République Algérienne Démocratique et Populaire

Ministère de l'Enseignement Supérieure et de la Recherche Scientifique

Université Abderrahmane Mira

Faculté de Technologie

Département d'Automatique, Télécommunication et d'Electronique

Projet de Fin d'Etudes

Pour l'obtention du diplôme de Master

Filière : Télécommunications.

Spécialité : Réseaux des télécommunications.

Thème

Identification de malaria par apprentissage profond

Préparé par :

BENSLIMANE Baya

HARAOUI Meriem

Dirigé par :

Mr R. KASMI

Mlle.MC BENABDELHAK

Examiné par :

Mr. M. SADJI

Mme GHENNAM

Année universitaire : **2022/2023**

Remerciements

Avant tout, nous remercions le bon DIEU de nous avoir donné la santé, le courage et la capacité pour mener ce travail.

De tout cœur nous exprimons nos profondes gratitudes aux membres de nos familles, nos parents, nos frères, sœurs et petits neveux, pour leur soutien tout au long de notre parcours.

Nous voudrions tout d'abord adresser toutes nos reconnaissances à notre encadrant docteur Kasmi Réda, d'avoir acceptée de nous encadrer dans ce travail de thèse et pour sa patience, sa disponibilité et son suivie pendant ce travail, et on tient à le remercier encore une fois pour la confiance qu'il nous a fait. Et pour tout le soutien, l'aide, l'orientation, la guidance, ainsi que pour ses précieux conseils et ses encouragements lors de la réalisation de notre mémoire. On a beaucoup appris à son contact et ce fut un grand plaisir de travailler avec lui.

Nous tenons à exprimer nos sentiments de reconnaissance aux co-encadrantes madame Benabdelhak Meriem Chahinez et madame Bensadi Lilia. Qui nous ont appris une infinité de choses et qui nous ont vraiment aidés.

Nous tenons également à remercier les membres du jury d'avoir consacré une partie de leur temps à examiner ce mémoire, pour l'intérêt qu'ils ont porté à notre travail et pour leurs contributions à l'enrichir.

Nous adressons nos sincères remerciements à tous ceux qui ont contribuée de diverses manières à l'aboutissement de ce travail.

En dernier lieu, nous pensons à tous nos amis qui nous ont soutenu d'une manière constante et aux personnes chères à nos yeux qui veillent sur nous tout là-haut.

Dédicace

Avec tous mes sentiments de respect, avec l'expérience de ma reconnaissance j'offre ce mémoire :

À l'être la plus chère de ma vie, à la source de ma joie et mon bonheur, au fil d'espoir qui allume mon chemin, ma mère KHOUKHA.

A celui qui a fait de moi une femme, à mon pilier, ma source de vie, à mon support qui a était toujours à mes côtés mon père MOHAMED.

Je dédie ce travail À ma grande sœur, à ma moitié DIHIA, qui a été toujours là avec moi, et qui n'a jamais cessé de me conseiller, qui m'a toujours soutenu au long de mes études et c'est grâce à elle que J'ai eu mon baccalauréat.

Je l'adresse aussi à mon grand frère KOUCEILA, pour l'amour qu'il me réserve.

Aucune dédicace ne saurait exprimer mon amour éternel, ma reconnaissance et ma considération pour les sacrifices que vous avez consenti pour mon éducation et notre bienêtre.

Je vous remercie pour tout le soutien et l'amour que vous m'avez porté depuis mon enfance et j'espère que votre bénédiction m'accompagnera toujours.et j'espère aussi que je vous ai rendu fière de moi.

Que ce modeste travail soit l'exaucement de vos vœux tant formulés, le fruit de vos innombrables sacrifices, bien que je ne vous en acquitte jamais assez.

Puisse ALLAH, le très haut, vous accorde la santé, le bonheur et longue vie et faire en sorte que jamais je ne vous décevrai.

Je le dédie aussi à mon encadreur Mr Kasmi Réda ainsi que les doctorantes pour tout laide, l'orientation, leurs suivis pendant ce travail et à leurs confiances qu'ils nous ont accorder.

Je le dédie aussi à mes deux professeurs aimés Mr HADJI SLIMANE et Mr ALLICHE ABEDENOUR

Sans oublie ma binôme MERIEM pour son soutien moral et sa compréhension tout au long de ce travail.

Je l'adresse aussi à mon cher ami proche MOUNIR et à mes amies SISSA, KENZA, AMINA et IMENE. Et a tous mes chers collègues de ma promo.

- Baya -

Dédicace

Je dédie ce modeste travail

A mes très chers parents TAHAR et FATMA

Pour leur amour, conseils et leurs sacrifices pour ma réussite. Quoi que je fasse ou je dise je ne saurai point remercier comme il se doit. Votre affection me couvre, votre bienveillance me guide et votre présence a mes cotes a toujours été ma force. Que Dieu vous protège pour moi.

A mon très cher époux NAZIM ALI

Tu as toujours été à mes cotes pour me soutenir et m'encourager. Je te souhaite que de la réussite dans ta vie, que ce travail traduit ma gratitude et mon affection. Dieu te garde pour moi.

A ma chère petite princesse MAELLE GHANIA

Tu es ma source de bonheur et de force je t'aime énormément. Que Dieu te protège ma chère fille.

A mes très cher(es) frères et sœurs

Vous n'avez jamais cessé d'être là pour moi, de me soutenir et de m'épauler. Dieu vous préserve et vous garde pour moi.

A mes cher(es) neveux et nièces

MELINA, MAELLE, NOAM, YOUNES, INES, THIZIRI, BADIS, LYNDA, AYOUB, DJESSIM, MELINA, IBRAHIM, AMIR, NELYA. Que dieu vous protèges.

A mes chers beaux frères

A ma belle famille

En particulier à ma chère belle-mère GHANIA qui mas toujours encourager pour mes études, que Dieu t'accueille dans son vaste paradis.

A mon encadrant Mr KASMI REDA

A mes cher(es) amis

A mon binôme Baya merci pour ton soutien et ta compréhension tout au long de ce travail, j'adresse aussi ce travail a mes chers amis SALAH, YANIS, TINO, MOUNA, KHATMA,

KOKO, KATIA, IMENE, ASMA. Et à tous ceux et celles qui mont accompagne et soutenue durant ce parcours.

- Meriem -

Sommaire

Sommaire

Chapitre III

Implémentation et résultat

Liste d'abréviation

ADAM : Adaptive Moment Estimation

AVG : Average pooling

CNN : Convolutional Neural Networks

DL : Deep Learning

DNN : Deep Neural Networks

FC : Fully Connected

IA : Intelligence Artificielle

LeNet : Convolutional Neural Network

ML : Machine Learning

PMC : Pool Maintenance

Relu : Rectified Linear Unit

ResNet : Residual Networks

RMSP rop : Root Mean Square Propagation

RNN : Recurrent Neural Networks

UCLA : Université Californie à Los Angeles

VIH : Virus Immunodéficience Humaine

ZFNet : Visualizing and Understanding Convolutional Network

Liste des tableaux

Liste des figures

Introduction Générale

La malaria est une maladie très dangereuse qui se produit quand les globules rouges sont infectés par le Plasmodium, un parasite qui est transmis par les moustiques femelles de la famille anophèles lorsqu'elles se nourrissent du sang. Selon les données de l'Organisation Mondiale de la Santé, l'évolution biologique du parasite responsable de la fièvre paludéenne a entraîné une augmentation des cas d'infection, qui se chiffrent désormais à trois millions par an avec un taux de mortalité d'environ quatre cent mille cas par an.

La détection de malaria nécessite un personnel qualifié qui est reste peu sur le terrain, spécialement dans les villages et les régions éloignées. En outre, les tests sont longs et fastidieux [1].

L'objectif de cette recherche est la détection des cellules, globules rouges, infectées par la malaria on se basant sur les techniques d'apprentissage profond (Deep Learning). Cette méthode fait partie de l'apprentissage machine et se fonde sur un réseau neuronal multicouche, capable d'assimiler de manière autonome des représentations de données complexes. Donc on se basant sur le réseau neuronal : Convulutional Neural Network (CNN), on propose de classifier automatiquement les cellules infectées par la malaria en se basant sur des images microscopiques des cellules du sang.

Le travail présenté dans ce mémoire est divisé en trois parties structurées de la manière suivante :

Le premier chapitre portera sur quelques définitions sur la maladie du paludisme en général et quelques notions de base sur le Deep learning.

Au cours du deuxième chapitre, nous allons exposer les réseaux de neurones, en mettant l'accent sur les réseaux de neurones convolutifs (CNN). Les réseaux CNN ont la faculté d'extraire les caractéristiques des images présentées en entrée et de les classifier.

Dans le troisième chapitre, nous allons décrire notre application qui est un système basé sur le CNN qui a pour but de classifier les cellules du sang infectées des cellules saines.

Enfin, nous finirons par une conclusion générale qui récapitule notre travail ainsi les résultats obtenus et nous examinerons les possibilités à venir.

Chapitre I

Généralité

Introduction

La malaria, est une maladie infectieuse aiguë causée par un parasite appelé Plasmodium transmis par une piqûre de moustique appelé Anophèle. Elle demeure l'un des problèmes de santé les plus répandus dans le monde [2], Certains types de malaria peuvent entraîner une maladie grave et mortelle. Les nourrissons, les enfants de moins de 5 ans, les femmes enceintes, les personnes vivant avec le VIH ou le sida sont plus susceptibles d'être touchés.

La figure I.1 représente une image microscopique des cellules du sang, globules rouges, infectées par la malaria.

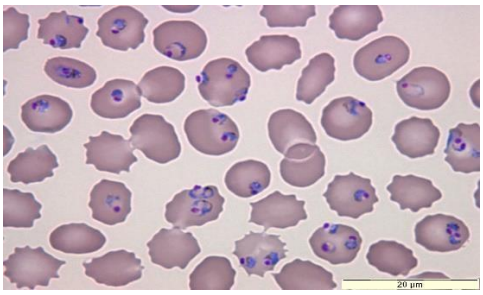

Figure I-1 : *Cellules infectés par le Paludisme*

I.1. Différents types de malaria

Il existe différents types de malaria :

a) La malaria falciparum (fièvre tropicale)

La malaria falciparum représente la variante la plus grave de toutes les formes de malaria. Elle se manifeste par une évolution soudaine, une fièvre généralement irrégulière, une anémie, une hypertrophie du foie et de la rate, une forte présence de parasites dans le sang, une variété de symptômes cliniques et des conséquences fâcheuses pouvant conduire à la mort. La malaria falciparum est à l'origine de 98% des décès liés à la malaria. **[2]**

b) Paludisme à vivax

Les caractéristiques distinctives de la malaria à vivax incluent, une maladie prolongée, de la fièvre intermittente qui se produit toutes les 48 heures ou quotidiennement, de la faiblesse et une tendance à la rechute qui se manifeste par des épisodes récurrents après une phase latente de plusieurs mois (3, 6, 12).

La progression de la maladie est généralement légère, mais chez les enfants et les

personnes non immunisées, la malaria à vivax peut parfois prendre une forme assez sévère. Les décès sont extrêmement rares et sont plutôt liés à des complications graves survenant en même temps. [2]

c) Malaria à ovale

Malaria à ovale est l'une des quatre variétés d'infection engendrées par les plasmodies du même nom. Elle est définie par une évolution répétitive, une fièvre intermittente de type tierce ou quotidienne, des accès fébriles en soirée et une anémie. C'est une maladie inoffensive qui a tendance à guérir spontanément après un nombre limité de crises. **[2]**

d) Paludisme à malariae (fièvre quarte)

Elle est identifiée par une fièvre intermittente qui survient toutes les quatre journées, une anémie légèrement marquée et une faible quantité de parasites dans le sang. Sa propagation est fréquente dans les zones tropicales et les états d'Afrique, d'Asie et d'Amérique latine. Toutefois, son taux de prédominance est relativement bas. **[2]**

I.2. Détection de malaria

La détection de malaria exige un personnel qualifié qui sont appelés, dans certains cas, à se déplacer dans des villages éloignés, en outre, les tests sont longs et fastidieux. Cela nécessite des outils d'aide au diagnostic pour accélérer l'opération de dépistage. Les outils d'aide au diagnostic se basent sur les techniques du traitement d'images d'extraction de caractéristiques et classification. Les techniques d'apprentissage profond arrivent à extraire des caractéristiques très intéressantes pour la classification.

I.3. Définition de l'apprentissage profond (Deep learning)

L'apprentissage profond est un sous domaine de l'intelligence artificielle qui reproduit le comportement du cerveau humain. C'est une des variantes de l'apprentissage automatisé qui peut être employé pour dans de nombreuses applications telles que la reconnaissance d'objets, la reconnaissance vocale, la biométrie, la traduction de langues et la prise de résolutions. Les programmes qui utilisent les techniques de l'apprentissage profond sont capables d'acquérir des connaissances en se basant sur des informations qui sont étiquetées. **[3]**

Figure I.2 représente un schéma du rapport entre IA, ML et DL

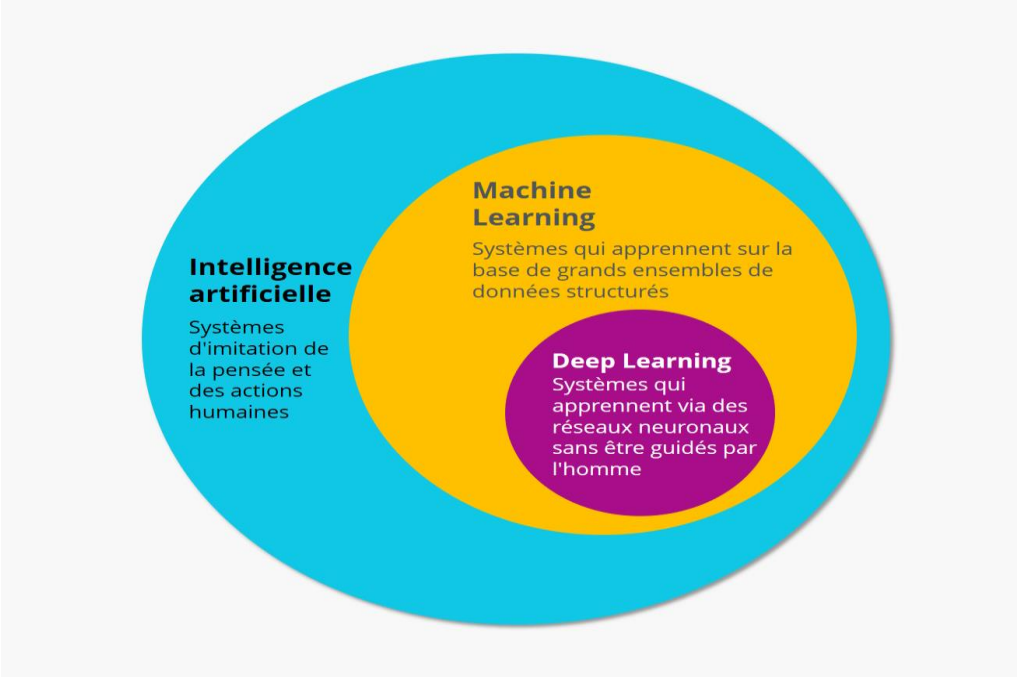

Figure 0-1 : *Schématisation du rapport entre IA, ML et DL*

I.4. Domaines d'application de l'apprentissage profond

Les domaines d'application de l'apprentissage profond sont très varié tel que :

I.4.1. Conduite automatisée

Les experts de l'industrie automobile utilisent l'apprentissage profond pour identifier automatiquement des éléments tels que les panneaux d'arrêt et les feux de signalisation. Ils ont également recours à cette méthode pour repérer les piétons et ainsi réduire le nombre d'accidents. **[4]**

I.4.2. Aérospatiale et défense

Le Deep Learning est utilisé pour la reconnaissance d'objets à partir de satellites, dans le but de repérer des zones d'intérêt et de déterminer les secteurs sécurisés ou risqués pour les troupes au sol. **[4]**

I.4.3. Recherche médicale

Le Deep Learning est également dans le domaine médical, les spécialistes en oncologie ont la capacité de détecter de façon automatisée les cellules malignes. Plusieurs

chercheurs ont utilisé le Deep Learning pour identifier de manière exacte les cellules cancéreuses. **[4]**

I.4.4. Automatisation industrielle

Le Deep Learning contribue à renforcer la sécurité des travailleurs qui opèrent à proximité de machines lourdes, en identifiant de manière automatique les circonstances où l'écart de sécurité entre les employés et les machines. **[4]**

I.4.5. Reconnaissance vocale

Le Deep Learning est employé pour la détection sonore et la reconnaissance vocale **[4]**

I.5. Les différents algorithmes de Deep Learning

Il existe différentes sortes d'algorithmes de Deep Learning, chacun de ces algorithmes à ses propres caractéristiques et utilisations, On peut citer par exemple :

I.5.1. Convolutional Neural Networks (CNN)

Appelé ConvNets, Il se base sur la convolution avec plusieurs filtres pour extraire des caractéristiques des objets étudiés. Ils sont utilisés pour l'identification et la localisation d'objets. **[5]**

I.5.2. Recurrent Neural Networks (RNN)

Ou réseaux de neurones récurrents sont des modèles d'apprentissage automatique performants qui permettent l'analyse de séquences de données, notamment de la parole, du texte ou des séries chronologiques. Ces réseaux autorisent les machines à "mémoriser" les informations antérieures et les utiliser pour des prises de décisions en temps réel. **[6]**

I.5.3. Deep Neural Networks (DNN)

Les réseaux de neurones profonds ressemblent aux réseaux perceptron multicouche MLP, mais ils possèdent davantage de couches cachées. L'accroissement du nombre de couches permet à un réseau de neurones de repérer des variations minimes dans le modèl d'apprentissage, ce qui peut favoriser le sur-apprentissage ou le sur-apprentissage. **[7]**

I.6. Les avantages du Deep Learning

- ✓ **Des résultats de grandes qualités :** Le principal avantage de cette technologie de l'intelligence artificielle est sans doute la qualité des résultats qu'il permet d'obtenir. Les

autres techniques d'extraction de caractéristiques automatique ne sont pas en mesure de produire des résultats aussi satisfaisants. Dans des domaines tels que le traitement d'images ou la reconnaissance faciale, cette forme d'IA est préférée aux autres types d'intelligence artificielle **[8]**

- ✓ **Une exécution optimale et stricte des tâches habituelles :** Du fait qu'il repose sur une assimilation habituelle, l'apprentissage profond ne montre aucun signe d'épuisement et assure constamment des performances élevées. **[8]**

- ✓ **La capacité de gérer les informations non organisées :** Contrairement à d'autres systèmes d'IA tels que l'apprentissage automatique, l'apprentissage profond est capable de traiter des données non structurées telles que des documents, des photos ou des courriels. Il est donc beaucoup plus intéressant que les technologies de l'IA qui se limitent uniquement au traitement des données structurées (carte bancaire, adresse, contacts téléphoniques…). **[8]**

Conclusion

Dans ce chapitre nous avons illustré le paludisme et sa dangerosité et montré la nécessite de la création d'un outil d'aide au diagnostic. Puis, nous avons défini le deep learning et introduit les différents types d'algorithmes.

.

Chapitre II

Réseau de neurone convolutif (CNN)

II.1. Définition du CNN

Les réseaux de neurones convolutifs, connus sous le nom de ConvNets, sont couramment utilisés dans le domaine du Deep Learning, ils sont composés d'un grand nombrede couches qui ont pour rôle de traiter et d'extraire les propriétés des données. De façon particulière, les ConvNets sont utilisés pour l'identification et la localisation d'objets. Par conséquent, ils peuvent être employés pour identifier des images de satellites, analyser des images médicales, détecter des anomalies ou anticiper des séries chronologiques... **[9]**

L'architecture CNN est constituée d'un certain nombre de couches comme le montre la figure II-1 qui est un exemple CNN avec 5 couches et deux couches entièrement connectées.

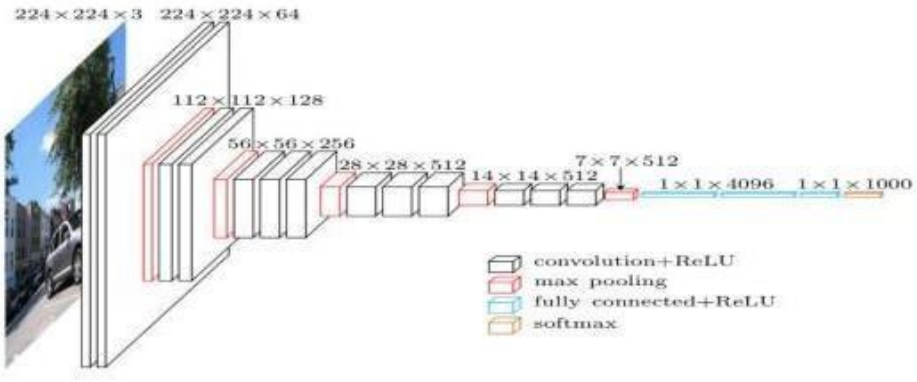

Figure III-1 : *Couche CNN (16)*

II.2. Les différentes couches de CNN

Un réseau de neurones convolutifs possède différents types de couches :

- La couche de convolution
- La couche de pooling
- Fonction Unité Linéaire Rectifiée (ReLu)
- La couche dense (dense layer)
- Couches entièrement connectée (FC : Fully Connected)

II.2.1. La couche de convolution

La couche de convolution est l'élément crucial des réseaux de neurones convolutifs. Son objectif est de détecter la présence d'un groupe de caractéristiques dans les images qui sont transmises à l'entrée de la couche suivante. En effet, une convolution de l'image avec différents filtres permet de mettre en évidence des caractéristiques des objets dans l'image (lignes, coins,…). **[16]** L'équation de convolution entre une image I est un filtre H :

$$Y[M,N] = I[M,N] * H[M,N] = \sum_{-N/2}^{\frac{N}{2}-1} \sum_{M/2}^{\frac{M}{2}-1} I[i,j]\, H[M - i, N - j] \quad (2.1)$$

Figure II.2 montre le principe de la convolution d'une image par un filtre.

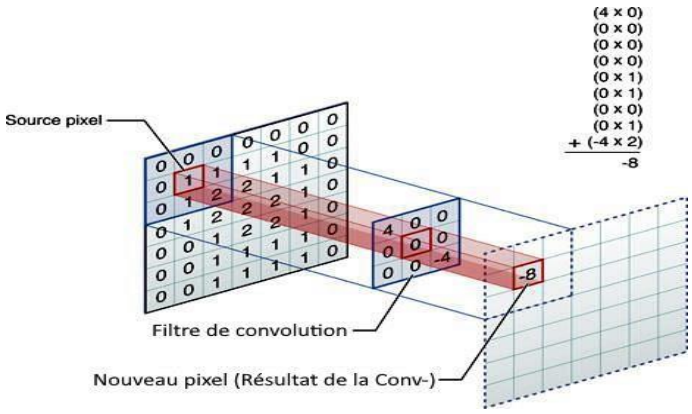

Figure III-2 : *Convolution d'une image avec un filtre 3x3.*

- **Les paramètres de convolution dans une architecture CNN**

Les réseaux de neurones convolutifs (CNN) emploient davantage de paramètres qu'un perceptron multicouche classique, donc il convient de tenir compte du nombre de filtres, de leur configuration ainsi que de celle du maxpooling.

- **Profondeur de la couche :** Correspond au nombre de filtres utilisés pour chaque couche.

- **Le pas (Stride) :** Le nombre de pixels par lesquels la fenêtre de convolution se déplace. Si le pas est réduit, les zones de convolution se superposent davantage, ce qui entraîne une intensification du volume de sortie.

- **La marge à zéro (zero padding) : cette** opération gère la dimension spatiale de

l'image de sortie. Elle consiste à ajouter des zéros aux bords de l'image d'entrée. Cette extension permet au résultat de convolution (image de sortie) d'avoir la même dimension d'image d'entrée. **[17]**

- **Nombre de filtres :** plus le traitement avance en profondeur, les images intermédiaires diminuent en taille, ce qui fait, le nombre de filtre augmente de plus en plus que les couches se rapprochent de la sortie. Ainsi, le nombre de caractéristiques et du nombre de pixels traités est maintenu à peu près constant d'une couche à l'autre.

➢ **Dimensions de l'image caractéristique**

La dimension de l'image à la sortie de la convolution dépend du pas de déplacement S (Stride), la quantité de zéro-padding P, la dimension de l'image d'entrée *Din* et la taille du filtre F.

$$D_{out} = \frac{D_{in}-F+P_{Start}+P_{End}}{S} + 1 \qquad (2.2)$$

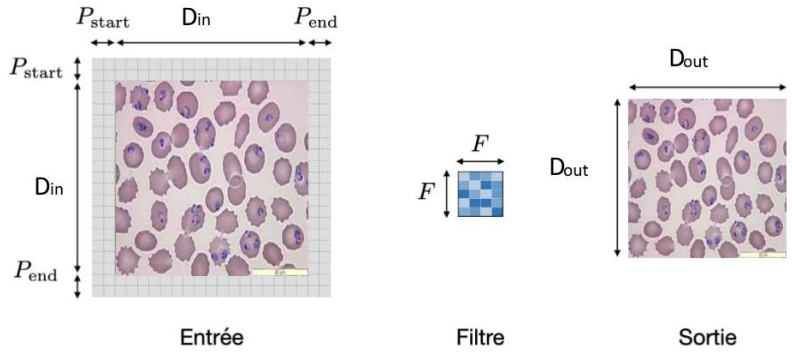

Figure III-3 : *Réduction de la dimension de l'image après convolution*

II.2.2. La couche de Pooling

Le sous-échantillonnage est une opération couramment effectuée après une couche de convolution, connue sous le nom de couche de pooling. Les types de pooling les plus couramment utilisés sont le max-pooling et average-pooling, qui sélectionnent respectivement les valeursmaximales et moyennes

- **Max-Pooling** : dans cette opération de pooling consiste à remplacer un bloc de pixels par leur maximum.

- **Average-Pooling** : Chaque opération de pooling choisi la valeur moyenne de chaque bloc de pixels. **[18]**

La Figure II.7 montre l'opération MaxPooling et Average-Pooling et la façon dont les dimensions de l'image sont réduites de moitié en choisissant des blocs de 2x2.

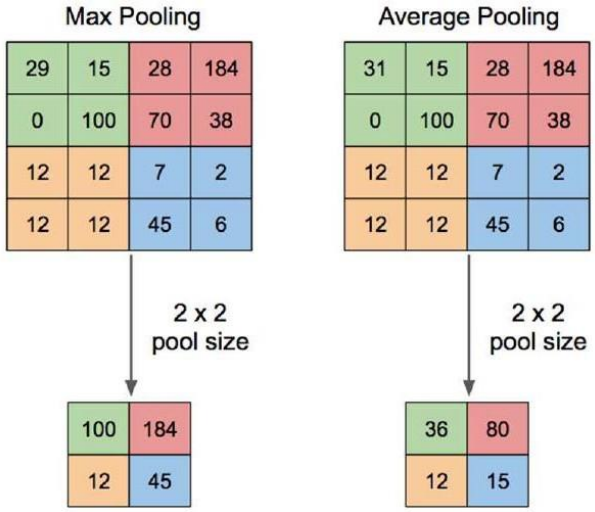

Figure III-4 : *Opération de Max-Pooling et Averag- Pooling*

La dimension des blocs peuvent être choisit en fonction des dimensions de l'image d'entrée. Pour des images larges, les blocs peuvent être choisit de dimensions plus grandes que 2x2. Les dimensions sont choit de sort à avoir un compromis entre la réduction de l'image par l'opération max-pooling et la conservation des caractéristiques de l'image.

II.2.3. Fonction d'activation

La fonction d'activation, suit directement la convolution. Ils existent plusieurs fonctions d'activation comme le montre la figure II.5

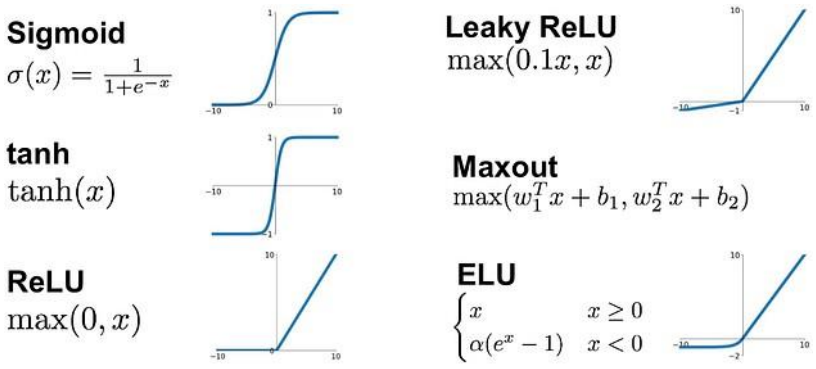

Figure III-5 : *Fonction d'activation*

II.2.4. La couche dense (dense layer)

C'est la couche qui vient juste avant le classificateur (couche complètement connectée). Cette couche permet de transformer les matrices caractéristiques d'entré (résultat des convolutions) en un vecteur.

II.2.5. La couche complètement connectée (FC : Fully Connected)

La couche complètement connectée (Fully Connected), où chaque entrée est reliée à tous les neurones, prend en entrée le vecteur caractéristique issu de la couche dense. **[19]**

FigureII.6 représente la couche complétement connecté FC.

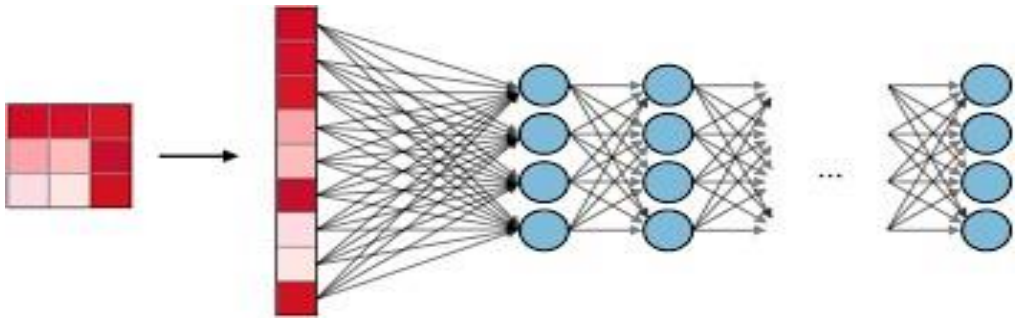

Figure III-61 : *La couche complètement connectée*

II.3. La fonction de coût

Mesure la similarité entre l'image générée $\hat{I}_l$ et l'image original $\boldsymbol{I_l}$ à la couche $\boldsymbol{l}$, noté J (Î, I) est défini :

$$J(\hat{I}, I) = \frac{1}{2} \|\hat{I}l - Il\| \quad (2.3)$$

II.3.1. Optimiseurs

Un optimiseur est essentiellement l'algorithme qui tente de contrôler la descente de gradient et la recherche du minimum dans la fonction de coût (Loss function). Le but d'un optimiseur est de faire converger la Descente de Gradient vers le minimum global en un minimum de temps. **[20]**

Figure II.7 représente le gradient descente.

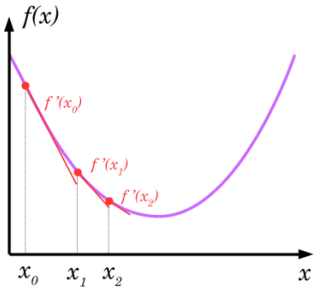

Figure III-2 : *Gradient descentePlusieurs optimiseurs sont proposés.*

- **Adadelta :** Est un optimiseur pour de diminuer l'effet de ralenti se produit au fil des itérations, Adadelta ne tient compte que des k derniers gradients pour calculer son facteur d'amortissement par paramètre. **[20]**

- **RMSProp :** est un optimiseur qui opte pour l'accélération de la descente de gradient, Il est employé dans le but de former des modèles fondés sur des réseaux de neurones profonds. **[20]**

- **Adam :** (Adaptive Moment Estimation) a intégré les améliorations d'Adadelta et RMSProp en incluant une gestion des moments de premier et de second ordre. Cet optimiseur est célèbre pour ses excellentes performances et sa capacité à converger rapidement. [**20]**

II.4. Les CNN connues

Parmi les réseaux convolutifs célèbres on trouve :

- **LeNet :** Il s'agit d'une architecture de réseaux de neurones convolutifs, employée pour la reconnaissance de codes postaux, de chiffres. **[10]**

- **GoogLeNet :** Son apport majeur a été la conception d'un module d'inception qui a significativement diminué le nombre de variables dans le réseau. Ce module utilise le Pooling global AVG ((Average Pooling calcule la valeur moyenne au lieu du PMC (Pooling Maximal Convolutionnels qui est le même que max pooling ou il prend la valeur maximal) a derrière couche, ce qui élimine beaucoup de paramètres.**[11]**

- **ResNet :** Residual network présente des sauts de connexion et une utilisation importante de la normalisation par lots. Il opte pour le pooling AVG global plutôt que le PMC en finde compte. **[12]**

Figure II.2 représente l'architecture ResNet

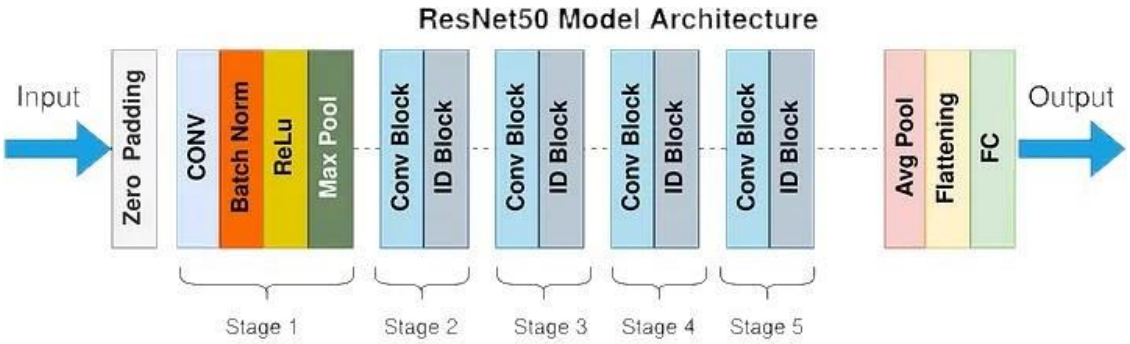

Figure III-8 : *ResNet architecture*

- **AlexNet**

AlexNet est un réseau neuronal convolutif de 8 couches de profondeur. La version pré- entraînée du réseau formé sur plus d'un million d'images à partir de la base de données ImageNet **[13]**.

Le réseau pré-entraîné peut classer les images en 1000 catégories d'objets. **[14]** Figure II.9 représente AlexNet architecture

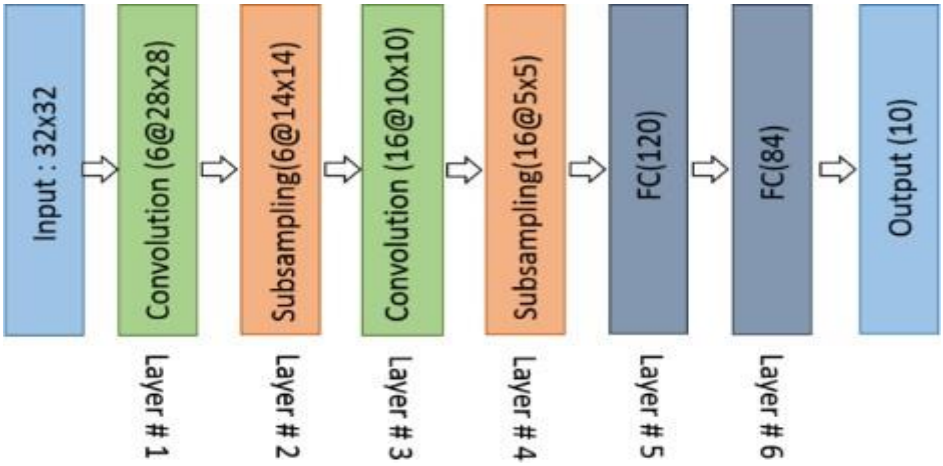

Figure III-9 : *AlexNet architecture*

- **ZFnet :** Il s'agit d'une évolution d'AlexNet obtenue par la modification des hyper-paramètres de l'architecture, notamment en augmentant la dimension des couches convolutives et en réduisant la taille du noyau pour la première couche. **[15]**

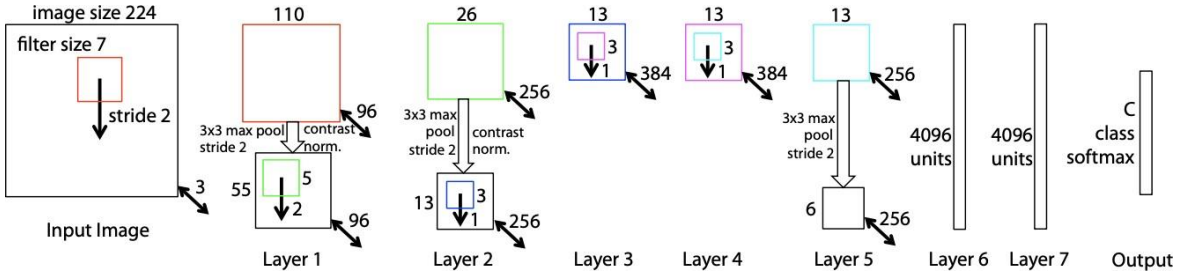

Figure III-10 : *ZFnet*

Conclusion

Au cours de ce chapitre, nous avons introduit les réseaux de neurones convolutifs. Ce dernier peut extraire des caractéristiques des images dans le but d'une classification.

Chapitre III

Implémentation et résultat

III.1. Introduction

Ce chapitre introduit l'architecture CNN proposée pour classifier les images de cellules de sang où les globules rouge sont infectés ou non infectée.

III.2. Implémentation

Pour l'implémentation nous avons utilisé le langage Python implémenté sur la plateforme virtuelle Kaggle destinée aux chercheurs Data Scientist.

Dans cette partie nous allons exposer les diverses phases de notre code source :

- Importation des bibliothèques nécessaires.
- Importation de la base donnée.
- Traitement des images de la base donnée.
- Affichage des courbes Loss et Accuray.

III.2.1. Importation des bibliothèques nécessaires

```
import numpy as np
from matplotlib import pyplot as plt
import numpy as np

import os
import cv2

import tensorflow as tf
```

- **Import numPy as np**

On utilise Cette instruction pour importer le package numpy ou toute ses fonctions seront préfixées par np.

- **from matplotlib import pyplot as plt**

C'est une médiathèque du langage de codage Python conçue pour dessiner et présenter des informations sous la forme de diagrammes.

- **Import os**

C'est une bibliothèque qui gère les opérations lié au système d'exploitation. Nous l'utilisant pour créer des liens.

- **Import cv2**

  Cette bibliothèque qui contient des fonctions de différent traitement d'images.

- **Import tensorflow as tf**

  Il gère les différents calcules liés aux matrices et tenseurs.

III.2.2. Choix des paramètres

L'image d'entrée est réduite si nécessaire vu le grand nombre de calcule. Donc il est important de choisir une dimension qui gardera les caractéristiques et réduire l'image pour optimiser les calcule.

- **Img-size = [150,150]**

C'est une variable qui met les images sur la taille qu'on souhaite et dans notre cas on a choisi [150,150] (taille originale des images).

- **Batch-size =32**

Batch size est utile pour mettre nos images sous forme de lots. Dans notre cas on a pris 32 images pour chaque lot, c'est-à-dire l'entrainement se fera sur des lots de 32 images. L'entrainement se fera plus rapidement sur des lots car la mise à jour des poids se fait à chaque propagation.

```
IMG_SIZE = [150, 150]
BATCH_SIZE = 32
#utiliser la fonction de tensorflow pour avoir les images à partir du lien(générateur)
data = tf.keras.preprocessing.image_dataset_from_directory('/kaggle/input/malaria-data/cell_images', shuffle=True, batch
_size=BATCH_SIZE, image_size=IMG_SIZE)

Found 27558 files belonging to 2 classes.
```

- **Data = tf. Keras.preprocessing.image_dataset_from_directory**

C'est une fonction de TensorFlow pour avoir directement les images à partir du lien ('/kaggle/input/malaria-data/cell_images',)

- **Shuffle =True**

C'est à dire que nos images des deux lésions (malaria et non malaria) seront mélangées pour permettre un bon entrainement

II. 2.3. Itérations des data et récupération des batch

```
data_iterator = data.as_numpy_iterator()
batch = data_iterator.next()
```

- **Data-itération =data.as-numpy-iterator ()**

Utilisé pour inspecter le contenu de la base de données, pour voir la forme et les types des éléments.

- **Batch = data-iterator. next ()**

Cette partie permet de récupérer les prochaines 32 lots d'images.

III.2.4. Affichages des images du batch

Avec cette instruction nous allons afficher quelques images du batch avec leurs étiquettes.

Chaque colonne de cette instruction contient 4 images et chaque image a une taille de [20,20] et une étiquette 0 pour malaria et 1 pour non malaria.

```
fig, ax = plt.subplots(ncols=4, figsize=(20,20))
for idx, img in enumerate(batch[0][:4]):
    ax[idx].imshow(img.astype(int))
    ax[idx].title.set_text(batch[1][idx])
```

- **fig, ax = plt. subplots (ncols=4, figsize= (20,20))**

Et on obtient :

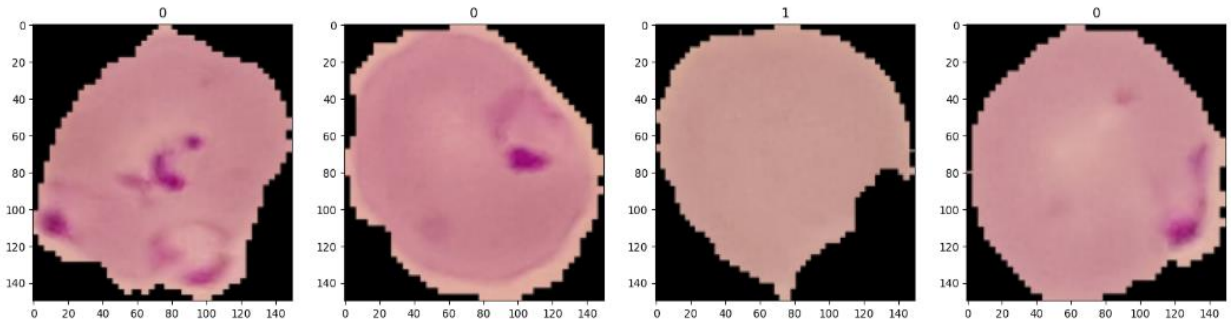

Figure III.1 :

III.2.5. Normalisation des images

Cette étape nous permet de normaliser les images en les devisant sur 255, les valeurs des pixels deviennent entre 0 et 1.

```
data = data.map(lambda x,y: (x/255, y))
```

III.2.6. Transformation en format tableau

Avec le code ci-dessus, nous avons converti les données sous format tableau.

```
data.as_numpy_iterator().next()
```

III.2.7. La division de la base de données

La base de données est deviser en trois sous base de données, 70% base d'entrainement, 20% de base de validation, et 10% de base de test.

```
train_size = int(len(data)*.7)
val_size = int(len(data)*.2)
test_size = int(len(data)*.1)
```

III.2.7. Utilisations des fonctions take et skip pour la division des images

On a utilisé les fonctions take et skip pour deviser les images et s'assurer de ne pas prendre les mêmes pour entrainement, validation et test.

```
train = data.take(train_size)
val = data.skip(train_size).take(val_size)
test = data.skip(train_size+val_size).take(test_size)
```

III. 2.8. Constructions de notre model

Séquentiel, est le moyen le plus simple de créer un modèle dans Keras. Il permet de construire un CNN couche par couche, séquence par séquence.

- **from tensorflow. keras.models import Sequential**
- **from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout**

- **Conv2 :** Convolution, on spécifie une dimension de fenêtre qui va se déplacer sur toute l'image.

- **MaxPooling2D :** est employé afin de diminuer les dimensions spatiales du volume de sortie.

- **Dense :** est Le contrôleur de couche permet à chaque couche de recevoir une entrée de chaque neurone de la couche précédente.

- **Flatten :** il aplatit les caractéristiques et les rend sous forme d'un vecteur.

- **Dropout:** il désactive quelques neurones dans la couche FC pour éviter un sur apprentissage.

➢ **Les couches**

- Dans cette étape nous avons mis 7 couches avec le nombre de filtres de 128, 256, 128, 64, 16, 32, 64 respectivement avec une dimension de (3,3) et de pool-size (3,3).

- Et on a choisi une fonction activation relu et un optimiseur ADAM.

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dense, Flatten, Dropout

model = tf.keras.models.Sequential()
model.add(tf.keras.layers.InputLayer(input_shape=(150,150,3)))

model.add(tf.keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(tf.keras.layers.Conv2D(256, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))


model.add(tf.keras.layers.Conv2D(128, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(tf.keras.layers.Dropout(0.25))
model.add(tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(tf.keras.layers.Conv2D(16, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(tf.keras.layers.Conv2D(32, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(tf.keras.layers.Conv2D(64, (3, 3), padding='same', activation='relu'))
model.add(tf.keras.layers.MaxPooling2D(pool_size=(2, 2)))

model.add(tf.keras.layers.Flatten())
model.add(tf.keras.layers.Dropout(0.3))
model.add(tf.keras.layers.Dense(128, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))

model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
model.summary()
```

Et après l'exécution on aura :

Notebook Input Output Logs Comments (0) Settings

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 128)     3584

 max_pooling2d (MaxPooling2D  (None, 75, 75, 128)      0
 )

 conv2d_1 (Conv2D)           (None, 75, 75, 256)       295168

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 256)      0
 2D)

 conv2d_2 (Conv2D)           (None, 37, 37, 128)       295040

 max_pooling2d_2 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 dropout (Dropout)           (None, 18, 18, 128)       0

 conv2d_3 (Conv2D)           (None, 18, 18, 64)        73792

 max_pooling2d_3 (MaxPooling  (None, 9, 9, 64)         0
 2D)

 conv2d_4 (Conv2D)           (None, 9, 9, 16)          9232

 max_pooling2d_4 (MaxPooling  (None, 4, 4, 16)         0
 2D)
```

```
max_pooling2d_3 (MaxPooling  (None, 9, 9, 64)         0
2D)

conv2d_4 (Conv2D)            (None, 9, 9, 16)         9232

max_pooling2d_4 (MaxPooling  (None, 4, 4, 16)         0
2D)

conv2d_5 (Conv2D)            (None, 4, 4, 32)         4640

max_pooling2d_5 (MaxPooling  (None, 2, 2, 32)         0
2D)

conv2d_6 (Conv2D)            (None, 2, 2, 64)         18496

max_pooling2d_6 (MaxPooling  (None, 1, 1, 64)         0
2D)

flatten (Flatten)            (None, 64)               0

dropout_1 (Dropout)          (None, 64)               0

dense (Dense)                (None, 128)              8320

dense_1 (Dense)              (None, 1)                129

=================================================================
Total params: 708,401
Trainable params: 708,401
Non-trainable params: 0
_________________________________________________________________
```

- La figure ci-dessus nous présente les différentes couches de notre architecture de CNN ainsi que leurs paramètres.
- Les résultats 0 signifie qu'elle ne retourne pas de paramètres.
- Le total de paramètres d'après les résultats obtenu est 708401.

III.3. Entrainement du model

Dans cette partie on a entrainé notre model avec ls sous-base de données 'train', nombre d'époque et la validation et on obtient :

```
hist = model.fit(train, epochs=20, validation_data=val)
```

```
603/603 [==============================] - 67s 111ms/step - loss: 0.1253 - accuracy: 0.9607 - val_loss: 0.1303 - val_accuracy: 0.9575
Epoch 9/20
603/603 [==============================] - 85s 141ms/step - loss: 0.1230 - accuracy: 0.9612 - val_loss: 0.1239 - val_accuracy: 0.9611
Epoch 10/20
603/603 [==============================] - 85s 141ms/step - loss: 0.1193 - accuracy: 0.9628 - val_loss: 0.1288 - val_accuracy: 0.9560
Epoch 11/20
603/603 [==============================] - 85s 141ms/step - loss: 0.1149 - accuracy: 0.9636 - val_loss: 0.1176 - val_accuracy: 0.9602
Epoch 12/20
603/603 [==============================] - 85s 141ms/step - loss: 0.1138 - accuracy: 0.9635 - val_loss: 0.1190 - val_accuracy: 0.9622
Epoch 13/20
603/603 [==============================] - 69s 115ms/step - loss: 0.1101 - accuracy: 0.9639 - val_loss: 0.1144 - val_accuracy: 0.9628
Epoch 14/20
603/603 [==============================] - 69s 114ms/step - loss: 0.1068 - accuracy: 0.9650 - val_loss: 0.1107 - val_accuracy: 0.9617
Epoch 15/20
603/603 [==============================] - 68s 112ms/step - loss: 0.1017 - accuracy: 0.9655 - val_loss: 0.1230 - val_accuracy: 0.9597
Epoch 16/20
603/603 [==============================] - 85s 141ms/step - loss: 0.1001 - accuracy: 0.9647 - val_loss: 0.1106 - val_accuracy: 0.9615
Epoch 17/20
603/603 [==============================] - 68s 112ms/step - loss: 0.0966 - accuracy: 0.9662 - val_loss: 0.1105 - val_accuracy: 0.9622
Epoch 18/20
603/603 [==============================] - 85s 141ms/step - loss: 0.0934 - accuracy: 0.9679 - val_loss: 0.1061 - val_accuracy: 0.9615
Epoch 19/20
603/603 [==============================] - 68s 112ms/step - loss: 0.0933 - accuracy: 0.9678 - val_loss: 0.1052 - val_accuracy: 0.9624
Epoch 20/20
603/603 [==============================] - 85s 141ms/step - loss: 0.0912 - accuracy: 0.9686 - val_loss: 0.1092 - val_accuracy: 0.9620
```

Taux de pression (val accuracy) est 0.9620 donc 96,20 %

Taux d'erreur (val loss) est de 0.1092 donc 10.92 %

III.4. Affichage des courbe loss et accuracy

Loss

```
fig = plt.figure()
plt.plot(hist.history['loss'], color='teal', label='loss')
plt.plot(hist.history['val_loss'], color='orange', label='val_loss')
fig.suptitle('Loss', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

Accuracy

```
fig = plt.figure()
plt.plot(hist.history['accuracy'], color='teal', label='accuracy')
plt.plot(hist.history['val_accuracy'], color='orange', label='val_accuracy')
fig.suptitle('Accuracy', fontsize=20)
plt.legend(loc="upper left")
plt.show()
```

Ces deux instructions nous permettent d'afficher les deux courbes de loss et accuracy.

FIGURES ?

III.5. Tests et résultats

III.5.1. La base donnée

Nous avons utilisé une base de données avec une dimension de (150 ,150,3) qui contient 27558 images appartenant à 2 classes (infectées et non infectées). Parmi ces images on a 19296 images d'entrainements, 5504 ensembles de validation et 2752 de test.

L'image suivante nous montre quelque image infectée (Parasitized) et non infectée (Uninfected)

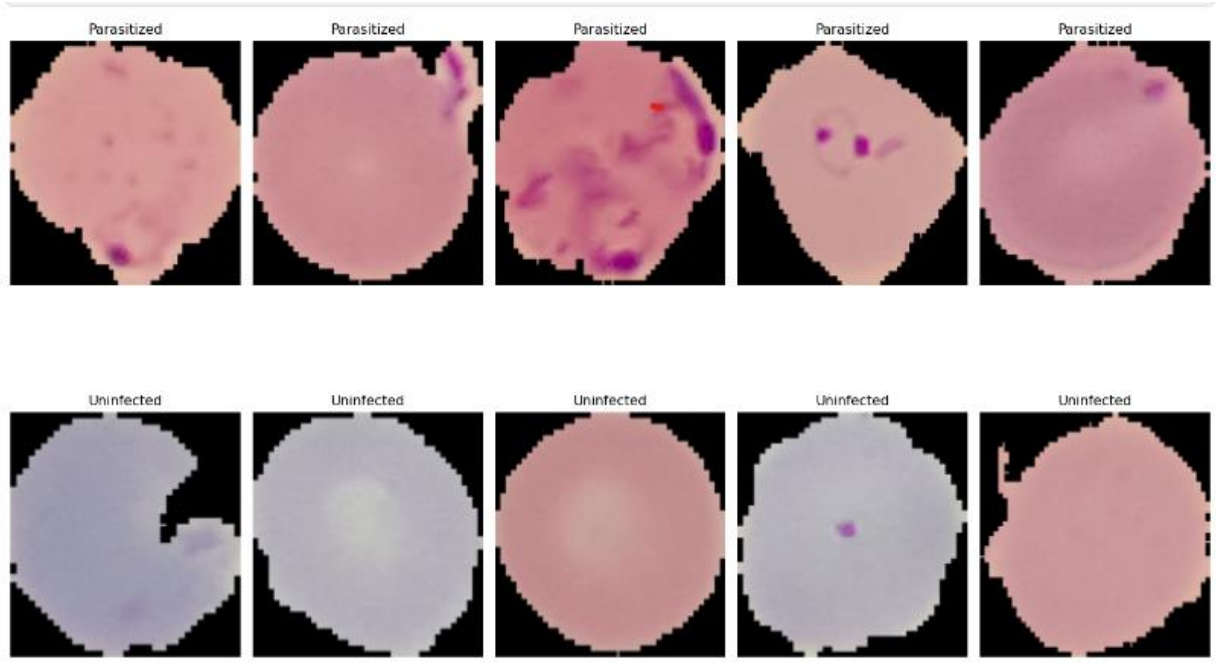

Figure II.2 :

III.1. Les métriques d'évaluations d'un modèle de classification

III.1.1. Accuracy

L'accuracy est une mesure d'évaluation de la performance des modèles de classification à deux classes ou plus qui indique le pourcentage de prédictions correctes. Elle est également le rapport entre le nombre de prédictions correctes et le nombre total de prédictions des individus positifs et négatifs.

Elle se définit comme suit :

$$\text{Accuracy}= \frac{TP+TN}{TP+TN+FP+FN} \qquad (3.)$$

Où :

- True positive (TP) : on prédit vrai et la réponse est vraie
- True negative (TN) : On prédit faux et la réponse est fausse
- False positive (FP) : On prédit vrai et la réponse est fausse
- False negative (FN) : On prédit faux et la réponse est vraie

Et Pour compléter l'accuracy, on calcule également le Recall

III.1.2. Métrique Recall

C'est un paramètre qui se focalise exclusivement sur les consommateurs qui ont effectivement annulé leur abonnement et fournit une évaluation de la proportion de résultats erronés négatifs. Il permet également une estimation du pourcentage de résultats positifs correctement prévus par le modèle.

Recall se donne sous la formule suivante :

$$\text{Recall}=\frac{TP}{TP+FN} \quad (3.n)$$

III.1.3. Métrique précision

La précision est une mesure complémentaire de l'accuracy et recall,

Elle se définie par :

$$\text{Précision}=\frac{\text{TP}}{\text{TP+FP}} \quad (3.n)$$

III.1.4. Le F1-score

Le score F1 correspond à la moyenne harmonique de la précision et du recall.

Il est ainsi déterminé par la formule suivante :

$$\text{F1}-\text{score}=2\times\frac{\text{recall}\times\text{precision}}{\text{recall}+\text{precision}} \quad (3.n)$$

III.1.5. La spécificité

Définie le pourcentage de négatifs bien prédits

Il se donne sous forme :

$$\text{Specificity} = \frac{TN}{TN+FP}$$

III.1.6. Métrique Loss

Loss est une fonction de perte/coût, qui est employée pour la classification binaire ou la classification multi-classes impliquant plusieurs étiquettes ou labels en sortie. Loss peut également être exploité pour la segmentation sémantique.

III.2. Test selon le nombre de couche

Evaluation des résultats en fonction de nombre de couches de CNN

Maintenant nous allons tester les couches de 1 jusqu'à 7 afin de trouver le meilleur résultat.

Tableau IV-1: *Représente les valeur d'accuracy et loss en fonction de nombre de couche*

Nombre de couche	Nombre de filtre	Loss	Accuracy
1 couche	128	Train loss 0.0294 Val loss 0.4540	Train accuracy 0.9910 Val accuracy 0.8959
2 couches	128 256	Train loss 0.1375 Val loss 0.1423	Train accuracy 0.9539 Val accuracy 0.9589
3 couches	128 256 128	Train loss 0.0450 Val loss 0.2378	Train accuracy 0.9845 Val accuracy 0.9511
4 couches	128 256 128 64	Train loss 0,0894 Val loss 0,1258	Train accuracy 0.9690 Val accuracy 0.9608
5 couches	128 256 128	Train loss 0.1100 Val loss 0.1259	Train accuracy 0.9608 Val accuracy 0.9588

	64 16		
6 couches	128 256 128 64 16 32	Train loss 0.1187 Val loss 0.1287	Train accuracy 0.9603 Val accuracy 0.9542
7 couches	128 256 128 64 16 32 64	Train loss 0.1112 Val loss 0.1213	Train accuracy 0.9626 Val accuracy 0.9609
8 couches	128 256 128 64 16 32 64 32	Train loss 0.1022 Val loss 0.1112	Train accuracy 0.9601 Val accuracy 0.9502

III.3. Test selon le nombre d'epoch

Après avoir testé les nombres de couches on passera aux nombres d'époch en gardant le nombre de couche qui donne le meilleur résultat.

Tableau IV-2: *Représente les résultats d'accuracy et loss en fonction de nombre d'epochs*

Nombre d'epochs	Loss	Accuracy

5	Train loss 0.6932 Val loss 0.6931	Train accuracy 0.4994 Val accuracy 0.5016
10	Train loss 0.1112 Val loss 0.1213	Train accuracy 0.9626 Val accuracy 0.9609
15	Train loss 0.0930 Val loss 0.1521	Train accuracy 0.9686 Val accuracy 0.9528
20	Train loss 0.0912 Val loss 0.1092	Train accuracy 0.9686 Val accuracy 0.9620
25	Train loss 0.6932 Val loss 0.6931	Train accuracy 0.4996 Val accuracy 0.5005
30	Train loss 0.0611 Val loss 0.1308	Train accuracy 0.9781 Val accuracy 0.9611
35	Train loss 0.6931 Val loss 0.6933	Train accuracy 0.5048 Val accuracy 0.4995
40	Train loss 0.0449 Val loss 0.1997	Train accuracy 0.9839 Val accuracy 0.9597
45	Train loss 0.6932 Val loss 0.6932	Train accuracy 0.4967 Val accuracy 0.5000
50	Train loss 0.0261 Val loss 0.1966	Train accuracy 0.9905 Val accuracy 0.9618

D’après les deux tableaux on constate que le meilleur résultat est avec 7 couches et 20 epochs et ont a obtenu les valeurs des métriques pour test et validation, comme le tableau ci de suite :

Résultats de validation :

Tableau IV-3: *Résultats des métriques de validation*

Accuracy	Précision	Recall	Specificity	F1
0.9622	0.9644	0.9594	0.9649	0.9619

Résultats du test:

Tableau IV-4: *Résultats des métriques de test*

Accuracy	Précision	Reccal	Specificity	F1
0.9622	0.9621	0.9592	0.9649	0.9606

Tableau IV-5: *Tableau représente l'architecture de Notre exemple CNN*

Input	Input (Li, Ci, Pi)	(150,150,3)
InputLayer	Output (Li,Ci,Pi)	(150,150,3)
Convolution	Input (TF)	(150,150,3)
Conv2D+Relu	Output (TF)	(150,150,128)
Pooling	Input (TF)	(150,150,128)
Maxpooling2D	Output (TF)	(75,75,128)
Convolution	Input (TF)	(75,75,128)
Conv2D+Relu	Output (TF)	(75,75,256)
Pooling	Input (TF)	(75,75,256)
Maxpooling2D	Output (TF)	(37,37,256)
Convolution	Input (TF)	(37,37,256)
Conv2D+Relu	Output (TF)	(37,37,128)
Pooling	Input (TF)	(37,37,128)
Maxpooling2D	Output (TF)	(18,18,128)
Dropout	Input (TF)	(18,18,128)
Dropout	Output (TF)	(18,18,128)
Convolution	Input (TF)	(18,18,128)
Conv2D+Relu	Output (TF)	(18,18,64)
Pooling	Input (TF)	(18,18,64)
Maxpooling2D	Output (TF)	(9,9,64)
Convolution	Input (TF)	(9,9,64)
Conv2D+Relu	Output (TF)	(9,9,16)
Pooling	Input (TF)	(9,9,16)
Maxpooling2D	Output (TF)	(4,4,16)
Convolution	Input (TF)	(4,4,16)
Conv2D+Relu	Output (TF)	(4,4,32)

Pooling	Input (TF)	(4,4,32)
Maxpooling2D	Output (TF)	(2,2,32)
Convolution	Input (TF)	(2,2,32)
Conv2D+Relu	Output (TF)	(2,2,64)
Pooling	Input (TF)	(2,2,64)
Maxpooling2D	Output (TF)	(1,1,64)
Flatten	Input (TF)	(1,1,64)
Flatten	Output (TF)	(64)
Dropout	Input	(64)
Dropout	Output	(64)
Dense	Input	(64)
Dense	Output	(128)
Dense	Input	(128)
Dense	Output	(1)

Où :

Li : nombre de lignes d'image d'entrée

Ci : nombre de colonnes d'image d'entrée

Pi : nombre de profondeur de l'image

TF : taille du filtre

Après avoir réalisé les tableaux on passe aux résultats des courbes accuracy loss

D'après les instructions citées en haut on a obtenu la courbe de loss comme si de suit :

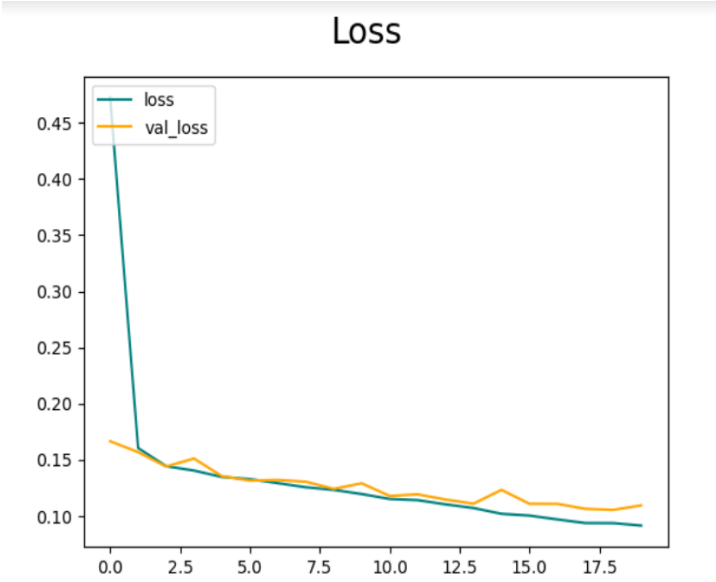

Figure IV-1 : *Représentation de loss en fonction d'epochs*

On remarque que les valeurs de loss diminuent avec l'augmentation du nombre d'epochs.

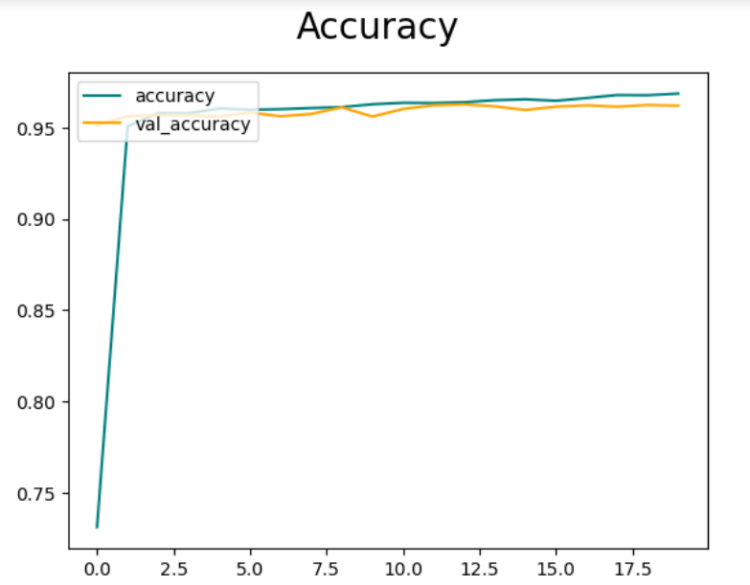

Figure IV-2 : *Représentation de l'accuracy en fonction d'epochs*

On remarque que les valeurs de l'accuracy augmentent avec l'augmentation d'epochs contrairement à loss.

Conclusion

Dans ce chapitre, nous avons présenté en premier lieu les métriques dévaluations ainsi que les métriques utilisées dans le programme.

Puis nous avons expliqué amplement notre étude. Et d'Après les résultats escomptés il s'est avéré que 0 correspond à la présence de malaria et le 1 pour le non malaria.

Et on conclut que l'identification de la malaria atteint un taux de classification de 96.20% d'après cette expérience.

Conclusion Générale

La malaria est une maladie très dangereuse et même mortelle qui se produit quand les globules rouges sont infectés par le plasmodium, sa détection chez les personnes atteintes par cette maladie peut diminuer le taux de mortalité et nous facilite de le détecter en un peu de temps contrairement aux tests qui sont longs.

L'objectif de cette étude est de pouvoir détecter cette maladie en employant les méthodes de CNN.

L'objectif principal de cette étude était de développer un modèle intelligent basé sur le Deep Learning pour détecter la malaria sur une base de données d'images des globules rouge du sang. Le but de l'utilisation de l'intelligence artificielle dans le domaine médical est la diminution detaux de mortalité et améliorer la qualité des soins.

En premier lieu, nous avons défini la malaria et ses différents types, puis nous avons procéder à la définition de l'apprentissage profond ainsi que ses domaines d'application et les différents algorithmes, au final nous avons cité quelques avantages du Deep Learning.

En deuxième lieu, nous avons commencé par nous avons commencé par, nous avons commencé par illustrer le CNN et citer ses célèbres réseaux de convolution, ensuite nous somme passer aux couches de CNN. Au final ont à clôturer ce chapitre par les optimiseurs.

En dernier lieu, La mise en œuvre a été effectuée via une plateforme web Kaggle. Ensuite le prototype a été examiné aléatoirement sur un groupe d'images et classifié comme étant contaminé ou non contaminé. Le prototype a montré une grande exactitude de 96,20 % lorsqu'il a été traité sur une partie des images, qui comprenait un ensemble de 27558 images.

Liste bibliographique

[1] M. GAUCHER. Encyclopédie médicale quillet.1965(288

[2] K. LOBAN et E. POLOZOK (1983). Le paludisme. Pervi Rijski péréoulok, Moscou, (101-147).

[3] [Alom et al., 2018] The history be- gan from alexnet : A comprehensive survey on deep learning approaches.

[4] https://fr.mathworks.com/discovery/deep-learning.html

[5] Alajrami, E., et al. (2019). "Blood Donation Prediction using Artificial Neural Network." International Journal of Academic Engineering Research (IJAER) 3(10): 1-7.

[6] https://datascientest.com/recurrent-neural-network

[7] https://datascientest.com/deep-neural-network

[8] https://www.jedha.co/formation-ia/algorithmes-deep-learning

[9] HUBEL David hunter, WIESEL Torsten nils : « Ferrier lecture-functional architecture of Macaque monkey visual cortex ». 1977.

[10] LECUN, Yann, *et al.* LeNet-5, convolutional neural networks. *URL: http://yann. lecun. com/exdb/lenet*, 2015, vol. 20, no 5, p. 14.

[11] SZEGEDY, Christian, LIU, Wei, JIA, Yangqing, *et al.* Going deeper with convolutions. In : *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015. p. 1-9.

[12] TARG, Sasha, ALMEIDA, Diogo, et LYMAN, Kevin. Resnet in resnet: Generalizing residual architectures. *arXiv preprint arXiv:1603.08029*, 2016.

[13] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg and Li Fei-Fei. **ImageNet Large Scale Visual Recognition Challenge**. *IJCV*, 2015

[14] KRIZHEVSKY, Alex, SUTSKEVER, Ilya, et HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 2017, vol. 60, no 6, p. 84-90

[15] ZEILER, Matthew D. et FERGUS, Rob. Visualizing and understanding convolutional networks. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer International Publishing, 2014. p. 818-833.

[16] Alex Krizhevsky,Ilya Sutskever and Geoffrey E. Hinton,ImageNet Classification

with Deep Convolutional Neural Networks,2012

[17] Pierre Buyssens, Fusion de différents modes de capture pour la reconnaissance du visage Appliquée aux e_ transactions DOCTORAT de l'UNIVERSIT´E de CAEN Le 4 Janvier 2011

[18] https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels#layer

[19] https://stanford.edu/~shervine/l/fr/teaching/cs-230/pense-bete-reseaux-neurones-convolutionnels

[20] https://perso.esiee.fr/~buzerl/IA/70%20optimisers/optimisers.html

Résume

Le paludisme est une maladie très dangereuse. Dans ce mémoire nous nous somme intéresser à une chaine de traitement d'image microscopique afin de les classifier en lésion infectés ou non infectés.

Le but de cette étude est de détecter le paludisme de manière automatisée à l'aide de CNN, en créant une vision informatique. La méthode utilisée repose sur l'apprentissage profond des données avec la structure CNN, pour classifier les images d'échantillons de sang en infectées et non infectées, à l'aide d'images standards divisées en trois fichiers : infectées, non infectées et un fichier de test. Ensuite, les images sont traitées sur une plateforme kaggle a l'aide du langage python.

Ensuite, les couches de CNN ont été appliquées aux images, et le modèle a été teste aléatoirement sur un ensemble d'images et classifier comme étant infectés ou non infectés. Le modèle a affiché une précision élevée de 96.20 % lorsqu'il a été traité sur une partie des images, comprenant un total de 27558 images. Le modèle a produit deux images aléatoires qui ont été dessinées et classées comme étant infectées ou non, ce qui démontre son efficacité. Cette étude préconise l'utilisation d'ordinateurs dotés d'un processeur puissant pour entraîner des images et des données de plus grande taille, afin d'obtenir une précision allant jusqu'à 100%.

Abstract

Malaria is a very dangerous disease. In this brief we look at a microscopic image processing chain to classify them as infected or uninfected lesions.

The purpose of this study is to automatically detect malaria using CNN, creating a computer vision. The method used is based on deep data learning with the CNN structure, to classify images of blood samples into infected and uninfected, using standard images divided into three files: infected, uninfected and a test file. Then, the images are processed on a kaggle platform using the python language.

Then the CNN layers were applied to the images, and the model was randomly tested on a set of images and classified as infected or not infected. The model displayed a high accuracy of 96.20% when processed on a portion of the images, comprising a total of 27558 images. The model produced two random images that were drawn and classified as infected or not, demonstrating its effectiveness. This study recommends the use of computers with a powerful processor to drive larger images and data to achieve accuracy of up to 100%.