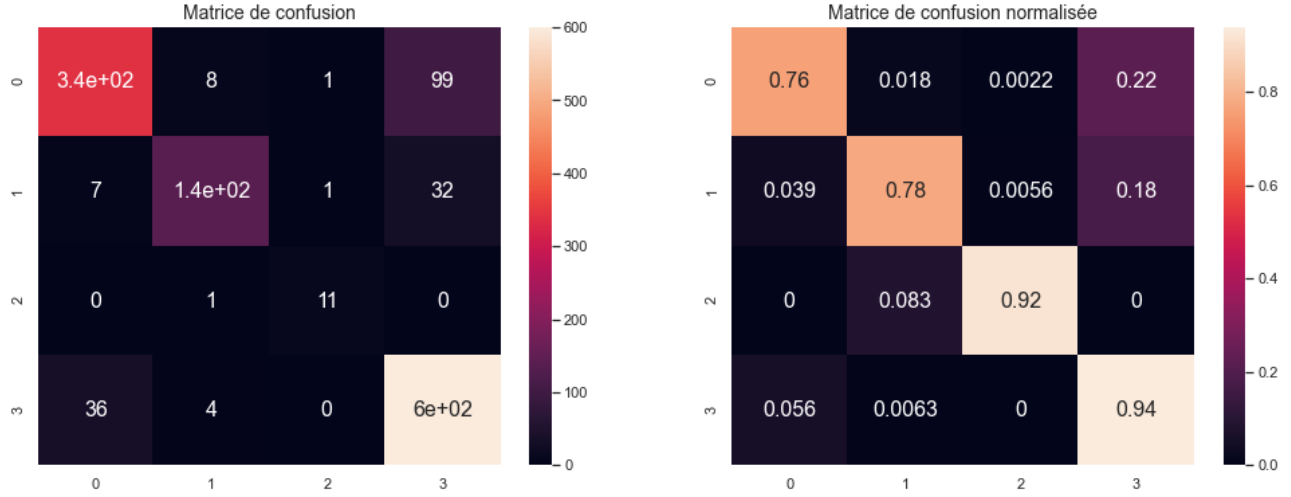


FIGURE 3.17 – Matrice de confusion du modèle de fusion de caractéristiques pour la classification multiclasse suivant la deuxième distribution de la base de données

3.6 Récapitulatif des résultats

Les figures 3.18 et 3.19 ci-dessous englobent les meilleurs résultats de chaque configuration pour les deux distributions et montre l'évolution des performances suivant les modèles sélectionnés dans chaque étape :

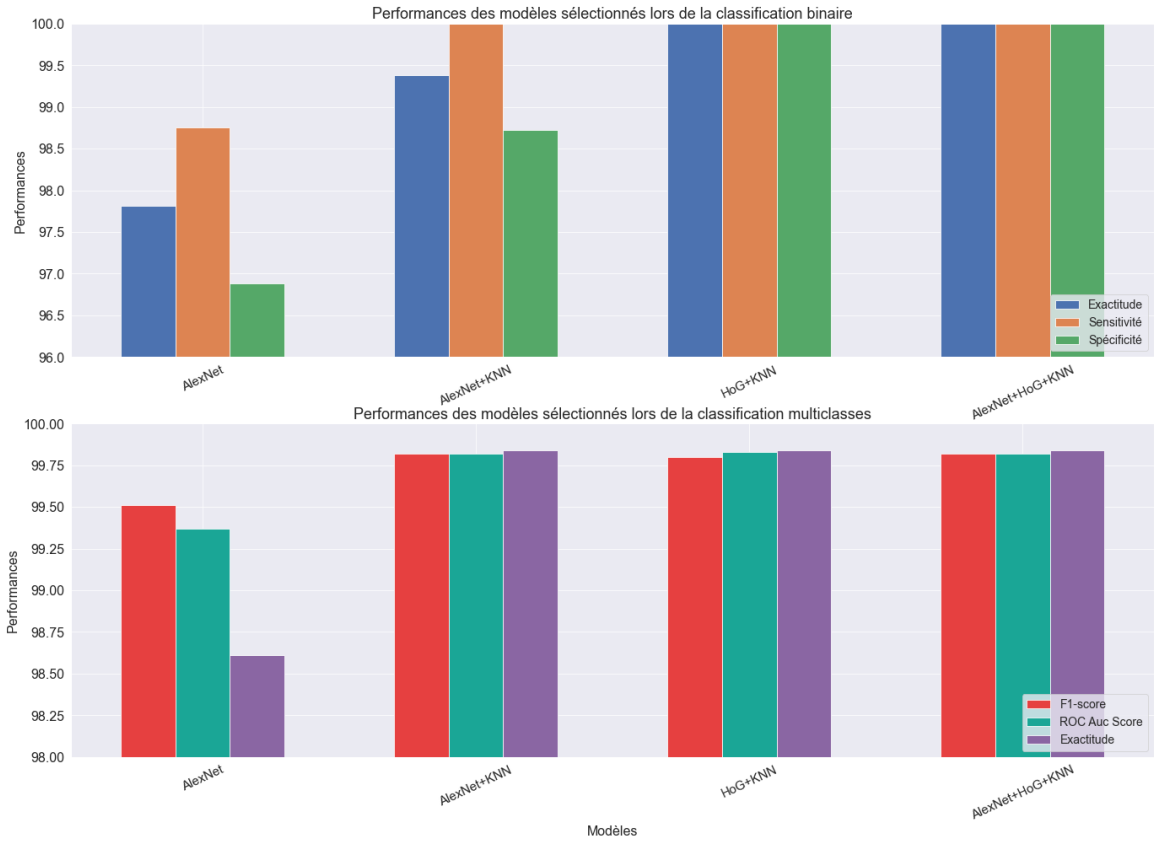


FIGURE 3.18 – Comparaison des performances des différents modèles sélectionnés suivant la première distribution

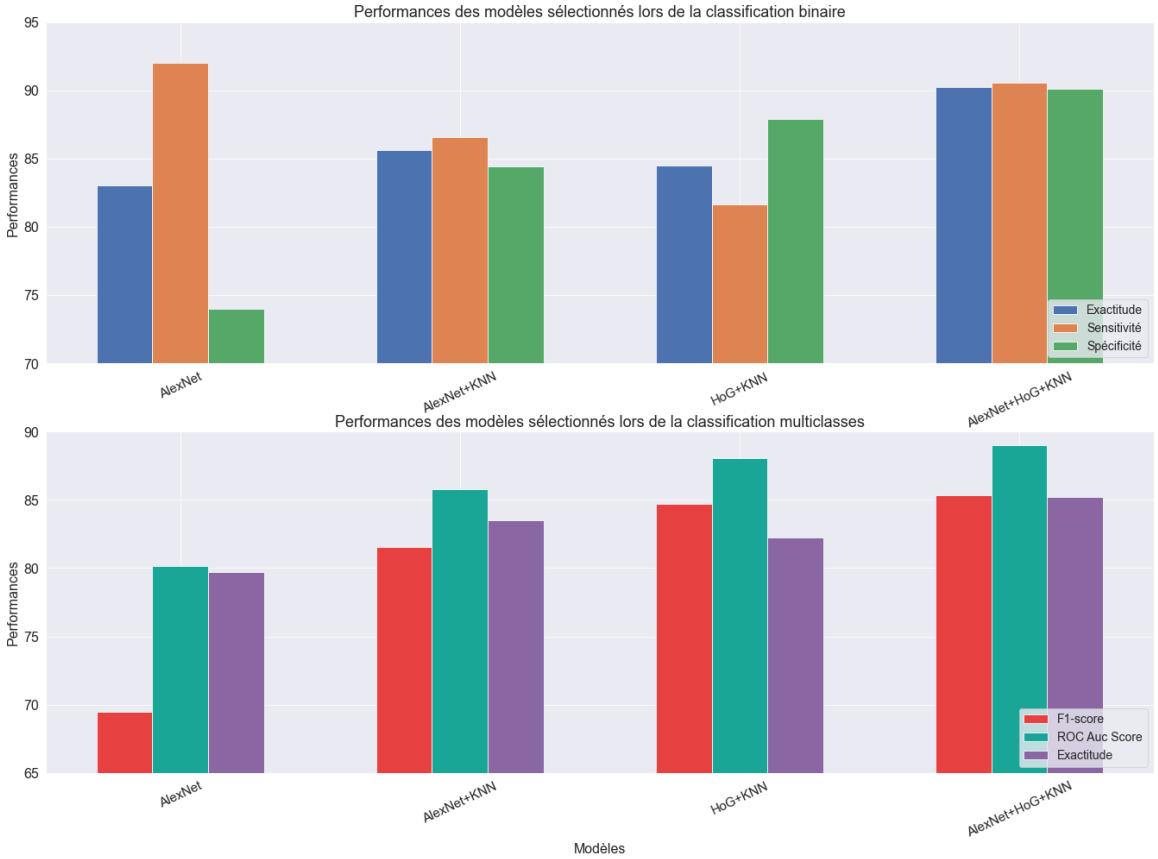


FIGURE 3.19 – Comparaison des performances des différents modèles sélectionnés suivant la seconde distribution

3.7 Conclusion

Dans ce chapitre on a commencé par présenter les outils logiciels et matériels utilisés dans notre projet en passant par les bases de données et les méthodes d'évaluation de nos différents modèles et on conclue par une présentation succinte des différents tests effectués commençant par l'utilisation des modèles pré-entraînés : AlexNet, ResNet-101 et Inception ResNet-V2. Par la suite, on a opté pour d'autres classifieurs que le conventionnel réseau de neurones observant une amélioration claire des performances, on a par la suite présenter la fusion des caractéristiques avec le HOG et qui représente l'architecture finale de notre projet et donnant les meilleurs résultats dépassant clairement les performances atteintes par A. Loddo et al. [5].

On peut alors conclure au bout de ce chapitre sur trois points importants :

- L'augmentation de données est bénéfique pour la classification multiclasse, mais elle est superflue pour la grande majorité des cas de classification binaire.
- La subdivision de la base de données selon Loddo et al. [5] donne de meilleurs résultats que la subdivision originale de l'auteur, mais la subdivision originale est plus représentative de la robustesse des modèles.
- La robustesse de la fusion de caractéristiques avancée dans ce projet pour la classification, autant binaire que multiclasse, des images IRM des différents niveaux de démence de la maladie d'Alzheimer, assurant à chaque fois et sur les deux distributions effectuées, les meilleures performances de classification.

CONCLUSION GÉNÉRALE

La maladie d'Alzheimer est une des maladies les plus graves au monde, elle touche des millions de personnes dans le monde et ce nombre ne fait qu'augmenter à travers les années ce qui incite les équipes de recherches du monde entier à travailler dans le combat contre cette maladie.

Dans ce mémoire on a retracé notre contribution dans ce combat. On a commencé par mettre en lumière certains parmi les travaux les plus marquants dans la classification des malades atteints d'Alzheimer se basant sur des images IRM. On a décidé d'aborder le plus grand jeu de données labelisé (et aussi le plus récent) d'images IRM de patients atteints de la maladie d'Alzheimer et disponible sur Kaggle en utilisant des techniques de traitement d'images et d'apprentissage automatique.

Notre approche a été alors sur une fusion de caractéristiques entre un CNN et un histogramme de gradients orientés (HOG) suivis d'un KNN donnant les meilleurs résultats à ce jour sur cette base de données avoisinant les 99% de précision dans certaines configurations.

Ce travail a donc montré la puissance de la fusion de caractéristiques entre l'apprentissage profond avec le CNN et les descripteurs statistiques de traitement d'images à travers le HOG, ce qui a donné des résultats extrêmement encourageants.

De plus, d'après les comparaisons entre les deux subdivisions de la base de données, nous posons l'hypothèse de fuite de données lors de l'utilisation de la méthode de A. Loddo et al. [5]. Et nous trouvons intéressant d'étudier cette hypothèse par la suite en tant que perspective future.

Bien-sûr ce travail qui représente une aide au diagnostique n'est pas une finalité en soit et ne représente que la première pierre dans un long combat contre cette maladie, parmi les perspectives éventuelles en vu de l'évolution de ce projet on peut citer :

- L'investigation et mise en place de méthodes basées sur l'études des scans IRM 3D avec des CNN 3D et autres méthodes d'apprentissage automatique.
- Confirmation de l'hypothèse de fuite de données en utilisant les modèles sur un nouveau jeu de données, notamment celui de la Alzheimer's Disease Neuroimaging Initiative (ADNI).
- Développement d'un modèle pouvant prendre les images IRM sous format DICOM (Digital Imaging and Communications in Medicine) ou NIfTI (Neuroimaging Informatics Technology Initiative), formats les plus courants dans l'imagerie médicale, permettant de prendre les images directement en sortie du scanner IRM et de concevoir un logiciel d'aide au diagnostique efficace et opérationnel.
- Généralisation de ce modèle au vu de la séparation entre les patients atteints d'Alzheimer

et d'autres patients atteints d'autres pathologies cérébrales : Parkinson...

- Mise en place d'une base de données d'images IRM de patients Algériens, permettant l'étude de cette maladie et son combat sur le plan national.

BIBLIOGRAPHIE

[1] World alzheimer report 2021 | alzheimer's disease international (adi). https://www.alzint.org/resource/world-alzheimer-report-2021/.

[2] El-Sayed Ahmed El-Dahshan, Tamer Hosny, and Abdel-Badeeh M Salem. Hybrid intelligent techniques for mri brain images classification. *Digital signal processing*, 20(2) :433–441, 2010.

[3] Naimul Mefraz Khan, Nabila Abraham, and Marcia Hon. Transfer learning with intelligent training data selection for prediction of alzheimer's disease. *IEEE Access*, 7 :72726–72735, 2019.

[4] Hina Nawaz, Muazzam Maqsood, Sitara Afzal, Farhan Aadil, Irfan Mehmood, and Seungmin Rho. A deep feature-based real-time system for alzheimer disease stage detection. *Multimedia Tools and Applications*, 80(28) :35789–35807, 2021.

[5] Andrea Loddo, Sara Buttau, and Cecilia Di Ruberto. Deep learning based pipelines for alzheimer's disease diagnosis : a comparative study and a novel deep-ensemble method. *Computers in biology and medicine*, 141 :105032, 2022.

[6] Alzheimer's dataset (4 class of images) | kaggle. https://www.kaggle.com/datasets/tourist55/alzheimers-dataset-4-class-of-images.

[7] Md Rishad Ahmed, Yuan Zhang, Zhiquan Feng, Benny Lo, Omer T Inan, and Hongen Liao. Neuroimaging and machine learning for dementia diagnosis : recent advancements and future prospects. *IEEE reviews in biomedical engineering*, 12 :19–33, 2018.

[8] Manhua Liu, Fan Li, Hao Yan, Kundong Wang, Yixin Ma, Li Shen, Mingqing Xu, Alzheimer's Disease Neuroimaging Initiative, et al. A multi-model deep convolutional neural network for automatic hippocampus segmentation and classification in alzheimer's disease. *Neuroimage*, 208 :116459, 2020.

[9] Imagerie par résonance magnétique — wikipédia. https://bit.ly/37tlhz0.

[10] WHAT DO WE KNOW. What is alzheimer's disease ? 1986.

[11] Jeffrey L Cummings and Greg Cole. Alzheimer disease. *Jama*, 287(18) :2335–2338, 2002.

[12] Rudy J Castellani, Raj K Rolston, and Mark A Smith. Alzheimer disease. *Disease-a-month : DM*, 56(9) :484, 2010.

[13] Ashley I Bush. The metallobiology of alzheimer's disease. *Trends in neurosciences*, 26(4) :207–214, 2003.

[14] Anil Kumar, Jaskirat Sidhu, Amandeep Goyal, and Jack W Tsao. Alzheimer disease. 2018.

[15] La maladie d'alzheimer. https://bit.ly/397vBNG.

[16] Les chiffres clés | fondation médéric alzheimer. https://bit.ly/3wbWPKO.

[17] Sérgio Pereira, Adriano Pinto, Victor Alves, and Carlos A Silva. Brain tumor segmentation using convolutional neural networks in mri images. *IEEE transactions on medical imaging*, 35(5) :1240–1251, 2016.

[18] Yudong Zhang, Shuihua Wang, Yuxiu Sui, Ming Yang, Bin Liu, Hong Cheng, Junding Sun, Wenjuan Jia, Preetha Phillips, and Juan Manuel Gorriz. Multivariate approach for alzheimer's disease detection using stationary wavelet entropy and predator-prey particle swarm optimization. *Journal of Alzheimer's Disease*, 65(3) :855–869, 2018.

[19] Shaik Basheera and M Satya Sai Ram. Convolution neural network–based alzheimer's disease classification using hybrid enhanced independent component analysis based segmented gray matter of t2 weighted magnetic resonance imaging with clinical valuation. *Alzheimer's & Dementia : Translational Research & Clinical Interventions*, 5 :974–986, 2019.

[20] Halebeedu Subbaraya Suresha and Srirangapatna Sampathkumaran Parthasarathy. Alzheimer disease detection based on deep neural network with rectified adam optimization technique using mri analysis. In *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, pages 1–6. IEEE, 2020.

[21] Ali Khazaee, Ata Ebrahimzadeh, and Abbas Babajani-Feremi. Application of advanced machine learning methods on resting-state fmri network for identification of mild cognitive impairment and alzheimer's disease. *Brain imaging and behavior*, 10(3) :799–817, 2016.

[22] Meysam Asgari, Jeffrey Kaye, and Hiroko Dodge. Predicting mild cognitive impairment from spontaneous spoken utterances. *Alzheimer's & Dementia : Translational Research & Clinical Interventions*, 3(2) :219–228, 2017.

[23] Kilian Hett, Vinh-Thong Ta, José V Manjón, Pierrick Coupé, Alzheimer's Disease Neuroimaging Initiative, et al. Adaptive fusion of texture-based grading for alzheimer's disease classification. *Computerized Medical Imaging and Graphics*, 70 :8–16, 2018.

[24] Iago RR Silva, Gabriela SL Silva, Rodrigo G de Souza, Wellington P dos Santos, and A de A Roberta. Model based on deep feature extraction for diagnosis of alzheimer's disease. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2019.

[25] J Ramírez, JM Górriz, F Segovia, R Chaves, D Salas-Gonzalez, M López, I Álvarez, and P Padilla. Computer aided diagnosis system for the alzheimer's disease based on partial least squares and random forest spect image classification. *Neuroscience letters*, 472(2) :99–103, 2010.

[26] Savita Dahiya, S Vijayalakshmi, and Munish Sabharwal. Alzheimer's disease detection using machine learning : A review. In *2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICAC3N)*, pages 288–293. IEEE, 2021.

[27] Tooba Altaf, S Anwar, Nadia Gul, N Majeed, and M Majid. Multi-class alzheimer disease classification using hybrid features. In *IEEE future technologies conference*, 2017.

[28] Bijen Khagi, Chung Ghiu Lee, and Goo-Rak Kwon. Alzheimer's disease classification from brain mri based on transfer learning from cnn. In *2018 11th biomedical engineering international conference (BMEiCON)*, pages 1–4. IEEE, 2018.

[29] Blessy C Simon, D Baskar, and VS Jayanthi. Alzheimer's disease classification using deep convolutional neural network. In *2019 9th International Conference on Advances in Computing and Communication (ICACC)*, pages 204–208. IEEE, 2019.

[30] Marcia Hon and Naimul Mefraz Khan. Towards alzheimer's disease classification through transfer learning. In *2017 IEEE International conference on bioinformatics and biomedicine (BIBM)*, pages 1166–1169. IEEE, 2017.

[31] Myfanwy Thomas and Michael Isaac. Alois alzheimer : A memoir. *Trends in Neurosciences*, 1987.

[32] Marjorie LeMay, Juliene L Stafford, Tamas Sandor, Marilyn Albert, Hani Haykal, and Amir Zamani. Statistical assessment of perceptual ct scan ratings in patients with alzheimer type dementia. *Journal of computer assisted tomography*, 10(5) :802–809, 1986.

[33] Truda K Shonk, Rex A Moats, Patricia Gifford, Thomas Michaelis, Jennifer C Mandigo, Judith Izumi, and Brian D Ross. Probable alzheimer disease : diagnosis with proton mr spectroscopy. *Radiology*, 195(1) :65–72, 1995.

[34] A Brand, C Richter-Landsberg, and D Leibfritz. Multinuclear nmr studies on the energy metabolism of glial and neuronal cells. *Developmental neuroscience*, 15(3-5) :289–298, 1993.

[35] Chetan Patil, MG Mathura, S Madhumitha, S Sumam David, Merwyn Fernandes, Anand Venugopal, and B Unnikrishnan. Using image processing on mri scans. In *2015 IEEE International Conference on Signal Processing, Informatics, Communication and Energy Systems (SPICES)*, pages 1–5. IEEE, 2015.

[36] Claudia Plant, Stefan J Teipel, Annahita Oswald, Christian Böhm, Thomas Meindl, Janaina Mourao-Miranda, Arun W Bokde, Harald Hampel, and Michael Ewers. Automated detection of brain atrophy patterns based on mri for the prediction of alzheimer's disease. *Neuroimage*, 50(1) :162–174, 2010.

[37] Lulu Yue, Xiaoliang Gong, Kaibo Chen, Mingze Mao, Jie Li, Asoke K Nandi, and Maozhen Li. Auto-detection of alzheimer's disease using deep convolutional neural networks. In *2018 14th international conference on natural computation, fuzzy systems and knowledge discovery (ICNC-FSKD)*, pages 228–234. IEEE, 2018.

[38] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4) :115–133, 1943.

[39] Donald Olding Hebb. *The organization of behavior : A neuropsychological theory*. Psychology Press, 2005.

[40] Herbert L Gelernter and Nathaniel Rochester. Intelligent behavior in problem-solving machines. *IBM Journal of Research and Development*, 2(4) :336–345, 1958.

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

[42] David E Rumelhart, Richard Durbin, Richard Golden, and Yves Chauvin. Backpropagation : The basic theory. *Backpropagation : Theory, architectures and applications*, pages 1–34, 1995.

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

[44] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[45] David H Hubel and Torsten N Wiesel. Receptive fields of cells in striate cortex of very young, visually inexperienced kittens. *Journal of neurophysiology*, 26(6) :994–1002, 1963.

[46] Torsten N Wiesel and David H Hubel. Single-cell responses in striate cortex of kittens deprived of vision in one eye. *Journal of neurophysiology*, 26(6) :1003–1017, 1963.

[47] David H Hubel and Torsten N Wiesel. Shape and arrangement of columns in cat's striate cortex. *The Journal of physiology*, 165(3) :559, 1963.

[48] David H Hubel and Torsten N Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of physiology*, 195(1) :215–243, 1968.

[49] Kunihiko Fukushima and Sei Miyake. Neocognitron : A self-organizing neural network model for a mechanism of visual pattern recognition. In *Competition and cooperation in neural nets*, pages 267–285. Springer, 1982.

[50] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11) :2278–2324, 1998.

[51] A comprehensive guide to convolutional neural networks — the eli5 way | by sumit saha | towards data science. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53. (Accessed on 04/22/2022).

[52] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv :1409.1556*, 2014.

[53] Bekir Karlik and A Vehbi Olgac. Performance analysis of various activation functions in generalized mlp architectures of neural networks. *International Journal of Artificial Intelligence and Expert Systems*, 1(4) :111–122, 2011.

[54] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv :1511.07289*, 2015.

[55] Imagenet. https://www.image-net.org/.

[56] Wikipedia. Apprentissage par transfert — Wikipedia, the free encyclopedia. https://bit.ly/381v0wF, 2022.

[57] Transfer learning : Qu'est-ce que c'est ? https://datascientest.com/transfer-learning.

[58] Wikipedia. AlexNet — Wikipedia, the free encyclopedia. https://bit.ly/3vIbuON.

[59] 7.1. deep convolutional neural networks (alexnet) — dive into deep learning 0.17.5 documentation. https://bit.ly/3LKvmGC.

[60] Resnet-101 - wolfram neural net repository. https://bit.ly/3vW6yEU.

[61] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[62] Inception module explained | papers with code. https://bit.ly/3kFutU0.

[63] Deep learning : Googlenet explained | by richmond alake | towards data science. https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765.

[64] A simple guide to the versions of the inception network | by bharath raj | towards data science. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.

[65] Keras - wikipedia. https://en.wikipedia.org/wiki/Keras.

[66] Mikel Galar, Alberto Fernandez, Edurne Barrenechea, Humberto Bustince, and Francisco Herrera. A review on ensembles for the class imbalance problem : bagging-, boosting-, and hybrid-based approaches. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(4) :463–484, 2011.

[67] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3) :273–297, 1995.

[68] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[69] Eugene M Kleinberg. Stochastic discrimination. *Annals of Mathematics and Artificial intelligence*, 1(1) :207–239, 1990.

[70] Leo Breiman. Random forests. *Machine learning*, 45(1) :5–32, 2001.

[71] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[72] Muhammad Irfan Sharif, Jian Ping Li, Muhammad Attique Khan, and Muhammad Asim Saleem. Active deep neural network features selection for segmentation and recognition of brain tumors using mri images. *Pattern Recognition Letters*, 129 :181–189, 2020.

[73] Tanzila Saba, Ahmed Sameh Mohamed, Mohammad El-Affendi, Javeria Amin, and Muhammad Sharif. Brain tumor detection using fusion of hand crafted and deep learning features. *Cognitive Systems Research*, 59 :221–230, 2020.