REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université Mohamed Khider – BISKRA
Faculté des Sciences Exactes, des Sciences de la Nature et de la Vie
Département d'informatique

N° d'ordre : IA09/M2/2022

Mémoire

Présenté pour obtenir le diplôme de master académique en

Informatique

Parcours : **Intelligence Artificielle (IA)**

Une approche Deep Learning pour le diagnostic préalable de la tuberculose

---

Par :
DJOUANI AYA

Soutenu le 27/06/2022 devant le jury composé de :

SAOULI RACHIDA	Professeur	Présidente
BELOUAAR HOUCINE	MCA	Rapporteur
MERIZIG ABDELHAK	MAB	Examinateur

Année universitaire 2021-2022

R e m e r c i e m e n t s

Je voudrais exprimer toute ma reconnaissance et adresser mes sincères remerciements à mon honorable encadrant Monsieur BELOUAAR HOUCINE pour sa disponibilité, son aide, ses orientations et ses conseils précieux ainsi que son encouragement et sa confiance qui m'ont permis de ne jamais faiblir et d'aller toujours plus loin dans mon travail.

J'adresse également mes grands remerciements aux honorables membres du jury qui ont bien voulu accepter d'évaluer ce modeste travail.

Je remercie ma famille pour son aide et son soutien moral qui ont été pour moi une source de courage et de confiance.

Enfin, merci à toutes les personnes qui, par un mot ou par un geste, m'ont donné la force de continuer à travailler afin d'atteindre mes objectifs.

Aya Djouani

D é d i c a c e

Je dédie ce mémoire :

- A la mémoire de ma mère, Allah yarhamha, qui m'avait toujours exhortée à persévérer et à aller plus loin dans mes études.

- A mon père et à ma sœur qui ont toujours été à mes cotés.

- A toutes mes professeures et à tous mes professeurs qui m'ont accompagnée durant mon cursus universitaire et m'ont permis d'acquérir les compétences scientifiques nécessaires pour la suite de mes parcours académique et professionnel.

Aya Djouani

Résumé

Le diagnostic précoce de la tuberculose pulmonaire (TBP) pourrait réduire le taux de mortalité de cette maladie, à des niveaux plus faibles. L'examen radiographique à Rayons X du thorax constitue une méthode efficace pour identifier la TBP. Mais la lecture précise de ces radiographies nécessite la disponibilité de radiologues qualifiés, ce qui n'est pas toujours le cas dans les pays en voie de développement.

L'Intelligence Artificielle, qui englobe les réseaux de neurones convolutifs (CNN) de l'apprentissage profond (Deep Learning), a le potentiel de diagnostiquer la maladie à un stade précoce et donne donc une solution adéquate.

En explorant le potentiel de l'apprentissage profond pour l'analyse des images, nous avons pu étudier les concepts du Deep Learning avec des algorithmes des réseaux neuronaux convolutifs (CNN) pour construire un modèle de classification binaire de ces images, qui peut donc, prédire la présence ou l'absence de la tuberculose sur une radiographie pulmonaire du patient.

En nous basant sur le concept de l'Apprentissage Ensembliste, nous avons proposé un modèle constitué d'un ensemble de trois (03) architectures CNN différentes à plusieurs paramètres et avec l'utilisation de la technique du Transfer Leaning. Pour l'entrainement et la validation, nous avons utilisé un jeu de données ou dataset de 800 radiographies pulmonaires.

Les résultats obtenus, en termes de classification, sont importants et prometteurs.

Mots-clés : Diagnostic, Tuberculose pulmonaire, Rayons X du Thorax, Intelligence Artificielle, Apprentissage profond, CNN, Apprentissage Ensembliste.

Abstract

Early diagnosis of pulmonary tuberculosis (PTB) may reduce the mortality rate of this disease to lower levels. Chest X-ray is an effective method of identifying PTB. But accurate reading of these X-rays requires the availability of trained radiologists, which is not always the case in developing countries.

Artificial Intelligence, which includes Deep Learning Convolutional Neural Networks (CNN), has the potential to diagnose the disease at an early stage and thus provide an adequate solution. Exploring the potential of deep learning for image analysis, we were able to study the concepts of Deep Learning with convolutional neural network (CNN) algorithms to build a binary classification model of these images, which can therefore, predict the presence or absence of tuberculosis on a patient's chest X-ray.

Based on the concept of Ensemble Learning, we have proposed a model consisting of a set of three (03) different CNN architectures with several parameters and using the Transfer Leaning technique. For training and validation, we used a dataset of 800 chests X-ray.

The results obtained, in terms of classification, are important and promising.

Key-Words: Diagnosis, Pulmonary tuberculosis, Chest X-ray, Artificial Intelligence, Deep Learning, CNN, Ensemble Learning.

Table des matières

Introduction générale **1**

1 État de l'art **3**

1.1 Introduction 3

1.2 Santé numérique 3

1.2.1 Définition 4

1.2.2 Domaines d'application de la santé numérique 4

1.3 Intelligence Artificielle (IA) 6

1.3.1 Définition 6

1.3.2 Principaux avantages de l'IA 6

1.3.3 Exemples de l'IA dans les domaines de la santé numérique 6

1.4 Apprentissage Automatique (Machine Learning) 8

1.4.1 Définition 8

1.4.2 Types d'apprentissage Automatique 9

1.5 Réseaux de Neurones Artificiels 11

1.5.1 Définition 11

1.5.2 Neurone Biologique 12

1.5.3 Neurone Artificiel 12

1.5.4 Constitution d'un réseau neuronal 13

1.5.5 Couches d'un réseau de neurones 14

1.5.6 Fonction d'activation 15

1.6 Apprentissage profond (Deep Learning) 15

1.6.1 Définition 15

1.6.2 Réseau de neurones convolutif (CNN ou ConvNet) 16

1.6.3 Autres Algorithmes de deep learning 20

1.6.4 Différence entre l'apprentissage automatique et l'apprentissage profond 21

1.6.5 Transfer Learning (apprentissage par transfert) 21
1.6.6 Ensemble Learning 22
1.6.7 Quelques architectures des CNNs 23
1.7 La maladie de la tuberculose 27
1.7.1 Définition 27
1.7.2 Cause de la tuberculose 27
1.7.3 Evolution du BK dans l'organisme 28
1.7.4 Symptômes de la tuberculose active 28
1.7.5 Transmission de la maladie 29
1.7.6 Diagnostic 29
1.7.7 Traitement 29
1.8 Imagerie médicale 29
1.9 Radiographie du thorax (Chest X-Ray) 31
1.10 Détection Assistée par Ordinateur de la tuberculose 32
1.11 Travaux Connexes 33
1.11.1 Tableau récapitulatif des travaux connexes 35
1.12 Conclusion 36

2 Conception et Implémentation 37
2.1 Introduction 37
2.2 Modèle proposé 37
2.3 Approches du Transfer Learning 39
2.4 Ensemble de données (Dataset) 40
2.5 Pré-traitement 41
2.5.1 Divison du dataset (Dataset splitting) 41
2.5.2 Redimensionnement des images 42
2.5.3 Augmentation des données (Data Augmentation) 42
2.6 Couches du modèle CNN 42
2.7 Hyper-paramètres d'apprentissage CNN 43
2.7.1 Taux d'apprentissage (Learning Rate) 43
2.7.2 Epoque (epoch) 43
2.7.3 Batch size 43
2.7.4 Etape par époque (Step per epoch) 43

2.7.5 Etapes de validation (Validation steps) . . . 43
2.7.6 Fonction de perte (Loss Function) . . . 44
2.7.7 Fonction de classification Softmax . . . 44
2.7.8 Optimiseur Adam . . . 44
2.8 Evaluation du modèle CNN (métriques) . . . 44
2.8.1 Matrice de confusion . . . 44
2.8.2 Mesures de performance . . . 45
2.9 Implémentation . . . 46
2.9.1 Langage de programmation utilisé (Python) . . . 46
2.9.2 Google Colab . . . 46
2.9.3 Frameworks, IDE et Bibliothèques Python . . . 46
2.9.4 Importation des bibliothèques . . . 50
2.9.5 Préparation des données . . . 50
2.9.6 Application du Transfer Learning . . . 51
2.9.7 Initialisation des paramètres . . . 53
2.9.8 Compilation du modèle . . . 54
2.9.9 Entrainement du modèle . . . 55
2.9.10 Evaluation du modèle . . . 55
2.9.11 Test du modèle . . . 56
2.9.12 Modèle ensembliste . . . 57
2.10 Conclusion . . . 58

3 Résultats et Discussion . . . 59
3.1 Introduction . . . 60
3.2 Résultats et discussion . . . 60
3.2.1 Premier modèle (VGG-16) . . . 61
3.2.2 Deuxième modèle (MobileNet-V2) . . . 64
3.2.3 Troisième modèle (DenseNet-169) . . . 66
3.2.4 Modèle final (Modèle ensembliste) . . . 68
3.2.5 Discussion . . . 69
3.3 Comparaison avec les travaux connexes . . . 69
3.4 Conclusion . . . 70

Conclusion Générale et Travaux Futurs **71**

Table des figures

1.1 Un médecin pratiquant son métier à distance [5]. 5
1.2 Glucomètre sans fils [8]. 5
1.3 Système "da Vinci Si HD" [11]. 7
1.4 Le logiciel InferRead CT Pneumonia.AI [13]. 8
1.5 Exemple d'apprentissage supervisé [18]. 9
1.6 Exemple d'apprentissage non-supervisé [9]. 10
1.7 Le scenario d'apprentissage par renforcement [21]. 11
1.8 Neurone biologique [24]. 12
1.9 Similitude entre neurone biologique et neurone artificiel [25]. 13
1.10 Principe de fonctionnement d'un neurone artificiel [26]. 13
1.11 Architecture d'un réseau de neurones [17]. 14
1.12 Fonctions d'activation : Sigmoid, Tanh, RELU [29]. 15
1.13 Correlation entre l'IA , ML et DL [31]. 16
1.14 Opération de convolution [34]. 18
1.15 Différence entre max pooling et average pooling [35]. 18
1.16 Couche fully-connected [36]. 19
1.17 Exemple d'une architecture CNN [37]. 20
1.18 Différence entre le ML et le DL [38]. 21
1.19 Transfer Learning [40]. 22
1.20 Schéma illustratif de l'ensemble learning [41]. 23
1.21 Architecture AlexNet [42]. 23
1.22 Architecture LeNet [44]. 24
1.23 Architecture du VGG-16 [45]. 25
1.24 Architecture du VGG-19 [46]. 25
1.25 Architecture du DensNet-169 [47]. 26
1.26 Bloc MobileNet-V2 [48]. 26

1.27 Bacille de Koch [50]. 28
1.28 Radiographie [51]. 30
1.29 Scanner [52]. 30
1.30 Echographie [53]. 31
1.31 Procédure de la radiographie du Thorax [55]. 32
1.32 RX du thorax, TBP étendue et bilatérale [55]. 32

2.1 Schéma illustratif du fonctionnement de notre modèle. 38
2.2 Diagramme du modèle CNN. 38
2.3 Utilisation du modèle VGG-16 comme extracteur de features. 40
2.4 Logo de Spyder. 47
2.5 Logo de Keras. 47
2.6 Logo de TensorFlow. 47
2.7 Outils de développement Python. 49
2.8 Code d'importation des bibliothèques. 50
2.9 Code du montage de Google Drive. 50
2.10 Code montrant le contenu du dataset. 51
2.11 Code de création d'une instance de la classe ImageDataGenerator. 51
2.12 Code de chargement du réseau VGG-16. 52
2.13 Code d'ajout des nouvelles couches (notre classifieur). 52
2.14 Gel des première couches de convolution. 53
2.15 Code de compilation. 54
2.16 Paramètres. 54
2.17 Code d'entrainement. 55
2.18 Code d'évaluation. 56
2.19 Visualisation d'un cas de test. 57
2.20 Code d'implémentation de notre modèle final. 57
2.21 Code du traçage de notre modèle final. 58
2.22 Traçage de notre modèle final. 58

3.1 Courbe d'évolution de la précision du premier modèle. 61
3.2 Courbe d'évolution de la perte du premier modèle. 61
3.3 Résultats du test du premier modèle. 62

3.4 Matrice de confusion du premier modèle. 63
3.5 Courbe d'évolution de la précision du deuxième modèle. 64
3.6 Courbe d'évolution de la perte du deuxième modèle. 64
3.7 Résultats de test du deuxième modèle. 65
3.8 Matrice de confusion du deuxième modèle. 65
3.9 Courbe d'évolution de la précision du troisième modèle. 66
3.10 Courbe d'évolution de la perte du troisième modèle. 66
3.11 Résultats de test du troisième modèle. 67
3.12 Matrice de confusion du troisième modèle. 67
3.13 Résultats de test du modèle final (modèle ensembliste). 68
3.14 Matrice de confusion du modèle final (modèle ensembliste). 68

Liste des tableaux

1.1 Tableau récapitulatif des travaux connexes 35

2.1 Tableau des paramètres des couches CNN 42

3.1 Tableau comparatif 70

Introduction générale

L'imagerie médicale est devenue une technique indispensable et incontournable dans le diagnostic de diverses maladies. Les bases d'images médicales n'ont cessé d'augmenter et constituent aujourd'hui un gigantesque volume de données qui requiert de nouvelles méthodes de traitement. Cependant, les grandes variations et la complexité des données d'imagerie médicale rendent difficile la recherche de solutions analytiques ou de méthodes simples pour décrire et représenter, par exemple, des lésions ou des anatomies sur ces images. Une telle tache nécessite un apprentissage à partir d'exemples d'imagerie médicale, autrement dit, l'utilisation de l'apprentissage automatique ou le Machine Learning qui est un sous-domaine de l'Intelligence Artificielle (IA).

L'apprentissage automatique donne des solutions pour développer des outils permettant d'aider les médecins à faire des diagnostics précoces des maladies et à les traiter à temps. L'apprentissage profond ou Deep Learning est un sous-domaine de l'apprentissage automatique qui englobe un large éventail d'architectures de réseaux élaborés pour exécuter plusieurs tâches d'analyse d'images médicales comme la segmentation, la classification, la détection, …etc.

Le réseau de neurones convolutif ou Convolutional Neural Network (CNN) est l'algorithme du Deep Learning le plus performant dans la classification des images, comme les radiographies pulmonaires. Nous pouvons donc utiliser cette technique pour le diagnostic de la tuberculose pulmonaire (TBP) qui est une maladie infectieuse causée par une mycobactérie Mycobacterium tuberculosis qui affecte les poumons. Elle est l'une des dix premières causes de mortalité dans le monde [1]. Les tests traditionnels de la TBP produisent souvent des résultats inexacts ou trop longs pour être définitifs.

La problématique est comment faire un diagnostic rapide, précis et précoce pour mieux contrôler cette maladie en vue de son éradication, notamment dans des pays du tiers

monde où les ressources humaines et matérielles sont encore insuffisantes. L'utilisation des techniques de l'intelligence artificielle (IA) dans le domaine de l'imagerie médicale permet de développer des logiciels de Détection Assistée par Ordinateur (DAO) capables d'analyser les radiographies à rayon X du thorax. Le concept de l'e-Healthcare est basé sur ces nouvelles techniques de l'informatique et du numérique pour la mise en œuvre de systèmes de santé performants.

Objectifs

Ce projet a pour but d'étudier et de proposer un modèle de Deep Learning CNN pour la classification automatique des radiographies RX du thorax selon l'absence ou la présence de la tuberculose pulmonaire (TBP). La performance du modèle constitue une partie cruciale de ce travail.

Pour atteindre notre objectif, nous avons organisé notre mémoire en trois chapitres suivis d'une conclusion générale et travaux futurs. Chaque chapitre représente une partie du travail de ce projet qu'elle soit théorique ou pratique :

Chapitre 1 : État de l'art

Chapitre 2 : Conception et Implémentation

Chapitre 3 : Résultats et Discussion

Chapitre 1

État de l'art

1.1 Introduction

Dans ce chapitre, nous présenterons le contexte de notre travail. Nous aborderons le concept de la santé numérique ou e-Healthcare et son champ d'application ainsi que les avantages de l'intelligence artificielle et ses applications dans le domaine médical. Nous parlerons également des différents types de l'apprentissage automatique ou Machine Learning, des réseaux de neurones artificiels et de l'apprentissage profond ou deep learning. Les sujets de l'imagerie médicale notamment la radiographie à rayon X du thorax (Chest X-Ray) et de la Détection Assistée par Ordinateur de la tuberculose pulmonaire (TBP) seront également abordés. Enfin, nous clôturerons ce chapitre par l'étude de quelques travaux liés à notre problématique.

1.2 Santé numérique

En 2019, l'Organisation mondiale de la santé (OMS) a publié de nouvelles recommandations sur l'utilisation des technologies numériques pour améliorer la santé des populations dans le monde [2].

1.2.1 Définition

La santé numérique, également appelé *e-Healthcare* en anglais, est définie par l'OMS comme « les services du numérique au service du bien-être de la personne » [3].

La santé numérique fait donc intervenir dans les domaines de la santé les technologies de l'information et de la communication (TIC) et se base sur les techniques informatiques appliquées au domaine médical ou l'Informatique Médicale.

1.2.2 Domaines d'application de la santé numérique

Le champ d'application de la santé numérique s'étend sur trois domaines :

Systèmes d'information

Les systèmes d'information dans la santé permettent, par exemple, de collecter des données sur des patients traités et de tenir à jour leurs dossiers médicaux numérisés ou de faire une déclaration obligatoire en ligne d'une personne atteinte d'une maladie contagieuse via une application comme (e-DO) [4].

Télémédecine

Le deuxième domaine d'application de la santé numérique est la télémédecine qui comprend la consultation à distance ou téléconsultation, la télésurveillance, la téléassistance, la télé-expertise, etc [4]. Dans la figure 1.1, un médecin pratique son métier à distance par le biais des TIC.

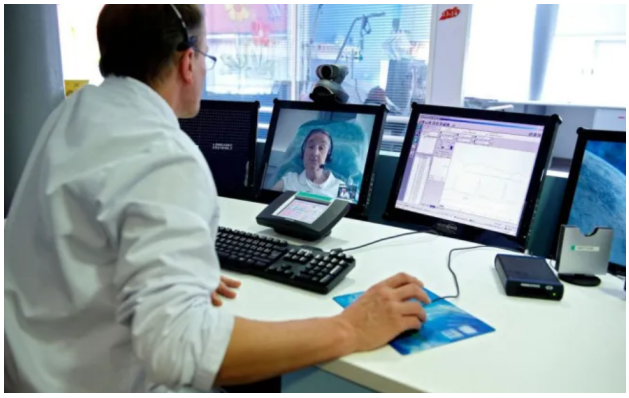

FIG. 1.1 : Un médecin pratiquant son métier à distance [5].

M-santé

Le troisième domaine est relatif à la m-santé ou mobile-santé (M-Health) qui comprend l'ensemble des objets connectés et des applications de santé pour le bien-être et la médecine. Il concerne l'ensemble des actes médicaux et de santé publique reposant sur des appareils mobiles tels que les smartphones, les dispositifs de surveillance des patients, les PDA (personal digital assistant) et d'autres appareils sans fil [6].

Le domaine de la mobile-santé est investi par le grand public conséquemment à la généralisation de l'usage de la téléphonerie mobile. La m-santé couvre donc une très grande variété de sujets de santé publique du plus global, comme le bien-être, au plus ciblé, comme la prise en charge des arrêts cardiaques via l'application SAUV Life [7].

Parmi les objets connectés, nous pouvons citer le thermomètre, le tensiomètre, le glucomètre (figure 1.2). La M-santé joue un grand rôle de prévention des maladies concernant les comportements susceptibles d'avoir des mauvaises conséquences sur la santé tels que la non observation des règles d'hygiène, le tabagisme, etc.

FIG. 1.2 : Glucomètre sans fils [8].

1.3 Intelligence Artificielle (IA)

1.3.1 Définition

Il s'agit d'un ensemble de systèmes informatiques développés en vue de simuler l'intelligence humaine dans divers domaines, comme celui de l'santé numérique.

1.3.2 Principaux avantages de l'IA

Les avantages de l'IA sont la résolution des problèmes complexes, l'aide à la prise de décision et la prédiction avec une grande précision. Son utilisation dans les domaines de santé tels que l'imagerie médicale ou la gestion des médicaments aide les professionnels à améliorer leur efficacité, leur productivité et leur constance dans la qualité des soins apportés aux patients.

Parmi les facteurs ayant aidé à l'introduction et à l'utilisation de l'IA dans les domaines de l'santé numérique, nous pouvons citer :

- **La disponibilité des données médicales :** L'intelligence artificielle est basée sur des technologies telles que l'apprentissage automatique (machine learning) et l'apprentissage profond (deep learning) qui nécessitent des données massives (big data). Ces données médicales sont disponibles sous forme d'antécédents médicaux et d'images médicales.
- **Le développement d'algorithmes complexes :** Avec l'introduction du Deep learning capable de traiter de grands ensembles de données variées, il est devenu possible d'analyser des données médicales de gros volume [9].

1.3.3 Exemples de l'IA dans les domaines de la santé numérique

Chirurgie assistée par robot

Le système « da Vinci Si HD » est une plate-forme chirurgicale robotisée et sophistiquée qui permet aux chirurgiens de réaliser des opérations chirurgicales complexes et délicates, de manière moins invasive [10]. La figure 1.3 montre le système « da Vinci Si HD ».

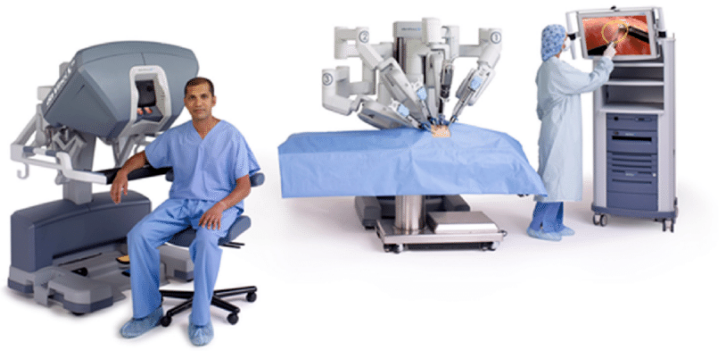

FIG. 1.3 : Système "da Vinci Si HD" [11].

Chatbots ou robots conversationnels

Une infirmière virtuelle qui est un chatbot (un robot conversationnel) peut interagir avec les patients plus régulièrement qu'une infirmière humaine. Puisque le chatbot est disponible en permanence. Il est aussi capable de répondre instantanément aux questions et de garder en mémoire les informations à fournir, sur demande, aux médecins. Ce sont des taches difficiles à exécuter par les infirmières humaines.

Diagnostic médical

Reposant sur le deep learning et le machine learning, l'IA est capable d'interpréter l'imagerie médical et de faire un diagnostic avec autant, voir plus de précision qu'un humain. Cette tendance propose que l'intelligence artificielle se concentre sur l'établissement des diagnostics médicaux pendant que le médecin s'occupe des interactions avec ces patients [12]. La figure 1.4 montre Le logiciel « InferRead CT Pneumonia.AI » qui utilise le diagnostic assisté par l'intelligence artificielle pour améliorer l'efficacité globale du service de radiologie.

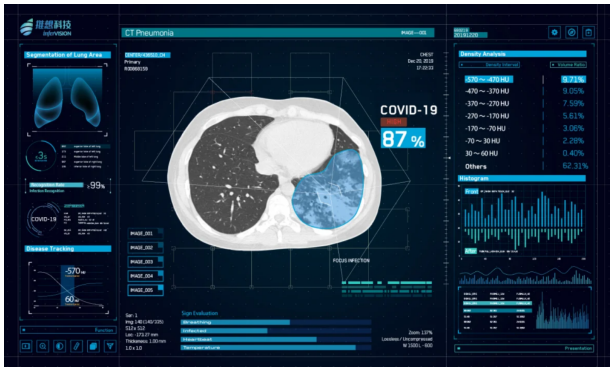

FIG. 1.4 : Le logiciel InferRead CT Pneumonia.AI [13].

1.4 Apprentissage Automatique (Machine Learning)

1.4.1 Définition

Plusieurs définitions formelles ont été données au domaine de l'apprentissage automatique.

Selon Tom Mitchell : *"Un programme qui apprend est un programme qui tire profit d'une expérience E, par rapport à une famille de tâches T, pour une mesure d'efficacité P, si son efficacité dans l'accomplissement des tâches de T augmente après l'expérience E"* [14].

Ethem Alpaydin a défini l'apprentissage automatique ainsi : *"la programmation d'ordinateurs pour optimiser leur performance en utilisant des données d'exemple ou des expériences déjà effectuées"* [15].

En résumé :

L'apprentissage automatique, également appelé apprentissage artificiel ou *machine learning* en anglais, est un sous-domaine de l'intelligence artificielle qui permet aux systèmes d'apprendre et de s'améliorer à partir de leur expérience sans être explicitement programmés. Ces systèmes, quand ils sont exposés à de nouvelles données, apprennent, changent et se développent de façon autonome [16].

1.4.2 Types d'apprentissage Automatique

Selon le type du problème abordé, l'apprentissage automatique est divisé en trois sous-domaines qui varient en fonction du type et du volume des données.

Apprentissage supervisé

L'apprentissage supervisé est le type d'apprentissage automatique le plus couramment utilisé. Dans ce type d'apprentissage, on alimente l'algorithme en entrée avec des données déjà étiquetées (*features*) et on lui indique les résultats attendus (*labels*) [17]. La tâche de l'algorithme est de prédire le résultat ou l'étiquetage correct. L'algorithme, lors de sa phase d'entraînement, génère une fonction qui lui permet de délivrer le résultat désiré.

Dans l'exemple montré dans la figure 2.22, Nous avons alimenté l'algorithme avec des images étiquetées de chats et de chiens, l'algorithme doit donc apprendre à reconnaitre le type d'animal dans les nouvelles photos qu'il rencontre.

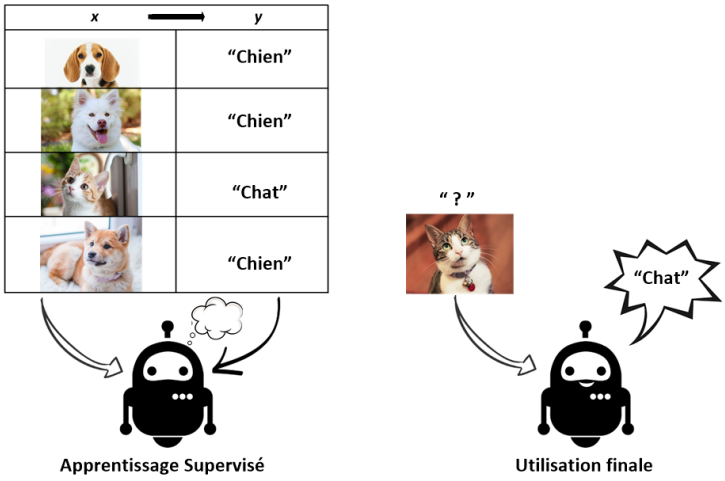

FIG. 1.5 : Exemple d'apprentissage supervisé [18].

Types d'apprentissage supervisé

L'apprentissage supervisé lui-même est divisé en deux types :

Classification

Dans un problème de classification, la variable de sortie prédite par l'algorithme est une classe précise, telle que l'exemple des chats et des chiens. Lorsqu'il n'y a que deux

classes présentes, nous parlons de classification binaire. Pour plus de deux valeurs de classe, il s'agit d'une classification multi-classes [19]. Ce type d'algorithmes est utilisé par exemple pour le filtrage des e-mails "spam" ou "pas de spam".

Exemples : les machines à vecteurs de support, les arbres de décision, les forêts aléatoires.

Régression

Dans un problème de régression, la valeur de sortie prédite par l'algorithme est une valeur réelle ou continue, telle que la prédiction d'une valeur comprise entre –infini et l'infini par exemple 'salaire' ou 'poids' [19]. Les algorithmes de régression sont utilisés pour prédire des valeurs sur la base de différents points de données.

Exemples : la régression linéaire, la régression logistique et la régression polynomiale.

Apprentissage non supervisé

Dans ce type d'apprentissage, on alimente l'algorithme avec des données non-étiquetées sans lui indiquer les résultats attendus. L'algorithme doit apprendre par lui-même à partir des données sans intervention humaine. Sa tâche consiste à découvrir des modèles et des structures dans les données afin de les regrouper en fonction de leur similitude [17].

Ce type d'apprentissage est utilisé par exemple dans des applications de réseaux sociaux telles que Facebook, Instagram, Twitter, pour exploiter une quantité massive de donnée non-étiquetées.

Exemple : Méthode des k plus proches voisins, la mise en cluster de k-moyennes, et les règles d'association.

Dans l'exemple montré dans la figure 1.6, nous avons alimenté l'algorithme par des images non-étiquetées des fruits, l'algorithme va apprendre à distinguer les trois fruits.

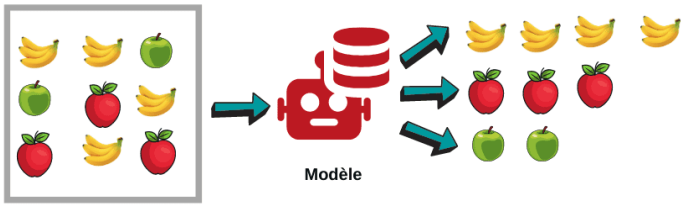

Fig. 1.6 : Exemple d'apprentissage non-supervisé [9].

Apprentissage par renforcement

Dans l'apprentissage par renforcement, l'algorithme apprend par un système de récompenses et de punitions. L'agent (algorithme) fonctionne dans un environnement et reçoit un retour de l'environnement. Ce retour est soit une récompense pour un bon choix ou une punition pour un mauvais choix. Autrement dit, l'agent apprend de ses expériences passées pour maximiser le cumul de récompenses et donc atteindre son objectif. Ce processus est aussi appelé ***processus d'essai et d'erreur*** [20]. L'apprentissage par renforcement est généralement utilisé dans la robotique, les véhicules autonomes et la théorie des jeux. Dans la figure 1.7, l'agent effectue une action sur l'environnement, et en conséquent, il reçoit une récompense et une représentation du nouvel état.

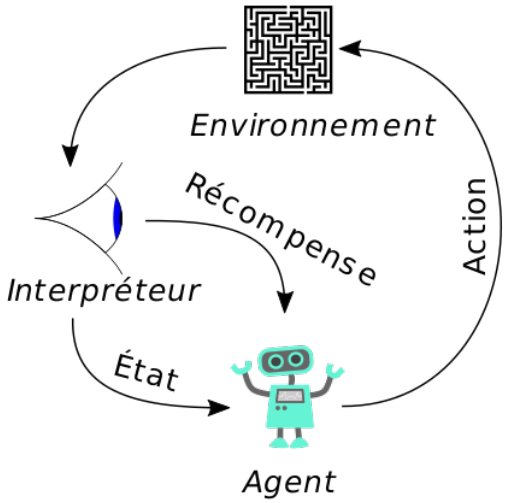

FIG. 1.7 : Le scenario d'apprentissage par renforcement [21].

1.5 Réseaux de Neurones Artificiels

1.5.1 Définition

Un Réseau de Neurones Artificiels, également appelé *Artificial Neural Network* en anglais, est formé d'un grand nombre de processeurs simples (neurones) qui fonctionnent en parallèle et fortement interconnectés comme les cellules nerveuses du cerveau humain [22].

Dans ce qui suit, nous allons donner les définitions des neurones biologiques (cellules nerveuses) et des neurones artificiels (formels), et aborder, par la suite, la constitution,

les couches d'un réseau neuronal et la fonction d'activation.

1.5.2 Neurone Biologique

L'unité de base du système nerveux est la cellule nerveuse ou neurone. Le neurone est constitué d'un grand corps cellulaire et de fibres nerveuses, l'axone et les dendrites. Le neurone biologique reçoit en entrées des signaux électriques (influx nerveux) transmis par d'autres neurones, grâce à une interaction qui se fait entre les dendrites et les synapses. Le neurone reçoit les signaux en entrée, les analyse et les traite en effectuant leur sommation, si le résultat est supérieur à un seuil d'activation, il envoie une décharge le long de son axone vers d'autres neurones. La synapse est l'espace d'interaction entre deux cellules nerveuses qui permet le passage du signal [23]. (figure 1.8)

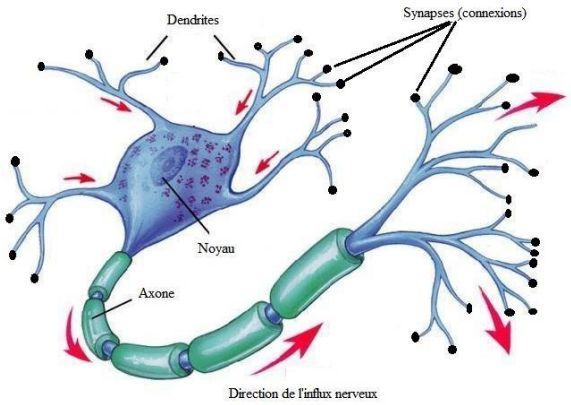

FIG. 1.8 : Neurone biologique [24].

1.5.3 Neurone Artificiel

Dans le neurone artificiel, les poids synaptiques, fonction d'activation et élément de sortie ont des fonctions similaires aux fonctions occupées par les synapses, corps cellulaire et axone du neurone biologique. Il y une similitude entre le neurone artificiel et le neurone biologique (figure 1.9).

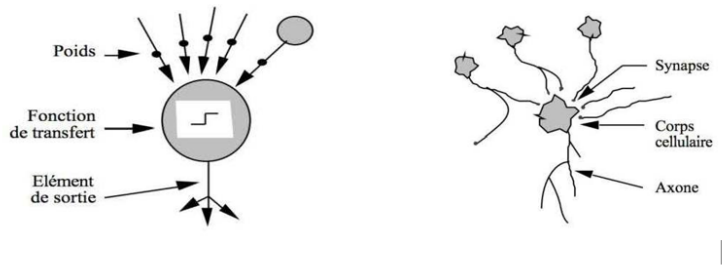

FIG. 1.9 : Similitude entre neurone biologique et neurone artificiel [25].

Le neurone artificiel prend en entrée un certain nombre de variables $x = \{x_1, x_2, x_3, ...x_n\}$ et chaque entrée est associée à un poids **w** représentant la valeur de la connexion. Une fonction d'activation f transforme la somme pondérée des variables d'entrée et de leurs poids.

$$S = \sum_{i=1}^{n} (w_i \times x_i)$$

à une valeur qui sera ensuite transmise à la couche de sortie pour être comparée à une valeur du seuil, puis fournir une réponse de sortie. (figure 1.10)

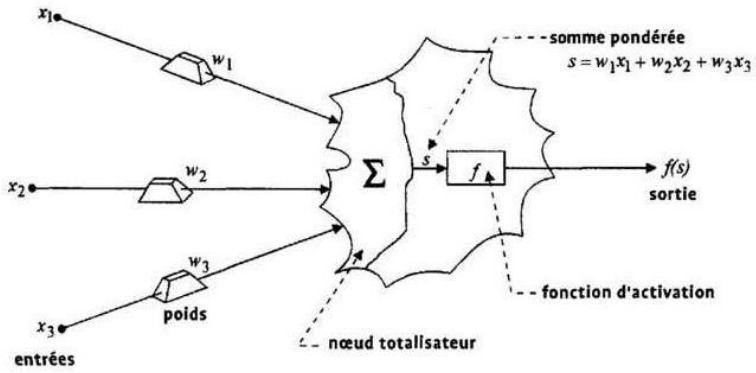

FIG. 1.10 : Principe de fonctionnement d'un neurone artificiel [26].

1.5.4 Constitution d'un réseau neuronal

Les nœuds de traitement qui forment le réseau neuronal ressemblent aux neurones du cerveau humain. Un nœud est connecté à différents nœuds en plusieurs couches qui se trouvent au-dessus et au-dessous. La fonction de ces nœuds est de déplacer les données à travers le réseau dans une seule direction [27].

1.5.5 Couches d'un réseau de neurones

Un réseau de neurones artificiel (perceptron multicouches ou *Multi Layers Perceptron* en anglais) est organisé en trois types de couches (figure 1.11) :

Couche d'entrée (Input layer)

Cette couche reçoit les données, aucun calcul n'est effectué .Les nœuds ici transmettent simplement les informations à la couche cachée.

Couches cachées (Hidden layers)

Les nœuds de cette couche ne sont pas exposés au monde extérieur. La couche cachée effectue toutes sortes de calculs sur les entités saisies via la couche d'entrée et transfère le résultat à la couche de sortie.

Couche de sortie (Output layer)

Cette couche fait remonter les informations apprises par le réseau vers le monde extérieur [28].

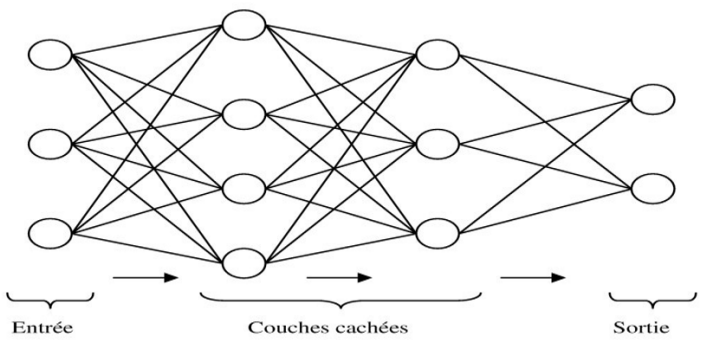

FIG. 1.11 : Architecture d'un réseau de neurones [17].

1.5.6 Fonction d'activation

En calculant la somme pondérée des entrées (x_i) et de leur poids (w_i) et en ajoutant un biais (b_i) à la somme pondérée,

$$\sum_{i=0}^{n}(w_i \times x_i) + b_i$$

la fonction d' activation détermine si un neurone doit être activé ou non. Elle introduit une non-linéarité dans la sortie d'un neurone. Le choix d'une fonction d'activation, parmi les fonctions qui existent, dépend du modèle adopté.

Les fonctions d'activation principalement utilisées sont montrées dans la figure 1.12.

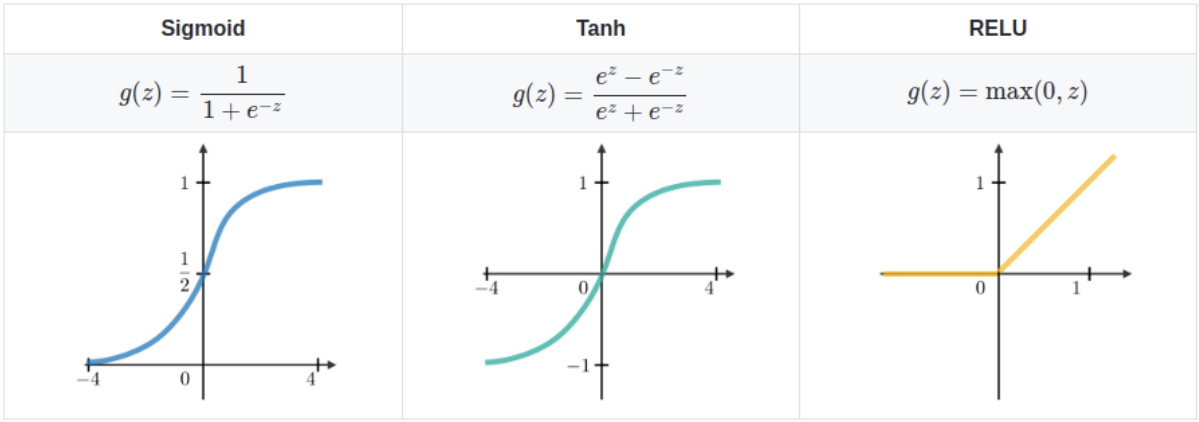

FIG. 1.12 : Fonctions d'activation : Sigmoid, Tanh, RELU [29].

1.6 Apprentissage profond (Deep Learning)

1.6.1 Définition

L'apprentissage profond, également appelé *deep learning* en anglais, est un sous-domaine de l'apprentissage automatique qui consiste en un ensemble d'algorithmes capables d'imiter le fonctionnement du cerveau humain grâce à des réseaux de neurones artificiels. Ces réseaux sont composés de dizaines voire des centaines de couches de neurones. Plus le nombre de couches est élevé, plus le réseau est qualifié « profond ». L'apprentissage profond est la technique la plus puissante dans le machine learning, notamment dans la classification des images [30]. Elle est utilisé dans :

- Le traitement du langage
- La reconnaissance d'image
- Les voitures autonomes
- Le diagnostic médical
- Les chatbots
- Les robots...

Figure 1.13 montre la corrélation entre l'intelligence artificielle, le machine learning et le deep learning.

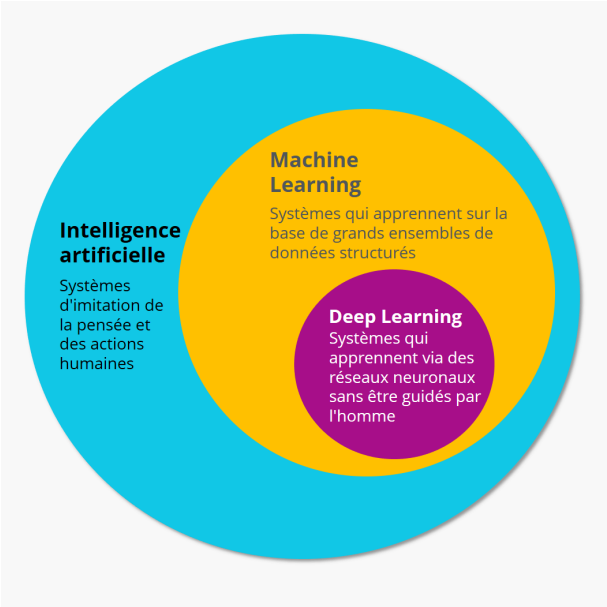

FIG. 1.13 : Correlation entre l'IA , ML et DL [31].

1.6.2 Réseau de neurones convolutif (CNN ou ConvNet)

En 1988, le premier réseau de neurones convolutif appelé LeNet a été développé par le chercheur français Yann LeCun [32].

En 2012, AlexNet, un autre réseau neuronal inspiré par LeNet a été développé par des chercheurs de Toronto [32].

Le CNN (*Convolutional Neural Network*), est l'un des algorithmes les plus performants de deep learning utilisé pour la détection d'objets ou le traitement d'images. Il est principalement utilisé pour identifier des images de satellite ou médicale ou pour détecter des

anomalies.

Couches de CNN

1- Couche de convolution

La couche de convolution est la couche de base d'un CNN. C'est là où la majorité des calculs sont effectués. Son but est de vérifier la présence d'un ensemble de caractéristiques (*features*) dans les images reçues en entrée et utilise pour cela des filtres. Un filtre, également appelé noyau, est une matrice 2D qui représente une caractéristique. L'opération de convolution consiste à faire glisser un filtre à travers toute l'image et de calculer le produit de convolution entre le filtre et chaque proportion de l'image. Une image est perçue par l'ordinateur comme une matrice de pixels avec des valeurs différentes. Une image en couleur est composée d'une matrice de pixels en 3D. Cela signifie que l'entrée aura trois dimensions (hauteur, largeur et profondeur) qui correspondent au RVB dans une image [33].

La figure 1.14 montre comment un filtre de taille 3x3 est appliqué sur une image de taille 7x7. Les 3x3 premiers pixels de l'image sont sélectionnée et chaque valeur de ces pixels est multipliée par la valeur de pixel correspondant du filtre pour créer un nouveau pixel dans l'image de sortie (carte de convolution). La valeur de ce dernier est égale à la somme des résultats de multiplications. Le filtre se déplace ensuite vers la droite avec un pas, également appelé « *stride* », pour que la convolution soit appliquée sur la deuxième proportion et ainsi de suite. Lorsque la fin de la première ligne est atteinte, le filtre descend d'un pas et à chaque proportion rencontrée de l'image, le même calcul s'effectue jusqu'à arriver au bout de l'image.

En résumé :

La couche de convolution calcule la convolution de chaque image reçue en entrée avec chaque filtre, cela signifie qu'on obtient une carte de convolution pour chaque paire (image, filtre). Dans cet exemple, les valeurs des pixels sont binaires mais en pratique elles varient entre 0 et 255.

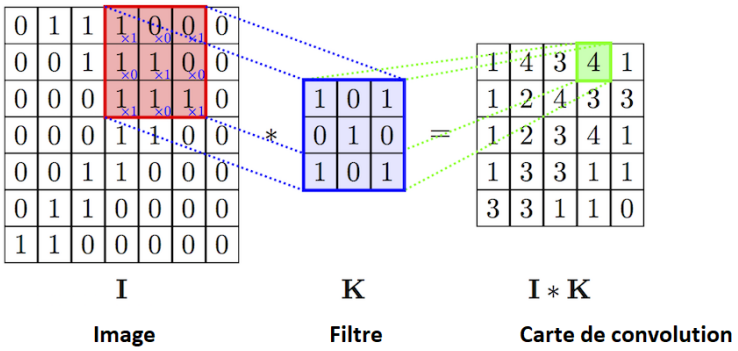

FIG. 1.14 : Opération de convolution [34].

2- Couche de pooling

La couche pooling consiste à remplacer chaque groupe de pixels par une nouvelle valeur. Comme l'opération de convolution, cette opération s'agit de faire glisser au-dessus de l'image une fenêtre (généralement de taille 2x2 ou 3x3) et de prendre la valeur maximale des valeurs présentes dans la fenêtre s'il s'agit d'un « max pooling » ou de calculer leur moyenne s'il s'agit d'un « Average pooling ». Cette valeur sera la valeur d'un nouveau pixel dans l'image de sortie. Cette opération réduit la taille de l'image et ne préserve que les informations importantes ce qui réduit la quantité de paramètres et de calculs dans le réseau [33] .

La figure 1.15 montre La différence entre max pooling et average pooling utilisant un filtre de taille 2x2.

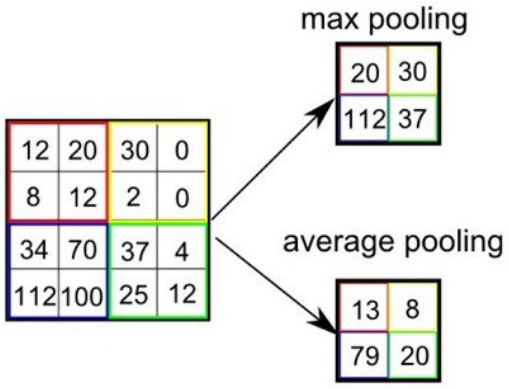

FIG. 1.15 : Différence entre max pooling et average pooling [35].

3- Couche fully-connected

La couche fully-connected (FC) constitue toujours la dernière couche d'un CNN (figure 1.16). Elle renvoie le nombre de classes, dans notre problème de classification d'images,

et indique la probabilité pour l'image en entrée d'appartenir à une classe. Cette dernière couche entièrement connectée fait donc une classification en combinant toutes les caractéristiques spécifiques détectées par les couches précédentes dans les données d'entrée [33].

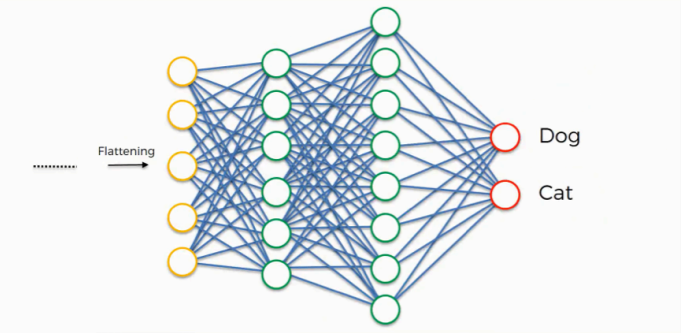

FIG. 1.16 : Couche fully-connected [36].

Fonctionnement des CNN

En résumé, dans les CNN nous distinguons deux parties :

- **La partie convolutive du modèle**

Cette partie est la particularité de ce type de réseau. C'est la partie responsable de l'extraction des caractéristiques des images. Chaque image passe à travers une série de filtres, appelés aussi « noyaux de convolution », créant de nouvelles images appelées cartes de convolutions ou *feature maps* en anglais. Certain de ces filtres réduisent la taille initiale de l'image. Au final, les dernières cartes de convolutions obtenues sont concaténées dans un vecteur de caractérisques qui sera ensuite fournie à l'entrée de la partie classification [32].

- **La partie classification du modèle**

Cette partie n'est pas caractéristique de CNN, elle est présente à la fin de tous les réseaux de neurones pour la classification. Elle est composée de couches entièrement connectées (perceptron multicouche). Elle prend en entrée un vecteur, combine ses caractérisques et renvoie un nouveau vecteur en sortie. Ce vecteur contient un élément par classe. Chaque élément est compris entre 0 et 1 et représente la probabilité d'une des classes. La dernière couche est celle qui calcule la probabilité de chaque catégorie et utilise comme fonction d'activation, une fonction logistique quand il s'agit d'une classi-

fication binaire ou une fonction softmax quand il s'agit d'une classification multi-classe [32]. (figure 1.17)

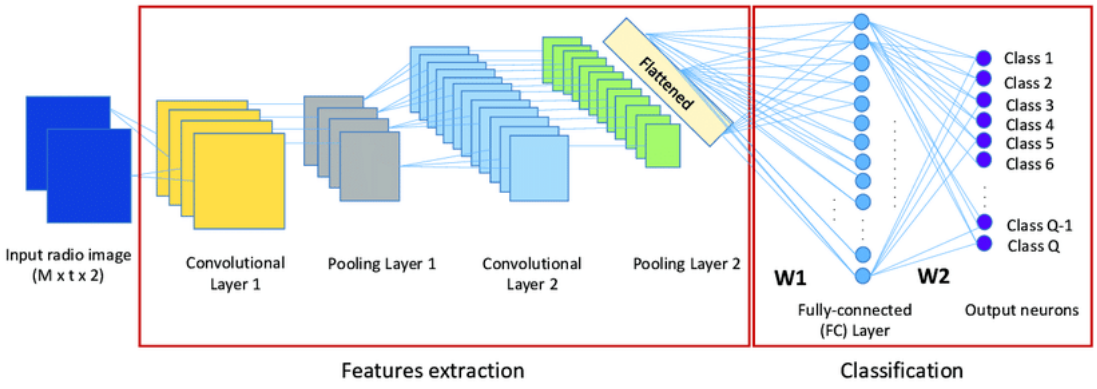

FIG. 1.17 : Exemple d'une architecture CNN [37].

1.6.3 Autres Algorithmes de deep learning

Réseaux de neurones récurrents (RNN)

Les réseaux de neurones récurrents (*Recurrent Neural Networks*) sont une catégorie de réseaux de neurones dédiée au traitement des séquences, c'est-à-dire aux signaux de taille variable. Ils sont utilisés pour la reconnaissance de la parole, le traitement du langage naturel, ...etc.

Réseaux de mémoire à long terme et à court terme (LSTM)

Long Short Term Memory sont des unités d'un réseau neuronal récurrent. Un RNN composé de ces unités est souvent appelé un réseau LSTM. Le but de ces unités cellules LSTM qui possèdent une mémoire interne est de donner aux réseaux de neurones récurrents la capacité de maintenir un état sur une longue période de temps.

Réseaux de neurones à propagation avant (feedforward)

Comme le perceptron multicouche (MLP). Ces réseaux sont adaptés aux données de tailles fixes, comme les images, et l'influx d'information va toujours des couches d'entrées aux couches de sorties. Les réseaux convolutionnels (CNN) rentrent dans cette catégorie.

1.6.4 Différence entre l'apprentissage automatique et l'apprentissage profond

Dans l'apprentissage automatique, la tâche d'extraction des charactéristiques se fait d'une manière manuelle. Autrement dit, cette technique nécessite des données structurés et organisées pour savoir comment classer les nouvelles données. Par contre, dans l'apprentissage profond, l'algorithme n'a pas besoins des données structurées. Il est capable d'identifier par lui-même les charactéristique sans l'intervention du développeur.(figure 1.18)

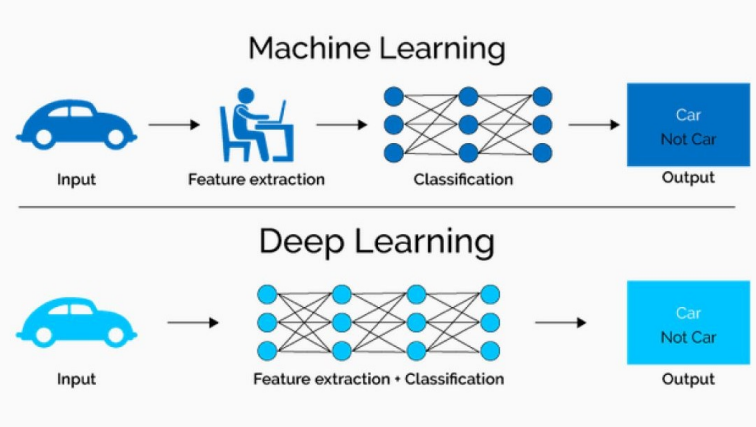

FIG. 1.18 : Différence entre le ML et le DL [38].

1.6.5 Transfer Learning (apprentissage par transfert)

Le CNN est très efficace pour la classification d'images mais nécessite une longue période d'entraînement. il fonctionne parce que les différents poids du réseau sont établis à partir de différentes entrées. Une fois que les données ont traversé les couches cachées, les poids sont calculés et évalués. Pour minimiser la fonction coût, le réseau passe par plusieurs phases de rétropropagation pendant lesquelles les poids sont ajustés plusieurs fois jusqu'à arriver à une configuration optimale des poids.

L'entrainement d'un réseau de neurones convolutif demande toujours plus de couches et donc un nombre élevé de convolutions et de paramètres à optimiser et des processeurs graphiques (GPU) très performants pour entraîner rapidement le réseau. Nous devons donc exploiter la possibilité de faire le transfert des connaissances acquises par un CNN déjà entrainé pour résoudre un problème plus ou moins similaire.

L'apprentissage par transfert, également appelé *Transfer Learning* en anglais, est une méthode permettant de réutiliser un modèle pré-entraîné pour réaliser une nouvelle tâche. En d'autres termes, les connaissances d'un modèle pré-entraîné sur un grand volume de données sont transférées à un nouveau modèle qui doit être entraîné sur des données comparativement moins nombreuses que nécessaire. Cela supprime la nécessité d'avoir un grand volume de données et réduit la longue période d'entraînement requise par le modèle lorsqu'il est développé "from scratch" (de A à Z) [39].

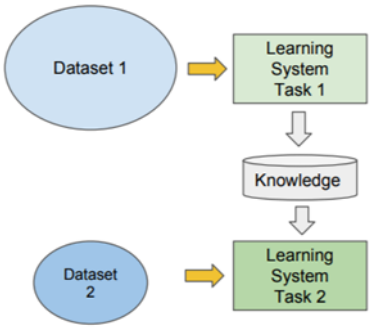

FIG. 1.19 : Transfer Learning [40].

1.6.6 Ensemble Learning

Ensemble Learning est une technique utilisée en apprentissage automatique qui vise à obtenir de meilleures performances prédictives en combinant les prédictions de plusieurs modèles. Il s'agit du processus de combinaison de plusieurs modèles d'apprentissage différents pour obtenir leur performance collective, ce qui permet d'obtenir un modèle robuste et fiable pour faire des prédictions. Le modèle final résultant aura une meilleure performance de généralisation et fera donc de meilleures prédictions par rapport aux prédictions faites par chaque modèle seul. Ce concept est analogue aux anecdotes telles que le fait de demander l'avis de plusieurs médecins. Les modèles sont empilés ensemble pour améliorer leurs performances et obtenir une prédiction finale, comme le montre la figure.

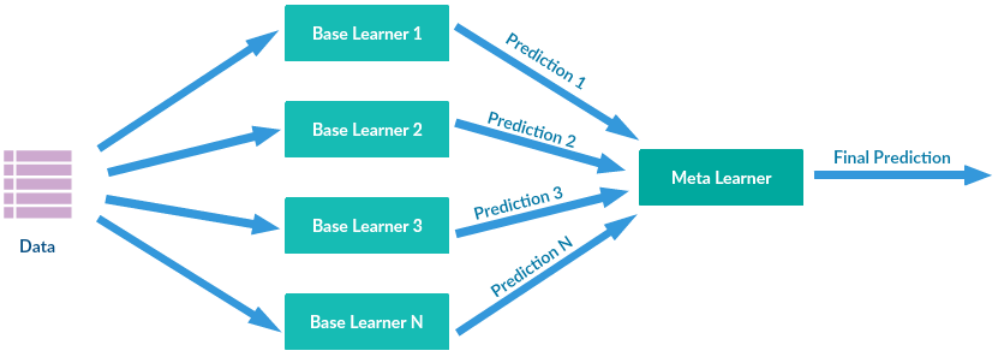

FIG. 1.20 : Schéma illustratif de l'ensemble learning [41].

1.6.7 Quelques architectures des CNNs

Il existe différents types d'architectures de CNN tels que AlexNet, LeNet , VGG... etc.

AlexNet

AlexNet a été conçu par le groupe SuperVision, composé d'Alex Krizhevsky, Geoffrey Hinton et Ilya Sutskever en 2012. Il est composé de :

- 05 couches de convolution ; les 1ère, 2ème et 5ème ayant des couches Max-Pooling.

- 02 couches entièrement connectées (FC) et une couche softmax à la fin pour les prédictions.

L'architecture AlexNet peut être vue sur la figure 1.21 [42].

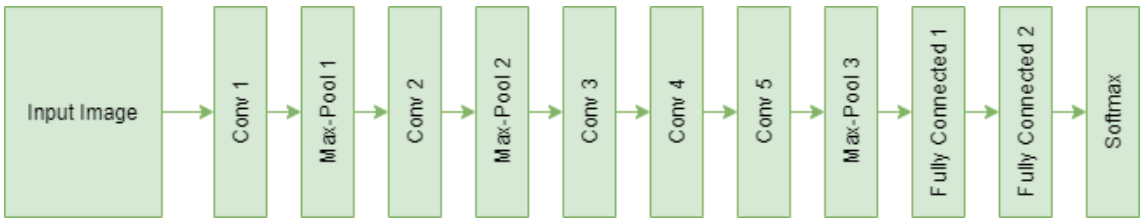

FIG. 1.21 : Architecture AlexNet [42].

LeNet

La configuration de LeNet est la suivante : Deux (2) couches convolutives, Deux (2) couches pooling, Deux (2) couches entièrement connectées (FC) et une couche de sortie.

LeNet a été développé par Yan Le Cun en 1998 pour la reconnaissance des chiffres [43]. (figure 1.22)

FIG. 1.22 : Architecture LeNet [44].

VGG

VGG est un réseau de neurones convolutif proposés par K. Simonyan et A. Zisserman du *Visual Graphics Group* de l'Université d'Oxford (d'où le nom VGG). Il a gagné la compétition ILSVRC (ImageNet Large Scale Visual Recognition Challenge) en 2014. Ce modèle utilise des noyaux de convolution de plus petites dimensions (3×3) ce qui permet d'atteindre une bonne précision.

Il existe deux algorithmes VGG16 et VGG19 dont les architectures sont très proches mais qui n'ont pas le même nombre de couches de convolution : VGG-19 en a 16 alors que VGA-16 en possède 13.

- Architecture du VGG-16

Le VGG-16 est composé de 13 couches de convolution (avec la fonction d'activation ReLu) accompagnées des couches Max Pooling et ,en sortie, de 3 couches de neurones Fully-Connected (FC) dont la dernière avec une fonction d'activation softmax [45]. (figure 1.23)

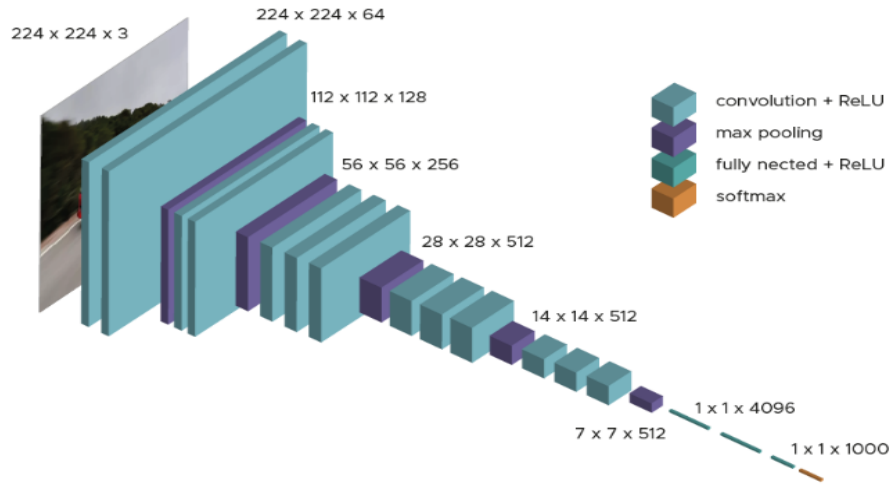

FIG. 1.23 : Architecture du VGG-16 [45].

- Architecture du VGG-19

Le modèle VGG-19 est un CNN à 19 couches comprenant 16 couches de convolution de 3x3 accompagnées de couches de pooling et 03 couches entièrement connectées (FC) comme le montre la figure ci-après [45]. (figure 1.24)

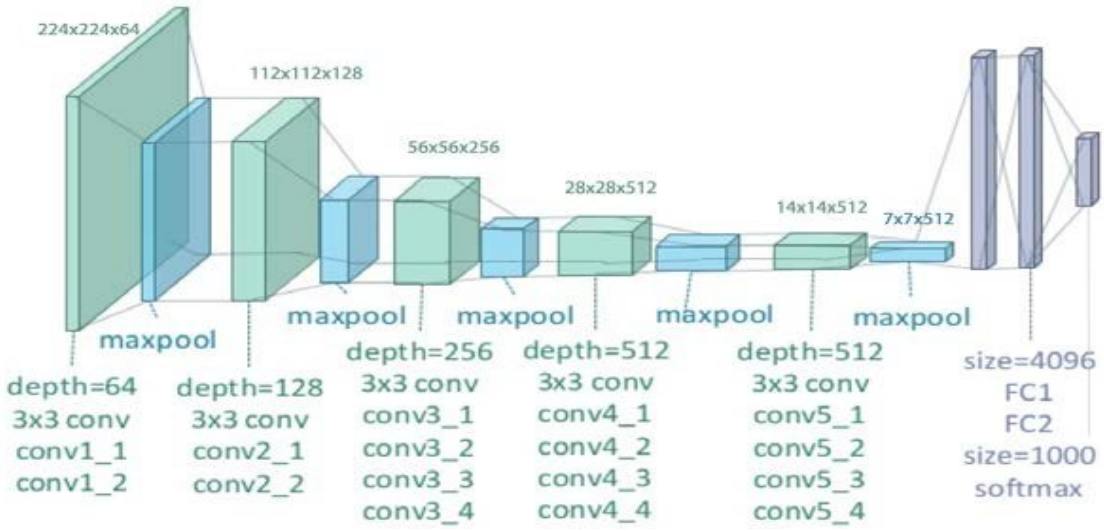

FIG. 1.24 : Architecture du VGG-19 [46].

DenseNet-169

L'architecture du DenseNet-169 est constituée de 04 Dense Blocks séparés par des couches de transition où sont effectuées des opérations de pooling et de convolution. Chaque Dense Block est constitué de 05 couches qui reçoivent chacune les caractéris-

tiques ou feature-maps de toutes les couches précédentes. Le réseau commence par une convolution et se termine par une couche de global pooling et une chouche softmax. (voir figure 1.25)

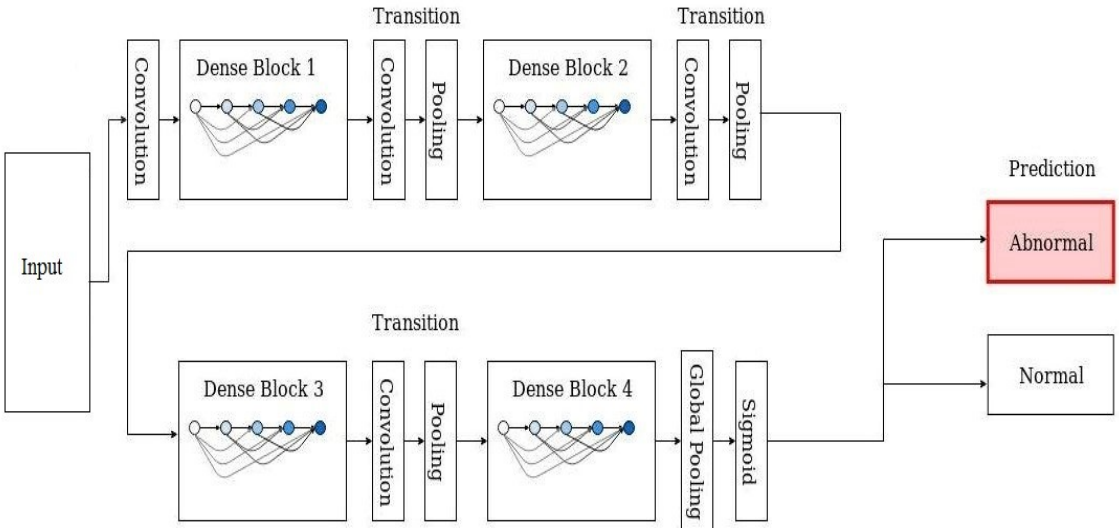

FIG. 1.25 : Architecture du DensNet-169 [47].

MobileNet-V2

MobileNet-V2 utilise des convolutions séparables en profondeur (depthwise separable convolutions), et son principal bloc de construction est schématisé comme suit :

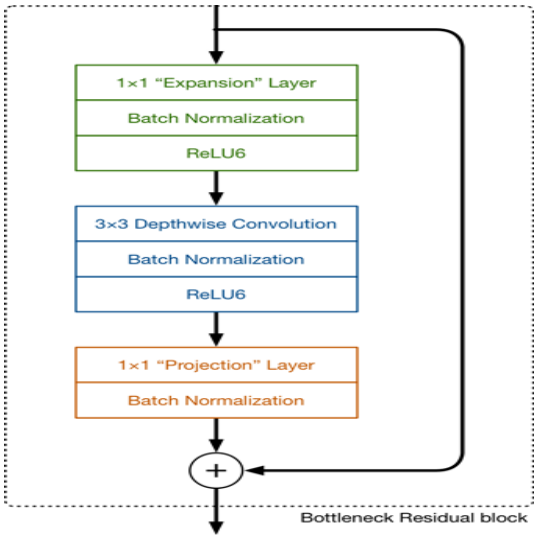

FIG. 1.26 : Bloc MobileNet-V2 [48].

Dans ce bloc, il y a trois couches de convolution. La première couche est une convolution 1×1 appelée couche d'expansion qui augmente le nombre de canaux avant le passage

des données dans la couche de convolution en profondeur (depthwise convolution). Cette dernière filtre les entrées et elle est suivie d'une couche de convolution ponctuelle 1×1 appelée couche de projection (projection layer) qui réduit le nombre de canaux à la sortie de ce bloc. C'est une couche "goulot d'étranglement" ou bottleneck layer. L'architecture complète de MobileNet-V2 se compose de 17 blocs en ligne. Ils sont suivis d'une convolution 1×1, d'une couche de global average pooling et d'une couche de classification [48].

1.7 La maladie de la tuberculose

1.7.1 Définition

La tuberculose (TB) est une maladie contagieuse qui touche le plus souvent les poumons. Si elle est devenue rare dans les pays riches, elle reste très fréquente dans de très nombreux pays en voie de développement.

Cette maladie est l'une des 10 premières causes de mortalité dans le monde et entraîne 1,4 million de décès chaque année. Actuellement, nous estimons qu'un quart de la population mondiale est atteint de tuberculose [1].

En 2021, notre pays a enregistré, 18.825 cas de tuberculose, dont 5423 cas de tuberculose pulmonaire, selon le Docteur S.A. Halassa, Responsable du programme national de lutte contre la tuberculose [49].

1.7.2 Cause de la tuberculose

La tuberculose humaine est une maladie infectieuse due à la mycobactérie *Mycobacterium tuberculosis* appelée aussi Bacille de Koch (BK). Le BK atteint le plus souvent les poumons et provoque la tuberculose pulmonaire, qui est très fréquente. Il peut atteindre également d'autres organes et provoquer une tuberculose extra-pulmonaire comme la tuberculose miliaire qui touche la moelle osseuse, la tuberculose génito-urinaire et la méningite tuberculeuse. La figure 1.27 montre une image de microscopie électronique de la tuberculose.

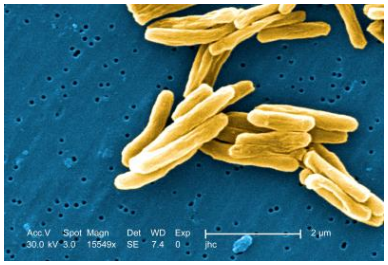

FIG. 1.27 : Bacille de Koch [50].

1.7.3 Evolution du BK dans l'organisme

Tuberculeuse latente

Le bacille tuberculeux s'installe dans l'organisme et la personne infectée développe une primo-infection, souvent sans symptômes, qui est contrôlée par le système immunitaire. Cette infection tuberculose devient latente (ITL) : le bacille reste présent dans le corps sans que la personne ne développe la maladie.

Tuberculose active

10 % environ des personnes infectées développent une tuberculose dite active qui va évoluer et provoquer des symptômes, le plus souvent au niveau des poumons (tuberculose pulmonaire, dans environ 75 % des cas).

Une personne atteinte de TB active pulmonaire est contagieuse et peut donc contaminer d'autres personnes.

1.7.4 Symptômes de la tuberculose active

Les symptômes de la tuberculose sont la fièvre, les douleurs des articulations, les sueurs nocturnes, l'amaigrissement et la fatigue chronique.

Dans les formes avancées de la tuberculose pulmonaire, la personne souffre de toux persistante, d'essoufflement et rejette des crachats sanglants [1].

1.7.5 Transmission de la maladie

Lorsque le malade tousse, crache ou éternue, il transmet l'agent infectieux par voie aérienne, via des gouttelettes contenant le BK. La contamination se fait par l'inhalation de ces gouttelettes. Les conditions sanitaires et sociales précaires favorisent la propagation de la TB et un malade non traité peut infecter jusqu'à 15 personnes en une année.

1.7.6 Diagnostic

Pour diagnostiquer la maladie, le médecin fait effectuer des analyses des crachats du patient à la recherche du bacille tuberculeux et fait également pratiquer une radiographie des poumons pour localiser les lésions. La culture du bacille est nécessaire pour tester sa sensibilité aux différents antibiotiques.

1.7.7 Traitement

Le traitement se fait par une association de plusieurs antibiotiques antituberculeux pendant 06 mois au minimum et il peut aller jusqu'à 02 ans dans le cas de tuberculose résistante.

1.8 Imagerie médicale

Pour le diagnostic de nombreuses maladies, comme la tuberculose, nous utilisons l'imagerie médicale en plus des examens cliniques et biologiques. Les différentes technologies d'imagerie médicale sont notamment la radiographie, le scanner, l'échographie et l'imagerie par résonance magnétique (IRM).

Radiographie à rayons X (X-ray radiography)

Elle repose sur l'utilisation des rayons X qui traversent le corps humain en étant plus ou moins absorbés par les tissus. La source émettrice de rayon X est placée devant le corps du patient et un détecteur est placé à l'arrière. Les images recueillies par ce biais sont imprimées sur un film photographique ou directement numérisées, grâce aux nouvelles

technologies.

FIG. 1.28 : Radiographie [51].

Tomodensitométrie (TDM) ou Scanner

Elle fonctionne selon le même principe que la radiologie, c'est-à-dire l'utilisation d'une source de rayons X et d'un détecteur de part et d'autre du corps du patient, mais il permet d'obtenir **des images en trois dimensions (3D)**.

FIG. 1.29 : Scanner [52].

Echographie ultrasonore

Elle utilise l'émission et la réflexion des ultrasons pour produire des images. L'échographe se compose d'un écran et d'une sonde émettrice et réceptrice des ondes appelée transducteur [53].

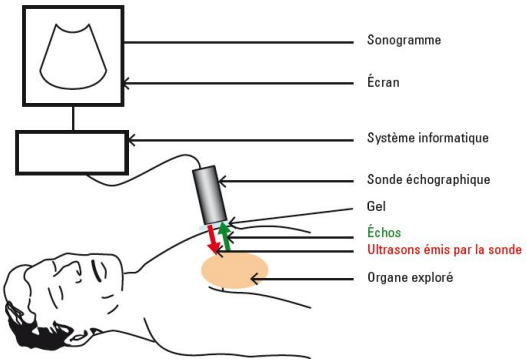

FIG. 1.30 : Echographie [53].

Imagerie par résonance magnétique (IRM)

Elle permet de visualiser la structure des tissus avec une grande précision. Cette technique d'imagerie repose sur les molécules d'eau qui composent 80% du corps du patient et qui ont des propriétés magnétiques. L'appareil IRM crée un champ magnétique qui fait vibrer les noyaux d'hydrogène qui émettent des signaux électriques. Ces signaux sont captés par une antenne placée sur la partie du corps étudiée et ensuite transformés en images [53].

1.9 Radiographie du thorax (Chest X-Ray)

La radiographie du thorax est utilisée pour diagnostiquer les affections de la cavité thoracique, notamment les voies respiratoires, les côtes, les poumons, le cœur et le diaphragme. Elle permet de déceler les anomalies pulmonaires comme la présence de cavités dans le cas de la tuberculose pulmonaire (TBP).

Les rayons X passent aisément à travers les tissus les moins denses qui apparaissent foncés sur la radio mais ils sont plus ou moins arrêtés par les tissus denses qui apparaissent clairs sur la radio. Les densités très différentes qui composent la structure du thorax permettent donc un bon examen radiographique puisqu'une structure anatomique est bien visible si elle est de densité différente de celle qui l'entoure. En fonction de la densité du tissu traversé par les rayons X, nous pouvons voir sur la radio :

- Gris très foncé à noir : l'air (arbre bronchique, alvéoles pulmonaires)

- Gris clair : la graisse (parois, graisse péricardique)

- Blanc à gris clair : l'eau (cœur, vaisseaux, médiastin, abdomen sous le diaphragme)

- Blanc : le calcium (os et calcifications)

Par exemple, les bronches du poumon ont une densité aérique et ne sont donc pas visibles dans le poumon qui a la même densité. En cas d'inflammation, les parois de ces bronches s'épaississent et elles deviennent visibles (figure 1.32).

Si la radio Rx confirme l'affection des poumons, nous procédons aux tests bactériologiques [54].

FIG. 1.31 : Procédure de la radiographie du Thorax [55].

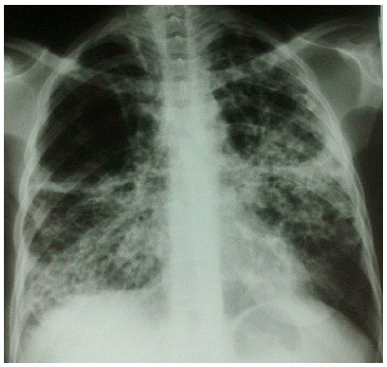

FIG. 1.32 : RX du thorax, TBP étendue et bilatérale [55].

1.10 Détection Assistée par Ordinateur de la tuberculose

La radiographie pulmonaire est largement utilisée comme outil de dépistage de la tuberculose dans les pays développés. Mais son utilisation reste limitée dans les pays

pauvres et très touchés par la TB, et dont les ressources sont insuffisantes pour faire face au coût élevé des films radiographiques et au manque de professionnels pour interpréter les images radiographiques. Selon l'Organisation mondiale de la santé (OMS), nous estimons qu'en 2019 seulement 7,1 millions de personnes ont été diagnostiquées sur un nombre de 10 millions de personnes ayant contracté la TBP dans le monde [56].

Le coût plus abordable de la technologie de la radiographie numérique et le développement récent de logiciels de Détection Assistée par Ordinateur (DAO) capables d'analyser les images RX du thorax, pourraient éliminer ces obstacles et permettre une plus grande utilisation de la RX du thorax et donc un meilleur diagnostic de la TBP.

C'est pour cette raison que l'Organisation mondiale de la santé (OMS) a recommandé en mars 2021 l'utilisation des logiciels de DAO pour l'interprétation des RX numériques à la place des lecteurs humains dans les deux cas de triage et de dépistage pour savoir si la personne doit subir ou non des tests bactériologique de confirmation.

1.11 Travaux Connexes

• **Classification automatique de la TB en utilisant un ensemble d'architectures profondes :**

Hooda et al. ont proposé un ensemble de trois architectures standard, à savoir AlexNet, GoogleNet et ResNet qui ont été entrainé "from scratch". Le dataset utilisé contient un ensemble de 1133 images. Il est formé de quatre datasets différents : Montgomery County (MC) dataset, Shenzhen dataset, Belarus dataset et JSRT dataset. Le dataset a été divisé en trois sous-ensembles : 70% (793 images) pour l'entrainement, 15% pour la validation et le reste 15% pour le test. Ils ont également utilisé la technique d'augmentation des données. L'ensemble atteint une précision (accuracy) de 88.24% et l'aire sous la courbe (AUC) est égale à 0.93 [57].

• **Détection des manifestations de la tuberculose pulmonaire sur les radiographies du thorax en utilisant différents modèles CNN :**

Meraj et al. ont testé quatre modèles pré-entraînés (VGG-16, VGG-19, RestNet50 et GoogLenet) sur deux datasets publics : Shenzhen Dataset et Montgomery Dataset. Pour l'entraînement, 75% des images ont été allouées et les 25% restantes ont été utilisées pour

la validation. Seule l'augmentation de l'image a été effectuée par des retournements, des rotations et des zooms. Aucune autre technique de prétraitement n'a été effectuée. Pour le premier dataset (Shenzhen), la plus haute précision (86.74%) a été atteinte par VGG-16 avec une AUC de 0.92. Pour le deuxième dataset (MC), VGG-16 et VGG-19 ont atteint une précision de 77.14% [58].

- **Comparaison de modèles de deep learning pour le dépistage de population utilisant des radiographies du thorax :**

Sivaramakrishnan et al. ont proposé une architecture CNN et ont utilisé cinq modèles pré-entrainés (AlexNet, VGG-16, VGG-19, Xception et ResNet-50) comme extracteurs de features pour la classification des images rayons X en images positives de tuberculose et en images noramles. Dans leur étude, ils ont utilisé deux datasets, Mountgomery County et Shenzhen gérés par la National Library of Medicine (NLM). L'architecture CNN proposée se compose de 12 couches, Chaque bloc CNN comporte une couche de convolution suivie d'une couche Batch normalization et d'une couche ReLU. 80% des images du dataset sont utilisées pour l'entrainement et les 20% pour les tests. Les images sont redimensionnées à 224×224 pixels. L'architecture CNN proposée a atteint une précision de 82% et une AUC de 0.89 pour le dataset de Shenzhen et une précision de 65.80% et une AUC de 0.74 pour le dataset de Montgomery.

Pour les modèles pré-entrainés, AlexNet a été le plus performant. Il a atteint précision de 84.20% et une AUC de 0.91 pour le dataset SH et une précision de 72.50% et une AUC de 0.80 pour le dataset MC [59].

- **Architecture d'apprentissage profond basé sur Inception pour le dépistage de la tuberculose en utilisant les radiographies pulmonaires :**

Das et al. ont proposé un réseau CNN basé sur le modèle Inception Net V3 qui a été pré-entrainé sur ImageNet. Ils ont entrainé ce réseau sur le dataset de Shenzhen en Chine ainsi que de Mountgomery County (MC) en USA. Les images de ces deux datasets ont été redimmensionnés à 224 x 224 pixels. Après le prétraitement numérique préliminaire, un algorithme de génération de masque a été exécuté sur les radiographies du thorax, et qui segmente les régions pulmonaires sur la radiographie entière. Ceci permet de délimiter la région d'intérêt (ROI), ce qui réduit le bruit des données et assure un meilleur entrainement du CNN. La segmentation est effectuée en utilisant un modèle d'apprentis-

sage profond basé sur U-Net qui est pré-entrainé sur des radiographies thoraciques pour produire des masques des poumons. Cette précision (accuracy) de segmentation est supérieure à 95%. Leur modèle a atteint une précision de 87.47% et une AUC de 0.92 pour le dataset de MC et une précision de 91.7% et une AUC de 0.96 pour le dataset de Shenzhen [60].

1.11.1 Tableau récapitulatif des travaux connexes

Les résultats obtenus par les travaux connexes sont résumé dans le tableau suivant :

Tab. 1.1 : Tableau récapitulatif des travaux connexes

Référence	Dataset	Méthode	Accuracy (%)	AUC	Classes
[57]	Combinaison de 4 datasets (dataset de Montgomery County (MC), dataset de Shenzhen, dataset de Belarus, dataset de JSRT).	Ensemble (AlexNet, GoogleNet, ResNet)	88.24	0.93	2
[58]	Dataset de Shenzhen	VGG-16 VGG-19 ResNet50 Googlenet	86.74 84.33 81.92 80.72	0.92 0.91 0.91 0.88	2
	Dataset de MC	VGG-16 VGG-19 ResNet50 Googlenet	77.14 77.14 71.42 71.42	0.75 0.90 0.76 0.75	
[59]	Dataset de Shenzhen	CNN AlexNet	82 84.20	0.89 0.91	2
	Dataset de MC	CNN AlexNet	65.80 72.50	0.74 0.80	
[60]	Dataset de Shenzhen	Inception-V3	91.7	0.96	2
	Dataset de MC	Inception-V3	87.47	0.92	

1.12 Conclusion

Dans ce chapitre, nous avons mis l'accent sur la santé numérique et ses domaines d'application qui sont basés sur les techniques de l'intelligence artificielle et nous avons abordé les points essentiels relatifs à l'objet de notre travail. Ainsi, nous avons défini le Machine Learning, le Deep Learning et les réseaux de neurones artificiels. Nous avons aussi abordé les réseaux de neurones convolutifs (CNN ou ConvNet) en définissant leur architecture ainsi que les fonctions des couches qui les composent. Les techniques du Transfer Learning et de l'Ensemble Learning ont été également traitées. Enfin des travaux connexes, ayant un rapport direct avec notre travail, ont été indiqués dans un tableau récapitulatif.

Chapitre 2

Conception et Implémentation

2.1 Introduction

Dans le chapitre précédent, nous avons évoqué le Deep Learning et les réseaux de neurones convolutifs (CNN) ainsi que leurs architectures. Dans ce chapitre, nous allons utiliser la technique de l'Ensemble Learning, que nous avons définie précédemment, pour construire un modèle basé sur la fusion de trois modèles CNN qui sont entrainés et capables de classifier les images, séparément.

Nous définirons, aussi, notre ensemble de données utilisées ou dataset et son prétraitement ainsi que les différents paramètres d'apprentissage. Nous traiterons également des outils et méthodes utilisés dans l'implémentation.

2.2 Modèle proposé

Afin de développer un modèle de Deep Learning basé sur les réseaux de neurones profonds convolutifs (CNN) capable de distinguer les images rayons-x de radiographie du thorax avec manifestation de la tuberculose pulmonaire des images normales, nous avons appliqué la technique du Transfer Learning pour construire trois modèles capables de classifier nos images. C'est-à-dire que nous avons fait un transfert de connaissances qui ont été déjà acquises par des CNNs pré-entrainés sur un dataset large vers nos nouveaux réseaux. Ensuite, nous avons fusionné ces trois modèles pour construire un ensemble de CNNs (modèle ensembliste ou Ensemble Model, en anglais) qui représente notre modèle

final.

Les modèles que nous avons utilisés dans le transfert des connaissances sont : VGG-16, MobileNet-V2, DenseNet-169 que nous avons définis précédemment.

Le principe du fonctionnement de notre modèle est illustré dans la figure 2.1.

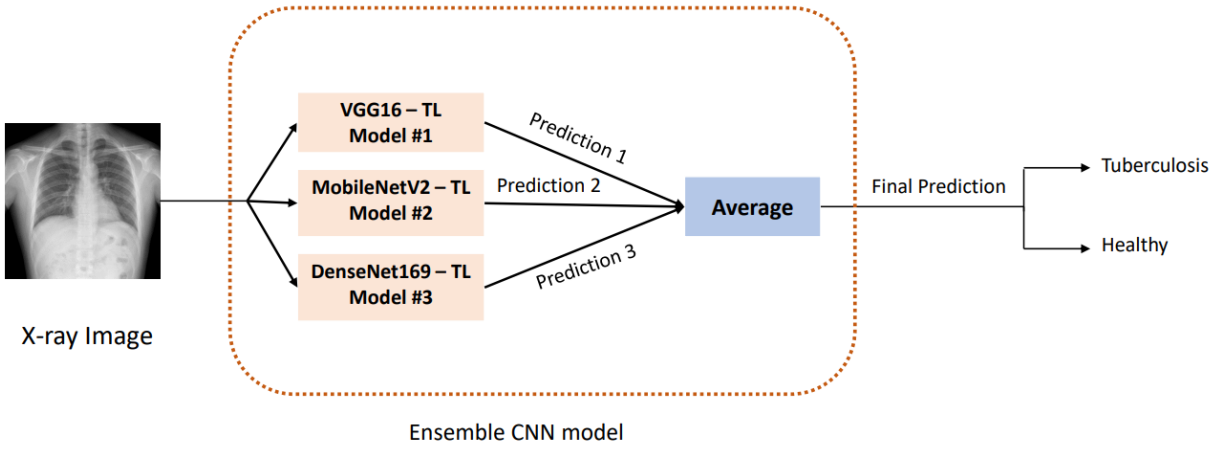

FIG. 2.1 : Schéma illustratif du fonctionnement de notre modèle.

Le développement d'un modèle CNN suit un ensemble d'étapes comme le montre la figure 2.2.

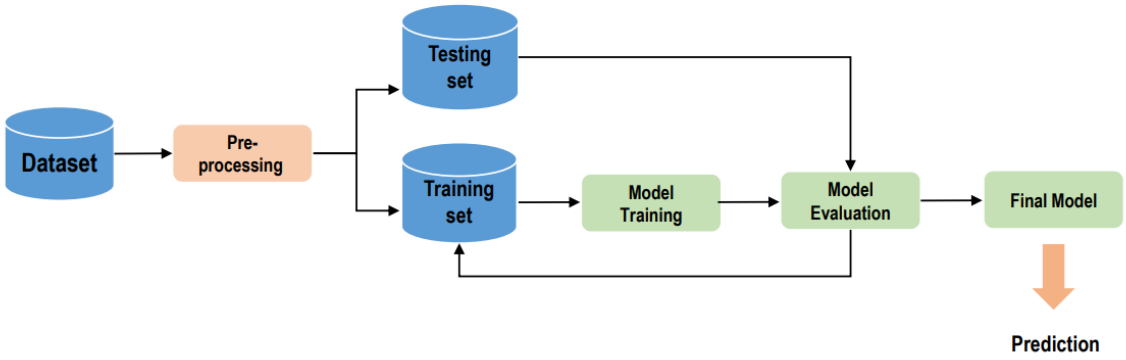

FIG. 2.2 : Diagramme du modèle CNN.

Nous commençons par le choix de notre ensemble de données (datatset) sur lequel nous effectuons un prétraitement et nous le divisons, par la suite, en deux sous-ensembles : un sous-ensemble pour l'entrainement de notre modèle (*train set*) et un sous-ensemble pour le

test (*test set*). Le modèle reçcoit en entrée les mêmes données d'entrainement de manière répétée et il continue à apprendre les caractéristiques de ces données. Nous pouvons utiliser le même sous-ensemble *test set* pour évaluer la performance du modèle durant son entrainement et après son entrainement (évaluation du modèle final). Nous obtenons à la fin un modèle capable de faire des prédictions sur les nouvelles images reçues.

2.3 Approches du Transfer Learning

Dans le premier chapitre, nous avons expliqué le principe du Transfer Learning. Nous citons à présent les deux approches existantes de cette technique.

1. Transfer Learning par extraction de features :

L'utilisation de modèles pré-entraînés comme extracteurs de features est basée sur l'idée de se servir des caractéristiques génériques déjà apprises par le modèle lors de son pré-entrainement sans avoir besoin de le ré-entrainer. Pour faire cela, nous gelons toutes les couches de ce modèle et nous remplaçons seulement le classifieur spécifique à la tâche pour laquelle ce modèle a été pré-entrainé par un nouveau classifieur spécifique à notre tâche.

Dans notre travail, nous avons suivi cette approche avec les trois modèles. La figure 2.3 montre comment nous l'avons appliquée avec le modèle VGG-16, par exemple.

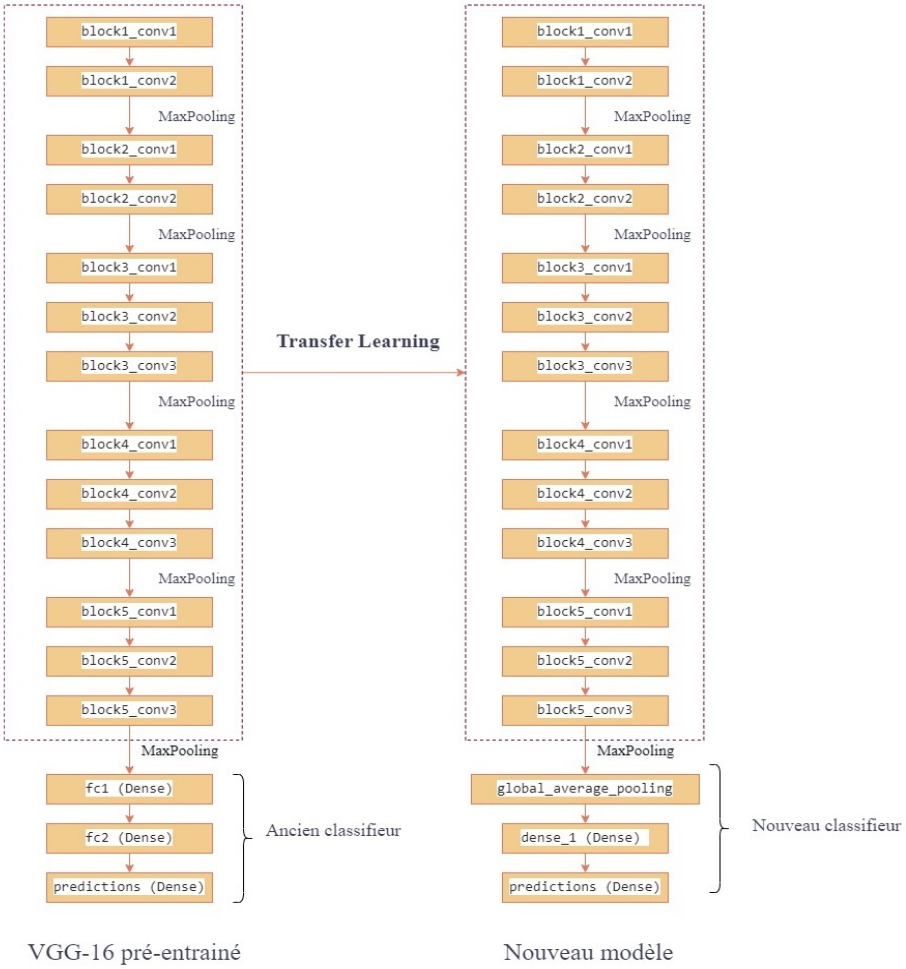

FIG. 2.3 : Utilisation du modèle VGG-16 comme extracteur de features.

2. Transfer Learning par fine-tuning :

Dans cette approche, au lieu de geler toutes les couches du modèle pré-entrainé, nous pouvons garder une partie des caractéristiques de ses premières couches et entrainer le reste avec le nouveau classifieur sur les nouvelles images de notre tâche. Les couches non-gelées correspondent généralement aux couches plus hautes du réseau.

2.4 Ensemble de données (Dataset)

Le dataset que nous avons utilisé contient 800 images des radiographies pulmonaires normales et anormales avec des manifestations de TBP, en plus des lectures de radiologues de ces clichés. Il est composé de deux datasets qui sont disponibles sur Kaggle :

- **Radiographies pulmonaires RX Montgomery County (MC)**

Cet ensemble contient 138 radiographies du thorax de face dont 80 sont des cas normaux et 58 des cas présentant des manifestations de tuberculose. Ce sont des images au format PNG et qui sont disponibles également au format DICOM. La taille des radiographies est de 4020×4892 ou 4892×4020 pixels. Tous les fichiers d'images sont nommés MCUCXR_****_X.png, où (****) représente un nombre de 4 chiffres, et X est " 0" pour une radiographie normale et " 1" pour une radiographie anormale [61].

- **Radiographies pulmonaires RX Shenzhen**

Cet ensemble contient 662 radiographies du thorax de face, dont 326 sont des cas normaux et 336 des cas présentant des manifestations de la tuberculose. Les radiographies sont fournies au format PNG. Leur taille est d'environ 3000 × 3000 pixels. Tous les fichiers d'images sont nommés CHNCXR_****_X.png, où (****) représente un nombre de 4 chiffres, et X est " 0" pour une radiographie normale et " 1" pour une radiographie anormale [61].

2.5 Pré-traitement

2.5.1 Divison du dataset (Dataset splitting)

Cette procédure consiste à prendre un dataset et à le diviser en deux sous-ensembles. Le premier sous-ensemble est utilisé pour entraîner le modèle et est appelé train set. Le second sous-ensemble n'est pas utilisé pour l'apprentissage du modèle, mais ses éléments sont fournis au modèle pour faire des prédictions qui sont comparées aux valeurs attendues. Ce deuxième ensemble de données est appelé test set. L'objectif de cette procédure est d'estimer la performance du modèle sur de nouvelles données qui n'ont pas été utilisées durant l'entrainement. Elle peut être utilisée pour des problèmes de classification ou de régression et pour tout algorithme d'apprentissage supervisé. Pour notre modèle, nous avons 80% des images du dataset pour l'entraînement et le reste 20% pour le test.

2.5.2 Redimensionnement des images

Les images du dataset MC ont comme dimension 4020 × 4892 pixels tandis que les images du dataset Shenzhen ont comme dimension 3000 × 3000 pixels. Les images de ces deux datasets ont été redimensionnées à 224 × 224 pixels.

2.5.3 Augmentation des données (Data Augmentation)

L'augmentation de données (data augmentation) est basée sur le principe d'augmenter artificiellement la quantité de données à utiliser pour un apprentissage. La technique de la data augmentation nous permet d'effectuer des opérations modifiant l'aspect de l'image, sans pour autant en modifier la sémantique. Cela consiste à transformer des données en effectuant un traitement sur chaque image, pour générer de nouvelles images, des variantes de l'image initiale. Il s'agit, par exemple, de contraste, de luminosité, de rognage, de rotation, …etc.

2.6 Couches du modèle CNN

Nous avons décrit dans le chapitre précédent les différentes couches qui composent le modèle CNN à savoir la couche de convolution, la couche de pooling et la couche fully-connected. Les paramètres et les hyper-paramètres de chaque couche sont indiqués dans le tableau suivant :

Tab. 2.1 : Tableau des paramètres des couches CNN

Couche	Paramètres	Hyper-paramètres
Convolution	Noyaux ou filtres	La taille et nombre de noyaux, le stride (le pas), le Zéro-padding et la fonction d'activation.
Pooling	/	la taille du filtre, le padding et le stride
Entièrement connectée	poids	Nombre de poids

2.7 Hyper-paramètres d'apprentissage CNN

Les algorithmes de Deep Learning impliquent l'optimisation des hyper-paramètres. Il s'agit de faire le choix d'un ensemble d'hyper-paramètres optimaux qui configure l'entrainement du modèle de Deep Learning. Nous allons, définir dans ce qui suit, quelques fonctions, optimiseurs et hyper-paramètres que nous utilisons pour notre modèle.

2.7.1 Taux d'apprentissage (Learning Rate)

A chaque mise à jour des poids du modèle, l'hyper-paramètres Learning Rate contrôle dans quelle mesure ce modèle doit être modifié en réponse à l'erreur estimée. Une valeur trop petite du Learning Rate peut donner un long processus d'apprentissage, tandis qu'une valeur trop grande peut avoir comme résultat un processus d'apprentissage trop rapide et instable [62].

2.7.2 Epoque (epoch)

Une époque d'apprentissage est une itération d'entraînement et le nombre d'époques correspond donc au nombre de fois où notre réseau est entraîné [62].

2.7.3 Batch size

Le batch size ou la taille du lot est un hyperparamètre qui définit le nombre d'échantillons d'images à traiter avant de mettre à jour les paramètres du modèle [62].

2.7.4 Etape par époque (Step per epoch)

C'est le nombre d'itérations d'un batch (lot) pendant une époque d'entrainement ou training epoch [62].

2.7.5 Etapes de validation (Validation steps)

C'est similaire à Step per epoch mis cela concerne les données de validation et non celles de l'entrainement [62].

2.7.6 Fonction de perte (Loss Function)

Lors de l'entrainement de notre modèle, nous cherchons à minimiser l'erreur. Pour cela, nous utilisons une fonction de perte ou Loss Function qui sert à estimer, de manière répétée, l'erreur qui serait commise sur les prédictions du modèle.

2.7.7 Fonction de classification Softmax

La fonction Softmax est fréquemment ajoutée à la dernière couche d'un réseau de classification d'images. Elle est utilisée dans la classification où l'on demande en général que le modèle de prédiction de renvoie :

- Un numéro identifiant la classe prédite, par exemple, 0 ou 1.

- Un vecteur de probabilité ou un vecteur de nombres compris entre 0 et 1 et dont la somme est égale à 1 et qui donnent les probabilités d'appartenance à chaque classe.

2.7.8 Optimiseur Adam

L'estimation adaptative du moment (adaptive moment estimation) ou Adam est un algorithme d'optimisation qui peut être utilisé en Deep Learning pour mettre à jour les poids de réseaux en fonction des données d'apprentissage [62].

2.8 Evaluation du modèle CNN (métriques)

2.8.1 Matrice de confusion

Le modèle de classification prédit si un objet appartient ou non à une classe. La matrice de confusion est en quelque sorte un résumé des résultats de prédiction pour un problème particulier de classification. Elle compare les données réelles pour une variable cible à celles prédites par un modèle. Il y a quatre (04) cas de prédiction que nous pouvons présenter dans la matrice de confusion (confusion matrix) suivante :

(+) : l'objet appartient à cette classe.

(-) : l'objet n'appartient pas à cette classe.

		Classe réelle	
		-	+
Prédiction	-	TN	FN
	+	FP	TP

(TP) : True Positives ou Vrais positifs.

(FP) : False Positives ou Faux Positifs.

(TN) : True negatives (TN) ou Vrais Négatifs.

(FN) : False Negatives (FN) ou Faux Négatifs.

2.8.2 Mesures de performance

A partir de cette matrice de confusion on peut calculer les mesures de performance souvent utilisées :

- **Précision (Accuracy) :** c'est la proportion des cas correctement prédits.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

- **Rappel (Recall) :** c'est le taux de vrais positifs (TP) appelée aussi Sensibilité (sensitivity).

$$Rappel = \frac{TP}{TP + FN}$$

- **Spécificité (Specificity) :** c'est le taux de vrais négatifs (TN).

$$Specificite = \frac{TN}{FP + TN}$$

- **Aire sous la croube ROC (ou Area Under the Curve, AUC) :**

La capacité du test à bien détecter les malades est la " sensibilité " tandis que sa capacité à bien détecter les non-malades est la " spécificité ". La sensibilité en fonction de (1 – spécificité) est représentée par une courbe appelée ROC (Receiver Operating Characteristic) pour toutes les valeurs seuils du marqueur étudié. L'aire sous la courbe ROC ou Area Under the Curve (AUC) représente la probabilité que la valeur du marqueur

soit plus élevée pour le malade que pour le non-malade, si on prend au hasard deux personnes, l'une malade et l'autre saine. Une AUC de 0,5 ou 50% indique que le marqueur est non-informatif. L'augmentation de l'AUC jusqu'à un maximum de 1,0 ou 100% traduit une amélioration des capacités discriminatoires. [63].

2.9 Implémentation

2.9.1 Langage de programmation utilisé (Python)

Nous utilisons dans notre projet le langage de programmation Python. Python est très utilisé notamment en Deep Learning et dans le domaine de l'Intelligence Artificielle en général parce qu'il contient un nombre important de bibliothèques très performantes dont nous avons besoin pour nos réseaux de neurones.

2.9.2 Google Colab

Nous utilisons la plateforme Google Colab qui nous permet d'entrainer des modèles de Deep Learning dans le Cloud en nous offrant une puissance de calcul considérable.

Les avantages de ce service gratuit sont :

- L'accès à un processeur graphique GPU et au TPU (Tensor Processor Unit).
- Un environnement de développement (Jupyter Notebook).
- La prise en charge de Python et des bibliothèques telles que Keras, TensorFlow et OpenCV.
- L'enregistrement des documents Colab (Jupyter Notebook) sur le compte Google Drive.

2.9.3 Frameworks, IDE et Bibliothèques Python

Spyder

Spyder est un environnement de développement (IDE) Python et fait partie de Anaconda [64].

FIG. 2.4 : Logo de Spyder.

Keras

Keras est écrit en langage Python. Il s'agit d'une API ou interface de programmation applicative open-source capable de s'exécuter sur TensorFlow, CNTK ou Theano. Keras est très utilisé pour développer et tester de réseaux de neurones et permet de mettre en place des architectures complexes CNN en créant très facilement les couches du réseau [64].

FIG. 2.5 : Logo de Keras.

TennsorFlow

TensorFlow est un framework écrit principalement en C++, avec une interface en Python et développé par l'équipe Google Brain de Google et permet de résoudre des problèmes mathématiques extrêmement complexes. Nous nous servons de ce framework pour entraîner et exécuter des réseaux de neurones profonds pour la classification et la reconnaissance d'images [64].

FIG. 2.6 : Logo de TensorFlow.

Autres bibliothèques Python

- **SciPy**

C'est un outil essentiel pour effectuer des calculs mathématiques et scientifiques : optimisation, algèbre linéaire, intégration, statistiques, etc.

- **NumPy**

C'est une bibliothèque essentielle pour les projets en Python, elle permet de faire des calculs matriciels et de générer des nombres aléatoires.

- **Scikit-learn**

Cette bibliothèque Python propose des algorithmes de classification pour la reconnaissance d'images et des algorithmes de régression pour la prédiction.

- **Pandas**

Pandas est utilisée pour analyser et organiser des données dans des structures sous forme de dataframes ou trames de données pour avoir une représentation claire des données ainsi que leur personnalisation.

- **Matplotlib**

Cette bibliothèque nous permet de faire une représentation visuelle des données et des résultats à laide des schémas 2D/3D et des graphiques à barres. Il faut noter que les fonctionnalités de Matlab sont similaires à celles de NumPy, SciPy, et Matplotlib [64].

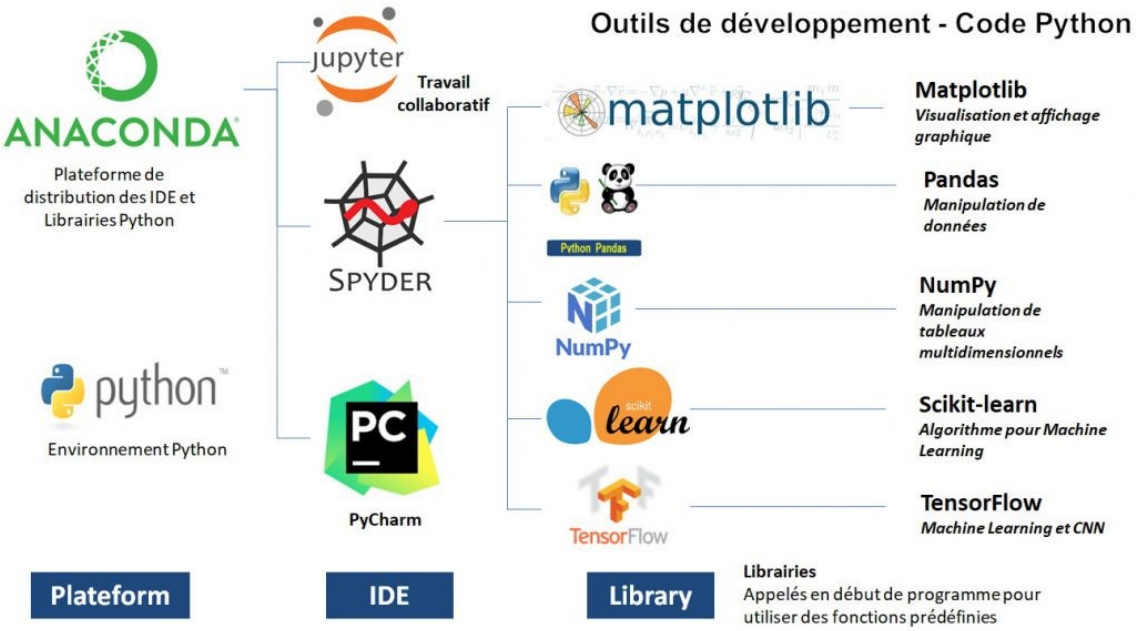

Fig. 2.7 : Outils de développement Python.

Nous présentons dans ce qui suit les étapes d'implémentation du Transfer Learning utilisant le modèle VGG-16. Ces mêmes étapes ont été suivies avec les deux autres modèles, MobileNet-V2 et DenseNet-169. Le code d'implémentation de notre modèle final sera également présenté à la fin.

2.9.4 Importation des bibliothèques

```
import os
import numpy as np
import tensorflow as tf
import scikitplot
import matplotlib.pyplot as plt
from matplotlib import cm
from sklearn.metrics import confusion_matrix
from keras.preprocessing import image
from collections import Counter
```

FIG. 2.8 : Code d'importation des bibliothèques.

2.9.5 Préparation des données

Après avoir importé nos données dans Google Drive, nous avons monté notre lecteur Google Drive dans l'interface de Google Colab.

```
from google.colab import drive
drive.mount('/content/drive')
```

FIG. 2.9 : Code du montage de Google Drive.

Comme nous avons mentionné, auparavant, toutes les images ont été précédemment redimensionnées à 224 x 224 pixels pour être compatibles avec les modèles pré-entrainés et pour réduire la complexité de calcul. La figure 2.10 montre que notre dataset est divisé en 600 images (80%) pour l'entrainement et 200 images (20%) pour le test.

```
print(' Images of training: {}'.format(train_x.shape))
print(' Images of tesing: {}'.format(test_x.shape))

print(' Labels of training: {}'.format(train_y.shape))
print(' Labels of tesing: {}'.format(test_y.shape))
```

```
Images of training: (600, 224, 224, 3)
Images of tesing: (200, 224, 224, 3)
Labels of training: (600,)
Labels of tesing: (200,)
```

FIG. 2.10 : Code montrant le contenu du dataset.

Comme le nombre d'images d'entrainement est trop faible dans notre cas, nous avons utilisé la classe "ImageDataGenerator" de la bibliothèque Keras pour générer de nouvelles images en temps réel en vue de prévenir un risque d'overfitting (sur-apprentissage).

```
trainAug = tf.keras.preprocessing.image.ImageDataGenerator(height_shift_range = 0.15,
                                                           width_shift_range = 0.15,
                                                           rotation_range = 10,
                                                           shear_range = 0.1,
                                                           fill_mode = 'nearest',
                                                           zoom_range=0.2
                                                           )
```

FIG. 2.11 : Code de création d'une instance de la classe ImageDataGenerator.

Classe	Description
ImageDataGenerator	Génère des lots de données d'images tensorielles et augmente les données en temps réel.

2.9.6 Application du Transfer Learning

Dans un premier temps, nous avons récupéré le modèle VGG-16 utilisant la classe VGG16 fournie par TensorFlow (nous avons chargé les poids "imagenet").

Include_top = False précise que nous voulons exclure les dernières couches denses (FC) et ne garder que les premières couches de convolutions. Il est également important de vérifier que l'image donnée en entrée est une image RVB de taille 224 x 224 pixels.

```
def VGG16_Model():

  baseModel = tf.keras.applications.VGG16(weights="imagenet",
                                          include_top=False,
                                          input_tensor=tf.keras.layers.Input(shape=(224, 224, 3)))
```

FIG. 2.12 : Code de chargement du réseau VGG-16.

Nous avons ajouté ensuite nos deux propres couches denses que nous allons entrainer par la suite. La dernière couche est la couche de prédiction avec une fonction d'activation "softmax" pour déterminer la classe de l'image "Tuberculose" ou "Normal".

```
output = baseModel.output
output = tf.keras.layers.BatchNormalization()(output)
output = tf.keras.layers.GlobalAveragePooling2D()(output)
output = tf.keras.layers.Dropout(0.2)(output)
output = tf.keras.layers.Dense(128, activation="relu")(output)
output = tf.keras.layers.Dropout(0.2)(output)
output = tf.keras.layers.Dense(2, activation="softmax")(output)
```

FIG. 2.13 : Code d'ajout des nouvelles couches (notre classifieur).

	Description
BatchNormalization	Batch normalization accélère l'entrainement, dans certains cas, en réduisant de moitié les époques ou mieux, et fournit une certaine régularisation, réduisant l'erreur de généralisation.
GlobalAveragePooling2D	GlobalAveragePooling2D effectue exactement la même opération que la couche Average Pooling sauf que la taille de la fenêtre est la taille de la matrice entière, c'est-à-dire qu'il calcule une seule valeur moyenne pour chacun des canaux. Elle est utilisée comme alternative à la couche Flatten après la dernière couche de Pooling du réseau CNN.
Dropout	La couche dropout permet de désactiver temporairement certains neurones dans le réseau, ainsi que toutes leurs connexions entrantes et sortantes pour réduire l'overfitting (sur-apprentissage) lors de l'entraînement du modèle.

Afin de ne pas écraser l'apprentissage déjà récupéré, nous avons gelé les premières couches de convolution du VGG-16. Le réseau que nous allons entrainer est composé des première couches gelées et de notre classifieur.

```
  model = tf.keras.Model(inputs = baseModel.input, outputs = output)

  for layer in baseModel.layers:
    layer.trainable = False
  return model

model = VGG16_Model()
```

FIG. 2.14 : Gel des première couches de convolution.

2.9.7 Initialisation des paramètres

Avant de procéder à l'entrainement, nous devons tout d'abord initialiser les hyper-paramètres : Learning Rate, Nombre des époques et Batch size.

Hyper-paramètre	Valeur
Learning Rate	0.001
Nombre des époques	100
Batch size	32

2.9.8 Compilation du modèle

Pour compiler notre modèle, nous avons choisi :

- L'optimiseur "Adam" qui prend comme argument "learning Rate".
- La fonction de perte "categorical_crossentropy" qui est utilisée pour les modèles de classification multi-classes où il y a deux ou plusieurs labels de sortie. une valeur de One-hot encoding sous la forme de 0 et de 1 est attribuée au label de sortie . Le label de sortie, si elle est présente sous forme de nombre entier, est convertie en encodage one-hot.
- La métrique "CategoricalAccuracy" qui calcule la fréquence à laquelle les prédictions correspondent aux labels one-hot et la métrique "AUC".

```
INIT_LR = 0.001
EPOCHS = 100
BATCHSIZE = 32

optimizer = tf.keras.optimizers.Adam(lr = INIT_LR)
model.compile(loss = "categorical_crossentropy",
              optimizer = optimizer,
              metrics = [tf.keras.metrics.CategoricalAccuracy(),
                         tf.keras.metrics.AUC()])
model.summary()
```

FIG. 2.15 : Code de compilation.

model.summary() nous permet de connaître le nombre total des paramètre et de ceux qui vont etre entrainés .

Nous voyons sur la figure 2.16 que sur 14,782,658 paramètres, nous allons entrainer seulement 66,946 paramètres, tandis que le reste (14,715,712 paramètres) sont exclus avec l'instruction *Include_Top= False* et seront utilisés directement avec leurs poids précédents et ne seront pas modifiés durant l'entrainement.

```
=================================================================
Total params: 14,782,658
Trainable params: 66,946
Non-trainable params: 14,715,712
_________________________________________________________________
None
```

FIG. 2.16 : Paramètres.

2.9.9 Entrainement du modèle

Une fois que nous avons terminé la compilation du modèle, nous pouvons procéder à son entrainement en appelant la méthode *fit()*.

Fonctions de rappel (callbacks)

Les Callbacks sont des outils dans keras qui permettent de suivre ce qui se passe pendant le processus d'apprentissage et éventuellement d'agir sur l'apprentissage. Le premier callback auquel nous pouvons facilement accéder est renvoyé lorsque la méthode *fit()* est appelée. Ce callback est un objet avec un attribut *history* qui est un dictionnaire dont les clés sont les métriques suivies pendant l'apprentissage. La mise en place d'autres callbacks se fait lors de l'appel à la méthode *fit()*.

```
reduceLROnPlat = tf.keras.callbacks.ReduceLROnPlateau(monitor='val_categorical_accuracy',
                                                      factor=0.8, patience=10,
                                                      verbose=1, mode='auto',
                                                      min_delta=0.0001, cooldown=5,
                                                      min_lr=0.0001)

model_checkpoint = tf.keras.callbacks.ModelCheckpoint(modelPath+'/vgg16-best-model.h5',
                                                      monitor='val_categorical_accuracy',
                                                      verbose=1, save_best_only=True,
                                                      mode='auto')

STEP_TRAIN = len(train_x) // BATCHSIZE

modelHistory = model.fit(trainAug.flow(train_x, train_y_oneHot, batch_size=BATCHSIZE),
                         steps_per_epoch=STEP_TRAIN,
                         validation_data= (test_x, test_y_oneHot),
                         epochs=EPOCHS, verbose=1,
                         callbacks=[model_checkpoint, reduceLROnPlat])
```

FIG. 2.17 : Code d'entrainement.

2.9.10 Evaluation du modèle

Ci-après le code d'évaluation du modèle :

Fonction Callback	Description
ModelCheckpoint	Utilisée pour enregistrer les poids (dans un fichier de point de contrôle), afin que ces poids puissent être chargés ultérieurement pour continuer l'entraînement à partir de l'état enregistré dans le cas où un plantage arriverait par exemple.
ReduceLROnPlateau	Permet de réduire le taux d'apprentissage lorsque l'apprentissage stagne. Ce rappel surveille une métrique et si aucune amélioration n'est constatée pendant un nombre d'époques que nous précisons par l'argument "patience", le taux d'apprentissage est réduit.

```
# Evaluate the Best Saved Model

loss, accuracy, auc= model.evaluate(x=test_x, y=test_y_oneHot, batch_size=32, verbose=1)
print('Model Accuracy: {:0.2f} | Model Loss: {:0.4f} | Model AUC: {:.02f}'.format(accuracy, loss, auc))
```

Fig. 2.18 : Code d'évaluation.

2.9.11 Test du modèle

Afin de tester le modèle, nous avons défini deux fonction, une qui prend en entrée une image parmi les images de test et renvoie la prédiction du modèle et une autre pour le traçage de l'image et la prédiction.

```
# visualizing a test case
img = image.load_img('/content/drive/MyDrive/valid/0/CHNCXR_0174_0.png', target_size=(224, 224))
preds = predict(model, img)

plot_preds(np.asarray(img), preds)
preds
```

```
array([1.000000e+00, 6.168563e-37], dtype=float32)
```

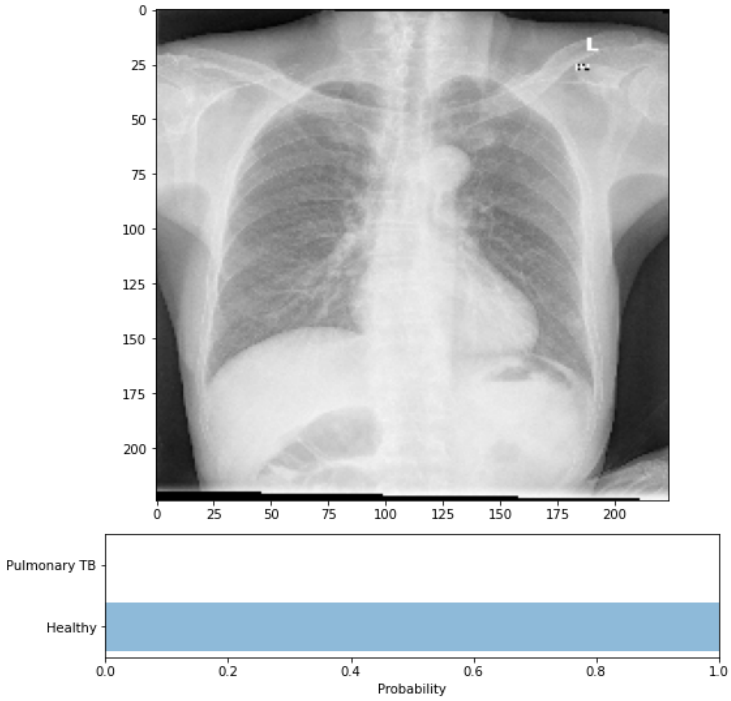

FIG. 2.19 : Visualisation d'un cas de test.

2.9.12 Modèle ensembliste

Nous montrons ci-après le code d'implémentation de notre modèle final (Modèle ensembliste) :

```
from tensorflow.keras.models import Model, load_model
from tensorflow.keras.layers import Input, Maximum, Average
model_1 = load_model('/content/drive/MyDrive/Tuber_new_direction/Pretrained_VGG16/vgg16-best-model.h5')
model_1 = Model(inputs=model_1.inputs,
                outputs=model_1.outputs,
                name='vgg16')
model_2 = load_model('/content/drive/MyDrive/Tuber_new_direction/Pretrained_MobileNetV2/mobilenetv2-best-model.h5')
model_2 = Model(inputs=model_2.inputs,
                outputs=model_2.outputs,
                name='Mobilenetv2')
model_3 = load_model('/content/drive/MyDrive/Tuber_new_direction/Pretrained_DenseNet169/denseNet169-best-model.h5')
model_3 = Model(inputs=model_3.inputs,
                outputs=model_3.outputs,
                name='DenseNet169')
models = [model_1, model_2, model_3]
model_input = Input(shape=(224, 224, 3))
model_outputs = [model(model_input) for model in models]
ensemble_output = Average()(model_outputs)
ensemble_model = Model(inputs=model_input, outputs=ensemble_output, name='ensemble')
```

FIG. 2.20 : Code d'implémentation de notre modèle final.

Le code qui fait le traçage de note modèle est le suivant :

```
from keras.utils.vis_utils import plot_model
plot_model(ensemble_model, to_file='ensemble_plot.png', show_shapes=True, show_layer_names=True)
```

FIG. 2.21 : Code du traçage de notre modèle final.

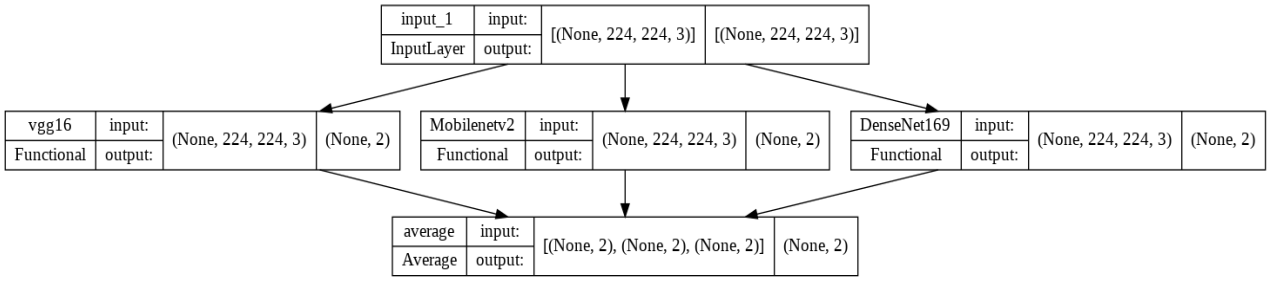

FIG. 2.22 : Traçage de notre modèle final.

2.10 Conclusion

Dans ce chapitre, nous avons décrit la conception de notre modèle de prédiction qui est basée sur l'utilisation de la technique de l'Ensemble Learning, et nous avons expliqué comment nous avons fusionné trois modèles entrainés séparément en un seul modèle ensembliste. Nous avons montré l'implémentation de la technique du Transfer Learning avec les modèles VGG-16, DenseNet-169 et MobileNet-V2. Nous avons fait la description de l'ensemble des données (dataset) utilisées et de leur prétraitement et nous avons énuméré les différents paramètres et hyper-paramètres qui interviennent dans le processus de l'apprentissage de nos modèles. Les méthodes utilisées dans l'implémentation et l'évaluation de nos modèles ont été également expliquées en détail.

Chapitre 3

Résultats et Discussion

3.1 Introduction

Dans le chapitre précédent nous avons montré comment nous avons implementé la technique du Transfer Learning avec les modèles VGG-16, DenseNet-169 et MobileNet-V2 ainsi que la conception et l'implémentation de notre modèle final (modèle ensembliste ou ensemble model). Dans ce chapitre, nous présentons et discutons les résultats obtenus pour chaque modèle et nous terminerons ensuite par une comparaison entre les résultats de notre travail et de ceux des travaux connexes mentionnés dans le premier chapitre.

3.2 Résultats et discussion

Grace à la bibliothèque Matplotlib, nous pouvons tracer les courbes d'évolution de la précision et de la perte de chaque modèle tout au long de son entraînement. Ces courbes nous permettent de visualiser les performances des modèles sur nos données d'entrainement ainsi que sur nos données de test. A chaque époque, un test de validation est effectué, ce qui nous permet de suivre l'évolution de la précision et de la perte de validation et de savoir l'efficacité d'apprentissage du modèle.

Comme nous avons mentionné dans le chapitre précédent, grâce au callback "checkpoint", nous pouvons enregistrer les poids du modèle à chaque fois qu'une amélioration de la précision se présente, cela nous permet de garder les poids du meilleur modèle dans un fichier et de les charger ultérieurement pour faire des prédictions. Afin de confirmer que ce modèle est réellement capable de classer les nouvelles images correctement, nous le testons sur notre sous-ensemble de test et nous l'évaluons par les métriques : Précision, Loss, AUC ainsi que par sa matrice de confusion.

Généralement, il y a deux problèmes qui peuvent être rencontrés lors de l'entrainement d'un CNN :

1- Sur-apprentissage (Overfitting)

Dans le cas du sur-apprentissage, le modèle CNN apprend par cœur les données et fonctionne donc bien sur les données d'entrainement. Mais il effectue de mauvaises prédictions sur les données de validation, car il ne sait pas généraliser à des données inconnues.

2- Sous-apprentissage (Underfittting)

En sous-apprentissage, le modèle CNN n'arrive pas à déduire des informations de l'ensemble de données et n'apprend donc pas assez. Il fait de mauvaises prédictions sur les données d'entrainement car il n'arrive pas à capter la relation entre ces données et leurs labels.

3.2.1 Premier modèle (VGG-16)

Courbes d'évolutions de la précision et de la perte :

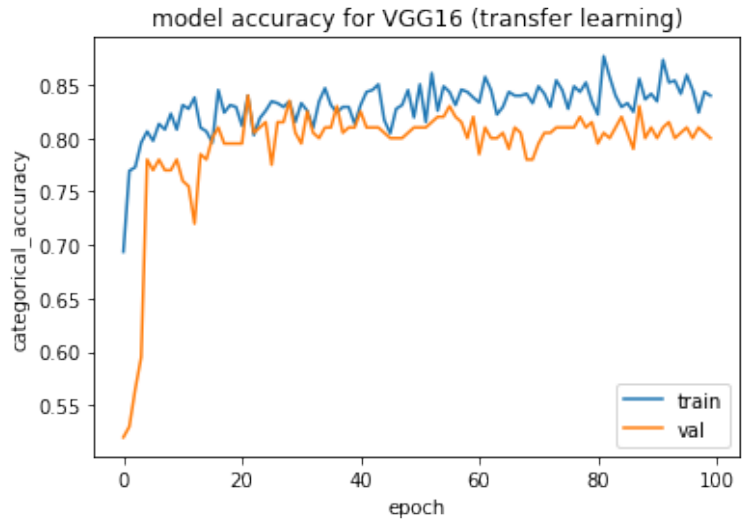

FIG. 3.1 : Courbe d'évolution de la précision du premier modèle.

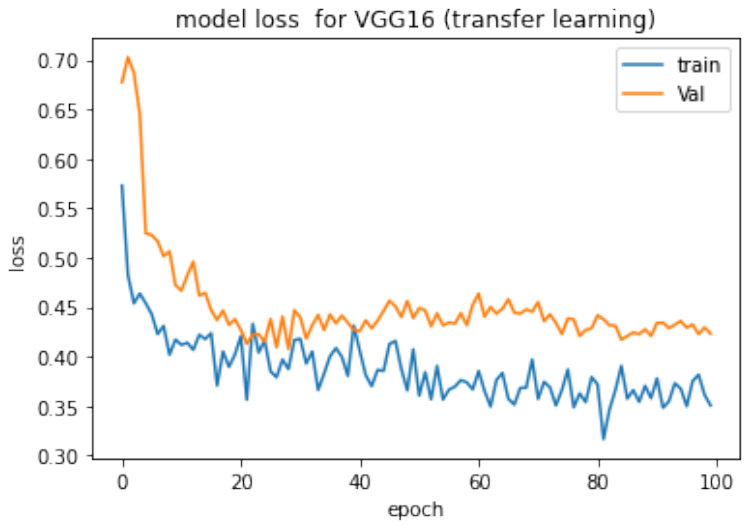

FIG. 3.2 : Courbe d'évolution de la perte du premier modèle.

En observant la figure 3.1, nous remarquons que les précisions d'entrainement et de validation augmentent parallèlement et rapidement lors des premières époques ce qui signifie que le modèle est en train d'apprendre les caractéristiques présentes dans les images pour la première fois, puis à un certain point, elles cessent d'augmenter et se stabilisent, car à ce stade le modèle est à la recherche de nouvelles caractéristiques à apprendre (des caractéristiques qu'il n'a pas apprises pendant les époques précédentes).

- La précision de validation atteint la valeur max de 84% à l'époque 22.

Sur la figure 3.2, nous constatons l'inverse. Les pertes d'entrainement et de validation diminuent parallèlement ce qui indique qu'il y a une amélioration de perte, puis à partir d'un certain point, elles cessent de diminuer et se stabilisent.

- Ces résultats confirment que ce modèle ne souffre ni du problème d'overfitting ni d'underfitting.

Résultats du test :

Après avoir testé ce modèle sur notre sous-ensemble de test, nous avons obtenu les résultats suivants :

```
7/7 [==============================] - 21s 394ms/step - loss: 0.4129 - categorical_accuracy: 0.8400 - auc: 0.8944
Model Accuracy: 0.84 | Model Loss: 0.4129 | Model AUC: 0.89
```

FIG. 3.3 : Résultats du test du premier modèle.

Matrice de confusion :

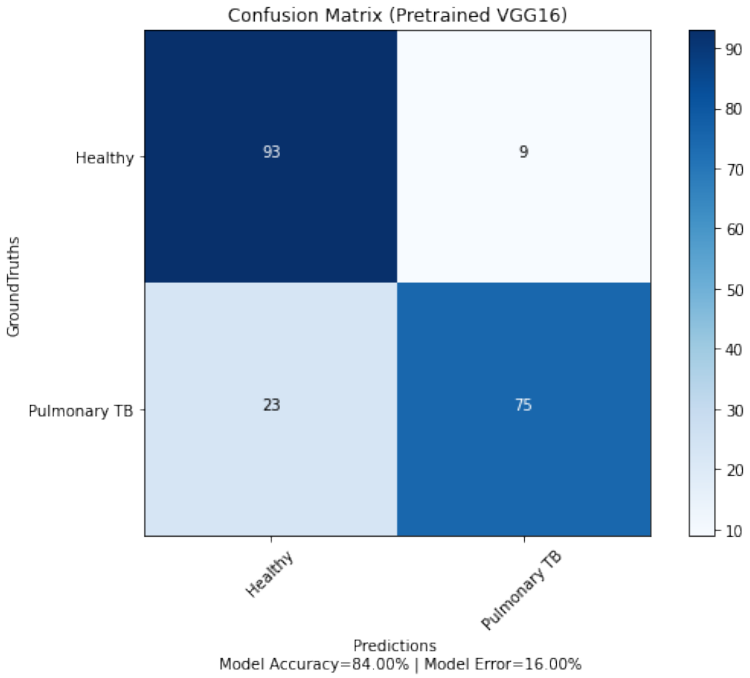

FIG. 3.4 : Matrice de confusion du premier modèle.

En observant la figure 3.4, nous remarquons que sur 200 images du test :

- 168 images ont été bien classées, donc un taux de précision de 84.00%.
- 32 images ont été mal classées, donc un taux d'erreur de 16.00%.

3.2.2 Deuxième modèle (MobileNet-V2)

Courbes d'évolutions de la précision et de la perte :

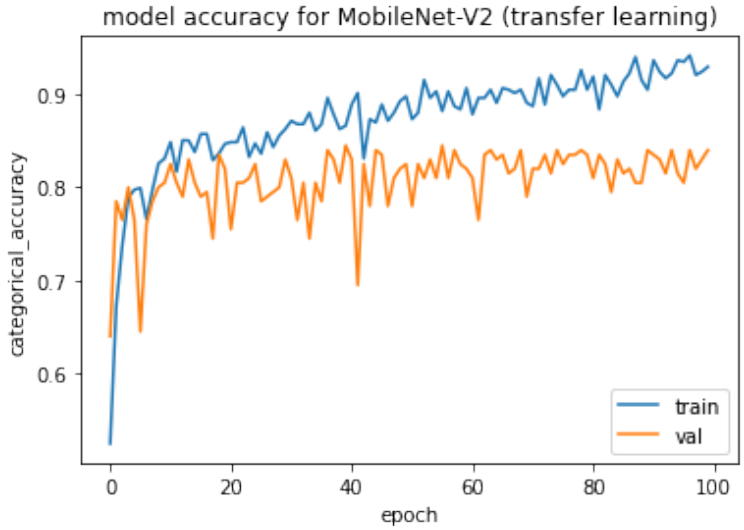

FIG. 3.5 : Courbe d'évolution de la précision du deuxième modèle.

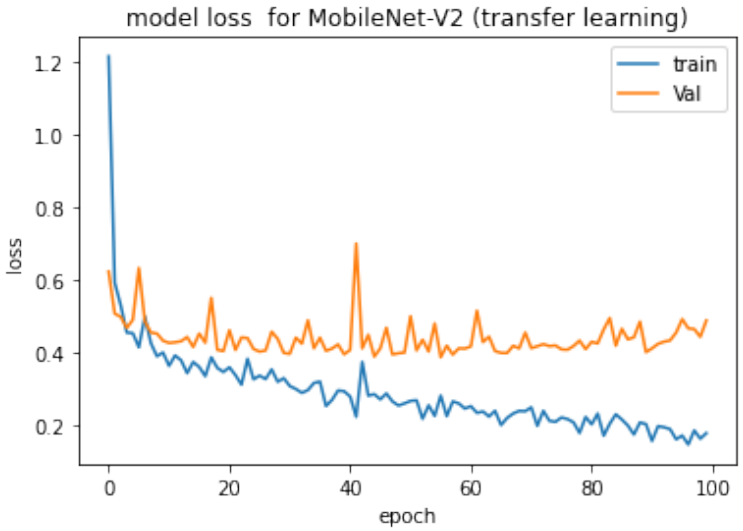

FIG. 3.6 : Courbe d'évolution de la perte du deuxième modèle.

En observant la figure 3.5, nous remarquons que les précisions d'entrainement et de validation augmentent parallèlement et rapidement lors des premières époques ce qui signifie que le modèle est en train d'apprendre, puis à un certain point, la précision de validation cesse d'augmenter et se stabilise, car à ce stade le modèle est à la recherche de nouvelles caractéristiques à apprendre.

- La précision de validation atteint la valeur max de 84.50% à l'époque 40.

Sur la figure 3.6, nous constatons l'inverse. Les pertes d'entrainement et de validation diminuent ce qui indique qu'il y a une amélioration de perte, puis à partir d'un certain point, la perte de validation se stabilise.

- Ces résultats confirment que ce modèle n'a pas de problème : ni overfitting, ni underfitting.

Résultats du test :

Après avoir testé ce modèle sur notre sous-ensemble de test, nous avons obtenu les résultats suivants :

```
7/7 [==============================] - 15s 79ms/step - loss: 0.3957 - categorical_accuracy: 0.8450 - auc: 0.9063
Model Accuracy: 0.85 | Model Loss: 0.3957 | Model AUC: 0.91
```

FIG. 3.7 : Résultats de test du deuxième modèle.

Matrice de confusion :

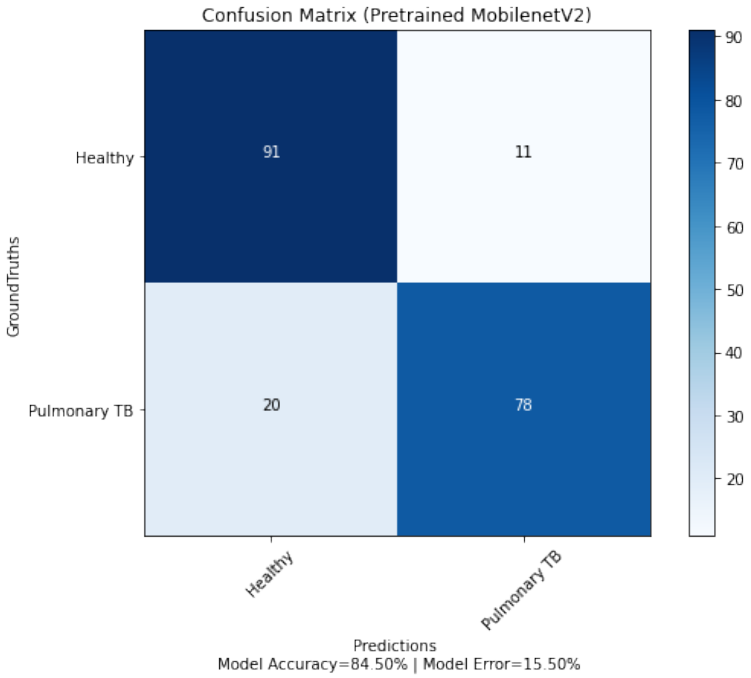

FIG. 3.8 : Matrice de confusion du deuxième modèle.

En observant la figure 3.8, nous remarquons que sur 200 images du test :

- 169 images ont été bien classées, donc un taux de précision de 84.50%.

- 31 images ont été mal classées, donc un taux d'erreur de 15.50%.

3.2.3 Troisième modèle (DenseNet-169)

Courbes d'évolutions de la précision et de la perte :

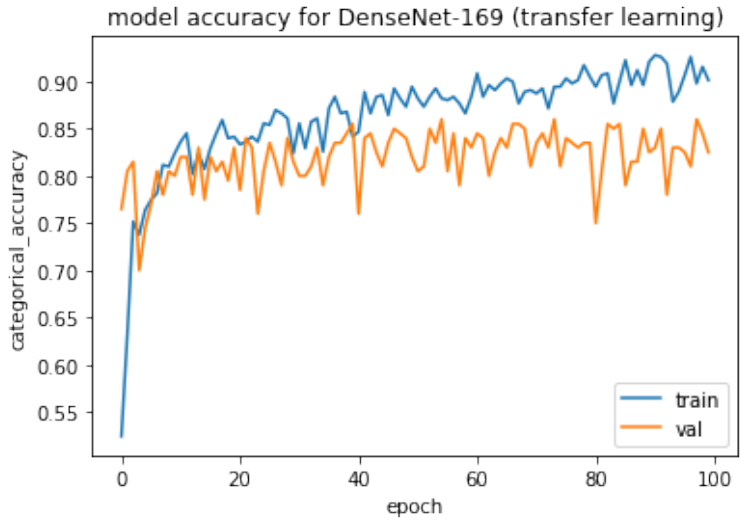

FIG. 3.9 : Courbe d'évolution de la précision du troisième modèle.

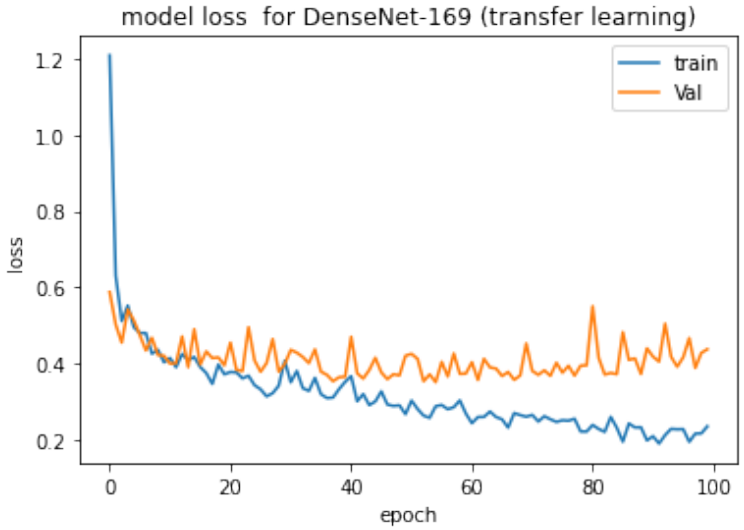

FIG. 3.10 : Courbe d'évolution de la perte du troisième modèle.

La figure 3.9 montre que les précisions d'entrainement et de validation augmentent en parallèle lors des premières époques, cela veut dire que le modèle est en train d'apprendre, puis à un certain point, la précision de validation cesse d'augmenter et se stabilise, car à ce stade le modèle est à la recherche de nouvelles caractéristiques à apprendre.

- La précision de validation atteint la valeur max de 86% à l'époque 55.

Sue la figure 3.10, nous constatons l'inverse. Les pertes d'entrainement et de validation diminuent ce qui indique qu'il y a une amélioration de perte, puis à partir d'un certain point, la perte de validation se stabilise.

- Ces résultats confirment que ce modèle ne souffre ni du problème d'overfitting ni du underfitting.

Résultats du test :

Après avoir testé ce modèle sur notre sous-ensemble de test, nous avons obtenu les résultats suivants :

```
7/7 [==============================] - 17s 325ms/step - loss: 0.3511 - categorical_accuracy: 0.8600 - auc: 0.9224
Model Accuracy: 0.86 | Model Loss: 0.3511 | Model AUC: 0.92
```

FIG. 3.11 : Résultats de test du troisième modèle.

Matrice de confusion :

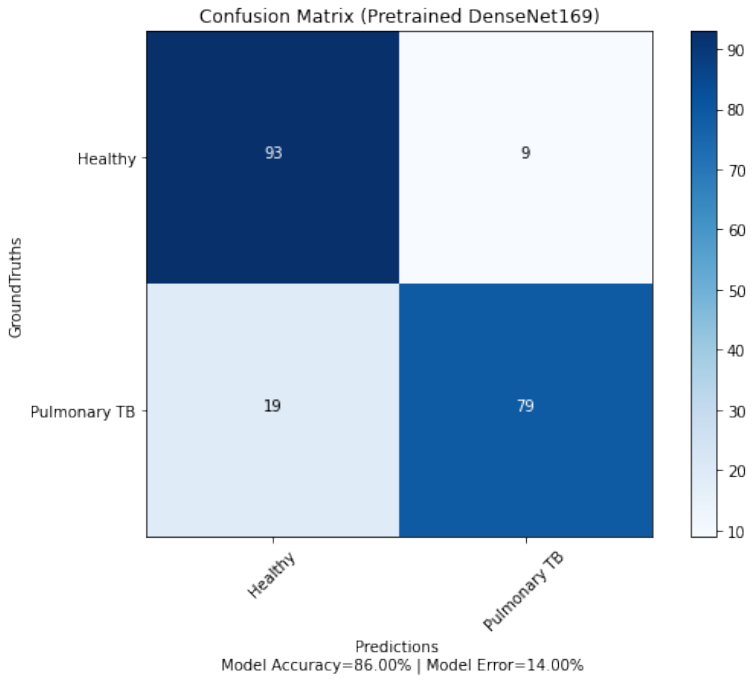

FIG. 3.12 : Matrice de confusion du troisième modèle.

En observant la figure 3.12, nous remarquons que sur 200 images du test :

- 172 images ont été bien classées, donc un taux de précision de 86.00%.
- 28 images ont été mal classées, donc un taux d'erreur de 14.00%.

3.2.4 Modèle final (Modèle ensembliste)

Résultats du test :

Nous avons testé notre modèle final sur le sous-ensemble du test, et nous avons obtenu les résultats suivants :

```
7/7 [==============================] - 27s 770ms/step - loss: 0.3397 - categorical_accuracy: 0.8750 - auc: 0.9305
Model Accuracy: 0.88 | Model Loss: 0.3397 | Model AUC: 0.93
```

FIG. 3.13 : Résultats de test du modèle final (modèle ensembliste).

Matrice de confusion :

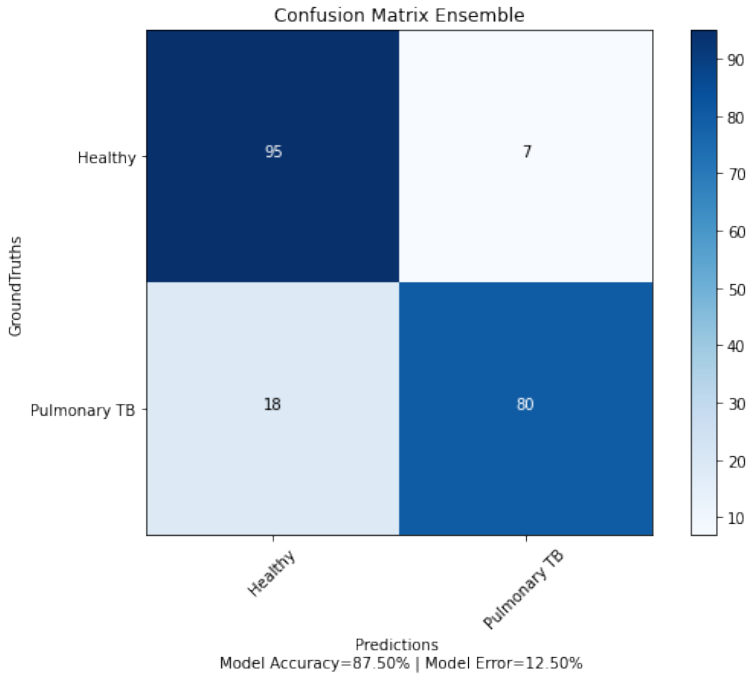

FIG. 3.14 : Matrice de confusion du modèle final (modèle ensembliste).

La figure 3.14 nous montre que sur 200 images du test :

- 175 images ont été bien classées, donc un taux de précision de 87.50%.
- 25 images ont été mal classées, donc un taux d'erreur de 12.50%.

3.2.5 Discussion

En se basant sur les expérimentations que nous avons faites sur les modèles, VGG-16, DenseNet-169 et MobileNet-V2, nous constatons que ces trois modèles ont donnés des résultats satisfaisants.

Nous avons choisi de tester ces trois architectures puisqu'elles ont prouvé leur efficacité sur d'autres tâches de classification d'images. Ces trois modèles ont été capables de généraliser et donc de faire la différence entre les radiographies avec manifestation de tuberculose et celles normales. DenseNet-169 a été le plus performant surpassant VGG-16 et MobileNet-V2 avec une précision de 86% et une AUC de 0.92. Il a été capable de classer correctement 172 images sur 200 images. Alors que, VGG-16 a été le moins précis, avec une précision de 84% et une AUC de 0.89. Cela est dû au fait que plus le réseau est profond (il possède plus de couches), plus il est précis. MobileNet-V2 a donné une précision de 84.50% et une AUC de 0.91.

Nous avons formé avec ces trois architectures un nouveau modèle ensembliste qui a donné une précision plus élevée que celles des modèles séparés. Ce modèle ensembliste a surpassé DenseNet-169 avec une précision de 87.50 et une AUC 0.93 et a été capable de classer correctement 175 images sur 200 images.

3.3 Comparaison avec les travaux connexes

Nous n'avons pas trouvé, dans nos recherches, des travaux connexes utilisant les deux datasets de Shenzhen et de Montgomery County (MC) en un seul ensemble comme dans notre cas. Néanmoins, en comparant nos résultats avec ceux des travaux connexes indiqués dans le tableau 3.1, qui ont utilisé également les datasets de Shenzhen et de MC, nous pouvons dire que nos expérimentations ont donné de bons résultats. La précision de notre modèle ensembliste final (87.50 %) a surpassé les précisions des deux travaux [58] et [59]. Elle est presque égale à celle du travail [57] et elle n'est pas loin de la plus haute précision (91.7 %) du travail [60] qui a utilisé la technique de segmentation.

Tab. 3.1 : Tableau comparatif

Référence	Dataset	Méthode	Accuracy (%)	AUC	Classes
[57]	Combinaison de 4 datasets (dataset de Montgomery County (MC) , dataset de Shenzhen, dataset de Belarus, dataset de JSRT).	Modèle ensembliste (AlexNet, GoogleNet, ResNet)	88.24	0.93	2
[58]	Dataset de Shenzhen	VGG-16 VGG-19 ResNet50 Googlenet	86.74 84.33 81.92 80.72	0.92 0.91 0.91 0.88	2
	Dataset de MC	VGG-16 VGG-19 ResNet50 Googlenet	77.14 77.14 71.42 71.42	0.75 0.90 0.76 0.75	
[59]	Dataset de Shenzhen	CNN AlexNet	82 84.20	0.89 0.91	2
	Dataset de MC	CNN AlexNet	65.80 72.50	0.74 0.80	
[60]	Dataset de Shenzhen	Inception-V3	91.7	0.96	2
	Dataset de MC	Inception-V3	87.47	0.92	
Notre approche	Combinaison de 2 datasets (dataset de Shenzhen et dataset de MC)	VGG-16 MobileNet-V2 DenseNet-169 **Ensemble**	84 84.50 86 **87.50**	0.89 0.91 0.92 **0.93**	2

3.4 Conclusion

Dans ce chapitre, nous avons comparé les précisions obtenues par les trois modèles séparément avec celle de notre modèle ensembliste et nous avons montré que ce modèle final a été le plus performant avec une précision de 87.50% supérieure aux précisions données par chaque modèle. Nous avons fait, également, une comparaison avec les travaux connexes, et nous avons constaté que la précision de notre modèle n'a été dépassée que par une seule valeur de précision (91.7%) du fait de l'utilisation de la technique de segmentation dans ce travail connexe.

Conclusion Générale et Travaux Futurs

Conclusion générale

Dans notre travail, nous avons réalisé un modèle de prédiction de la présence ou non de la tuberculose pulmonaire, basé les réseaux de neurones convolutifs (CNN ou ConvNet) qui constituent une classe des algorithmes du Deep Learning.

Après avoir étudié plusieurs travaux qui ont été proposés dans ce domaine, nous avons retenu les CNNs pour leur excellente performance dans le domaine de la classification d'images. Ainsi, nous avons utilisé trois modèles CNN d'architectures profondes différentes pré-entrainés sur ImageNet : DenseNet-169, MobileNet V2 et VGG-16 pour appliquer le Transfer Learning. Nous avons également utilisé la technique de Data Augmentation pour élever les nombres des données d'entrainement.

Afin d'obtenir un modèle plus performant, nous avons utilisé la technique de l'Ensemble Learning en construisant un modèle ensembliste ou Ensemble Model composé de ces trois modèles. Des tests sur un ensemble de données d'entrée de patients infectés et sains ont été effectués et montré de bons résultats en termes de classification. Ce qui montre que les modèles de réseaux de neurones convolutifs (CNN) peuvent être utilisés dans le diagnostic médical.

Comme perspectives, nous envisagerons d'améliorer notre présent travail par la technique de la segmentation utilisant les CNNs. Il s'agit de la délimitation du contour des anomalies dans les structures d'intérêt au niveau du pixel.

Références

[1] *Tuberculose : informations et traitements - Institut Pasteur*. URL : https://www.pasteur.fr/fr/centre-medical/fiches-maladies/tuberculose#traitemen.

[2] World Health Organization et al. *WHO guideline : recommendations on digital interventions for health system strengthening*. World Health Organization, 2019.

[3] Olivier Peraldi et Françoise Rouch. *Pour une filière des services à la personne en environnement numérique*. 2015.

[4] *Qu'est ce que l'e-santé ?* fr-FR. URL : https://www.fondationdelavenir.org/e-sante-definition/.

[5] *La Télémédecine pourrait-elle bien redonner l'envie d'un médecin traitant ? Réponse du Docteur Pierre SIMON*. fr-FR. Nov. 2019. URL : https://managersante.com/2019/11/04/la-telemedecine-peut-elle-redonner-lenvie-dun-medecin-traitant/.

[6] *Santé mobile : des applications de qualité*. fr. URL : https://www.has-sante.fr/jcms/p_3106528/fr/sante-mobile-des-applications-de-qualite.

[7] Linda Cambon Louise Bourdel. "Les domaines de l'e-santé". In : *Actualité et dossier en santé publique* (2019).

[8] *Comment utiliser un glucomètre ?* fr-FR. Août 2016. URL : https://www.glucoguide.top/comment-utiliser-un-glucometre/.

[9] *Intelligence artificielle et santé Inserm, La science pour la santé*. fr-FR. URL : https://www.inserm.fr/dossier/intelligence-artificielle-et-sante/.

[10] *Comment fonctionne le robot chirurgical da Vinci Si HD ® ? - Centre Hospitalier Universitaire (CHU) de Toulouse*. URL : https://www.chu-toulouse.fr/comment-fonctionne-le-robot.

[11] Patrick Aïdan et Maroun Bechara. “Gasless trans-axillary robotic thyroidectomy : the introduction and principle”. In : *Gland Surgery* 6.3 (2017), p. 229.

[12] *L’IA peut réaliser un diagnostic médical avec plus de précision qu’un humain*. fr-FR. Sept. 2019. URL : https://siecledigital.fr/2019/09/25/lia-peut-realiser-un-diagnostic-medical-avec-plus-de-precision-quun-humain/.

[13] *Artificial Intelligence Assisted Radiology Technologies Aid COVID-19 Fight in China*. en. Mar. 2020. URL : http://www.itnonline.com/article/artificial-intelligence-assisted-radiology-technologies-aid-covid-19-fight-china.

[14] *Une introduction à l’apprentissage automatique - Geek Madagascar*. URL : https://geek.mg/fr/tutoriel/une-introduction-a-lapprentissage-automatique/.

[15] MAÎTRE VÉRONIQUE RONDEAU- ABOULY. *Définition de l’Intelligence Artificielle (IA)*. fr. URL : https://www.cabinetrondeauabouly.com/actualites/84/2020-10-19-definition-de-lintelligence-artificielle-ia.htm.

[16] *What Is Machine Learning and How Does It Work | Simplilearn*. en-US. Oct. 2015. URL : https://www.simplilearn.com/what-is-machine-learning-and-why-it-matters-article.

[17] *Les 3 étapes essentielles de l’apprentissage automatique (Machine Learning)*. fr. URL : https://www.spiria.com/fr/blogue/intelligence-artificielle/3-etapes-essentielles-apprentissage-automatique-machine-learning/.

[18] *Apprentissage Supervisé : Introduction*. fr-FR. Juil. 2019. URL : https://machinelearnia.com/apprentissage-supervise-4-etapes/.

[19] *Types d’apprentissage automatique : apprentissage supervisé ou non supervisé - Autre*. fr. URL : https://fr.myservername.com/types-machine-learning.

[20] *Machine Learning : les 9 types d’algorithmes les plus pertinents en entreprise*. fr. URL : https://www.lemagit.fr/conseil/Machine-Learning-les-9-types-dalgorithmes-les-plus-pertinents-en-entreprise.

[21] *Apprentissage par renforcement*. fr. Page Version ID : 194552770. Juin 2022. URL : https://fr.wikipedia.org/w/index.php?title=Apprentissage_par_renforcement&oldid=194552770.

[22] Éditions Larousse. *neurone formel - LAROUSSE*. fr. URL : https://www.larousse.fr/encyclopedie/divers/neurone_formel/187288.

[23] Éditions Larousse. *neurone - LAROUSSE*. fr. URL : https://www.larousse.fr/encyclopedie/divers/neurone/73210.

[24] Samir Chtita. "Modélisation de molécules organiques hétérocycliques biologiquement actives par des méthodes QSAR/QSPR. Recherche de nouveaux médicaments". Thèse de doct. Université Moulay Ismaïl, Meknès, 2017.

[25] Claude Touzet. *les réseaux de neurones artificiels, introduction au connexionnisme*. Ec2, 1992.

[26] *Comprendre les réseaux de neurones - MonCoachData*. fr-FR. Section : Deep Learning. Juin 2019. URL : https://moncoachdata.com/blog/comprendre-les-reseaux-de-neurones/.

[27] *Qu'est-ce qu'un réseau de neurone ? - Actualité Informatique*. URL : https://actualiteinformatique.fr/intelligence-artificielle/definition-reseau-de-neurone.

[28] *Fonctions d'activation dans les réseaux de neurones – Acervo Lima*. URL : https://fr.acervolima.com/fonctions-d-activation-dans-les-reseaux-de-neurones/.

[29] *Activation Functions in Neural Network*. en-US. Section : Machine Learning Model. Oct. 2019. URL : https://studymachinelearning.com/activation-functions-in-neural-network/.

[30] Céline Deluzarche. *Définition | Deep Learning - Apprentissage profond | Futura Tech*. fr. Section : intelligence artificielle. URL : https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/.

[31] *Quelles sont les différences entre le Deep learning et le Machine learning ?* fr. URL : https://www.ionos.fr/digitalguide/web-marketing/search-engine-marketing/deep-learning-vs-machine-learning/.

[32] *Qu'est ce qu'un réseau de neurones convolutif (ou CNN) ?* fr. URL : https://openclassrooms.com/fr/courses/4470531-classez-et-segmentez-des-donnees-visuelles/5082166-quest-ce-quun-reseau-de-neurones-convolutif-ou-cnn.

[33] *What are Convolutional Neural Networks ?* en-us. URL : https://www.ibm.com/cloud/learn/convolutional-neural-networks.

[34] Humanités Numériques Learning et Big Data. "Détection et classification de visages pour la description de l'égalité homme-femme dans les archives télévisuelles". In : ().

[35] Pooja Mahajan. *Max Pooling*. en. Juil. 2020. URL : https://poojamahajan5131.medium.com/max-pooling-210fc94c4f11.

[36] *Convolutional Neural Networks (CNN) : Step 4 - Full Connection - Blogs - SuperDataScience | Machine Learning | AI | Data Science Career | Analytics | Success.* URL : https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-4-full-connection.

[37] Wafa Njima et al. "Deep CNN for indoor localization in IoT-sensor systems". In : *Sensors* 19.14 (2019), p. 3127.

[38] Sourish Dey. *CNN application on structured data-Automated Feature Extraction*. en. Sept. 2018. URL : https://towardsdatascience.com/cnn-application-on-structured-data-automated-feature-extraction-8f2cd28d9a7e.

[39] Omar Faruk et al. "A Novel and Robust Approach to Detect Tuberculosis Using Transfer Learning". In : *Journal of Healthcare Engineering* 2021 (2021).

[40] Dipanjan (DJ) Sarkar. *A Comprehensive Hands-on Guide to Transfer Learning with Real-World Applications in Deep Learning*. en. Nov. 2018. URL : https://towardsdatascience.com/a-comprehensive-hands-on-guide-to-transfer-learning-with-real-world-applications-in-deep-learning-212bf3b2f27a.

[41] Mehmet Akturk. *What is Ensemble Learning ?* en. Avr. 2021. URL : https://mathchi.medium.com/what-is-ensemble-learning-b2cf66cc7172.

[42] *ML | Premiers pas avec AlexNet – Acervo Lima*. URL : https://fr.acervolima.com/ml-premiers-pas-avec-alexnet/.

[43] *CNN Architectures : LeNet, AlexNet, VGG, GoogLeNet, ResNet and more - coderz.py*. URL : https://coderzpy.com/cnn-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more/.

[44] Hassan RAZA. *CNN Architectures : LeNet, AlexNet, VGG, GoogLeNet, ResNet and more*. en-US. Août 2020. URL : http://coderzpy.com/cnn-architectures-lenet-alexnet-vgg-googlenet-resnet-and-more/.

[45] *VGG : en quoi consiste ce modèle ? Daniel vous dit tout !* URL : https://datascientest.com/quest-ce-que-le-modele-vgg.

[46] Yufeng ZHENG, Clifford YANG et Alex MERKULOV. "Breast cancer screening using convolutional neural network and follow-up digital mammography". In : *Computational Imaging III*. T. 10669. SPIE. 2018, p. 1066905.

[47] Kwun Ho NGAN et al. "Making densenet interpretable a case study in clinical radiology". In : *medRxiv* (2019), p. 19013730.

[48] *MobileNet version 2*. URL : https://machinethink.net/blog/mobilenet-v2/.

[49] Boudjedri MOUNIA. *La tuberculose extrapulmonaire représente 70% des cas de tuberculose en Algérie*. fr-fr. URL : https://www.aps.dz/sante-science-technologie/137670-la-tuberculose-extrapulmonaire-represente-70-des-cas-de-tuberculose-en-algerie.

[50] *Mycobacterium tuberculosis*. fr. Page Version ID : 192787613. Avr. 2022. URL : https://fr.wikipedia.org/w/index.php?title=Mycobacterium_tuberculosis&oldid=192787613.

[51] DOCTISSIMO. *La radiographie standard*. fr. Section : Radiographie. Avr. 2018. URL : https://www.doctissimo.fr/html/sante/imagerie/radiographie_standard.htm.

[52] *Scanner thoracique : interprétation, effets secondaires, comment ça se passe ?* fr. URL : https://sante.journaldesfemmes.fr/fiches-anatomie-et-examens/2615955-scanner-thoracique-interpretation-resultat-effets-secondaires-deroule-risques/.

[53] CEA. *L'imagerie médicale*. fr. Articles Dossiers :L'essentiel sur. Publisher : CEA. Mar. 2022. URL : https://www.cea.fr/comprendre/Pages/sante-sciences-du-vivant/essentiel-sur-imagerie-medicale.aspx.

[54] World Health Organization et al. *Chest radiography in tuberculosis detection : summary of current WHO recommendations and guidance on programmatic approaches*. Rapp. tech. World Health Organization, 2016.

[55] *Definition of chest x-ray - NCI Dictionary of Cancer Terms - NCI*. en. nciAppModulePage. Archive Location : nciglobal,ncienterprise. Fév. 2011. URL : https://www.cancer.gov/publications/dictionaries/cancer-terms/def/chest-x-ray.

[56] World Health Organization et al. "Global tuberculosis report 2020: executive summary". In : (2020).

[57] Rahul Hooda, Ajay Mittal et Sanjeev Sofat. "Automated TB classification using ensemble of deep architectures". In : *Multimedia Tools and Applications* 78.22 (2019), p. 31515-31532.

[58] Syeda Shaizadi Meraj et al. "Detection of pulmonary tuberculosis manifestation in chest X-rays using different convolutional neural network (CNN) models". In : *Int. J. Eng. Adv. Technol.(IJEAT)* 9.1 (2019), p. 2270-2275.

[59] R Sivaramakrishnan et al. "Comparing deep learning models for population screening using chest radiography". In : *Medical Imaging 2018: Computer-Aided Diagnosis*. T. 10575. SPIE. 2018, p. 322-332.

[60] Dipayan Das, KC Santosh et Umapada Pal. "Inception-based Deep Learning Architecture for Tuberculosis Screening using Chest X-rays". In : *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021, p. 3612-3619.

[61] Stefan Jaeger et al. "Two public chest X-ray datasets for computer-aided screening of pulmonary diseases". In : *Quantitative imaging in medicine and surgery* 4.6 (2014), p. 475.

[62] Pranoy Radhakrishnan. *What are Hyperparameters ? and How to tune the Hyperparameters in a Deep Neural Network ?* en. Oct. 2017. URL : https://towardsdatascience.com/what-are-hyperparameters-and-how-to-tune-the-hyperparameters-in-a-deep-neural-network-d0604917584a.

[63] UBKOV. *Tutoriel : Comment lire une courbe ROC et interpréter son AUC ?* fr-FR. Avr. 2021. URL : https://www.idbc.fr/tutoriel-comment-lire-une-courbe-roc-et-interpreter-son-auc/.

[64] *Best Python libraries for Machine Learning*. en-us. Section : Technical Scripter. Jan. 2019. URL : https://www.geeksforgeeks.org/best-python-libraries-for-machine-learning/.