MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DELA RECHERCHESCIENTIFIQUE

UNIVERSITE ABDELHAMID IBN BADIS - MOSTAGANEM

Faculté des Sciences Exactes et d'Informatique

Département de Mathématiques et informatique

Filière : Informatique

RAPPORT DE PROJET FIN D'ETUDE

Option : **Ingénierie des Systèmes d'Information**

Projet soutenu dans le cadre de l'arrêté 1275

THEME :

Détection automatique des maladies gastro-intestinales par l'apprentissage profond optimisé.

Etudiant(e) : « SEBBAR Sara»

« SERRADJ Chaimaa »

Encadrant(e) : « Mr, BESNASSI Miloud »

Année Universitaire 2022-2023

Résumé

Le diagnostic précoce de la maladie gastro-intestinale pourrait réduire le taux de mortalité de cette maladie, à des niveaux plus faibles. Les systèmes de classification automatique des maladies gastro-intestinales peuvent aider à réduire ces risques en signalant les cadres et les lésions suspects. L'endoscopie gastro-intestinale consiste une surveillance de plusieurs organes et produit de nombreuses images. Cependant, une interprétation précise de ces images nécessite la disponibilité de professionnels compétents, ce qui n'est pas toujours le cas dans les pays en voie de développement.

L'intelligence artificielle, qui inclut les réseaux de neurones convolutifs (CNN) et l'apprentissage profond (Deep Learning), présente un potentiel prometteur pour le diagnostic précoce des maladies gastro-intestinales, offrant ainsi une solution adéquate. En explorant les capacités de l'apprentissage profond dans l'analyse d'images, nous avons étudié les concepts du Deep Learning en utilisant des algorithmes de réseaux neuronaux convolutifs (CNN) pour construire un modèle de classification multi classe de ces images. Ce modèle peut donc prédire différentes maladies gastro-intestinales, permettant une meilleure identification et un traitement plus ciblé.

En nous basant sur le concept de sélection d'attributs, nous avons utilisé des techniques d'optimisation dans le processus de sélection d'attributs afin de détecter les maladies gastro-intestinales. Cette démarche nécessite une modélisation appropriée pour trouver un compromis entre un taux de reconnaissance élevé et un nombre réduit de caractéristiques extraites à partir de « deep features », nous avons utilisé un jeu de données ou dataset « KVASIR V2 » de 8000 images d'endoscopie gastro-intestinale. Les résultats obtenus en termes de classification sont significatifs et prometteurs.

Mots-clés: Diagnostic, gastro-intestinale, endoscopie, Intelligence Artificielle, Apprentissage profond, CNN, techniques d'optimisation.

Abstract

Early diagnosis of gastrointestinal disease could reduce the mortality rate associated with this condition to lower levels. Automatic classification systems for gastrointestinal diseases can help reduce these risks by identifying suspicious patterns and lesions. Gastrointestinal endoscopy involves monitoring multiple organs and generating numerous images. However, an accurate interpretation of these images requires the availability of competent professionals, which is not always the case in developing countries.

Artificial intelligence, including convolutional neural networks (CNN) and deep learning, holds promising potential for the early diagnosis of gastrointestinal diseases, providing an appropriate solution. By exploring the capabilities of deep learning in image analysis, we have studied the concepts of deep learning and used CNN algorithms to construct a multiclass classification model for these images. This model can predict different gastrointestinal diseases, enabling better identification and more targeted treatment.

Based on the concept of attribute selection, we have utilized optimization techniques in the process of detecting gastrointestinal diseases. This approach requires appropriate modeling to strike a balance between high recognition rates and a reduced number of features extracted from deep features. We used a dataset called « KVASIR V2 » consisting of 8000 gastrointestinal endoscopy images. The obtained results in terms of classification are significant and promising.

Keywords: Diagnostics, gastrointestinal, endoscopy, Artificial Intelligence, Deep learning, CNN, optimization techniques.

Liste des figures

Liste des tableaux

Liste des abréviations

Abréviation	Expression Complète	Page
IA	Intelligence Artificielle	4
CNN	Réseau de neurones convolutif	5
GI	Gastro Intestinal	7
SVM	Support Vector Machine	13
MCC	Matthews Correlation Coefficient	13
SGD	Stochastic Gradient Descent	14
RBF	Radial Basis Function	15
JC	Jaccard Coefficient	16
DSC	Dice Similarity Coeficient	16
JSC	Jaccard Similarity Coefficient	17
ASPP	Atrous Spatial Pyramid Pooling	18
DL	Deep Learning	23
ML	Machine Learning	23
CPU	Unité Centrale de traitement	23
CONV	Convolutionnelles	26
RGB	Red Green Blue	26
POOL	Pooling	27
ReLu	Rectified Linear Units	28
FC	Fully Connected	29
VGG	Visual Geometry Group	30
ILSVRC	ImageNet Larger Scale Visual Recognition Challenge	32
KNN	K-Nearest Neighbors	33
MLP	Multilayer Perceptron	35
GA	Genetic Algorithm	39
PSO	Particle Swarm Optimization	40
AO	Aquila Optimzer	42
CGO	Chaos Game Optimisation	42

GHZ	gigahertz	65
HD	High Definition	65
RAM	Random Access Memory	65
HDD	Hard Disk Drive	66
GPU	Graphics Processing Unit	67
HTML	HyperText Markup Language	68
CSS	Cascading Style Sheets	68
XML	eXtensible Markup Language	68
JS	JavaScript	68
TP	True Positive	73
TN	True Négative	73
FP	False Positive	73
FN	False Négative	73
max	Maximum	85
min	Minimum	85
var	Variance	85
std	Standard Deviation	85

Table des matières

Introduction Générale

L'imagerie médicale a considérablement évolué ces dernières années est devenue un outil indispensable pour le diagnostic de nombreuses maladies. Elle permet aux médecins de visualiser les organes internes et les structures corporelles sans avoir recours à la chirurgie, ce qui est non seulement moins invasif pour les patients, mais également plus sûr et plus efficace. Les différentes techniques d'imagerie médicale, comme l'endoscopie, permettent aux médecins de visualiser les organes internes dans différents plans et de détecter les anomalies qui pourraient indiquer une maladie. Les avancées technologiques ont également permis de développer des techniques d'imagerie de plus en plus précises et d'améliorer les images obtenues grâce à l'utilisation de l'Intelligence Artificielle (IA), autrement dit, l'utilisation de l'apprentissage automatique ou le Machine Learning qui est un sous-domaine de l'Intelligence Artificielle (IA).

L'endoscopie vidéo médicale est une technique d'imagerie utilisée pour visualiser les organes internes de l'appareil digestif. Elle permet aux médecins de détecter les anomalies, de diagnostiquer les maladies et de suivre l'évolution des maladies telles que les ulcères, les tumeurs comme le cancer de l'estomac, cancer de colorectal et les maladies inflammatoires de l'intestin. Il produit de nombreuses images qui nécessitent une précision et une rapidité bien supérieures à celles d'un être humain.

L'utilisation des algorithmes de Deep Learning peuvent être utilisés pour détecter les maladies gastro-intestinales avec plus de précision et de rapidité. Ces algorithmes peuvent être entraînés à reconnaître les signes de maladies spécifiques en utilisant des images d'entraînement. Une fois entraînés, ils peuvent être utilisés pour détecter automatiquement les signes de maladies dans les images produites par l'endoscopie vidéo.

Cela permet aux médecins de gagner du temps et de la précision dans le diagnostic des maladies, en automatisant la détection des anomalies dans les images, et de réduire les erreurs humaines. Cela peut également permettre de diagnostiquer des maladies plus rapidement, ce qui peut améliorer les chances de récupération pour les patients.

Ce projet a pour but d'étudier et de proposer un modèle d'apprentissage profond optimisé (CNN) pour la détection automatique des maladies gastro-intestinales selon l'absence ou la présence des signes spécifiques. La performance du modèle constitue une partie cruciale de ce travail.

Nous avons organisé notre mémoire en quatre 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 : Présente les généralités sur les maladies Gastro-intestinales.
- Chapitre 2 : Concerne réseau de neurones convolutif (CNN) et modèles Pré-Entrainés.
- Chapitre 3 : Consacré aux algorithmes d'optimisations.
- Chapitre 4 : Présente l'implémentation et résultats expérimentaux.

Chapitre 1

Généralité sur les maladies Gastro-intestinales

1.1 Introduction

Dans ce chapitre, nous présenterons des informations générales sur les maladies Gastro-intestinale, concernant leur origine, le fonctionnement et les types. Enfin, nous clôturerons ce chapitre par l'étude de quelques travaux liés à notre problématique.

1.2 Explication de l'origine des maladies Gastro-intestinales

Les historiens et les médecins ont documenté la présence de troubles gastro-intestinaux fonctionnels tout au long de l'histoire humaine enregistrée. Cependant, jusqu'à récemment, une attention limitée a été accordée à ces troubles en raison du manque de pathologie identifiable et de l'absence d'un cadre conceptuel pour les comprendre et les catégoriser. L'investigation systématique des troubles gastro-intestinaux fonctionnels n'a commencé qu'au milieu du 20e siècle, et avant cette époque, seuls des rapports occasionnels de symptômes gastro-intestinaux fonctionnels ont été publiés, le premier apparaissant il y a seulement 200 ans.

Au cours des 25 dernières années, l'attention scientifique portée à la compréhension et à la prise en charge adéquate des patients atteints de troubles fonctionnels gastro-intestinaux s'est progressivement développée. Avec la compréhension vient la justification de l'utilisation de médicaments dirigés contre les récepteurs intestinaux ainsi que des formes de traitement psychopharmacologiques, comportementales et psychologiques. De plus, il y a eu une augmentation du taux de publications scientifiques et une plus grande exposition médiatique au public par le biais de la télévision, de la radio et d'Internet. [1]

1.3 Maladies gastro-intestinales

Les maladies gastro-intestinales désignent l'ensemble des affections qui affectent les organes de la digestion, c'est-à-dire l'estomac, l'intestin, le foie, la vésicule biliaire, le pancréas et le côlon. Ces maladies peuvent avoir différentes causes et peuvent provoquer des symptômes tels que des douleurs abdominales, des troubles de la digestion, des vomissements, de la diarrhée, de la constipation, des saignements rectaux, des troubles de la motilité intestinale.

Les maladies gastro-intestinales peuvent être classées en différentes catégories en fonction de leur cause, comme les maladies infectieuses, les maladies immunitaires, les maladies héréditaires, les maladies liées à l'environnement, les maladies liées à un mode de vie malsain. Les exemples de maladies gastro-intestinales comprennent :

- Les maladies infectieuses telles que la gastro-entérite, la dysenterie et l'hépatite.
- Les maladies immunitaires telles que la maladie de Crohn et la rectocolite hémorragique.
- Les maladies héréditaires telles que la maladie de Hirschsprung et la maladie coeliaque.
- Les maladies liées à l'environnement telles que la maladie de Lyme et les cancers colorectaux.
- Les maladies liées à un mode de vie malsain telles que l'obésité et les maladies cardiaques.

Il est important de noter que certains de ces symptômes peuvent aussi être causés par d'autres maladies qui n'ont pas de lien direct avec les organes digestifs. Il est donc important de consulter un médecin pour un diagnostic approprié.

1.4 Fonctionnement du tractus gastro-intestinal

Le tractus gastro-intestinal (GI) est composé d'un tube qui va de la bouche à l'anus, et qui comprend l'œsophage, l'estomac, le petit intestin, le gros intestin et l'anus, comme le montre la Figure 1. Il joue un rôle important dans la digestion et l'absorption des aliments.

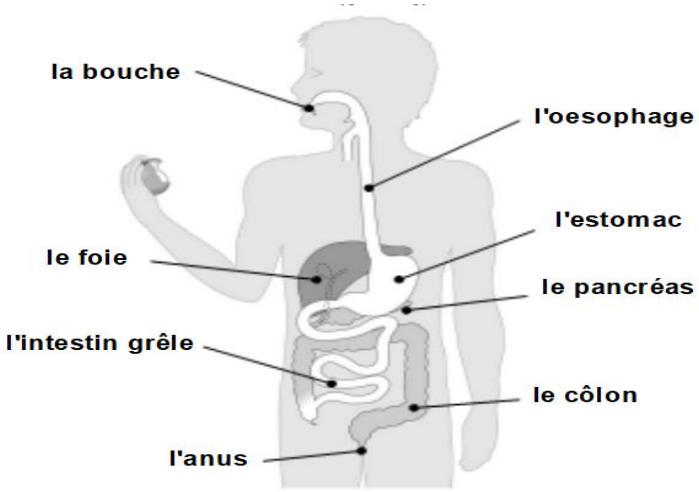


Figure 1 – Le tractus gastro-intestinal

Lorsque vous mangez, la nourriture est mâchée et mélangée avec la salive dans la bouche, puis déglutie et passe dans l'œsophage. L'œsophage utilise des contractions musculaires appelées péristaltisme pour propulser la nourriture dans l'estomac.

Dans l'estomac, la nourriture est mélangée avec des sucs gastriques qui décomposent les protéines et les graisses. L'estomac stocke également la nourriture et la libère lentement dans le petit intestin.

Le petit intestin est le principal organe d'absorption des nutriments. Il contient des villosités et des microvillosités qui augmentent la surface d'absorption, ainsi que des enzymes produites par le pancréas et la vésicule biliaire qui aident à la digestion.

Le gros intestin, ou côlon, absorbe l'eau et les sels minéraux restants, et compacts les aliments en selles. Les selles sont ensuite expulsées de l'organisme par l'anus.

La fonction de chaque organe est régulée par les hormones et les neurotransmetteurs, qui assurent la coordination et l'harmonie des mouvements musculaires et la sécrétion des sucs digestifs. [2]

1.5 Les types des maladies les plus courantes du gastro-intestinales

Avec les changements de style de vie générés dans l'ère actuelle, les maladies gastro-intestinales sont devenues très communs et souffrent d'une grande proportion de la population dans la société. Il existe de nombreux types de maladies gastro-intestinales, mais voici quelques-unes des plus courantes :

1.5.1 Cancer de l'estomac

Le cancer de l'estomac (parfois appelé cancer gastrique) prend naissance dans les cellules de l'estomac. La tumeur cancéreuse (maligne) est un groupe de cellules cancéreuses qui peuvent envahir les tissus voisins et les détruire. La tumeur peut également se propager (métastases) à d'autres parties du corps.

Les symptômes :

Il est possible que le cancer de l'estomac ne cause aucun signe ni symptôme aux tout premiers stades de la maladie. Les signes et symptômes comprennent ceux-ci :

- Douleur ou inconfort dans l'abdomen (difficulté à digérer).
- Brûlures d'estomac (sensation de brûlure en profondeur dans la poitrine).
- Perte d'appétit.
- Sensation de plénitude, même après un repas léger.
- Ballonnement dans l'abdomen.
- Nausées ou vomissements (dans lesquels il peut y avoir du sang).
- Sang dans les selles.
- Anémie.
- Fatigue et faiblesse.
- Troubles de la déglutition.
- Accumulation de liquide dans l'abdomen (ascite).
- Jaunissement de la peau et du blanc des yeux (jaunisse).

1.5.2 Cancer de colorectal

Le cancer colorectal est une maladie dans laquelle les cellules du côlon ou du rectum deviennent incontrôlables. Il est parfois appelé cancer du côlon en abrégé. Le côlon est le gros intestin ou gros intestin. Le rectum est le passage qui relie le côlon à l'anus.

Parfois, des tumeurs anormales, appelées polypes, se forment dans le côlon ou le rectum. Au fil du temps, certaines tumeurs bénignes peuvent se transformer en cancer. Les tests de dépistage peuvent détecter les polypes afin qu'ils puissent être retirés avant qu'ils ne se transforment en cancer. Le dépistage permet également de détecter précocement le cancer colorectal, lorsqu'il s'agit de la meilleure forme de traitement.

Les symptômes :

Les signes et symptômes du cancer colorectal comprennent ceux-ci :

- Sang dans les selles ou rectification de selles de couleur noir.
- Changements fréquents de la fréquence des selles.
- Diarrhée, constipation ou selles plus étroites que d'habitude.
- Douleur abdominale ou rectale.
- Ballonnements ou gaz fréquents.
- Perte de poids ou perte d'appétit.
- Fatigue ou anémie.
- Sentiment de satiété rapide ou de plénitude.
- Masses ou excroissances dans l'abdomen.
- Douleur ou gêne dans la région de l'abdomen, du dos, des fesses ou des jambes.

1.5.3 Brulures d’estomac

Les brûlures d’estomac se traduisent une sensation douloureuse dans la région de l'estomac qui irradie parfois derrière le sternum voire au niveau de la gorge. Elles peuvent être accompagnées de remontées acides de l’estomac vers l’œsophage, à l’origine d’un goût désagréable.

Les brûlures d'estomac sont bénignes et passagères, elles surviennent habituellement après la prise d'un repas ou la nuit, en cas de stress, de faim ... Lorsque le sujet a fait un repas trop copieux, qu'il n'a pas mâché suffisamment les aliments, un excès d'acidité enflamme les parois de l'estomac et quelquefois de l'œsophage, provoquant une brûlure.

Les symptômes :

- Sensation de cuisson ou de gêne dans la portion supérieure de votre estomac ou dans la région inférieure du thorax.
- Ballonnement.
- Des rots plus fréquents.
- Accumulation de gaz dans l'estomac entraînant de la flatulence.
- Faim plus rapidement satisfaite que de coutume.
- Goût aigre ou amer dans la bouche.
- La nausée.

1.5.4 L'inflammation de la muqueuse gastro-intestinale (gastrite)

La gastrite est l'inflammation aiguë ou chronique de la muqueuse gastrique de l'estomac. L'acide gastrique qu'il contient (qui sert à la digestion des aliments) attaque parfois la paroi de l'estomac. Résultat : une réaction inflammatoire se déclenche. Cette irritation de la muqueuse de l'estomac ne doit pas être confondue avec le reflux Castro-œsophagien ou des douleurs d'ulcère gastrique.

En médecine, le terme de gastrite est d'ailleurs souvent employé à tort sans aucune preuve. Or ce diagnostic repose sur des arguments cliniques, endoscopiques, biologiques et histologiques précis.

Les symptômes :

- Brûlures d'estomac ou douleur à l'estomac.
- Nausées ou vomissements.
- Perte d'appétit.
- Douleur ou gêne à l'abdomen supérieur.
- Goût amer ou métallique dans la bouche.
- Selles foncées ou sang dans les selles (en cas de saignement gastrique).

1.6 Etat de l’art sur la détection de la maladie gastro-intestinale

Dans l'état de l'art, Nous avons envisager les expériences ou les travaux qui sont déjà réalisées en utilisant les bases de données KVASIR V2, HYPERKVASIR, KVASIR-INSTRUMENT et CVC-CLINICDB où les anomalies du tractus gastro-intestinal ont été automatiquement détectées et classées à l'aide de cadres endoscopiques.

1.6.1 La base de données KVASIR V2, 2017

KVASIR V2 est une base de données d'images d'endoscopie qui a été développée pour la recherche en détection automatique des anomalies du tractus gastro-intestinal. Il s'agit d'une version mise à jour de la base de données KVASIR originale, qui contient des images d'endoscopie du tractus digestif supérieur et inférieur obtenues à partir de différents types d'endoscopes. Il comprend des images de haute résolution de l'œsophage, de l'estomac et du côlon, ainsi que des annotations de référence pour les anomalies détectables, telles que les ulcères, les polypes et les tumeurs. Les images ont été obtenues à partir de patients réels et ont été annotées par des experts en endoscopie pour garantir la qualité et la fiabilité des données.

La base de données contient 8 000 images divisé en 8 classes avec une résolution différente de 720x576 à 1920x1072 pixels et organisés de manière à être triés dans des dossiers séparés nommés en fonction du contenu. Il existe plusieurs travaux de recherche qui ont utilisé la base de données KVASIR V2 pour détecter et classer automatiquement les anomalies du tractus gastro-intestinal à l'aide de cadres d'apprentissage automatique. Ces travaux ont généralement utilisé des techniques d'apprentissage profond pour détecter les anomalies dans les images d'endoscopie, en utilisant des réseaux de neurones convolutionnels pour extraire des caractéristiques des images et des réseaux de neurones récurrents pour les classer. [3]

Parmi les travaux les plus récents, on peut citer :

- **Travaux de Taruna Agrawal et al, 2017**

Taruna Agrawal et son équipe du Laboratoire de parole et de communication de l'Université du Maryland ont travaillé sur un projet intitulé "Transfer learning based Classification of Medical Images" en 2017. Leur objectif était de faire progresser l'application des outils d'apprentissage automatique dans le domaine médical en se concentrant spécifiquement sur la détection de repères gastro-intestinaux à partir d'images.

Ils ont proposé une technique d'extraction des caractéristiques basée sur l'apprentissage par transfert avec des modèles CNN pré-entraînés, tels que VGGNet et Inception-v3, entraînés sur ImageNet. Les caractéristiques de base fournies ont été entraînées avec seulement 3200 échantillons d'entraînement en utilisant des algorithmes supervisés. Le meilleur modèle était la combinaison des caractéristiques Inception-V3 et les caractéristiques VGGNet. Le classificateur SVM multi-classes a été entraîné sur ces caractéristiques extraites avec un noyau linéaire. L'architecture proposée a limité la profondeur de l'extraction des données qui provoque le problème des minima locaux. De plus, VGGNet possède un nombre extrêmement élevé de paramètres entraînables. La méthode proposée a atteint un score MCC de 82,57%. Les résultats de performance de classification à l'aide des architectures profondes sont 81,6%, 78,5%, 82,6% pour Baseline + Inception-V3, Baseline + VGGNet, Baseline + Inception-V3 + VGGNet, respectivement. [4]

- **Travaux de Yogapriya Jaganathan et al, 2021**

L'équipe de Yogapriya Jaganathan du Collège d'ingénierie et de technologie en Inde a travaillé sur un projet de classification des maladies gastro-intestinales à partir d'images d'endoscopie sans fil en utilisant l'apprentissage par transfert. Ils ont utilisé des modèles CNN pré-entraînés tels que VGG16, ResNet-18 et GoogleNet pour extraire des caractéristiques à partir d'images d'endoscopie. Les modèles ont été entraînés sur un ensemble de données d'entraînement et testés sur un ensemble de données de validation. Le modèle VGG16 a obtenu le taux de reconnaissance le plus élevée de 96,33% et a également obtenu une précision et un rappel parfaits. Les résultats ont montré que les modèles VGG16 et GoogleNet offrent une meilleure précision dans la classification des maladies gastro-intestinales. [5]

- **Travaux de Zenebe Markos Lonseko et al, novembre 2021**

Zenebe Markos Lonseko et son équipe du Centre de biologie informationnelle de l'université des sciences et technologies électroniques de Chine ont travaillé sur un projet en Novembre 2021 pour utiliser des réseaux de neurones convolutionnels (CNN) pour classer les maladies gastro-intestinales en utilisant des images endoscopiques. Ils ont utilisé des techniques d'augmentation de données et une descente de gradient stochastique (SGD) pour optimiser leur réseau.

Ils ont comparé leur méthode (appelée ECA-Net) à d'autres méthodes apparentées et ont démontré que leur méthode a obtenu des résultats significativement meilleurs avec un taux de reconnaissance de 94,33%. [6]

1.6.2 La base de données HYPERKVASIR, 2019

La base de données HYPERKVASIR est une base de données de référence pour l'analyse d'images médicales gastro-intestinales. Elle a été créée par l'hôpital de Bærum en Norvège et contient 110 079 images et 374 vidéos qui ont été collectées lors d'examens réels de gastro-intestinale et de coloscopie. Les données ont été étiquetées par des endos copistes expérimentés et comprennent des repères anatomiques et des résultats pathologiques et normaux. L'ensemble de données comprend quatre enregistrements principaux : des images étiquetées, des images segmentées, des images non étiquetées et des vidéos étiquetées, avec un total de 40 classes différentes. Certaines images et vidéos contiennent également une image dans l'image utilisée pour obtenir une vue topographique du côlon. Il existe plusieurs travaux de recherche qui ont utilisé la base de données HYPERKVASIR pour des applications de diagnostic médical. Ces travaux ont souvent utilisé des techniques d'apprentissage automatique pour classer les images ou les vidéos dans différentes catégories, telles que les repères anatomiques normaux ou pathologiques. [7]

Parmi les travaux les plus récents, on peut citer :

- **Travaux de Melaku Bitew Haile et al, 2022**

Melaku Bitew Haile et son équipe de l'université de Gondar en Éthiopie ont travaillé sur un projet intitulé "Detection and classification of gastrointestinal disease using convolutional neural network and SVM" en juin 2022. L'objectif était de proposer un nouveau modèle pour l'extraction séquentielle de caractéristiques des modèles d'apprentissage en profondeur utilisant des architectures séquentielles VGGNet et InceptionNet pour modéliser les caractéristiques, puis les classer à l'aide d'une machine vectorielle multi support basée sur le noyau RBF.

Les résultats ont montré que le modèle proposé avec un classificateur SVM a atteint un taux de reconnaissance de 98,7%, surpassant les derniers modèles VGGNet16 et Inceptionv3 préformés. [8]

- **Travaux de Akella S. Narasimha Raju et al, 2022**

Akella S. Narasimha Raju et son équipe de département des réseaux et des communications, École d'informatique, Institut SRM des sciences et technologies, Inde, ont travaillé sur un projet intitulé "Dexterous Identification of Carcinoma through ColoRectalCADx with Dichotomous Fusion CNN and UNet Semantic Segmentation" en 2022. Le but principal était de construire un système pour détecter la maladie colorectale en tant que carcinome avec une reconnaissance automatique et qualifiée des lésions. Le système ColoRectalCADx comprend cinq étapes, au cours desquelles plusieurs expériences ont été menées pour la reconnaissance du carcinome colorectal.

Les résultats montrent que le modèle DenseNet-201 a surpassé les six autres modèles CNN avec un taux de reconnaissance de 84%. Le modèle de fusion DaRD-22 CNN a obtenu les résultats les plus élevés pour les données d'entraînement, de test, d'entraînement SVM et de test SVM. L'étape finale du système ColoRectalCADx consiste à identifier et reconnaître les vrais polypes malins avec l'ensemble de données HYPERKVASIR, en utilisant une structure UNet CNN. Les images originales avec les masques d'image correspondants des polypes malins sont reconnues avec précision avec des pertes d'entraînement. [9]

1.6.3 La base de données KVASIR-INSTRUMENT, 2020

La base de données KVASIR-INSTRUMENT est un ensemble de données open source d'images endoscopiques gastro-intestinales qui sont utilisé pour évaluer la charge de morbidité et le volume de résection pathologique. Elle contient 590 images d'outils endoscopiques et leur masque de vérité au sol. Les images ont été collectées à l'aide d'un équipement endoscopique standard Olympus et Pentax et contiennent des instruments chirurgicaux tels que des collets, des ballons et des pinces à biopsie. La résolution des images varie de 523X613 px à 1072X1920 px.

Il existe plusieurs travaux de recherche qui ont utilisé la base de données KVASIR-INSTRUMENT pour des tâches de segmentation d'images en utilisant des méthodes d'apprentissage automatique. [10]

Parmi les travaux les plus récents, on peut citer :

- **Travaux de Debesh Jha et al, 2020**

L'équipe de recherche dirigée par Debesh Jha de l'Université de Norvège a travaillé sur un projet visant à développer des systèmes automatisés pour la segmentation des outils de diagnostic et d'endoscopie thérapeutique du tractus gastro-intestinal. Ils ont utilisé les architectures U-Net et DoubleUNet en utilisant le Framework Keras avec TensorFlow comme back end. Ils ont utilisé une augmentation de base pour l'entraînement et ont divisé l'ensemble de données en 80% pour l'entraînement et 20% pour les tests.

Les résultats ont montré que l'architecture U-Net classique a eu de meilleures performances que DoubleUNet, avec un JC et un DSC plus élevés et une vitesse de calcul deux fois plus rapide. Les scores de précision étaient également comparables pour les deux architectures. Pour U-Net, nous avons obtenu un taux de reconnaissance de 94,87%, tandis que pour DoubleUNet, le taux de reconnaissance était de 92,75%. [11]

- **Travaux de Taira Watanabe et al, Janvier 2022**

En Janvier 2022, l'équipe de Taira Watanabe de l'Université de Doshisha au Japon a travaillé sur un projet intitulé "Performance Comparison of Deep Learning Architectures for Artifact Removal in Gastrointestinal Endoscopic Imaging". Leur objectif principal était d'utiliser un système CADx pour la segmentation sémantique afin de détecter automatiquement les régions cancéreuses et les polypes dans les images endoscopiques. Ils ont utilisé des architectures de réseaux de neurones profonds pour améliorer la segmentation des images et ont comparé les résultats obtenus avec ceux des architectures de base telles qu'U-Net et DoubleU-Net.

Les résultats de l'expérience ont montré que les sept architectures de codeur utilisées ont toutes amélioré les résultats de segmentation par rapport aux études existantes en termes de JSC, DSC, de précision, de rappel et d'exactitude. Parmi celles-ci, le réseau qui a utilisé Inception-ResNet-v2 a montré les meilleures performances, avec un taux de reconnaissance de 95,4 %. [12]

1.6.4 La base de données CVC-CLINICDB, 2015

CVC-CLINICDB est un ensemble de données composé de 612 images ayant une résolution de 384x288 provenant de 31 séquences de coloscopie. Il est utilisé pour des tâches de segmentation d'images médicales, en particulier pour la détection de polypes dans les vidéos de coloscopie. Les images de ce jeu de données sont disponibles au public et peuvent être utilisées à des fins de recherche.

Il existe plusieurs travaux de recherche qui ont utilisé la base de données CVC-CLINICDB pour des tâches de segmentation d'images en utilisant des méthodes d'apprentissage automatique. [13]

Parmi les travaux les plus récents, on peut citer :

- **Travaux de Olaf Ronnebergeret al, Mai 2015**

En Mai 2015, l'équipe de Olaf Ronneberger, Philipp Fischer, et Thomas Brox de département d'informatique et ESBIO Centre d'études sur la signalisation biologique, Université de Fribourg, Allemagne a travaillé sur un projet intitulé "U-Net: Convolutional Networks for Biomedical Image Segmentation". Leur objectif était de développer une architecture de réseau de neurones convolutionnel pour la segmentation d'images biomédicales.

Ils ont présenté U-Net pour remédier aux limitations des méthodes de segmentation d'images antérieures qui manquaient de capacité à capturer les informations contextuelles et spatiales importantes pour la segmentation précise. U-Net était conçu pour fournir une segmentation précise en utilisant une architecture d'apprentissage profond capable de capturer efficacement les informations de haut niveau et de bas niveau à partir de l'image d'entrée. Les résultats obtenus avec U-Net avec un taux de reconnaissance de 82,3% ont montré que leur objectif était atteint avec succès et qu'U-Net est devenu un outil largement utilisé pour diverses tâches de segmentation d'images biomédicales. [14]

- **Travaux de Debesh Jhaet al, Juin 2020**

En Juin 2020, l'équipe de Debesh Jha d'université de Norvège a travaillé sur un projet intitulé "DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation".

L'objectif de cette équipe dans cet article est de proposer une nouvelle architecture de réseau neuronal convolutionnel appelée DoubleU-Net pour la segmentation d'images médicales. La DoubleU-Net comprend cinq composants principaux, à savoir deux réseaux U-Net, VGG-19, un bloc squeeze-and-excite et ASPP. Les auteurs montrent que la performance de la DoubleU-Net est significativement meilleure que les réseaux de base et U-Net car il attient un taux de reconnaissance de 92,39%. Ils croient que les résultats de segmentation peuvent être améliorés en intégrant d'autres blocs de CNN et en utilisant des techniques de post-traitement. [15]

- **Travaux de Danish Ahmadet al, Décembre 2022**

En Décembre 2022, l'équipe de Danish Ahmad d'école de génie électrique et d'informatique a travaillé sur un projet intitulé "ESFPNet: efficient deep learning architecture for real-time lesion segmentation in autofluorescence bronchoscopic video".

L'objectif de ce projet était de concevoir une architecture de réseau neuronal profond pour la segmentation en temps réel des lésions dans les vidéos bronchoscopiques à auto fluorescence. Les chercheurs ont utilisé l'architecture ESFPNet pour atteindre ce but et ont effectué des expériences pour évaluer les performances de la méthode. Les résultats ont montré qu'ESFPNet peut fournir des résultats de segmentation de haute qualité avec un taux de reconnaissance de 94,9%, ce qui en fait une solution efficace pour la segmentation des lésions dans les vidéos bronchoscopiques. [16]

1.7 Synthèse

Dans la section précédente (État de l'art), nous avons abordé certaines bases de données annotées et vérifiées par des médecins, fournissant des images endoscopiques du tractus gastro-intestinal. Pour chaque base de données, nous avons examiné les travaux entrepris afin de diagnostiquer et détecter les maladies gastro-intestinales. Il est évident que l'approche la plus avancée jusqu'à présent repose sur l'utilisation d'algorithmes de deep learning, notamment les réseaux neuronaux convolutifs (CNN).

Cette affirmation est étayée par l'ensemble des études qui utilisent des modèles inspirés de l'architecture CNN, ainsi que par les résultats prometteurs de ces algorithmes. Certains ont même atteint l'état de l'art et sont utilisés dans le domaine médical comme outils d'aide au diagnostic ou de diagnostic. Le tableau 1 présente une comparaison des études récentes utilisant des méthodes CNN pour la classification et la segmentation des maladies gastro-intestinales.

Etude	Base de données	Méthode	Taux de reconnaissance
Olaf Ronnebergeret al, Mai 2015	CVC-CLINICDB	UNet	82.3%
Taruna Agrawal et al, 2017	KVASIR V2	Baseline + Inception-V3 + VGGNet	82.6%
Debesh Jhaet al, Juin 2020	CVC-CLINICDB	DoubleU-Net	92.39%
Debesh Jha et al, Octobre 2020	KVASIR-INSTRUMENT	UNet	94.87%
Yogapriya Jaganathan et al, Septembre 2021	KVASIR V2	VGG16	96.33%
Zenebe Markos Lonseko et al, Novembre 2021	KVASIR V2	ECA-Net	94.33%
Taira Watanabe et al, Janvier 2022	KVASIR-INSTRUMENT	Inception-ResNet-v2	95.4 %
Melaku Bitew Haile et al, juin 2022	HYPERKVASIR	VGGNet+InceoptionNet + SVM	98,7%
Akella S. Narasimha Raju et al,Octobre2022	HYPERKVASIR	DenseNet201	84%
Danish Ahmadet al, Décembre 2022	CVC-CLINICDB	ESFPNet	94.9%

Tableau 1 - Table comparative des études récentes.

1.8 Conclusion

Dans ce chapitre, nous avons présenté un aperçu général des maladies gastro-intestinales, qui se réfère à toute affection touchant le système gastro-intestinal, qui comprend l'ensemble de l'appareil digestif, allant de la bouche à l'anus. Nous avons abordé les causes de la maladie gastro-intestinale. Nous avons également expliqué le fonctionnement du système gastro-intestinal, qui commence par l'ingestion des aliments, leur décomposition chimique et leur absorption dans l'intestin grêle, la formation de selles et leur évacuation par le rectum. Ainsi, nous avons décrit les différents types de maladies gastro-intestinales courantes, notamment le cancer de l'estomac, cancer de colorectal, brulures d'estomac et gastrite, ainsi que leurs symptômes. Enfin, vous avez examiné les dernières avancées dans le domaine de la détection des maladies gastro-intestinales à partir d'images médicales. Ces avancées permettent aux médecins de détecter plus précocement les maladies gastro-intestinales, ce qui permet un traitement plus rapide et efficace.

Dans le prochain chapitre, nous allons explorer les concepts du Deep Learning et les réseaux de neurones convolutifs (CNN) en définissant leur architecture ainsi que les fonctions des couches qui les composent.

Chapitre 2

Réseau de neurones convolutif (CNN) et modèles pré-entrainés.

2.1 Introduction

L’apprentissage profond a permis la découverte de nouveaux médicaments ainsi que la détection de maladies. Il augmente considérablement notre compréhension de la biologie, notamment de la génomique, de la protéomique, de la métabolomique et de l'immunomique.

L’apprentissage profond s'est révélé particulièrement efficace dans l'imagerie médicale, grâce à la disponibilité d'images de haute qualité et à la capacité des réseaux neuronaux convolutifs à classer les images. L’apprentissage profond est également en train de réaliser des avancées majeures dans l'amélioration de la qualité des services de santé en anticipant des événements médicaux grâce aux dossiers médicaux électroniques.

Ce chapitre explique les concepts associés à l'apprentissage profond pour la reconnaissance des objets en utilisant des architectures de réseaux de neurones convolutifs.

2.2 Deep Learning

Le Deep Learning est un sous-domaine de l'apprentissage automatique qui consiste à utiliser des réseaux de neurones profonds pour résoudre des tâches complexes, comme illustré sur la Figure 2. Les réseaux de neurones profonds sont des architectures de réseaux de neurones qui ont plusieurs couches cachées, contrairement aux réseaux de neurones traditionnels qui n'ont qu'une seule couche cachée. [17]

Les couches cachées permettent aux réseaux de neurones de capturer des caractéristiques plus complexes et de les utiliser pour effectuer des tâches de reconnaissance d'images, de traitement du langage naturel, de reconnaissance de la parole. Les techniques de Deep Learning ont connu un grand succès dans de nombreuses applications, en raison de leur capacité à traiter des données volumineuses et à obtenir des résultats précis.

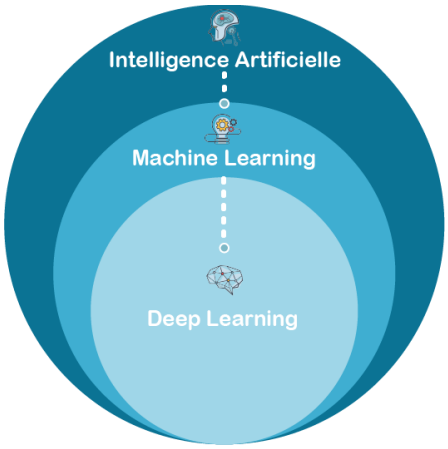


Figure 2 - Comprendre le machine Learning et le Deep Learning

2.2.1 Différence entre Deep Learning et Machine Learning

Deep Learning (DL) et Machine Learning (ML) sont deux types d'IA. La principale différence que nous remarquons entre ces deux concepts provient de la manière dont les données sont présentées au modèle. La Figure 3 montre la différence entre eux.

- ML est un sous-ensemble de l'IA et DL est un sous-ensemble de Machine Learning.
- ML peut s'entraîner sur des ensembles de données plus petits, contrairement DL nécessite de grandes quantités de données.
- ML nécessite plus d'intervention humaine pour corriger et apprendre, mais DL apprend par lui-même à partir de l'environnement et des erreurs passées.
- ML entraînement plus court et moins précis, DL entraînement plus long et plus précis.
- ML peut s'entraîner sur un CPU (unité centrale de traitement), alors que DL a besoin d'un GPU spécialisé (unité de traitement graphique) pour s'entraîner [18]

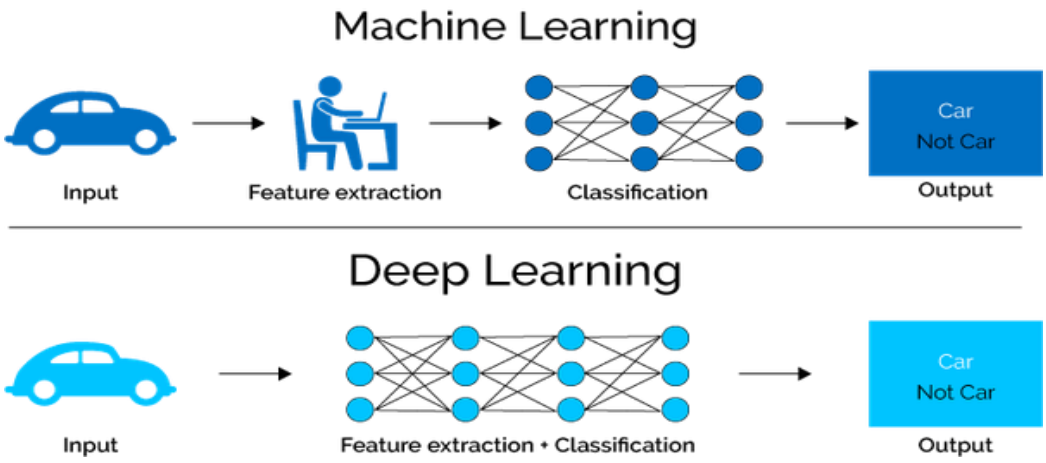


Figure 3 – Différence entre Deep Learning et Machine Learning

2.2.2 Pourquoi Deep Learning

Le Deep Learning est devenu de plus en plus populaire pour plusieurs raisons :

- **Flexibilité :** Les architectures profondes sont très flexibles et peuvent modéliser des données très différentes.

- **Automatisation :** En Deep Learning, le processus d'apprentissage englobe tout le pipeline, de l'extraction de variables explicatives jusqu'au modèle prédictif.

- **Complexité :** Le nombre de paramètres estimés dans un réseau de neurones profond peut facilement être de dizaines de millions.

- **Opacité :** Cette complexité et cet assemblage profond donnent au Deep Learning un caractère de "boîte noire" beaucoup plus important que pour les méthodes traditionnelles.

- **Transfer Learning :** Il est possible d'utiliser un modèle de Deep Learning déjà entraîné sur des données génériques, ce qui peut économiser du temps et des ressources.

- **Fine-tuning :** Les paramètres n'ont pas besoin d'être ré entraînés à partir de zéro, seulement ajustés.

2.3 Réseaux de Neurones Convolutifs (CNN)

2.3.1 Définition

Les réseaux de neurones convolutionnels ou (CNN pour Convolutional Neural Network en anglais), sont des réseaux de neurones multicouches spécialisés dans la reconnaissance de formes comme les systèmes de recommandation, le traitement du langage naturel, et bien sûr la classification des images.

Ils comportent deux parties distinctes, la partie convolutionnelle son architecture repose sur des couches de convolution alternant avec des couches d'agrégation (pooling). La partie perceptron convolutionnels multicouche qui combiner les caractéristiques du code CNN pour classer l'image.

2.3.2 Historique

Le premier réseau de neurones convolutif, appelé LeNet, a été développé en 1998 par le chercheur français Yann LeCun [19]. Il a permis de bonnes performances en reconnaissance des caractères, mais les progrès ont été limités par la technologie et l'accès aux données. Certains chercheurs ont continué à travailler sur ce modèle pendant deux décennies et ont finalement réussi à améliorer la technique grâce aux évolutions technologiques et à l'accès accru aux données. En 2012, un algorithme révolutionnaire de Deep Learning appelé AlexNet a remporté avec succès le concours de reconnaissance d'image ILSVRC et a révolutionné la reconnaissance d'image [20]. Aujourd'hui, les réseaux de neurones convolutifs sont les modèles les plus performants pour la classification d'images et sont utilisés par de nombreuses entreprises de technologie de pointe.

2.3.3 Architecture et le fonctionnement de CNN

Les réseaux de neurones convolutifs (CNN) sont des réseaux de neurones multicouches leur fonctionnement inspirés de processus biologiques. Les réseaux de neurones convolutifs ont de nombreuses applications dans les systèmes de recommandation de reconnaissance d'images et de vidéos.

L'architecture d'un réseau de neurones convolutif consiste à différents types de couches comme illustré à la Figure 4.

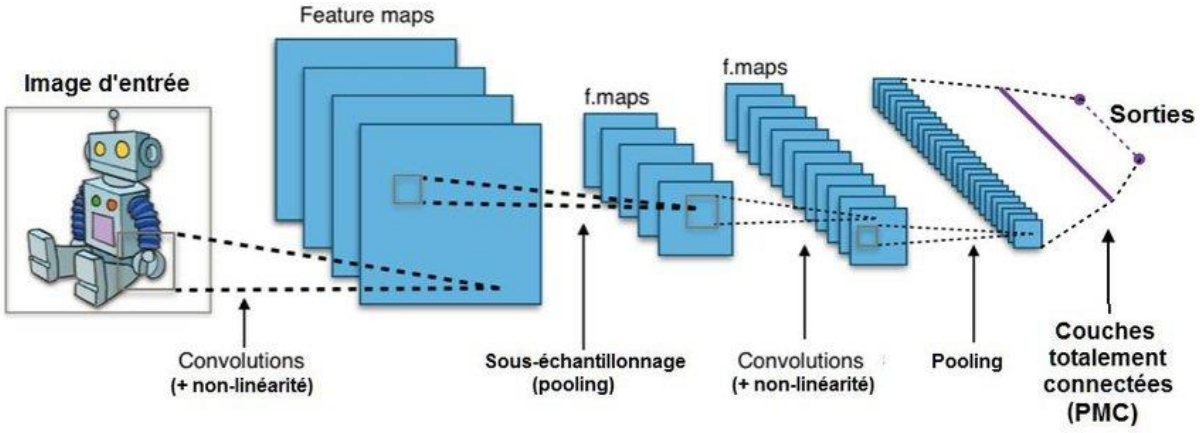


Figure 4 - L'architecture d'un réseau de neurones convolutif.

2.3.3.1 Couches convolutionnelles (CONV)

Cette couche est la première couche utilisée pour extraire les différentes caractéristiques des images d'entrée. Il effectue un produit entre deux matrices, une matrice étant l'ensemble de paramètres pouvant être appris, également appelée noyau ou filtre, et l'autre matrice étant la partie restreinte du champ récepteur [21]. Dans cette couche, nous utilisons un filtre ou une méthode Kernel pour extraire les caractéristiques de l'image d'entrée, comme illustré à la Figure 5.

$$\frac{W - F + 2P}{S} + 1 \quad (1)$$

Le produit de convolution est comme l'équation (2):

$$O(x, y) = \sum_{i=0}^{2} \sum_{j=0}^{2} K(i, j) \times I(x - 1 + j, y - 1 + i) \quad (2)$$

Telle que : *K* Le filtre, *I* la partie restreinte du champ récepteur

Jusqu'à ce que l'image complète soit numérisée, le noyau effectue des ajustements horizontaux et verticaux en fonction de la vitesse de foulée. [21]

Le noyau est moins volumineux qu'une image, mais il a plus de profondeur. Cela signifie que si l'image a trois canaux (RGB), la hauteur et la largeur du noyau seront modestes dans l'espace, mais la profondeur couvrira les trois.

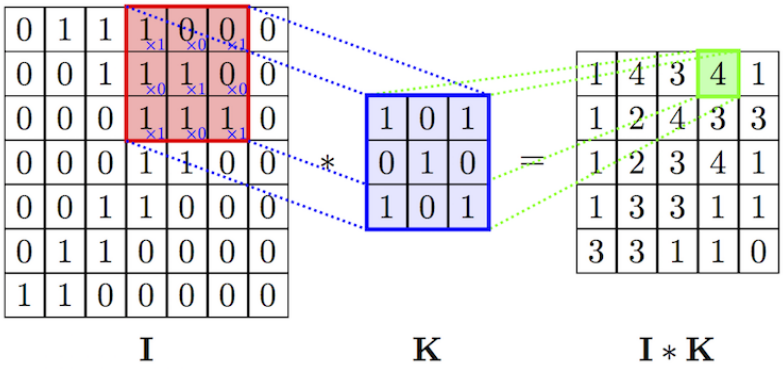


Figure 5 - Le produit de convolution.

Il y' a trois paramètres permettent de dimensionner le volume de la couche de convolution :

- **Profondeur de la couche :** nombre de noyaux de convolution (ou nombre de neurones associés à un même champ récepteur).
- **Le pas:** contrôle le chevauchement des champs récepteurs. Plus le pas est petit, plus les champs récepteurs se chevauchent et plus le volume de sortie sera grand.
- **La marge (à 0) ou zero padding :** parfois, il est commode de mettre des zéros à la frontière du volume d'entrée. La taille de ce 'zéro-padding' est le troisième hyper paramètre. Cette marge permet de contrôler la dimension spatiale du volume de sortie. En particulier, il est parfois souhaitable de conserver la même surface que celle du volume d'entrée [22].

2.3.3.2 Couches de pooling (POOL)

L'objectif principal de cette couche est de réduire la taille de la carte d'entités convoluées afin de réduire les coûts de calcul. Ceci est réalisé en diminuant les connexions entre les couches et en opérant indépendamment sur chaque carte d'entités.

En particulier, les types de pooling les plus populaires sont le max et l'average pooling, où les valeurs maximales et moyennes sont prises, respectivement [21].

Max pooling : Chaque opération de pooling sélectionne la valeur maximale de la surface

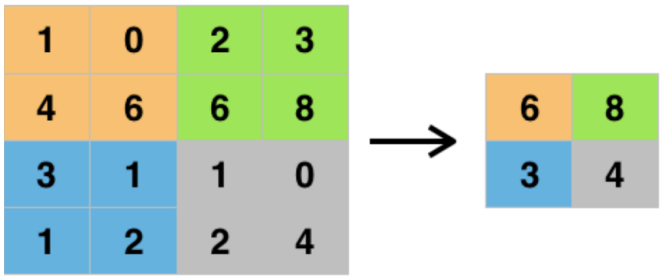


Figure 6 – Max pooling.

Average pooling : Chaque opération de pooling sélectionne la valeur moyenne de la surface

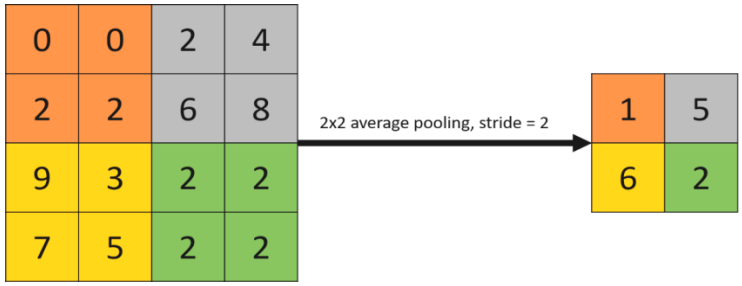


Figure 7 – Average pooling.

Si nous avons une carte d'activation de taille W x W x D, un noyau de regroupement de taille spatiale F et stride S, la taille du volume de sortie peut être déterminée à l'aide de l'équation (3) suivante:

$$\frac{W-F}{S} + 1 \qquad \textbf{(3)}$$

2.3.3.3 Couche de correction (ReLu)

Il est possible d'améliorer l'efficacité du traitement en intercalant entre les couches de traitement une couche qui va opérer une fonction mathématique (fonction d'activation) sur les signaux de sortie.

ReLU (Rectified Linear Units) désigne la fonction réelle non-linéaire définie par l'équation (4) : $$\boldsymbol{ReLU(x) = \max(0, x)} \qquad \textbf{(4)}$$

La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation, comme illustré à la Figure 8.

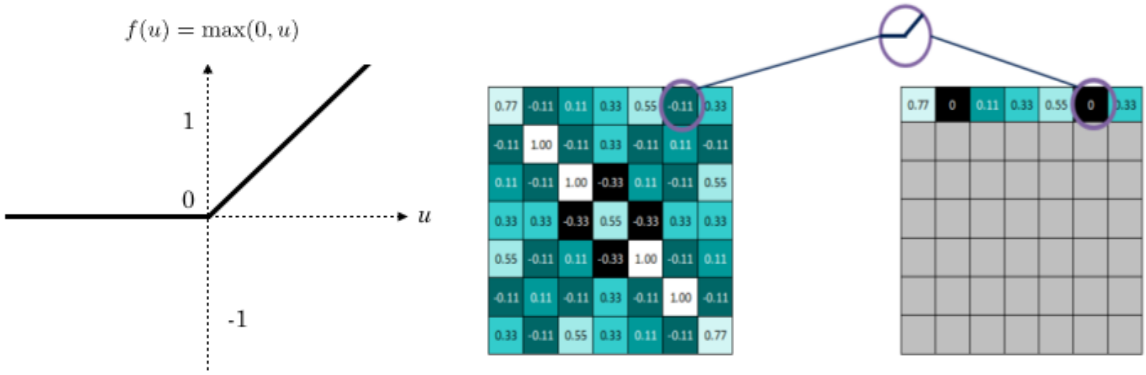


Figure 8 – Fonction d'activation.

2.3.3.4 Couche entièrement connectées (FC)

La couche entièrement connectée (FC) comprend les poids et les biais ainsi que les neurones sont utilisée pour connecter les neurones entre deux couches différentes. La représentation d'entrée est aplatie dans un vecteur de caractéristiques et transmise à travers un réseau de neurones afin de prédire les probabilités de sortie [21]. L'image suivante décrit l'opération d'aplatissement:

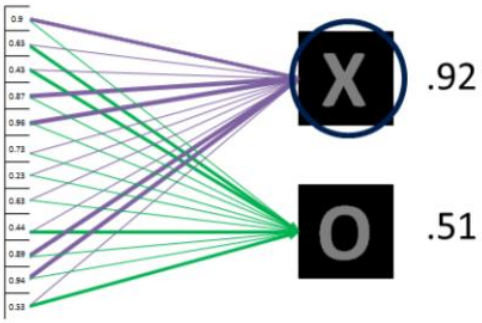


Figure 9 - L'opération d'aplatissement.

Les lignes sont concaténées pour former un vecteur long. Si plusieurs couches d'entrée sont présentes, ses lignes sont également concaténées pour former un vecteur de caractéristiques encore plus long.

Le vecteur de caractéristique est ensuite passé à travers plusieurs couches denses (entièrement connectées). À chaque couche dense, le vecteur de caractéristiques subit les mêmes opérations que dans un réseau de neurones classique. Ces couches sont généralement placées avant la couche de sortie et forment les dernières couches d'une architecture CNN.

2.3.3.5 Couche de perte (Loss)

La couche de perte spécifie comment l'entrainement du réseau pénalise l'écart entre le signal prévu et réel. Elle est normalement la dernière couche dans le réseau.

Diverses fonctions de perte adaptées à différentes tâches peuvent y être utilisées. La fonction « Softmax » permet de calculer la distribution de probabilités sur les classes de sortie.

2.3.4 Les modèles de CNN (pré-entrainés)

Dans cette section, les réseaux de neurones convolutifs utilisés dans ce projet sont :

- VGG16
- VGG19
- ResNet50
- GoogleNet
- MobileNet

2.3.4.1 VGG16

Il s'agit d'une structure du Visual Geometry Group d'Oxford réalisée par Andrea Vedaldi et Andrew Zisserman (en 2017). Une version du réseau de neurones convolutif très connu appelé VGG-Net. C'est un réseau de type encodeur-décodeur, adapté aux images de petite taille, pour le nombre et les dimensions des couches de convolution. Le VGG-16 est constitué de plusieurs couches, dont 13 couches de convolution et 3 entièrement connectés. Il doit donc apprendre les poids de 16 couches. Il prend en entrée une image en couleurs de taille 224 × 224 pixels et la classifie dans une des 1000 classes. Il renvoie donc un vecteur de taille 1000, qui contient les probabilités d'appartenance à chacune des classes. L'architecture du réseau VGG-16 est illustrée par la figure 10. [23]

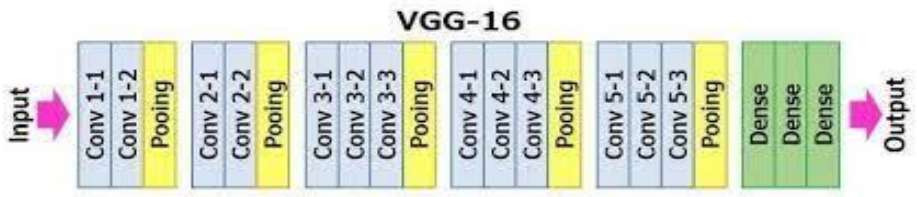


Figure 10 - L'architecture du modèle VGG16.

2.3.4.2 VGG19

VGG19 est un modèle de réseau de neurones convolutif développé par l'équipe de recherche Visual Geometry Group (VGG) de l'Université de Oxford. Il a été publié en 2014 et a remporté plusieurs compétitions de reconnaissance d'images à l'époque.

Le modèle est composé de 19 couches de traitement convolutionnel et de couches fully connected pour effectuer la classification des images [24], comme illustré à Figure 11.

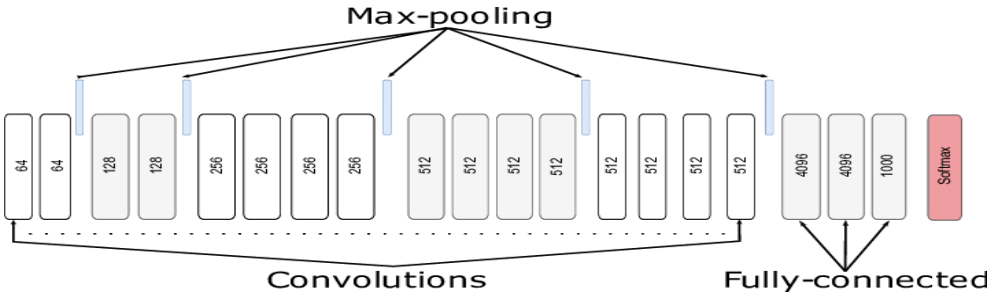


Figure 11 – L'architecture du modèle VGG19.

VGG19 est souvent utilisé comme référence pour évaluer les performances des autres modèles de réseaux de neurones convolutifs et est considéré comme un modèle de base solide pour la reconnaissance d'images. Cependant, étant donné que VGG19 est assez profond et a un nombre élevé de paramètres, il peut être coûteux en termes de ressources pour l'entraîner et le faire fonctionner en temps réel sur des appareils mobiles.

2.3.4.3 ResNet50

ResNet50 (Réseau de Récupération) est un modèle de réseau de neurones profond développé par Microsoft Research en 2015. Il a été présenté pour la première fois dans le papier scientifique intitulé "Deep Residual Learning for Image Recognition".

Différents types de ResNet peuvent être développés en fonction de la profondeur du réseau, illustré par la Figure 12. Le nombre à la fin de ResNet50 spécifie le nombre de couches dans le réseau ou la profondeur du réseau. Nous pouvons concevoir des ResNets de profondeur arbitraire en utilisant les éléments de base des ResNets [25].

La conception de l'architecture est inspirée du VGG-19 et dispose d'un réseau simple à 34 couches auquel des connexions de raccourci et de saut sont ajoutées.

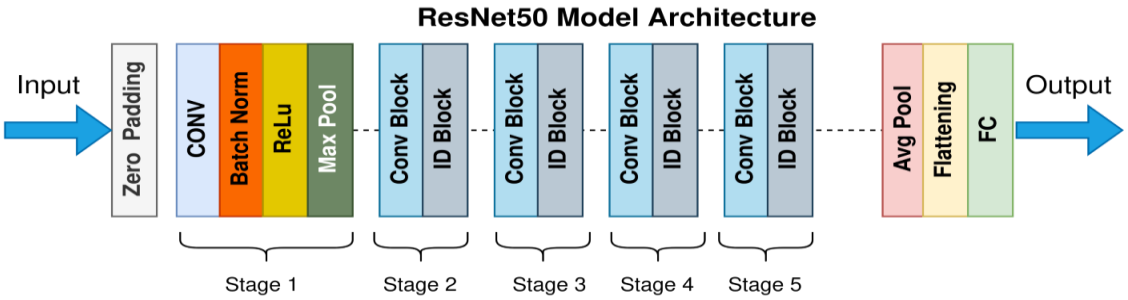


Figure 12 - L'architecture du modèle ResNet50.

2.3.4.4 GoogleNet

GoogleNet est un modèle de réseau de neurones profond développé par Google en 2014 et présenté dans le papier scientifique intitulé "Going deeper with convolutions". Il a remporté le concours d'object detection ImageNet Large Scale Visual Recognition Challenge (ILSVRC) en 2014.L'architecture de GoogleNet est basée sur une architecture en Inception qui utilise plusieurs filtres de convolution de tailles différentes pour capturer des informations à différentes échelles. Cette approche permet de traiter efficacement des images de différentes tailles et de différents angles [26].

GoogleNet comporte également plusieurs couches de moyen-âge de pooling qui sont utilisées pour réduire la taille des images et extraire des caractéristiques globales, comme illustré à la Figure 13. Les couches fully connected utilisent des informations à grande échelle pour effectuer la classification d'images.

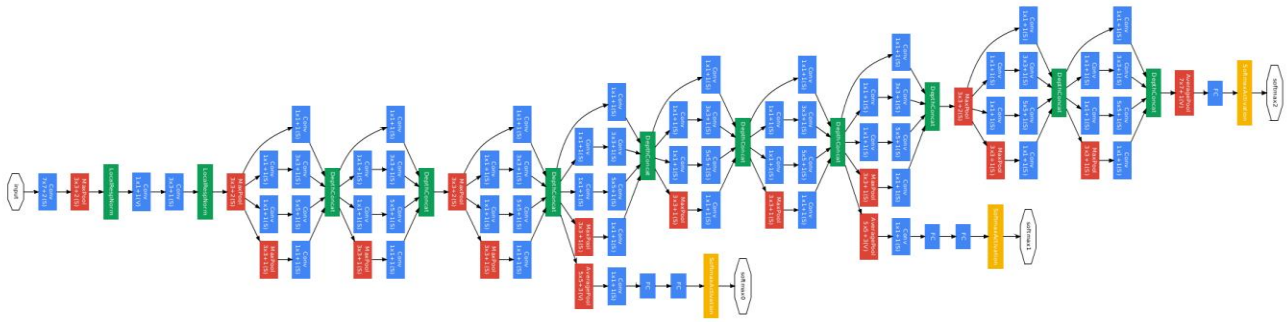


Figure 13 - L'architecture du modèle GoogleNet.

2.3.4.5 MobileNet

MobileNet est une architecture légère développée par « Howard et al. » de Google, le modèle MobileNet est conçu pour être utilisé dans des applications mobiles, cette architecture utilise des convolutions séparables en profondeur. Il réduit considérablement le nombre de paramètres par rapport au réseau avec des convolutions régulières avec la même profondeur dans les filets. Il en résulte des réseaux de neurones profonds légers [27].

Le MobileNet a une architecture de 28 couches, avec une taille d'image d'entrée standard de 224 x 224 x 3. Il est souvent utilisé pour des tâches de classification d'images en temps réel sur des appareils mobiles, mais peut également être utilisé pour des tâches de détection d'objets et de segmentation d'images, comme illustré à la Figure 14.

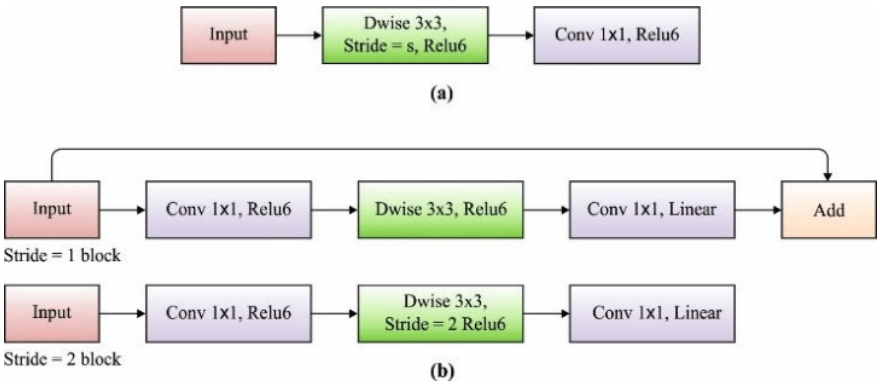


Figure 14 - L’architecture du modèle MobileNet.

2.4 Choix des classificateurs

Dans cette section, nous allons examiner le classificateur utilisés dans ce projet :

2.4.1 K plus proches voisins (KNN)

K nearest neighbors (KNN) ou K plus proche voisins est une technique d'apprentissage supervisé utilisée pour l'estimation statistique et la reconnaissance de modèles. Elle est considérée comme une méthode non paramétrique car elle ne suppose aucune distribution particulière des données. Cette méthode est simple et ne nécessite pas de phase d'entraînement explicite, ce qui la rend rapide à mettre en œuvre.

La méthode KNN suppose que les données sont représentées dans un espace de caractéristiques, où chaque point de données est un vecteur de caractéristiques. Cette méthode utilise une fonction de similarité pour mesurer la proximité entre deux points de données dans l'espace de caractéristiques [28]. Le nombre k est utilisé pour déterminer le nombre de voisins les plus proches qui vont influencer la classification d'un point donné. La similarité entre deux vecteurs est mesurée en utilisant la distance, qui peut être calculée à l'aide de plusieurs types de distance parmi lesquels on trouve [29]:

- La distance Euclidienne : $\boldsymbol{d(x,y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}}$ **(5)**
- La distance de Minkowsky : $\boldsymbol{d(x,y) = [\sum_{i=1}^{n}|x_i - y_i|^p]^{1/p}}$ **(6)**
- La distance de Manhattan : $\boldsymbol{d(x,y) = \sum_{i=1}^{n}|x_i - y_i|}$ **(7)**

Où : x, y sont des vecteurs. p : paramètre

2.4.1.1 L'algorithme des k plus proches voisins

Soit :

- X : ensemble d'entrainement.
- Y: étiquettes de classe de X
- x : individu inconnu

Algorithme 1 k plus proches voisins (KNN)	
1 :	Pour $i = 1$ a m faire
2 :	Calculer la distance d (Xi, x)
3 :	Fin pour
4 :	Construire l'ensemble I contenant des indices pour k plus petite distance d (Xi, x)
5 :	Retourner Étiquette majoritaire pour $\{Yi, ou\ i \in I\}$
6 :	

2.4.2 Random Forest (forêts aléatoires)

La forêt aléatoire (Random Forest) est un algorithme d'apprentissage supervisé très populaire qui peut être utilisé pour résoudre les problèmes de régression ou de classification. Cet algorithme est basé sur un ensemble d'algorithmes d'apprentissage, qui est un processus de combinaison de plusieurs algorithmes pour résoudre un problème complexe et améliorer

les performances du modèle. Il crée de nombreux arbres de décision sur divers sous-ensembles de l'ensemble de données, d'où le nom de "forêt".

Pour prédire la classe d'un nouvel échantillon, chaque arbre de décision donne une prédiction. Les prédictions sont combinées en prenant le vote majoritaire des prédictions des arbres de décision pour obtenir la prédiction finale. Cette méthode est appelée "voting" [30].

La figure suivant explique le fonctionnement et la structure d'algorithme :

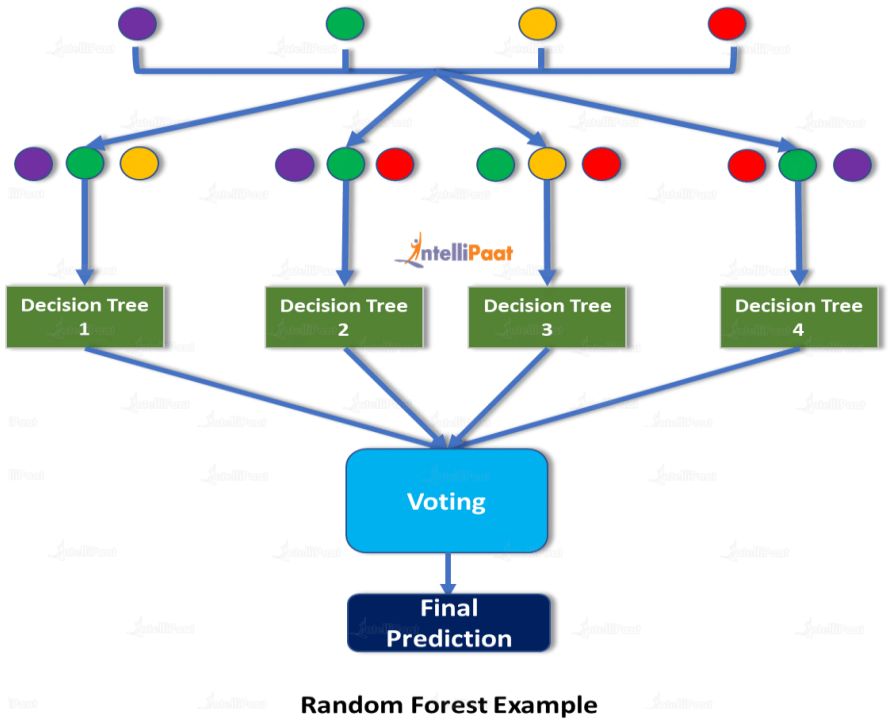


Figure 15 - Structure de l'algorithme RandomForest.

2.4.3 Le perceptron multicouche

Le Multi Layer Perceptron (MLP), ou perceptron multicouches, est un type de classifieur basé sur les réseaux de neurones. Il est composé de plusieurs couches, avec chaque couche étant constituée d'un ou plusieurs neurones formels. Ce modèle est généralement utilisé dans des cas d'apprentissage supervisé, et il utilise l'algorithme de rétro propagation du gradient pour ajuster les poids des connexions.

Le fonctionnement d'un neurone formel est représenté dans la Figure 16. Il effectue une somme pondérée des signaux reçus en entrée, puis applique une fonction d'activation pour obtenir une sortie, notée Y.

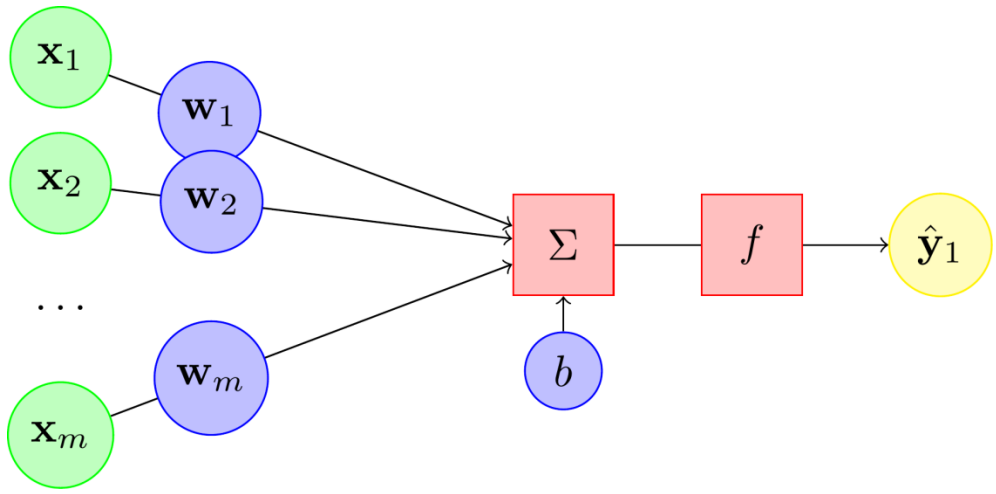


Figure 16 - Le fonctionnement d'un neurone formel.

Les caractéristiques principales du modèle MLP sont les suivantes :

- Il possède une seule couche d'entrée et une seule couche de sortie.
- Il peut également comporter une ou plusieurs couches cachées, situées entre la couche d'entrée et la couche de sortie.
- Chaque neurone est connecté uniquement à tous les neurones de la couche suivante.
- Chaque connexion entre la couche i et la couche suivante j propage l'activation ai de la couche i à la couche j. Cette connexion possède un poids w_{ij} qui détermine l'intensité du signal transmis.

2.5 Conclusion

Dans ce chapitre, nous avons exploré les fondamentaux du Deep Learning, en mettant particulièrement l'accent sur les réseaux de neurones convolutifs (CNN). Nous avons examiné l'architecture et le fonctionnement des différentes couches qui composent ces réseaux.

De plus, nous avons présenté cinq des architectures pré-entraînées les plus populaires, à savoir VGG16, VGG19, ResNet, MobileNet et GoogleNet. Nous avons discuté de leur structure et des avantages qu'elles offrent pour des tâches spécifiques de reconnaissance d'images. Nous avons également intégré trois classifieurs : KNN (K-Nearest Neighbors), RandomForest (Forêt aléatoire) et le Multi-Layer Perceptron (MLP).

Dans le prochain chapitre, nous aborderons les techniques d'optimisation pour la sélection des attributs pour minimiser le nombre total de caractéristiques dans notre processus de détection des maladies gastro-intestinales.

Chapitre 3

Les algorithmes d’optimisations

3.1 Introduction

Ces dernières années, les techniques d'optimisation ont connu une croissance significative, ce qui les rend très utilisées dans de nombreux domaines tels que l'apprentissage en profondeur, résoudre le problème de la sélection d’attributs et l'optimisation des poids synaptiques. Cela implique que les techniques d'optimisation sont devenues des outils essentiels pour améliorer les performances de diverses applications, en permettant de trouver les meilleures solutions possibles aux problèmes complexes. Le développement de nouveaux algorithmes d'optimisation peut donc contribuer à améliorer encore davantage les résultats obtenus dans ces domaines.

Ce chapitre présente deux types d'algorithmes d'optimisation qui ont montré leur efficacité dans le processus de sélection des attributs pour minimiser le nombre total des caractéristiques : les algorithmes bio-inspirés qui ont puisé leur inspiration dans la nature et les systèmes biologiques, les algorithmes d'optimisation méta-heuristiques qui sont des techniques générales d'optimisation qui ne dépendent pas d'un modèle mathématique spécifique du problème. Ces algorithmes explorent l'espace de recherche de manière intelligente et efficace en utilisant des mécanismes de recherche heuristiques.

3.2 Les algorithmes bio-inspirés

Les algorithmes bio-inspirés sont des techniques d'optimisation qui s'inspirent du comportement des organismes vivants pour résoudre des problèmes complexes. Ces techniques sont particulièrement utiles dans des cas où les méthodes classiques d'optimisation ne sont pas efficaces ou ne fournissent pas de solutions satisfaisantes.

Les algorithmes bio-inspirés ont été développés à partir de l'étude de la biologie, de la physique, de la chimie et de la neuroscience. Ils sont basés sur des concepts tels que la sélection naturelle, la coopération, la compétition, la communication, l'adaptation et l'apprentissage.

3.2.1 Les algorithmes génétiques (GA)

Les algorithmes génétiques(GA), proposés par Holland en 1973, sont considérés comme étant les algorithmes évolutionnaires les plus populaires et les plus efficaces pour la résolution de problèmes d'optimisation combinatoire. Ces algorithmes génétiques ont été largement appliqués pour des problèmes d'optimisation en recherche et en ingénierie [31].

Les algorithmes génétiques (GA) simulent de manière détaillée le processus de l'évolution des espèces, présentée dans la théorie de Charles Darwin. Les GA simulent tous les mécanismes de la théorie de l'évolution et non pas que le principe global de mutation et remplacement comme c'est le cas des algorithmes de stratégie d'évolution [31]. En effet, comme nous pouvons le voir sur la Figure 17 dans cette famille d'algorithmes cinq parties essentielles sont à définir en fonction du problème à résoudre l'encodage des solutions en chromosomes, la politique de sélection, les fonctions de croisement et de mutation et une fonction fitness pour évaluer la qualité des chromosomes (solutions).

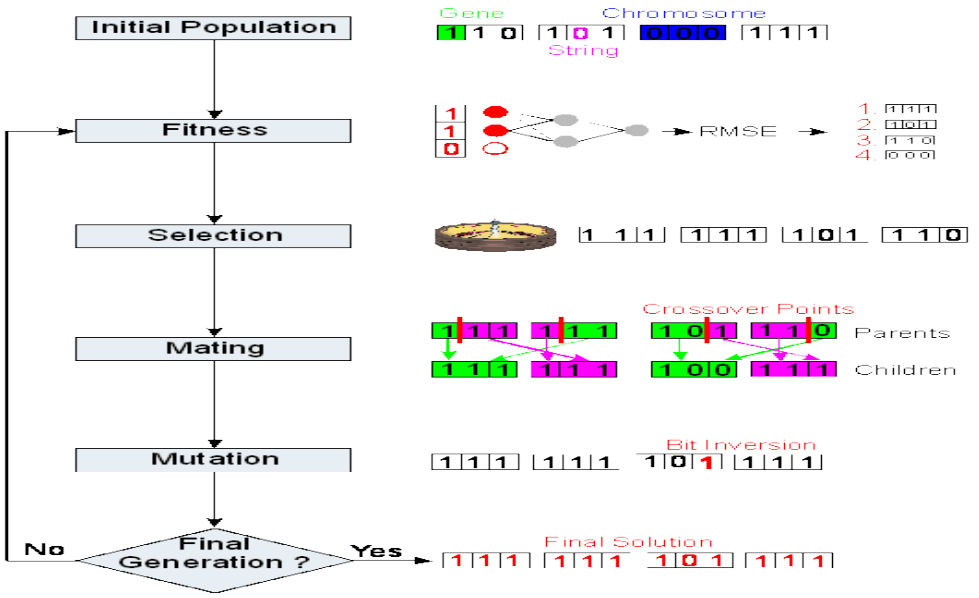


Figure 17 - Organigramme d'un algorithme génétique.

3.2.1.1 Pseudo-code d'algorithme génétique

Voici le pseudo-code pour l'algorithme génétique de base :

Algorithme 2 Genetic Algorithm (GA)

```
1 : Create randomly population P of Popsize individuals
2 : While (stop_condition) FALSE
3 :    P* = ∅
4 : While (Sizeof(P*) ≠ Popsize)
5 : Select p1 and p2 from P
6 :        Crossover p1 and p2 to obtain h1 and h2
7 :        Mutate h1 and h2 to obtain c1 and c2
8 : If [Dis(p1, c1) + Dis(p2, c2)] ≤ [Dis(p1, c2) + Dis(p2, c1)]
9 : If c1 < p1 then P* = P* ∪ {c1} else P* = P* ∪ {p1}
10 : If c2 < p2 then P* = P* ∪ {c2} else P* = P* ∪ {p2}
11 : Else
12 : If c1 < p2 then P* = P* ∪ {c1} else P* = P* ∪ {p2}
13 : If c2 < p1 then P* = P* ∪ {c2} else P* = P* ∪ {p1}
14 : EndWhile
15 :    P = P*
16 :    Evaluate individuals in P using the Diagnostic Sheme
17 : EndWhile
```

Avec :

- P : une population initiale.
- Popsize : un nombre donné d'individus
- stop_condition : la condition d'arrêt.
- $P*$: la nouvelle population.
- p_1, p_2 : deux individus de la population P.
- h_1, h_2 : deux descendants qui obtenu avec une opération de croisement.
- c_1, c_2 : deux descendants modifiés qui obtenu avec une opération de mutation.

3.2.2 L'optimisation par essaim particulaire (PSO)

La méta-heuristiques basée sur les « interactions sociales » entres des « agents » appelés « particules », dans le but d'atteindre un objectif donné dans un espace de recherche commun où chaque particule a une certaine capacité de mémorisation et de traitement de l'information. Cette méta-heuristique d'optimisation stochastique a été proposée en 1995 par James Kennedy socio-psychologue et Russell Eberhart ingénieur électricien [32].

Dans PSO le comportement social est modélisé par une équation mathématique permettant de guider les particules durant leur processus de déplacement [33]. Le déplacement d'une particule est influencé par trois composantes : la composante d'inertie, la composante cognitive et la composante sociale. Chacune de ces composantes reflète une partie de l'équation, figure 18 [34] :

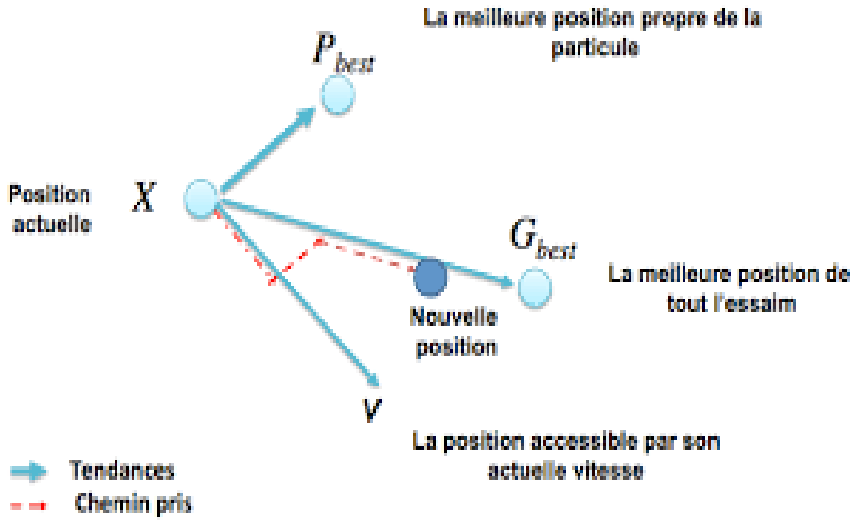


Figure 18 - Le déplacement d'une particule.

Une particule i de l'essaim dans un espace de dimension D est caractérisée, à l'instant t, par :

- X : sa position dans l'espace de recherche.
- V : sa vitesse.
- Pb : la position de la meilleure solution par laquelle elle est passée.
- Pg : la position de la meilleure solution connue de tout l'essaim.
- $f(Pb)$: la valeur de fitness de sa meilleure solution.
- $f(Pg)$: la valeur de fitness de la meilleure solution connue de tout l'essaim.

Le déplacement de la particule i entre les itérations t et $t+1$ se fait selon les deux équations (8) et (9) [32] :

$$\boldsymbol{V(t+1) = V(t) + C1r1\big(Pb(t) - X(t)\big) + C2r2(Pg(t) - X(t))} \quad \textbf{(8)}$$

$$\boldsymbol{X(t+1) = X(t) + V(t+1)} \quad \textbf{(9)}$$

- $C1$ et $C2$: deux constantes qui représentent les coefficients d'accélération, elles peuvent être non constantes dans certains cas selon le problème d'optimisation posé.
- $r1$ et $r2$: deux nombres aléatoires tirés de l'intervalle $[0,1]$.

L'algorithme de base de la méthode PSO proposé, commence par une initialisation aléatoire des particules dans leur espace de recherche, en leurs attribuant une position et une vitesse initiales. À chaque itération de l'algorithme les particules se déplacent selon les équations (8) et (9) et les fonctions objectives (fitness) des particules sont calculées afin de pouvoir calculer la meilleure position de toutes Pg. La mise à jour des Pb et Pg est faite à chaque itération suivant l'Algorithme 3 [34]. Le processus est répété jusqu'à satisfaction du critère d'arrêt.

Algorithme 3 Optimisation par essaim particulaire (PSO)

Initialize population
for $t = 1$: maximum generation
  for $i = 1$: population size
if $f\left(x_{i,d}(t)\right) < f\left(p_i(t)\right)$ **then** $p_t(t) = x_{i,d}(t)$
    $f\left(p_g(t)\right) = \min_t(f(p_t(t)))$
  end
  for $d = 1$: dimension
$v_{i,d}(t+1) = wv_{i,d}(t) + c_1r_1(p_t - x_{i,d}(t)) + c_2r_2(p_g - x_{i,d}(t))$
$x_{i,d}(t+1) = x_{i,d}(t) + v_{i,d}(t+1)$
if $v_{i,d}(t+1) > v_{max}$ **then** $v_{i,d}(t+1) = v_{max}$
elseif $v_{i,d}(t+1) < v_{min}$ **then** $v_{i,d}(t+1) = v_{min}$
end
    if $x_{i,d}(t+1) > x_{max}$ **then** $x_{i,d}(t+1) > x_{max}$
    else if $x_{i,d}(t+1) > x_{min}$ **then** $x_{i,d}(t+1) > x_{min}$
    end
  end
 end
end

3.3 Les algorithmes d’optimisation méta-heuristiques

Dans cette partie, nous allons introduire deux algorithmes d’optimisations méta-heuristiques : Aquila Optimizer (AO) des comportements de l'Aquila dans la nature pendant le processus de capture de la proie, et l'algorithme Chaos Game Optimization (CGO) est inspire de la théorie du chaos pour créer une méta-heuristique d'optimisation basée sur les fractales et les propriétés du chaos.

3.3.1 Aquila Optimizer (AO)

Aquila Optimizer (AO) s'inspire du comportement de l'aigle Aquila dans la nature lorsqu'il chasse sa proie. L'aigle Aquila est connu pour sa capacité à chasser efficacement en utilisant des stratégies de recherche et de capture sophistiquées.

L'algorithme AO s'appuie sur ces comportements pour créer une méta-heuristique d'optimisation qui imite le processus de capture de proie de l'aigle Aquila. Par exemple, AO peut inclure des stratégies de recherche adaptatives, des mouvements de balayage pour explorer l'espace de recherche de manière efficace et des mécanismes de sélection pour choisir les meilleures solutions.

3.3.1.1 Inspiration et comportement d'Aquila pendant la chasse

L'algorithme AO s'inspire des méthodes de chasse de l'Aquila, connue pour sa rapidité, son agilité et sa polyvalence dans la capture de différents types de proies.

L'algorithme émule ces méthodes dans son processus d'optimisation en utilisant quatre techniques : le vol en haute altitude avec une plongée verticale, le vol en contour avec une attaque en plané court, le vol bas avec une attaque en descente lente, et la marche et saisie de proie. L'algorithme exploite ces méthodes pour explorer et exploiter efficacement l'espace de recherche, améliorant ainsi les performances dans la résolution de problèmes d'optimisation complexes.

3.3.1.2 Modèle mathématique de l'AO

L'algorithme AO proposé simule le comportement de l'Aquila lors de la chasse, en montrant les actions de chaque étape de la chasse. Ainsi, les procédures d'optimisation de l'algorithme AO proposé sont représentées dans quatre méthodes : la sélection de l'espace de recherche par un vol haut avec une descente verticale, l'exploration dans un espace de recherche divergent par un vol de contour avec une attaque de courte glissade, l'exploitation dans un espace de recherche convergent par un vol bas avec une attaque de descente lente, et la capture de la proie par une descente rapide.[35]

L'algorithme AO peut passer des étapes d'exploration aux étapes d'exploitation en utilisant différents comportements en fonction de la condition suivante : si $t \leq (\frac{2}{3}) \times T$, les étapes d'exploration seront exécutées ; sinon, les étapes d'exploitation seront exécutées.

Nous avons modélisé les comportements de l'Aquila comme un paradigme d'optimisation mathématique, qui consiste à déterminer la meilleure solution en fonction de contraintes spécifiques. Le modèle mathématique de l'AO est proposé comme suit.

- **Étape 1 : Exploration étendue (X1)**

Dans la première méthode (X1), l'Aquila reconnaît la zone de la proie et sélectionne la meilleure zone de chasse en effectuant un vol haut avec une descente verticale. Ici, l'AO explore largement depuis un vol haut pour déterminer la zone de l'espace de recherche où se trouve la proie. La figure 19 montre le comportement de l'Aquila lors d'un vol haut avec une descente verticale. Ce comportement est présenté mathématiquement par l'équation (10).

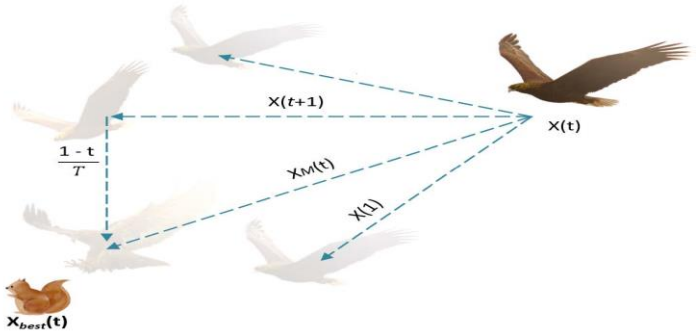


Figure 19 - Le comportement de l'Aquila s'envole avec le vol vertical. [35]

$$\boldsymbol{X_1\,(t+1) = X_{best}(t) \times \left(1 - \frac{t}{T}\right) + (X_M(t) - X_{best}(t) * rand)} \quad \textbf{(10)}$$

Où $X_1\,(t+1)$ est la solution de la prochaine itération t, qui est générée par la première méthode de recherche (X_1). $X_{best}(t)$ est la meilleure solution obtenue jusqu'à l'itération t, cela reflète l'endroit approximatif de la proie. L'équation $\left(1 - \frac{t}{T}\right)$ est utilisée pour contrôler la recherche étendue (exploration) à travers le nombre d'itérations. $X_M(t)$ représente la valeur moyenne des emplacements des solutions actuelles connectées à l'itération t, qui est calculée à l'aide de l'équation (11). $rand$ est une valeur aléatoire entre 0 et 1. t et T présentent le courant itération et le nombre maximal d'itérations, respectivement.

$$\boldsymbol{X_M(t) = \frac{1}{N}\sum_{i=1}^{N} X_i(t), \forall j = 1, 2, \ldots, Dim} \quad \textbf{(11)}$$

Où, Dim est la taille de la dimension du problème et N est le nombre de solutions candidates (taille de la population). [35]

- **Étape 2 : Exploration restreinte (X2)**

Dans la deuxième méthode (X2), lorsque la zone de la proie est trouvée à partir d'un plan élevé, l'Aquila tourne au-dessus de la proie cible, prépare le terrain, puis attaque. Cette méthode s'appelle vol de contour avec attaque en plané court. Ici, AO explore de près la zone sélectionnée de la proie cible en préparation de l'attaque. La figure 20 montre le comportement du vol de contour d'Aquila avec une courte attaque en plané. Ce comportement est présenté mathématiquement comme dans l'équation (12).

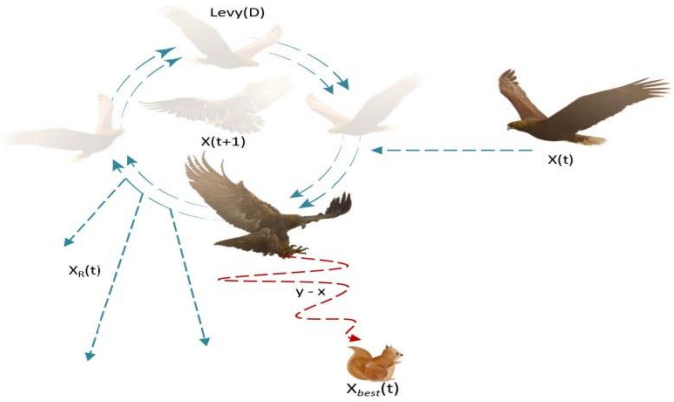


Figure 20 - Le comportement du vol de contour d'Aquila. [35]

$$\boldsymbol{X_2(t+1) = X_{best}(t) \times Levy(D) + X_R(t) + (y - x) * rand} \quad \textbf{(12)}$$

Où $X_2(t+1)$est la solution de la prochaine itération de t, qui est générée par la deuxième méthode de recherche (X_2). D est l'espace des dimensions et $Levy(D)$ est la fonction de distribution des vols de prélèvement, qui est calculée à l'aide de l'équation (13). $X_R(t)$ est une solution aléatoire prise dans l'intervalle de [1 N] à la $i^{\text{ème}}$ itération. [35]

$$\boldsymbol{Levy\ (D) = s \times \frac{u \times \sigma}{|v|^{\frac{1}{\beta}}}} \quad \textbf{(13)}$$

Où s est une constante fixée à 0,01,u et v sont des nombres aléatoires compris entre 0 et 1, σ est calculé à l'aide de l'équation (14).

$$\boldsymbol{\sigma = \left(\frac{T(1+\beta) \times sine(\frac{\pi\beta}{2})}{T(\frac{1+\beta}{2}) \times \beta \times 2^{(\frac{\beta-1}{2})}} \right)} \quad \textbf{(14)}$$

Où $\boldsymbol{\beta}$est une valeur constante fixée à 1,5. Dans l'équation (12), y et x sont utilisés pour présenter la forme en spirale dans la recherche, qui sont calculées comme suit.

$$\boldsymbol{y = r \times \cos(\theta)} \quad \textbf{(15)}$$

$$\boldsymbol{x = r \times \sin(\theta)} \quad \textbf{(16)}$$

Où :

$$\boldsymbol{r = r_1 + U \times D_1} \quad \textbf{(17)}$$

$$\boldsymbol{\theta = -\omega \times D_1 + \theta_1} \quad \textbf{(18)}$$

$$\boldsymbol{\theta_1 = \frac{3 \times \pi}{2}} \quad \textbf{(19)}$$

$\mathbf{r_1}$ prend une valeur comprise entre 1 et 20 pour fixer le nombre de cycles de recherche, et U est une petite valeur fixée à 0,00565, $\mathbf{D_1}$ est un nombre entier de 1 à la longueur de l'espace de recherche (Dim), et $\boldsymbol{\omega}$ est une petite valeur fixée à 0,005. La figure 21 montre le comportement de l'AO en forme de spirale.

- **Étape 3 : Exploitation étendue (X3)**

Dans la troisième méthode ($\boldsymbol{X_3}$), lorsque la zone de la proie est spécifiée avec précision et que l'Aquila est prêt pour l'atterrissage et l'attaque, l'Aquila descend verticalement avec une attaque préliminaire pour découvrir la réaction de la proie.

Figure 21 - Le comportement de l'AO en forme de spirale. [35]

Cette méthode appelée vol bas avec attaque en descente lente. Ici, AO exploite la zone sélectionnée de la cible pour se rapprocher de la proie et attaquer. La figure 22 montre le comportement de l'Aquila en vol à basse altitude avec une attaque en descente lente. Ce comportement est représenté mathématiquement comme dans l'équation (20).

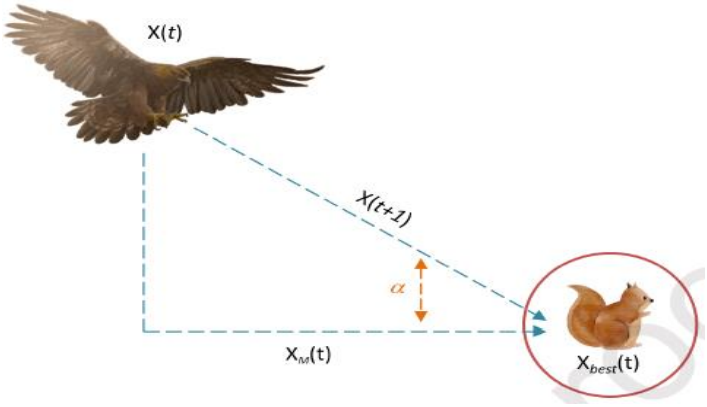


Figure 22 - La dépression d'Aquila léger avec une attaque en descente lente.

$$\boldsymbol{X_3(t+1) = \left(X_{best}(t) - X_M(t)\right) \times \propto - rand + ((UB - LB) \times rand + LB) \times \delta} \quad \textbf{(20)}$$

Où $\boldsymbol{X_3(t+1)}$ est la solution de la prochaine itération de t, qui est générée par la troisième méthode de recherche ($\boldsymbol{X_3}$), $\boldsymbol{X_{best}(t)}$ fait référence à l'emplacement approximatif de la proie jusqu'à la $\boldsymbol{t^{ème}}$ itération (la meilleure solution obtenue), et $\boldsymbol{X_M(t)}$ désigne la valeur moyenne de la solution actuelle à la $\boldsymbol{t^{ème}}$ itération, qui est calculée à l'aide de l'équation (11). $\boldsymbol{rand}$ est une valeur aléatoire comprise entre 0 et 1. $\boldsymbol{\delta}$ et $\propto$ sont les paramètres d'ajustement de l'exploitation fixés dans cet article à une petite valeur (0,1),$\boldsymbol{LB}$ désigne la borne inférieure et $\boldsymbol{UB}$ désigne la borne supérieure du problème donné. [35]

- **Étape 4 : Exploitation restreinte (X4)**

Dans la quatrième méthode (X_4), lorsque l'Aquila s'approche de la proie, l'Aquila attaque la proie au-dessus de la terre selon ses mouvements stochastiques. Cette méthode s'appelle marcher et attraper une proie. Ici, et enfin, AO attaque la proie au dernier emplacement. La figure 23 montre le comportement de l'Aquila qui marche et attrape sa proie. Ce comportement est présenté mathématiquement comme dans l'équation (21).

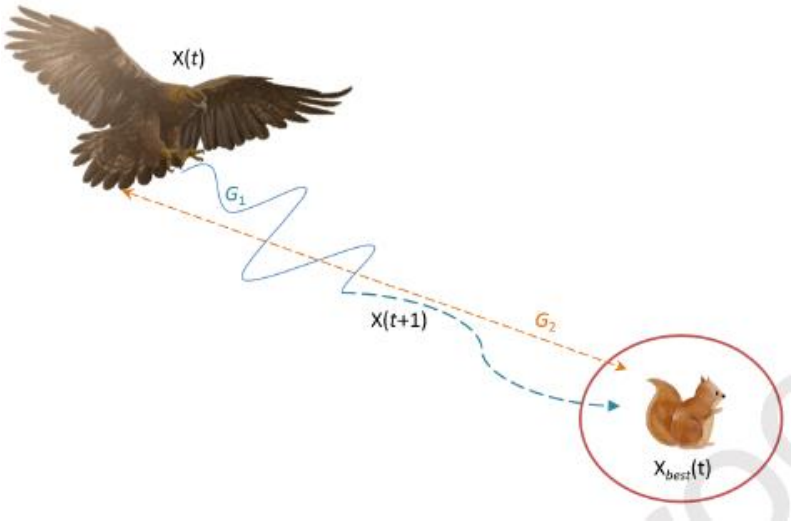


Figure 23 - Le comportement de l'Aquila marche et attrape sa proie. [35]

$$\boldsymbol{X_4(t+1) = QF \times X_{best}(t) - (G_1 \times X(t) \times rand) - G_2 \times Levy(D) + rand \times G_1} \quad \textbf{(21)}$$

Où $\boldsymbol{X_4(t+1)}$est la solution de la prochaine itération de t, qui est générée par la quatrième méthode de recherche ($\boldsymbol{X_3}$). $\boldsymbol{QF}$ désigne une fonction de qualité utilisée pour équilibrer les stratégies de recherche, qui est calculée à l'aide de l'équation (24). $\boldsymbol{G_1}$ désigne divers mouvements de l'AO qui sont utilisés pour suivre la proie pendant l'élope, qui est généré à l'aide de l'équation (25). $\boldsymbol{G_2}$ présente des valeurs décroissantes de 2 à 0, qui dénotent la pente de vol de l'AO qui est utilisée pour suivre la proie pendant l'élope du premier emplacement (1) au dernier emplacement (t), qui est généré à l'aide de l'équation (24). $\boldsymbol{X(t)}$ est la solution courante à la $\boldsymbol{t^{ème}}$ itération.

$$\boldsymbol{QF(t) = t^{\frac{2\times rand()-1}{(1-T)^2}}} \qquad \textbf{(22)}$$

$$\boldsymbol{G_1 = 2 \times rand() - 1} \qquad \textbf{(23)}$$

$$\boldsymbol{G_2 = 2 \times (1 - \frac{t}{T})} \qquad \textbf{(24)}$$

$\boldsymbol{QF(t)}$ est la valeur de la fonction de qualité à la $\boldsymbol{t^{ème}}$ itération, et $\boldsymbol{rand}$ est une valeur aléatoire entre 0 et 1. $\boldsymbol{t}$ et $\boldsymbol{T}$ présentent respectivement l'itération courante et le nombre maximal d'itérations. $\boldsymbol{Levy(D)}$ est la fonction de répartition des vols de prélèvement calculée à l'aide de l'équation (15). La figure 24 montre les effets de la fonction de qualité ($\boldsymbol{QF}$), $\boldsymbol{G1}$ et $\boldsymbol{G2}$ sur le comportement de l'AO. [35]

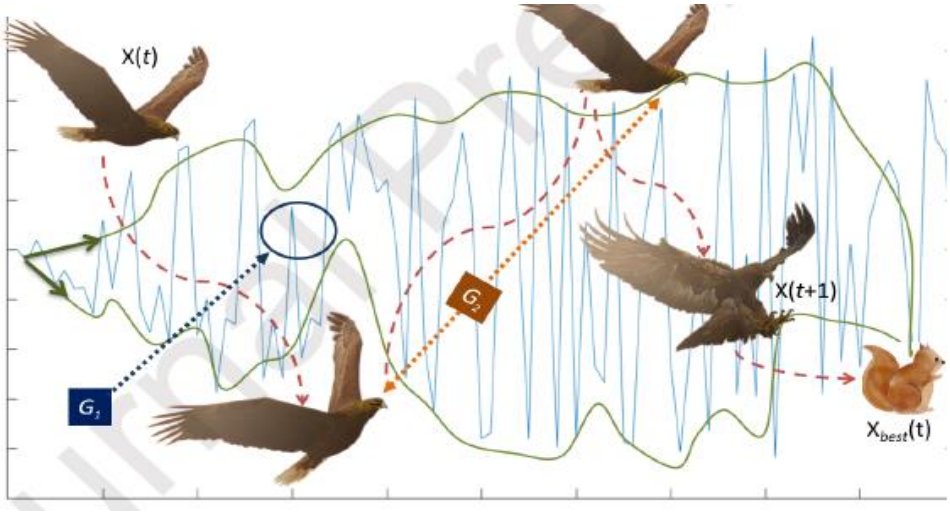


Figure 24 - Les effets de la fonction de qualité (QF), G1 et G2. [35]

3.3.1.3 Pseudo-code de l'Aquila Optimizer (AO)

Le pseudo-code de l'algorithme est détaillé dans l'algorithme 4. [35]

Algorithme 4 Aquila Optimizer

Initialization phase:
Initialize the population X of the GEO.
Initialize the parameters of the GEO (i. e. , $\delta, \propto$, etc).
while (The end condition is not met) **do**
    Calculate the tness function values.
$X_{best}(t) =$ Determine the best obtained solution according to the tness values.
for ($i = 1,2\ldots, N$) **do**
        Update the mean value of the current solution $X_M(t)$.
        Update the x, y, G1, G2, Levy(D), etc.

If $t \leq \left(\frac{2}{3}\right) * T$ **then**
If rand ≤ 0.5 **then**
{**Step 1**: **Expanded exploration** (**X1**)}
            Update the current solution using Equation (10).
if Fitness $(X_1(t+1)) < Fitness(X(t))$ **then**
                $X(t) = (X_1(t+1))$
if Fitness$(X_1(t+1)) < Fitness(X_{best}(t))$ **then**
$X_{best}(t) = X1(t+1)$
                end if
            end if
        else
{**Step 2**: **Narrowed exploration** (**X2**)}
            Update the current solution using Equation (12).
if Fitness$(X_2(t+1)) < Fitness(X(t))$ **then**
$X_{best}(t) = X2(t+1)$
end if
        end if
    end if
else
if rand $\leq$ 0.5 then
{**Step 3**: **Expanded exploitation** (**X3**)}
            Update the current solution using Equation (20).
if Fitness$(X_3(t+1)) < Fitness(X(t))$ **then**
                $X(t) = (X_3(t+1))$
if Fitness$(X_3(t+1)) < Fitness(X_{best}(t))$ **then**
$X_{best}(t) = X_3(t+1)$
                end if
            end if
        else
            { **Step 4**: **Narrowed exploitation** (**X4**)}
            Update the current solution using Equation (21).
if Fitness$(X_4(t+1)) < Fitness(X(t))$ **then**

```
                    X(t) = (X_4(t + 1))
if Fitness(X_4(t + 1)) < Fitness(X_best(t)) then
X_best(t) = X_4(t + 1)
                end if
            end if
        end if
    end if
end for
end while
return The best solution (X_best()).
```

3.3.2 Chaos Game Optimization (CGO)

Chaos Game Optimization (CGO) est un algorithme méta heuristique basé sur les principes de la théorie du chaos et le concept de fractales. L'algorithme utilise la méthodologie Chaos Game pour générer de nouvelles solutions et explorer l'espace de recherche. Les fractales ont été utilisées dans divers algorithmes méta heuristiques car elles présentent une autosimilarité, qui est une propriété importante dans les problèmes d'optimisation. L'utilisation de fractales dans CGO permet l'exploration de l'espace de recherche d'une manière unique et efficace.

3.3.2.1 Inspiration et comportement Chaos Game

La théorie du chaos est une branche des mathématiques qui étudie les systèmes dynamiques sensibles aux conditions initiales. Ces systèmes présentent des motifs primaires tels que des boucles, des modèles répétés et des fractales, qui les rendent auto-similaires et auto-organisés. La théorie du chaos montre que de petits changements dans les conditions initiales peuvent entraîner des différences extrêmes dans les conditions futures des systèmes. Les fractales, qui sont des formes géométriques se répétant à différentes échelles, sont souvent observées dans les processus chaotiques. Dans la figure 25 un exemple célèbre est l'ensemble de Mandelbrot, qui présente une autosimilarité à différentes échelles. Le jeu du chaos est une méthode utilisée pour créer des fractales en utilisant des formes polygonales et des points initiaux aléatoires. En répétant un processus itératif basé sur des fractions de distances entre les points, une fractale est générée. Par exemple, en utilisant un triangle de Sierpinski, une fractale avec un nombre arbitraire de sommets peut être créée, appelée Simplexe de Sierpinski [36].

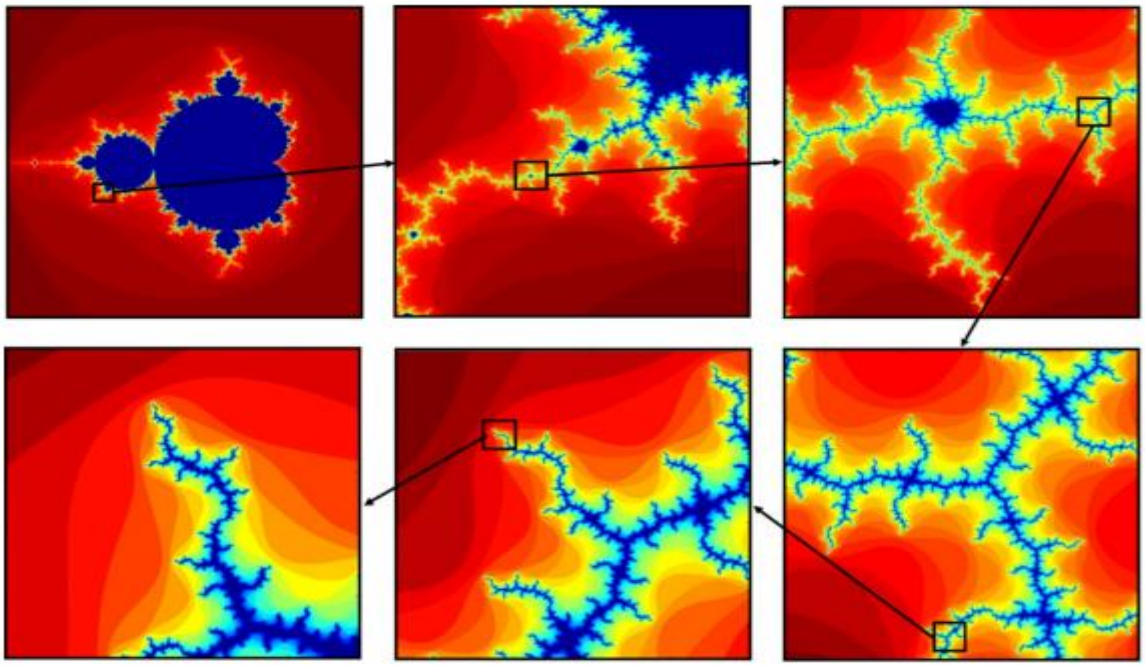


Figure 25 - Auto-similitude de Mandelbrot à différentes échelles. [36]

Comme exemple simple, la création étape par étape d'un triangle de Sierpinski par la méthodologie du jeu du chaos est présentée. Tout d'abord, trois sommets sont sélectionnés afin de créer la forme principale du fractal qui résulte dans une forme de triangle dans ce cas. Chacun des sommets sélectionnés est marqué par l'une des couleurs rouge, bleue et verte. Un dé est pris qui a deux faces rouges, deux faces bleues et deux faces vertes. Un point initial aléatoire est sélectionné comme point de départ du fractal qui est considéré comme une graine dans cet exemple. Lorsque le dé est lancé, en fonction de la couleur qui apparaît, la graine dans le point initial est déplacée vers le sommet correspondant à la moitié de la distance entre la graine et le sommet. La nouvelle position de la graine est utilisée dans l'itération suivante comme point de départ où le dé est lancé à nouveau et la graine est déplacée vers le sommet prévu en conséquence. En lançant le dé plusieurs fois, le triangle de Sierpinski est obtenu comme forme finale. [36]

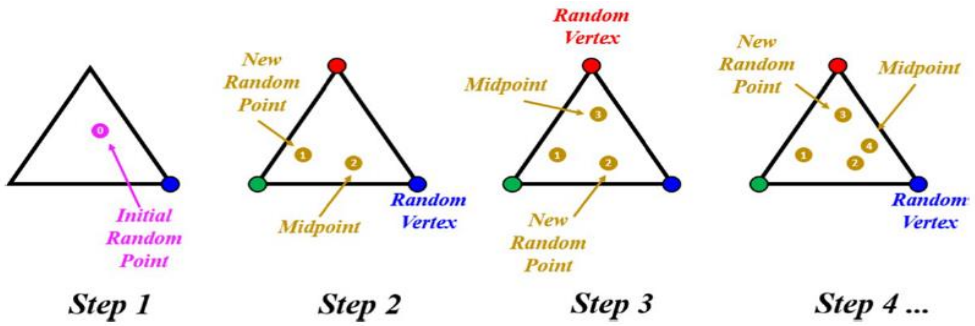


Figure 26 - La méthodologie du jeu du chaos pour créer le triangle de Sierpinski. [36]

La vue schématique de la méthodologie présentée est représentée dans la figure 26 tandis que la forme finale du triangle de Sierpinski et sa similarité d'autorépétition dans différentes échelles sont présentées dans la figure 27.

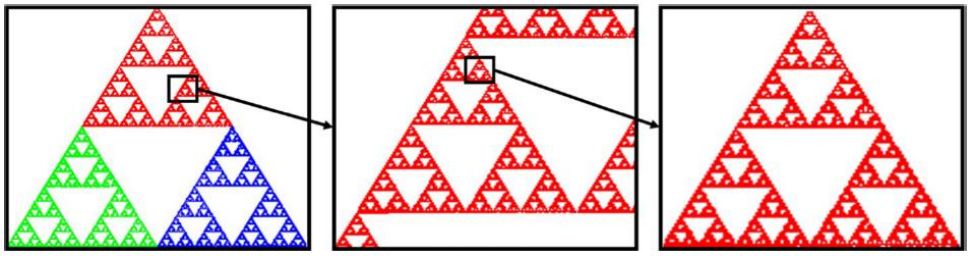


Figure 27 - L'autosimilarité du triangle de Sierpinski à différentes échelles. [36]

3.3.2.2 Modèle mathématique

Les concepts de base des fractales et du jeu du chaos sont utilisés pour formuler un modèle mathématique pour l'algorithme CGO. Étant donné que de nombreux algorithmes d'évolution naturelle maintiennent une population de solutions qui évoluent grâce à des altérations aléatoires et une sélection, l'algorithme CGO considère un certain nombre de candidats de solutions (X) qui représentent des graines éligibles à l'intérieur d'un triangle de Sierpinski. Dans cet algorithme, chaque candidat de solution (X_i) se compose de certaines variables de décision ($X_{i,j}$) qui représentent la position de ces graines éligibles à l'intérieur d'un triangle de Sierpinski. [36]

Le triangle de Sierpinski est considéré comme l'espace de recherche pour les candidats de solutions dans l'algorithme d'optimisation. La présentation mathématique de ces aspects est la suivante :

$$\mathbf{X}=\begin{bmatrix}\mathbf{X_1}\\ \mathbf{X_2}\\ \vdots\\ \mathbf{X_i}\\ \vdots\\ \mathbf{X_n}\end{bmatrix}=\begin{bmatrix}\mathbf{x_1^1}\ \mathbf{x_1^2}\ \cdots\ \mathbf{x_1^j}\ \cdots\ \mathbf{x_1^d}\\ \mathbf{x_2^1}\ \mathbf{x_2^2}\ \cdots\ \mathbf{x_2^j}\ \cdots\ \mathbf{x_2^d}\\ \vdots\quad\vdots\quad\vdots\quad\ddots\ \vdots\\ \mathbf{x_i^1}\ \mathbf{x_i^2}\ \cdots\ \mathbf{x_i^j}\ \cdots\ \mathbf{x_i^d}\\ \vdots\quad\vdots\quad\vdots\quad\ddots\ \vdots\\ \mathbf{x_n^1}\ \mathbf{x_n^2}\ \cdots\ \mathbf{x_n^j}\ \cdots\ \mathbf{x_n^d}\end{bmatrix}, \quad \begin{cases}\mathbf{i=1,2,\dots,n.}\\ \mathbf{i=1,2,\dots,d.}\end{cases} \qquad \textbf{(25)}$$

Où n est le nombre de graines éligibles (solutions candidates) à l'intérieur du triangle de Sierpinski (espace de recherche), et d est la dimension de ces graines.

Les positions initiales de ces graines éligibles sont déterminées aléatoirement dans l'espace de recherche comme suit :

$$\mathbf{x_i^j(0)=x_{i,min}^j+rand.(x_{i,max}^j-x_{i,min}^j)},\begin{cases}\mathbf{i=1,2,\dots,n.}\\ \mathbf{j=1,2,\dots d.}\end{cases} \qquad \textbf{(26)}$$

Où $x_i^j(0)$ détermine la position initiale des graines éligibles ; $x_{i,min}^j$ et $x_{i,max}^j$ sont les valeurs minimales et maximales admissibles pour la $j^{ème}$ variable de décision de la $i^{ème}$ solution candidate ; rand est un nombre aléatoire dans l'intervalle de [0,1].

Comme décrit précédemment, les principes de la théorie du chaos concernent l'existence de certains modèles primaires dans le comportement des systèmes dynamiques qui les représentent comme des systèmes auto-similaires et auto-organisés. Les graines initiales créées (graines éligibles) représentent les modèles primaires des systèmes dynamiques basés sur la théorie du chaos. L'éligibilité de ces graines à être les modèles primaires (l'autosimilarité) peut être modélisée avec des solutions candidates (X) pour un problème d'optimisation. Les solutions candidates avec les niveaux d'éligibilité les plus élevés et les plus bas sont respectivement équivalentes aux meilleures et aux pires valeurs de fitness. [36]

Le concept principal de ce modèle mathématique est de créer différentes graines éligibles à l'intérieur de l'espace de recherche afin de compléter la forme globale d'un triangle de Sierpinski. À cet égard, la méthodologie de création de nouvelles graines à l'intérieur d'un

triangle de Sierpinski est également utilisée. Pour chacune des graines éligibles dans l'espace de recherche (X_i), un triangle temporaire est dessiné avec trois graines comme suit :

- La position du Global Best trouvé jusqu'à présent (GB),
- La position du groupe moyen (MG_i),
- La position de la $i^{ème}$ solution candidate (X_i) en tant que graine sélectionnée.

Le GB fait référence à la meilleure solution candidate trouvée jusqu'à présent qui a les niveaux d'éligibilité les plus élevés et le MG_i fait référence aux valeurs moyennes de certaines graines éligibles sélectionnées au hasard avec une probabilité égale d'inclure la graine éligible initiale actuellement considérée (X_i). La vue schématique de la création de triangles temporaires est illustrée à la figure 26 tandis que la description schématique détaillée de cet aspect est présentée à la figure 29. [36]

L'objectif principal de la création de triangles temporaires dans la méthodologie du jeu du chaos est de générer de nouvelles graines éligibles dans l'espace de recherche. Chaque triangle temporaire, représenté par trois sommets d'un triangle de Sierpinski et trois graines (GB, MG_i, X_i), utilise un dé pour déterminer les déplacements des graines. La première graine est déplacée vers le GB ou le MG_i en fonction du résultat du lancer de dé (vert ou rouge). Ce processus est contrôlé par une fonction de génération d'entiers aléatoires qui ne produit que deux entiers (0 et 1) pour choisir les faces vertes ou rouges. De plus, des facteurs aléatoires sont utilisés pour limiter les déplacements des graines dans l'espace de recherche. Cette approche vise à créer de nouvelles graines et à contrôler leurs mouvements dans le cadre de la méthodologie du jeu du chaos. Une présentation schématique du processus décrit pour la première graine est présentée sur la figure 30 tandis que la présentation mathématique de ce processus est la suivante :

$$\boldsymbol{Seed_i^1 = X_i + \alpha_i \times (\beta_i \times GB - \gamma_i \times MG_i),\ i = 1, 2, \dots, n.} \qquad \textbf{(27)}$$

Où X_i est la $i^{ème}$ solution candidate, GB est la meilleure solution mondiale trouvée à ce jour et MG_i est la valeur moyenne de certaines graines éligibles sélectionnées. i est le factoriel généré aléatoirement pour modéliser les limitations de mouvement des graines

tandis que chacun des i et i représente un nombre entier aléatoire de 0 ou 1 pour modéliser la possibilité de lancer un dé. [35]

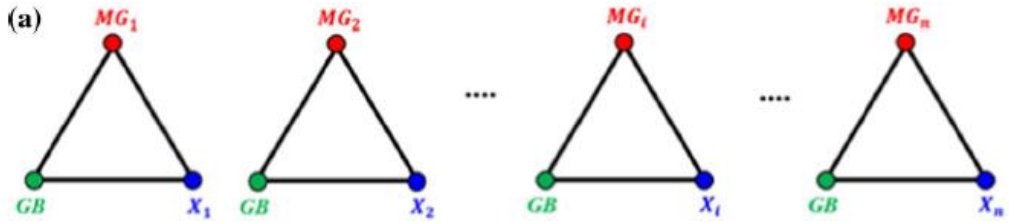


Figure 28 - La vue schématique de la création de triangles temporaires. [36]

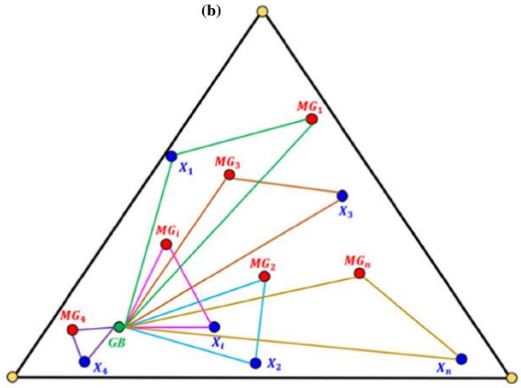


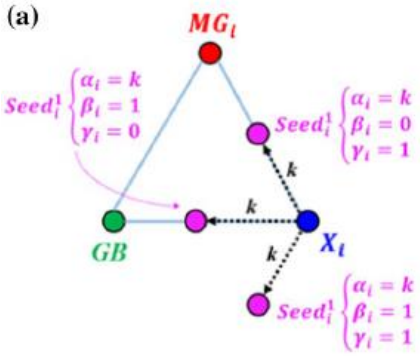
Figure 29 - triangles temporaires dans l'espace de recherche. [36]

Pour la deuxième graine (GB), un dé avec trois faces bleues et trois faces rouges est utilisé. Le dé est lancé et en fonction de la couleur qui sort (bleu ou rouge), la graine dans GB est déplacée vers X_i (face bleue) ou MG_i (face rouge). Cet aspect est modélisé comme décrit pour la première graine. Si la face bleue sort, cette graine se déplace vers X_i, mais si la face rouge sort, la graine se déplace vers MG_i.

Tout comme le processus de déplacement de la première graine, la deuxième graine peut se déplacer vers un point des lignes connectées entre X_i et MG_i et ce mouvement est limité en utilisant des factorielles générées de manière aléatoire. Une présentation schématique du processus décrit pour la deuxième graine est présentée dans la figure 31, tandis que la présentation mathématique de ce processus est la suivante:

$$\boldsymbol{Seed_i^2 = GB + \alpha_i \times (\beta_i \times X_i - \gamma_i \times MG_i), i = 1, 2, \dots, n.} \quad \textbf{(28)}$$

Où α_i est le factoriel généré aléatoirement pour modéliser les limitations de mouvement des graines tandis que chacun des γ_i et β_i représente un nombre entier aléatoire de 0 ou 1 pour modéliser la possibilité de lancer un dé. Les autres paramètres sont tels que décrits pour la première graine. [36]



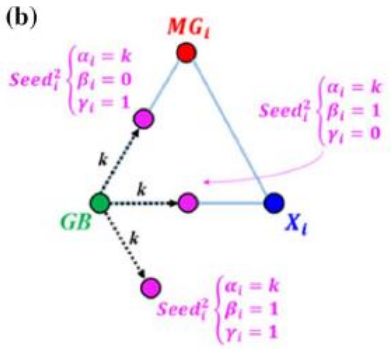


Figure 30 - La vue schématique pour le premier [36]. Figure 31- La vue pour la seconde [36].

MG_ien tant que troisième graine, un dé a des faces bleues et vertes et il est lancé et en fonction de la couleur qui apparaît (bleu ou vert), la graine se déplace vers le X_i (face bleue) ou le GB (face verte). Cet aspect est modélisé par une fonction de génération d'entiers aléatoires qui ne crée que deux entiers comme 0 et 1 pour la possibilité de sélectionner des faces bleues ou vertes. Il convient de noter que la graine peut également se déplacer dans la direction des lignes connectées entre le X_i et le GB. Certains factoriels aléatoires sont également utilisés pour atteindre cet objectif, comme :

$$\boldsymbol{Seed_i^3 = MG_i + \alpha_i \times (\beta_i \times X_i - \gamma_i \times GB), i = 1, 2, \dots, n.} \qquad \textbf{(29)}$$

Une présentation schématique du processus décrit pour la troisième graine est présentée sur la figure 32. Afin de mettre en œuvre la phase de mutation dans les mises à jour de position des graines éligibles dans l'espace de recherche, un autre processus est également utilisé pour générer la quatrième graine. Les mises à jour de position pour cette graine sont effectuées sur la base de certaines modifications aléatoires des variables de décision choisies au hasard. [36] Une présentation schématique du processus décrit pour la quatrième graine est présentée sur la figure 33 tandis que cet aspect est modélisé mathématiquement comme suit :

$$\boldsymbol{Seed_i^4 = X_i\left(x_i^k = x_i^k + R\right),\ \ k = [1, 2, \dots, d,]} \qquad \textbf{(30)}$$

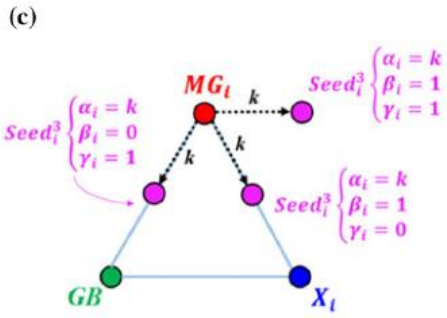


Figure 32 - La vue schématique pour le troisième.

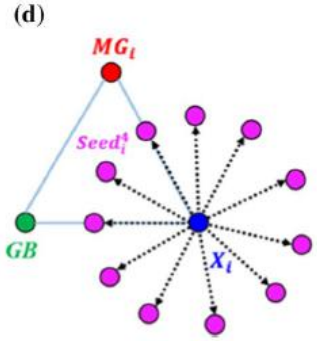


Figure 33 - La vue pour le dernier [36].

3.3.2.3 Pseudo-code de Chaos Game Optimization (CGO)

Le pseudo-code de l'algorithme est détaillé dans l'algorithme 5. [36]

Algorithme 5 Chaos Game Optimization (CGO)

Create random values for initial positions (X_i^j)of eligible seeds (X_i)
Evaluate fitness values for each eligible seed
Find GB, So for found best eligible seed
While (t < *maximum number of iterations*)
    For i = 1 : number of initial eligible seeds
        Find MG_i
        Create temporary triangles with X_i) , GB, and MG_i
        Calculate the α_i, β_i, and γ_i values
        Create new seeds by Eqs, (27)to (30)
    if new seed violate boundary conditions
        Control the position constraints for new seesd and amen it
    end if
    Evaluate the fitness values for new seeds
    if news seeds have better fitness values than the worst initial eligible seeds
        Substitute the worst initial eligible seeds by the new seeds
    end if
    Update GB if a better solution is found
    end for
    t = t + 1
end while
return GB

3.4 La sélection d’attributs

La sélection d'attributs, de caractéristiques ou feature selection, est un processus utilisé pour identifier les caractéristiques et les variables ou les mesures les plus importantes et les plus intéressantes, d’un système donné, dans le but de réaliser la tâche pour laquelle il a été conçu. Dans le domaine de l'apprentissage profond et de la classification, il est crucial de choisir attentivement les caractéristiques à utiliser afin d'éviter l'inclusion de données non pertinentes ou redondantes. Ces caractéristiques indésirables peuvent rendre l'apprentissage plus difficile et affecter négativement les performances de généralisation des modèles d'apprentissage.

3.4.1 Objectifs

Dans le contexte de la détection des maladies gastro-intestinales, il est important de prendre en compte les performances du système de détection et le temps de calcul lorsqu'il y a un grand nombre de caractéristiques. Il est fréquent de trouver des caractéristiques redondantes ou non pertinentes qui n'apportent aucune valeur ajoutée à la classification. C'est pourquoi il est nécessaire d'effectuer une sélection des caractéristiques les plus pertinentes afin de simplifier et accélérer le processus d'apprentissage. Les objectifs principaux de la sélection des caractéristiques sont les suivants :

- Identifier les caractéristiques pertinentes.
- Réduire les bases d’apprentissage et de test.
- Améliorer les performances en classification.
- Réduire le temps d’apprentissage.
- Améliorer la vitesse de la classification.

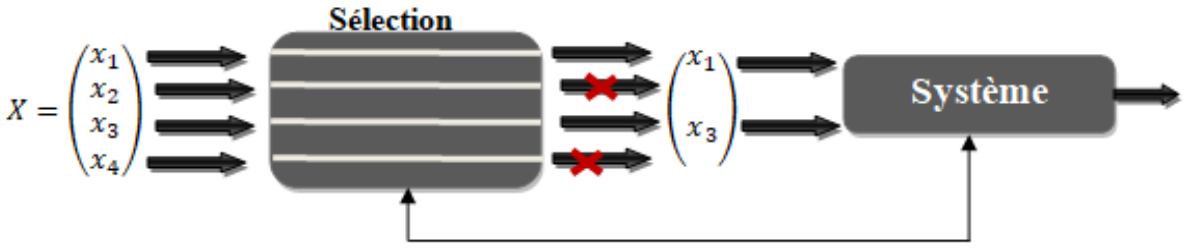


Figure 34 -Principe de la sélection d’attributs.

3.4.2 Le concept général de la sélection d'attributs

La sélection d'attributs en utilisant des méthodes d'optimisation requiert un encodage approprié de la solution et un choix judicieux de la fonction d'évaluation.

- **La structure de l'individu**

Pour sélectionner les attributs les plus pertinents, il est nécessaire de générer aléatoirement des solutions qui contiennent des valeurs dans la plage de 0 à 1. Si la valeur d'un attribut dépasse 0,5, alors cet attribut est considéré, sinon il est éliminé.

- **La fonction d'évaluation (fitness)**

Il existe différents types décrits dans la littérature. L'un des types couramment utilisés est la fonction qui cherche à trouver un compromis entre le taux de sélection des attributs et l'erreur de classification.

$$\boldsymbol{Fitness} = \propto (\mathbf{1} - \boldsymbol{acc}) + \boldsymbol{\beta} \times \frac{|\boldsymbol{T}|}{\boldsymbol{D}} \qquad \textbf{(31)}$$

Avec :$\boldsymbol{\alpha \epsilon R}$; $\boldsymbol{\beta = 1 - \alpha}$

$\boldsymbol{acc}$: Taux de reconnaissance.

$\boldsymbol{D}$: Représente la dimension globale tandis que d indique le nombre d'attributs sélectionnés

$\boldsymbol{T}$: Nombre d'attributs sélectionnés.

3.5 Système de détection avec les modèles pré-entrainés et les algorithmes d'optimisation

Cette section explique notre système de détection automatique des maladies Gastro-intestinales à l'aide des algorithmes d'optimisation Aquila Optimzer (AO), Chaos Game Optimisation (CGO), l'algorithme génétique (GA) et l'optimisation par essaim particulaire (PSO), qui est basé sur la sélection d'attribut. Ce système nécessite trois modules essentiels :

- Module d'extraire les caractéristiques par les modèles pré-entrainés.

- Module d’évaluation du score.
- Module de l'architecture d’un système de détection.

Ces trois modules sont détaillés au suit.

3.5.1 Module d’extraire les caractéristiques par les modèles pré-entrainés

Dans ce module, nous utilisons des modèles pré-entrainés (VGG16, VGG19, ResNet50, MobileNet et GoogleNet) qui ont été développés à partir d'un ensemble de données d'apprentissage annotées. Ces modèles sont capables d'extraire automatiquement des caractéristiques significatives à partir des images ou des signaux médicaux liés aux maladies gastro-intestinales.

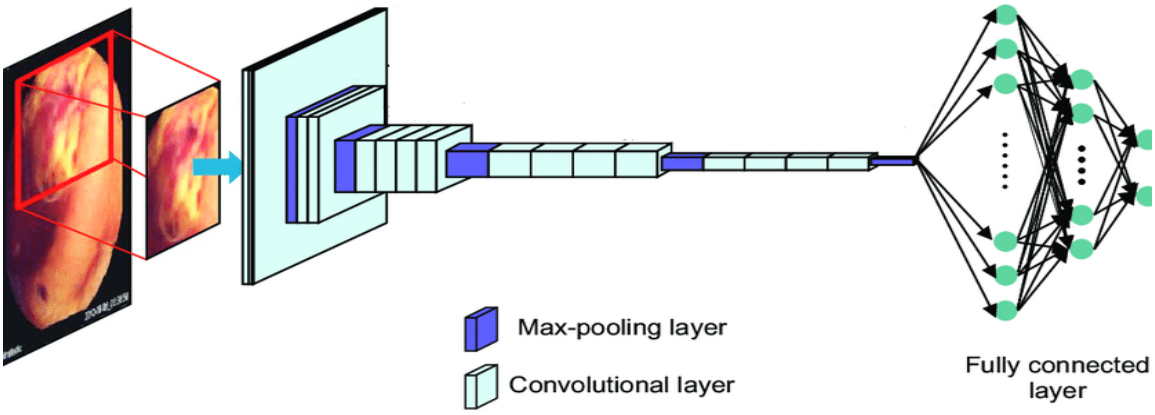


Figure 35 - Extraire les caractéristiques par les modèles pré-entrainés.

3.5.2 Module d’évaluation du score

Afin d'appliquer le processus de détection des maladies gastro-intestinales à l'aide d'une sélection en utilisant la technique wrapper, qui est assistée par AO, CGO, GA et PSO. Ces algorithmes sont appliqués pour rechercher l'ensemble optimal des attributs qui maximise la performance du système de détection et réduit le nombre de caractéristiques.

Donc, le score pour chaque objet est calculé par :

$$\boldsymbol{Sc = 099 \times Acc + 0,01 \times \left(\frac{D-d}{D}\right)} \quad \textbf{(32)}$$

Où : (Acc) et (d) représentent le taux de classification obtenu par KNN, RandomForest et MLP. Dans équation (32), D est le nombre total des caractéristiques à base de plusieurs blocs extraits de l'image originale.

3.5.3 Module de l'architecture d'un système de détection

Cette partie représente le cœur de notre travail, qui consiste à appliquer les algorithmes AO, CGO, GA et PSO dans la détection des maladies Gastro-intestinales à base de la sélection des attributs. Pour mieux comprendre l'architecture proposée, nous avons préféré d'expliquer les ingrédients suivants:

Chargement de la base de données KVASIR V2 : Nous avons chargé la base de données KVASIR V2, qui contient des images et des annotations de différentes maladies gastro-intestinales.

Redimensionnement des images : Nous avons redimensionné les images de la base de données aux dimensions requises, par exemple 224x224 pour certains modèles (VGG16, VGG19, ResNet50) et 299x299 pour d'autres modèles (MobileNet, GoogleNet).

Extraction des caractéristiques : Nous avons utilisé les modèles pré-entrainés (VGG16, VGG19, ResNet50, MobileNet, GoogleNet) pour extraire les caractéristiques des images redimensionnées. Chaque modèle génère un ensemble de caractéristiques représentant les informations significatives présentes dans les images.

Division des données en ensembles d'entraînement et de test : Nous avons divisé les données en 80% pour l'ensemble d'entraînement et 20% pour l'ensemble de test. Cette division permettra d'évaluer la performance du système de détection sur des données non utilisées lors de l'entraînement.

Initialisation des algorithmes d'optimisation : Initialiser les algorithmes AO, CGO, GA et PSO avec les paramètres appropriés. Ces algorithmes seront utilisés pour sélectionner les attributs les plus pertinents pour la détection des maladies gastro-intestinales.

Sélection des attributs avec un seuil de 0,5 : Appliquer les algorithmes d'optimisation pour rechercher l'ensemble optimal d'attributs qui maximise la performance de détection et minimise le nombre total du caractéristiques. Nous avons utilisé un seuil de 0,5 pour évaluer la pertinence de chaque attribut sélectionné.

Évaluation du score avec la fonction du fitness : Évaluer les ensembles d'attributs proposés en utilisant une fonction du fitness appropriée.

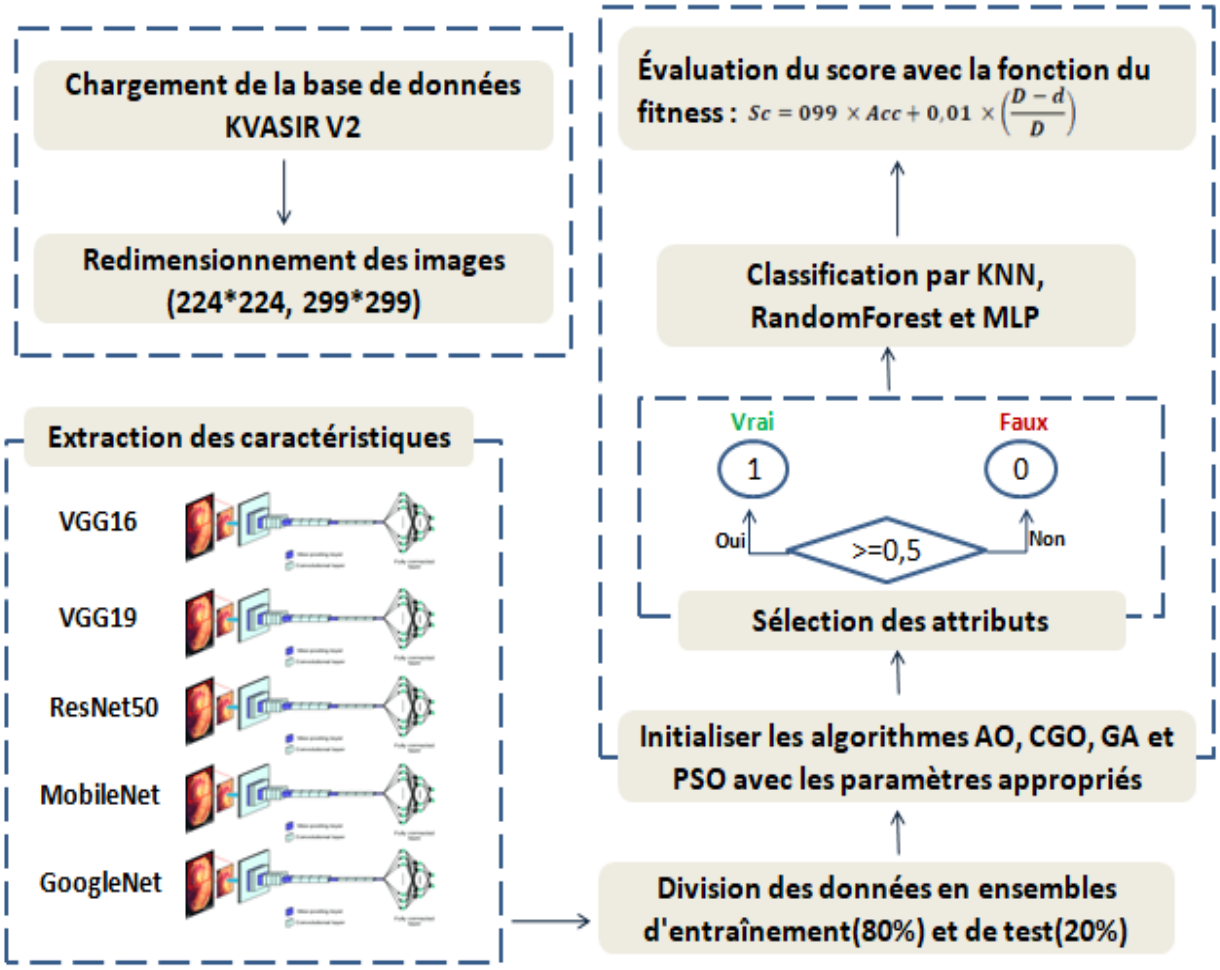


Figure 36 - l'architecture du système de détection des maladies Gastro-intestinales.

3.6 Conclusion

Ce chapitre a abordé les techniques d'optimisation utilisées pour résoudre le problème de la sélection d'attributs. Nous avons exploré différentes approches telles que les algorithmes génétiques (GA), l'optimisation par essaim particulier (PSO), Aquila Optimizer (AO) et Chaos Game Optimization (CGO). Grâce à ces méthodes, nous avons pu découvrir des solutions optimales pour la sélection d'attributs, améliorant ainsi les performances des systèmes de classification. De plus, nous avons clarifié le concept général de la sélection d'attributs et présenté le processus qui sous-tend cette démarche.

Le chapitre suivant sera principalement consacré à l'implémentation de ces méthodes, mettant en évidence leur efficacité. De plus, nous montrons les résultats obtenus pour chaque architecture proposée et effectuons une comparaison entre elles afin de définir la mieux adaptée à notre approche.

Chapitre 4

Implémentation et résultats expérimentaux

4.1 Introduction

Dans ce chapitre, nous abordons la mise en œuvre pratique de notre approche en présentant les outils logiciels et les bibliothèques utilisées, telles que Tensorflow et Keras. Nous décrivons également en détail la construction des jeux de données, comprenant les ensembles d'entraînement, de validation et de test, ainsi que les étapes de prétraitement et de représentation des données. Ensuite, nous présentons les résultats obtenus pour chaque architecture proposée, mettant en évidence leur performance respective. Une comparaison approfondie est effectuée afin de déterminer l'architecture la mieux adaptée à notre approche.

Enfin, nous présentons notre application Django pour la détection des maladies gastro-intestinales, qui utilise l'apprentissage profond optimisé. Nous expliquons son fonctionnement et mettons en avant ses fonctionnalités clés.

4.2 Présentation des outils

4.2.1 Configuration matérielle utilisée

La configuration matérielle utilisée pour l'entraînement et l'évaluation de nos modèles est la suivante :

- Modèle HP ProBook 430 G1.
- Processeur intel Core i5-4300U CPU contenant quatre cœurs à 2.50 GHZ chacun.
- Carte graphique intel HD Graphics 5500 2 GO.
- RAM d'une taille de 4 GO.

- Disque dur (HDD).
- Système d’exploitation Windows 10 Pro 64 bits.

4.2.2 Environnement de développement

4.2.2.1 Google colab

Colab est une plateforme offerte par Google qui permet d'écrire et d'exécuter du code Python via un navigateur web. Basée sur Jupyter Notebook, elle est spécifiquement conçue pour la formation et la recherche en apprentissage automatique (Machine Learning). Colab offre la possibilité d'entraîner des modèles de Machine Learning directement dans le cloud [37].

Figure 37 – Logo de Google Colab [37].

Voici quelques avantages et fonctionnalités de Colab :

- **Amélioration des compétences de codage en Python :** Colab permet aux utilisateurs de pratiquer et d'améliorer leurs compétences en programmation Python.

- **Développement d'applications en Deep Learning : Colab** offre un environnement de développement pour créer des applications en Deep Learning en utilisant des bibliothèques populaires telles que Keras, TensorFlow, PyTorch et OpenCV.

- **Utilisation d'un environnement de développement prêt à l'emploi :** Colab est basé sur Jupyter Notebook, ce qui signifie qu'il offre une interface conviviale et interactive pour écrire et exécuter du code sans nécessiter de configuration supplémentaire.

- **Accès gratuit à un GPU :** La fonctionnalité qui distingue Colab des autres services est l'accès gratuit à un processeur graphique GPU. Cela permet d'accélérer les calculs liés à l'entraînement des modèles de Machine Learning, qui peuvent être très intensifs en termes de calculs matriciels.

4.2.2.2 Visual Studio Code

Visual Studio Code est un éditeur de code open-source et extensible développé par Microsoft. Il est disponible pour les systèmes d'exploitation Windows, Linux et macOS. Visual Studio Code offre une large prise en charge de langages de programmation grâce à son écosystème d'extensions [38].

Les fonctionnalités principales de Visual Studio Code comprennent l'auto-complétion du code, la coloration syntaxique, le débogage intégré et la prise en charge des commandes Git. Ces fonctionnalités facilitent l'écriture et la gestion du code, améliorant ainsi l'efficacité des développeurs. En plus de ses fonctionnalités, Visual Studio Code est reconnu pour sa performance. Il est réactif et offre une expérience utilisateur fluide, même lors de la manipulation de projets volumineux.

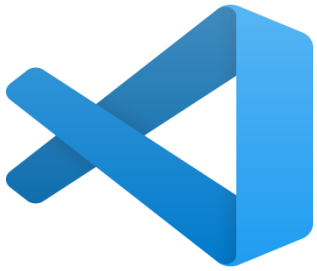


Figure 38–Logo de Visual Studio Code [38].

4.2.3 Langages de programmation

4.2.3.1 Python

Python est un langage de programmation interprété, orienté objet et de haut niveau avec une sémantique dynamique, développé par Guido van Rossum. Il a été initialement publié en 1991.

Python est utilisé pour le développement web côté serveur, le développement de logiciels, les mathématiques et la création de scripts système. Il est populaire pour le développement rapide d'applications et en tant que langage de script ou de liaison pour lier des composants existants en raison de ses structures de données intégrées de haut niveau, de sa typage dynamique et de sa liaison dynamique. Les coûts de maintenance des programmes sont réduits avec Python en raison de la syntaxe facilement apprise et de l'accent mis sur la lisibilité [39].

Figure 39 – Logo de Python [39].

4.2.3.2 HTML

HTML (HyperText Markup Language) est le bloc de construction le plus fondamental du Web. Il définit la signification et la structure du contenu web. D'autres technologies, en plus de HTML, sont généralement utilisées pour décrire l'apparence/présentation d'une page web (CSS) ou sa fonctionnalité/comportement (JavaScript) [40].

4.2.3.3 CSS

CSS (Cascading Style Sheets) est le langage permettant de décrire la présentation des pages Web, y compris les couleurs, la mise en page et les polices de caractères. Il permet d'adapter la présentation à différents types d'appareils, tels que les grands écrans, les petits écrans ou les imprimantes. CSS est indépendant de HTML et peut être utilisé avec n'importe quel langage de balisage basé sur XML [40].

4.2.3.4 JavaScript

JavaScript (JS) est un langage de programmation léger, interprété ou compilé à la volée, avec des fonctions de premier ordre.

Bien qu'il soit principalement connu comme le langage de script des pages Web, de nombreux environnements autres que les navigateurs l'utilisent également, tels que Node.js, Apache CouchDB et Adobe Acrobat. JavaScript est un langage basé sur des prototypes, multi-paradigme, à thread unique et dynamique, prenant en charge les styles de programmation orientée objet, impératif et déclaratif (par exemple, la programmation fonctionnelle) [40].

4.2.4 Frameworks et bibliothèques :

4.2.4.1 Django

Django est un framework web Python de haut niveau qui permet de créer rapidement des sites web sécurisés et faciles à maintenir. Développé par des experts, Django simplifie grandement le processus de développement web, vous permettant ainsi de vous concentrer sur l'écriture de votre application sans avoir à partir de zéro. Il est gratuit, open source, bénéficie d'une communauté dynamique et active, dispose d'une documentation complète et offre de nombreuses options de support gratuit et payant.

4.2.4.2 Bootstrap

Bootstrap est un framework de développement front-end open source et gratuit pour la création de sites web et d'applications web. Bootstrap inclut les bases du développement web responsive, de sorte que les développeurs n'ont qu'à insérer le code dans un système de grille prédéfini. Le framework Bootstrap est construit sur le langage de balisage hypertexte (HTML), les feuilles de style en cascade (CSS) et JavaScript. Les développeurs web utilisant Bootstrap peuvent créer des sites web beaucoup plus rapidement sans passer du temps à se soucier des commandes et fonctions de base.

4.2.4.3 Keras

Keras est une interface de programmation d'apprentissage profond de haut niveau, écrite en Python et compatible avec TensorFlow. Elle propose un ensemble d'abstractions de niveau supérieur et intuitives qui simplifient la configuration des réseaux neuronaux, indépendamment de la bibliothèque de calcul sous-jacente.

4.2.4.4 TensorFlow

TensorFlow est un framework open source de programmation pour le calcul numérique, lancé par Google en novembre 2015. Il est largement utilisé pour l'apprentissage profond et les réseaux neuronaux. Son nom provient de l'utilisation courante de tableaux de données multidimensionnels appelés tenseurs pour les opérations sur les réseaux neuronaux. Un tenseur à deux dimensions équivaut à une matrice.

Figure 40 – Logo de TensorFlow.

4.2.4.5 Scikit-learn

Scikit-learn est une bibliothèque d'apprentissage automatique (machine learning) en Python, construite sur d'autres bibliothèques Python telles que NumPy, SciPy et Matplotlib. Scikit-learn offre une gamme complète d'algorithmes pour les tâches d'apprentissage statistique, notamment la classification, la régression, la réduction de dimension et le regroupement. Grâce à ses fonctionnalités étendues, Scikit-learn facilite l'implémentation et l'évaluation de modèles d'apprentissage automatique dans le langage de programmation Python [41].

4.2.4.6 Pandas

Pandas est une bibliothèque Python conçue pour la manipulation et l'analyse de données. Elle offre des structures de données et des opérations permettant la manipulation de tableaux numériques et de séries temporelles. Pandas est un logiciel open source sous licence BSD, ce qui signifie qu'il peut être réutilisé avec peu de restrictions.

4.2.4.7 Numpy

NumPy est une bibliothèque essentielle pour le calcul scientifique avec Python. Elle est conçue pour la manipulation de matrices ou tableaux multidimensionnels, ainsi que pour l'application de fonctions mathématiques à ces tableaux (arrays). Les calculs réalisés avec

NumPy sont hautement optimisés, car les tableaux sont homogènes (contenant des valeurs du même type) et de taille fixe lors de leur création.

4.2.4.8 Matplotlib

Matplotlib est une bibliothèque de visualisation en deux dimensions pour Python, permettant de créer des images de haute qualité. Elle peut être utilisée dans des scripts Python, des serveurs d'applications Web, et bien d'autres applications. Matplotlib offre la possibilité de générer rapidement des histogrammes, des graphiques à barres, des diagrammes d'erreur, des diagrammes de dispersion, et bien plus encore, avec seulement quelques lignes de code. Nous avons utilisé cette bibliothèque pour tracer des graphiques afin de présenter les résultats de notre approche.

4.2.4.9 MealPy

Mealpy est une bibliothèque open source offrant un code bien documenté et une interface utilisateur simple. Elle nécessite peu de dépendances pour fonctionner. Mealpy propose une variété d'algorithmes méta-heuristiques populaires et récents, conçus pour optimiser les fonctions de benchmark couramment utilisées, telles que CEC-2017. Cette bibliothèque peut être utilisée pour résoudre des problèmes d'optimisation complexes en exploitant les capacités des algorithmes méta-heuristiques inclus dans Mealpy.

4.2.5 Stockage et partage des données

Pour assurer le stockage et le partage des données liées à notre approche, nous avons opté pour l'utilisation de Google Drive.

4.2.5.1 Google Drive

Google Drive est un service de stockage en ligne qui offre aux utilisateurs la possibilité de sauvegarder leurs fichiers et de les partager avec d'autres personnes. Grâce à Google Drive, nous avons pu télécharger nos fichiers de données, tels que des ensembles de données, des fichiers de configuration ou des résultats d'expérimentation, sur le cloud. Cela nous a permis d'y accéder facilement depuis n'importe quel appareil connecté à Internet.

4.2.6 Base de données

Pour évaluer et tester notre approche, nous avons utilisé la base de données KVASIR V2. Cette base de données a été créée par une équipe de chercheurs et de médecins spécialisés dans le domaine de la gastro-entérologie en 2017.La base de données KVASIR V2 est une ressource essentielle pour la recherche en détection automatique des anomalies du tractus gastro-intestinal. Elle est composée de 8 000 images réparties en 8 classes comme il illustre dans la Figure 41 et le Tableau 2, avec des résolutions variant de 720x576 à 1920x1072 pixels. Les images sont organisées dans des dossiers séparés, nommés en fonction de leur contenu, ce qui facilite leur utilisation et leur analyse [3].

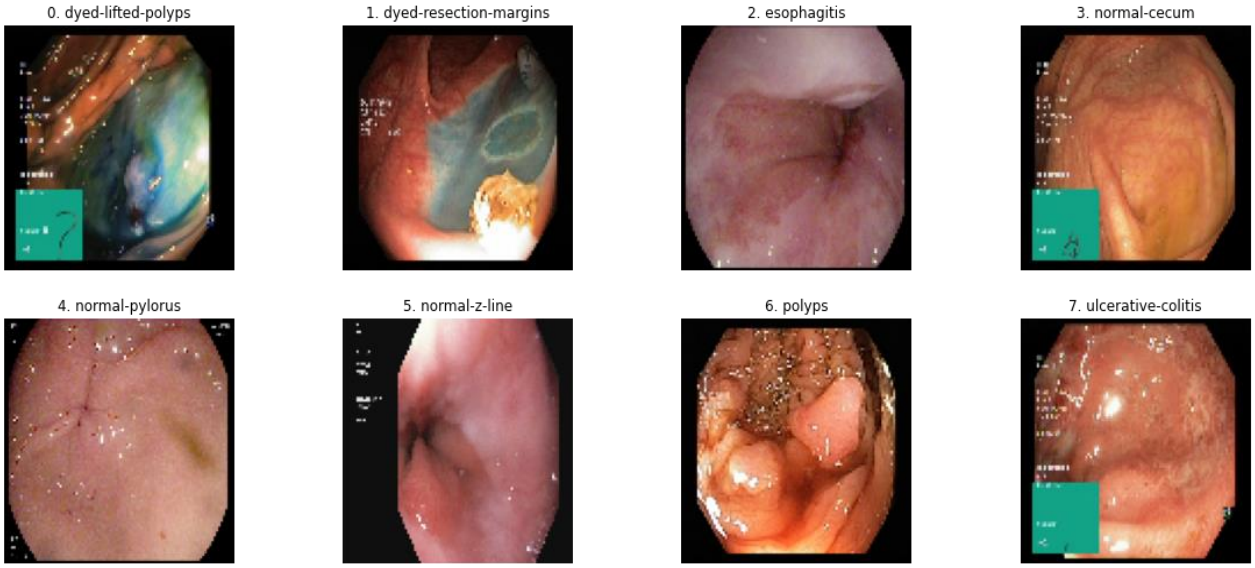


Figure 41 – Les classes de KVASIR V2.

Class	Nombre de fichiers	Explication
dyed-lifted-polyps	1000	Polypes soulevés teintés
dyed-resection-margins	1000	Marges de résection teintées
esophagitis	1000	Œsophagite
normal-cecum	1000	Cecum normal
normal-pylorus	1000	Pylore normal
normal-z-line	1000	Ligne Z normale
polyps	1000	Polypes
ulcerative-colitis	1000	Colite ulcéreuse

Tableau 2 - Tableau des classes de la base de données KVASIR V2.

4.2.7 Les méthodes d'évaluation

Dans le cadre de l'évaluation des performances, nous utilisons la matrice de confusion pour calculer des métriques telles que l'accuracy, la précision, le F1-score, le rappel.

- **Matrice de confusion**

La matrice de confusion est présentée sous la forme d'un tableau qui croise les prédictions du modèle avec les vraies valeurs des données de test comme la figure 42. Elle se compose de quatre éléments principaux :

– True Positive (TP) : la prédiction et la valeur réelle sont positives.
– True Negative (TN) : la prédiction et la valeur réelle sont négatives.
– False Positive (FP) : la prédiction est positive alors que la valeur réelle est négative.
– False Negative (FN) : la prédiction est négative alors que la valeur réelle est négative.

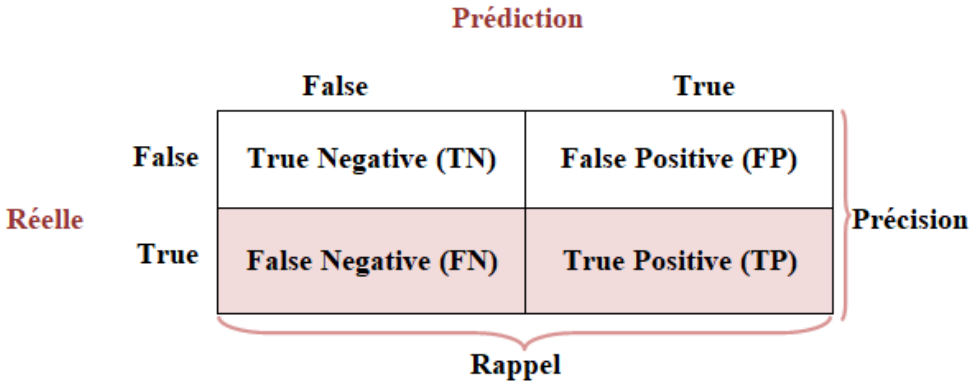


Figure 42 - Matrice de confusion.

Cette matrice permet de déduire les paramètres suivants :

$$\textbf{Accuracy} = \frac{\textbf{TP+TN}}{\textbf{TP+TN+FN+FP}} \quad \textbf{(32)}$$

$$\textbf{Précision} = \frac{\textbf{TP}}{\textbf{TP+ FP}} \quad \textbf{(33)}$$

$$\textbf{Rappel} = \frac{\textbf{TP}}{\textbf{TP+ FN}} \quad \textbf{(34)}$$

$$\textbf{F1 - Score} = 2 * \frac{\textbf{Précision*Rappel}}{\textbf{Précision+Rappel}} \quad \textbf{(35)}$$

4.3 Etude expérimentale et Résultats

À ce stade, on a divisé notre ensemble de données en deux sous-ensembles, où 80 % constitue l'ensemble d'apprentissage à 20 % l'ensemble de test.

4.3.1 Résultats sans optimisation

Nous avons évalué les performances de chaque modèle pré-entraîné en utilisant les classificateurs KNN, RandomForest et MLP. Les métriques telles que l'accuracy, la précision, le rappel et le score F1 ont été utilisées pour mesurer la performance de chaque modèle. De plus, nous avons généré des matrices de confusion pour examiner les résultats en détail et identifier les erreurs de classification. Ces évaluations nous ont permis de comprendre les performances de chaque modèle et d'analyser les erreurs fréquentes.

Les résultats sont présentés ci-dessous :

- **VGG16**

Le tableau et le graphique ci-dessous présente les résultats différents avec KNN, RandomForest et MLP :

Métrique	**KNN**	**RandomForest**	**MLP**
Accuracy	71,63%	77,25%	82,87%
F-Score	71,36%	77,16%	82,89%
Rappel	71,63%	77,25%	82,87%
Precision	72,27%	77,22%	83,76%

Tableau 3 - Tableau de comparaison de KNN, RandomForest et MLP pour VGG16.

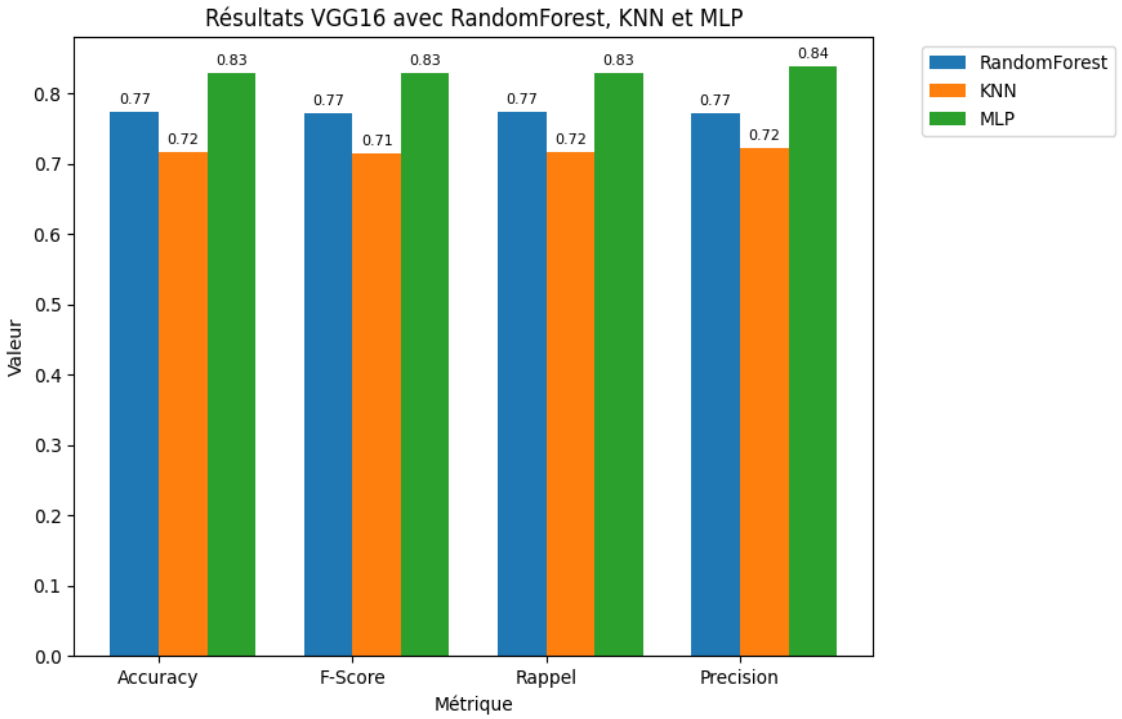


Figure 43 - Graphique à barres de KNN, RandomForest et MLP pour VGG16.

- **VGG19**

Le tableau et le graphique ci-dessous présente les résultats différents avec KNN, RandomForest et MLP :

Métrique	**KNN**	**RandomForest**	**MLP**
Accuracy	71,19%	75,19%	81,62%
F-Score	71,00%	75,09%	81,57%
Rappel	71,19%	75,19%	81,62%
Precision	71,98%	75,15%	81,80%

Tableau 4 - Tableau de comparaison de KNN, RandomForest et MLP pour VGG19.

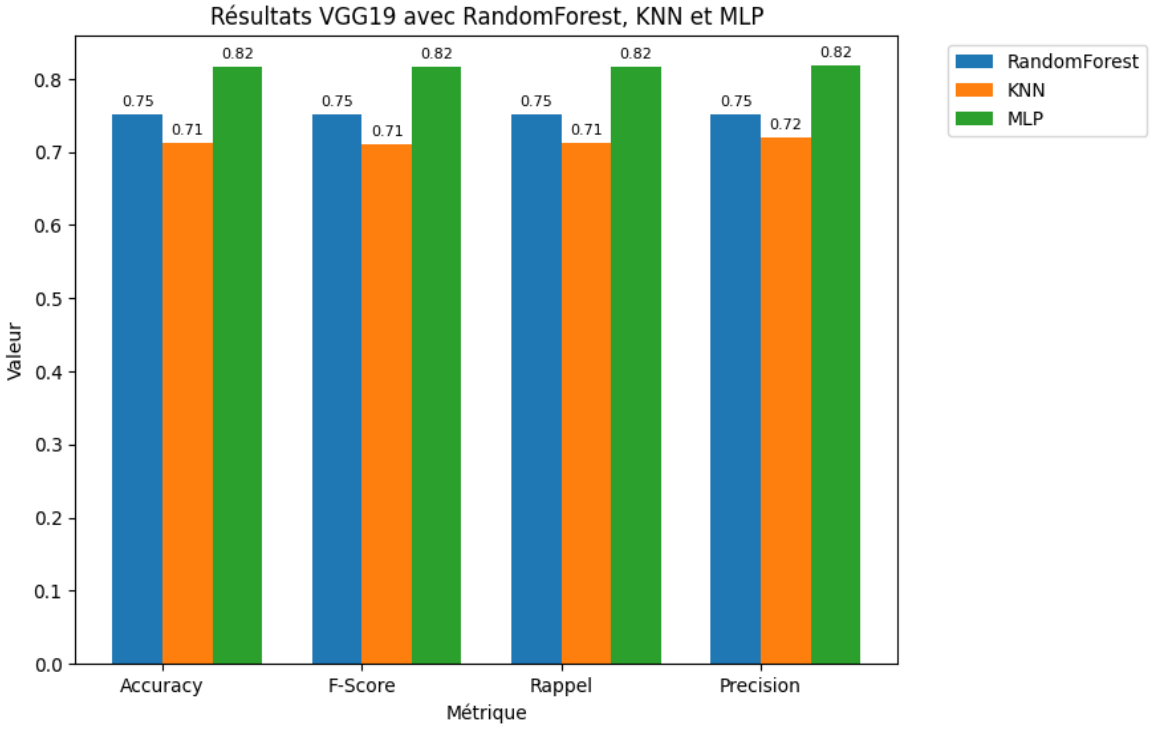


Figure 44 - Graphique à barres de KNN, RandomForest et MLP pour VGG19.

- **ResNet50**

Le tableau et le graphique ci-dessous présente les résultats différents avec KNN, RandomForest et MLP:

Métrique	**KNN**	**RandomForest**	**MLP**
Accuracy	51,81%	55,56%	66,50%
F-Score	50,69%	54,57%	65,15%
Rappel	51,81%	55,56%	66,50%
Precision	51,72%	55,01%	67,52%

Tableau 5 - Tableau de comparaison de KNN, RandomForest et MLP pour ResNet50.

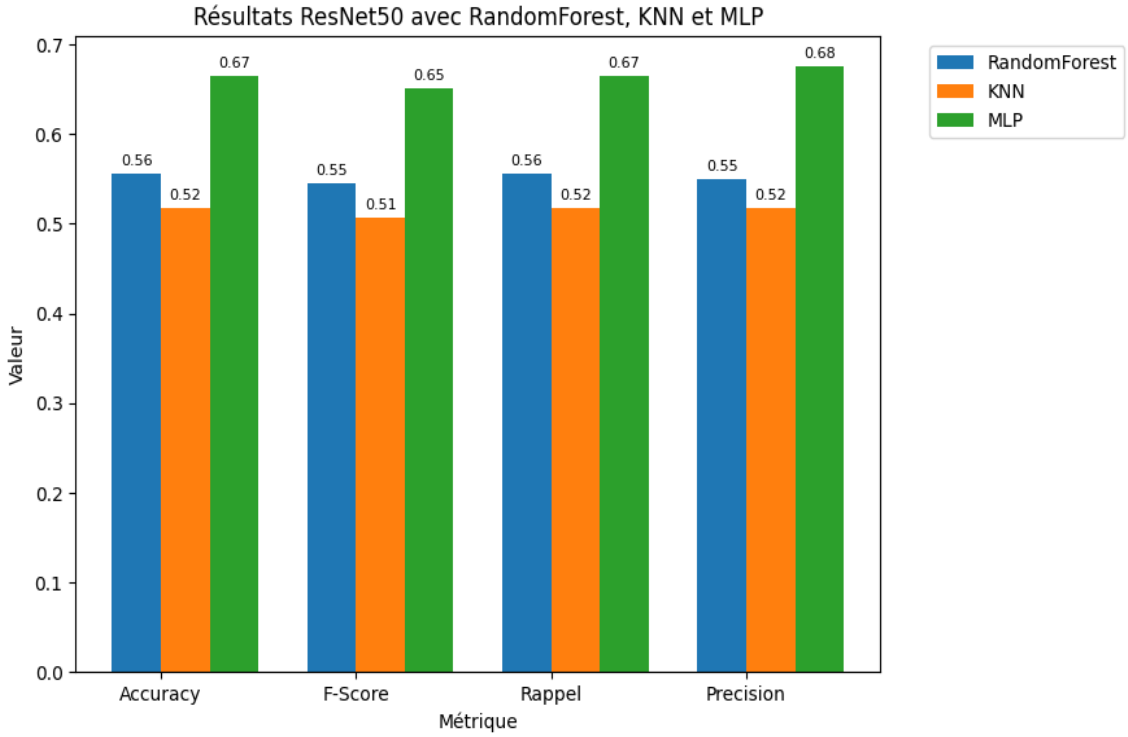


Figure 45 - Graphique à barres de KNN, RandomForest et MLP pour ResNet50.

- **MobileNet**

Le tableau et le graphique ci-dessous présente les résultats différents avec KNN, RandomForest et MLP :

Métrique	**KNN**	**RandomForest**	**MLP**
Accuracy	76,94%	82,06%	86,81%
F-Score	76,46%	82,01%	86,78%
Rappel	76,94%	82,06%	86,81%
Precision	78,09%	82,32%	86,90%

Tableau 6 - Tableau de comparaison de KNN, RandomForest et MLP pour MobileNet.

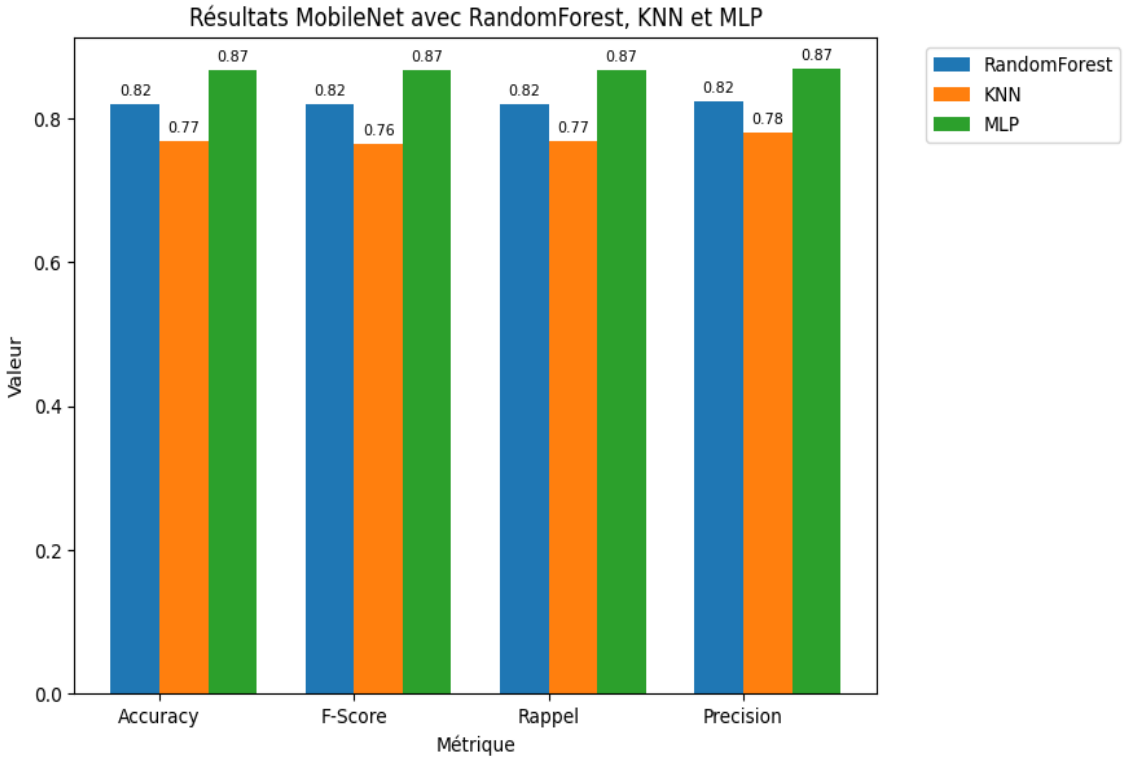


Figure 46 - Graphique à barres de KNN, RandomForest et MLP pour MobileNet.

- **GoogleNet**

Le tableau et le graphique ci-dessous présente les résultats différents avec KNN, RandomForest et MLP:

Métrique	**KNN**	**RandomForest**	**MLP**
Accuracy	75,19%	82,00%	86,06%
F-Score	74,81%	81,87%	86,05%
Rappel	75,19%	82,00%	86,06%
Precision	75,75%	81,85%	86,06%

Tableau 7 - Tableau de comparaison de KNN, RandomForest et MLP pour GoogleNet.

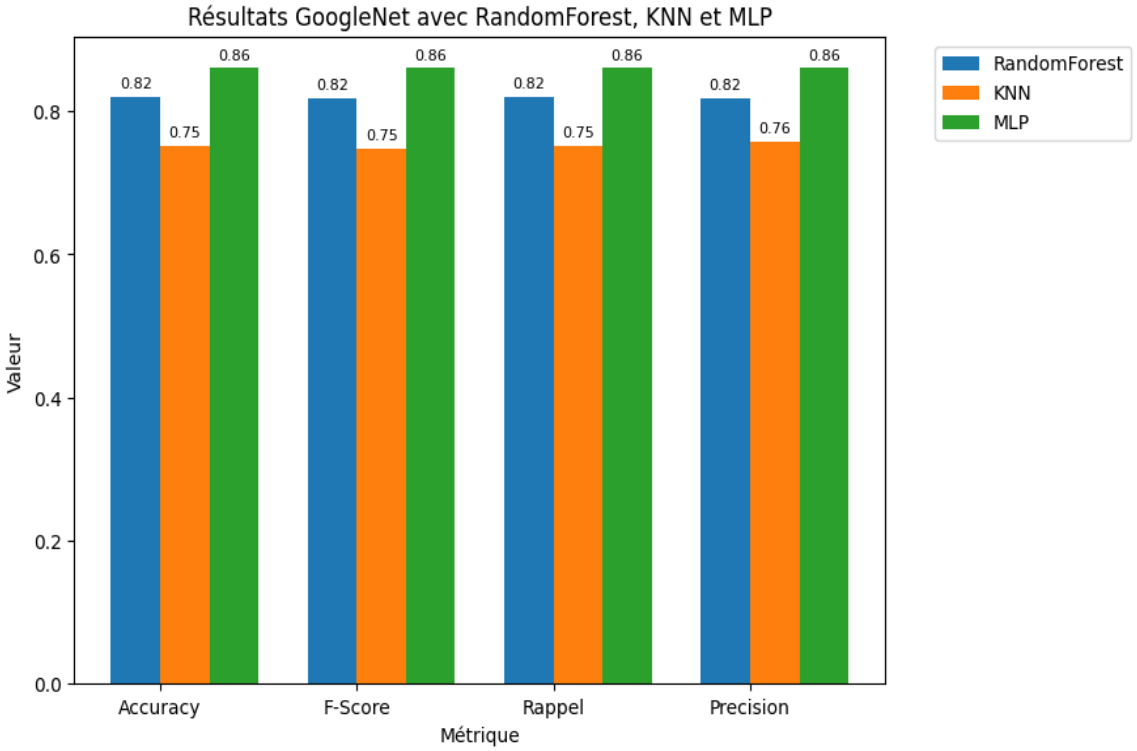


Figure 47 - Graphique à barres de KNN, RandomForest et MLP pour GoogleNet..

- **Modèle proposé**

Pour augmenter les performances et arrivée a la meilleure classification possible on propose notre modèle, appelé « combined-model », est conçu pour combiner les caractéristiques extraites de cinq modèles pré-entraînés : VGG16, VGG19, GoogleNet, MobileNet et ResNet50, comme illustre dans la figure 48.

En utilisant cette approche, nous tirons parti de la diversité des architectures et des capacités de ces modèles pour obtenir une représentation plus riche et complète des données.

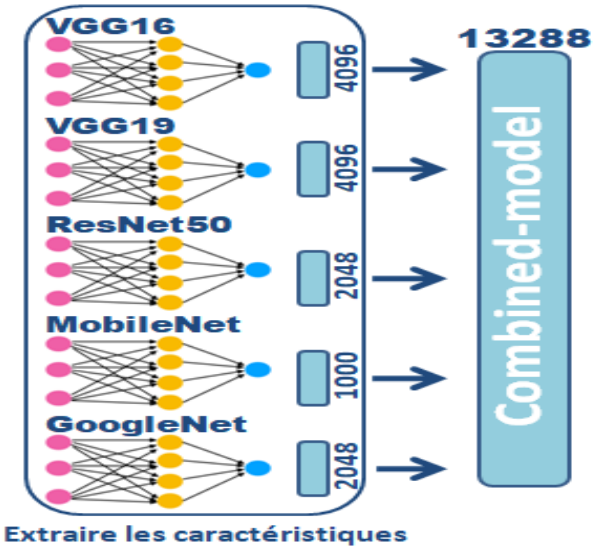


Figure 48 - Combiner les caractéristiques extraites.

Le tableau et le graphique à barre ci-dessous montre la variation du nombre de paramètres entre les modèles :

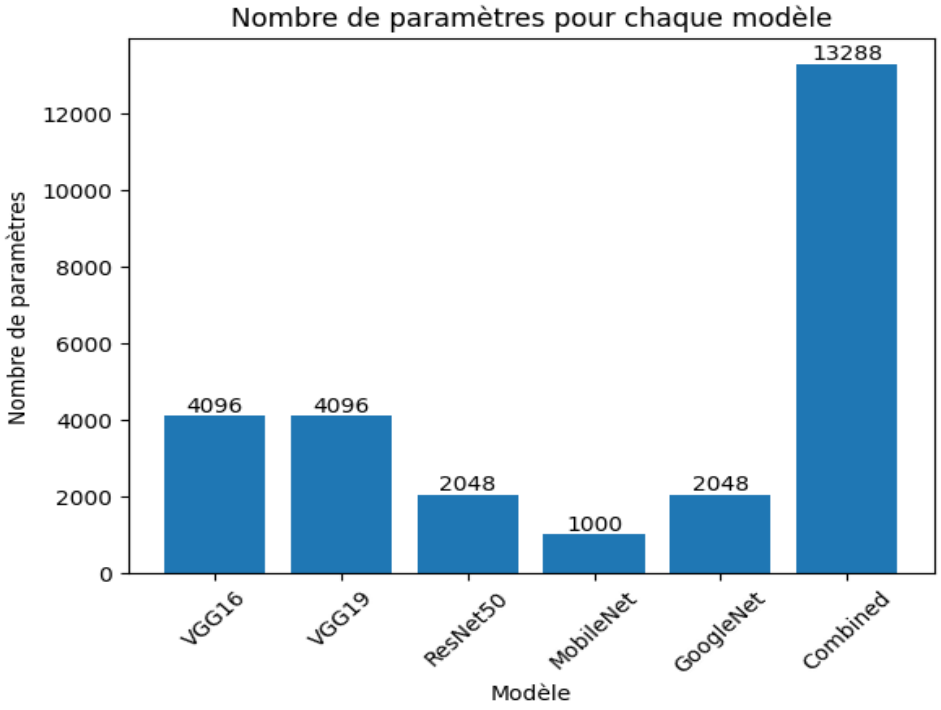


Figure 49 - Variation du nombre de paramètres entre les modèles.

Le tableau et le graphique ci-dessous présente les résultats de notre modèle proposé avec KNN, RandomForest et MLP :

Métrique	KNN	RandomForest	MLP
Accuracy	82,00%	94,19%	97,37%
F-Score	81,71%	94,17%	97,37%
Rappel	82,00%	94,19%	97,37%
Precision	83,38%	94,18%	97,37%

Tableau 8 - Tableau de comparaison de KNN, RandomForest et MLP pour notre modèle.

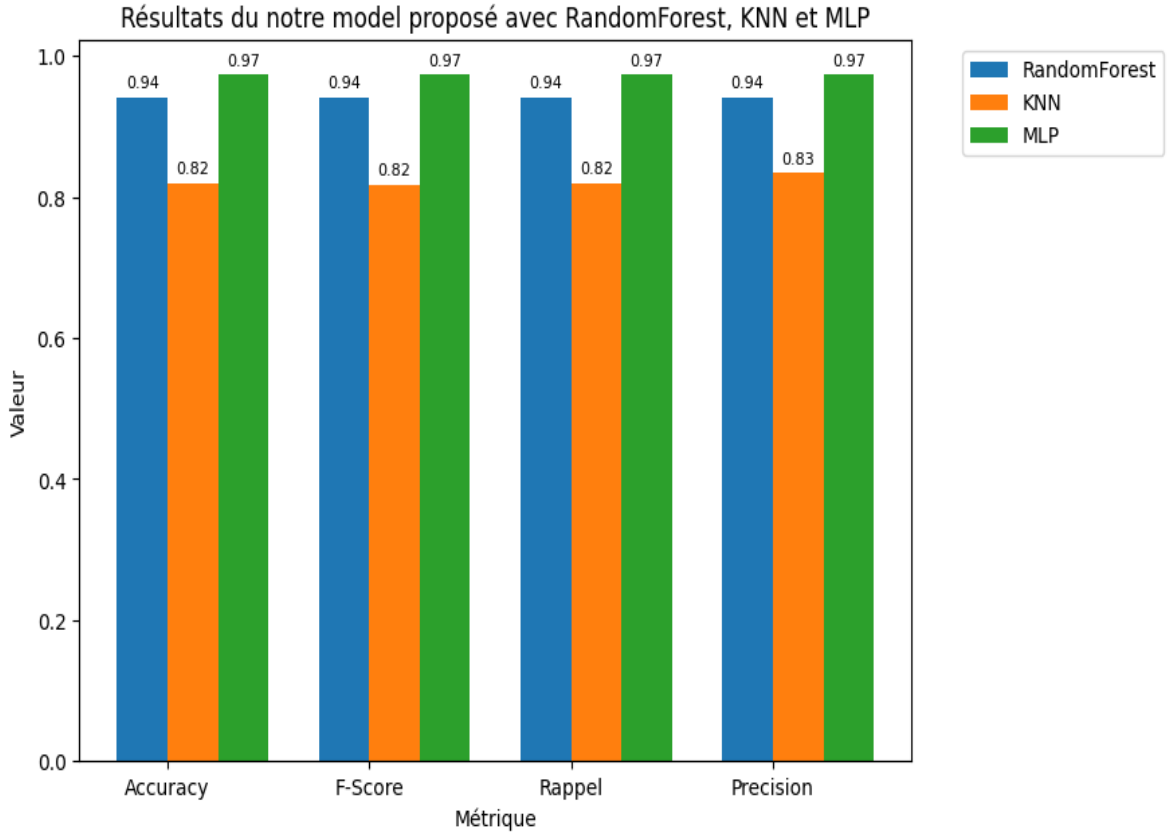


Figure 50 - Graphique à barres de KNN, RandomForest et MLP pour notre modèle.

Ici on voit les matrices de confusion obtenus pour notre modèle en utilisant les classificateurs RandomForest et KNN:

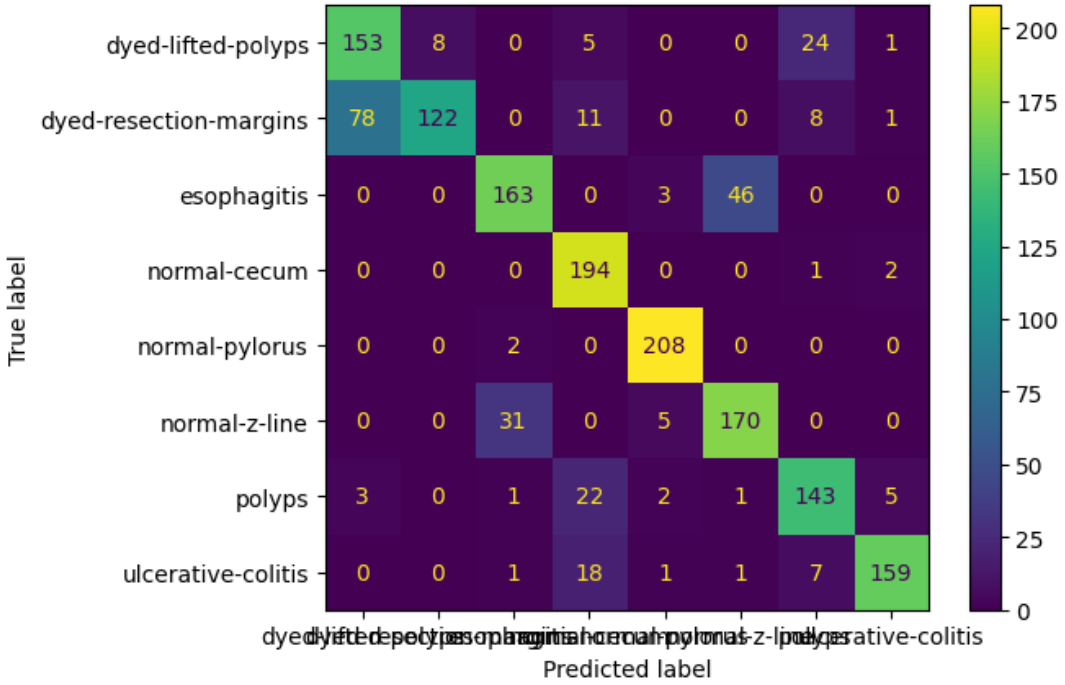


Figure 51 - Matrice de confusion pour notre modèle avec KNN.

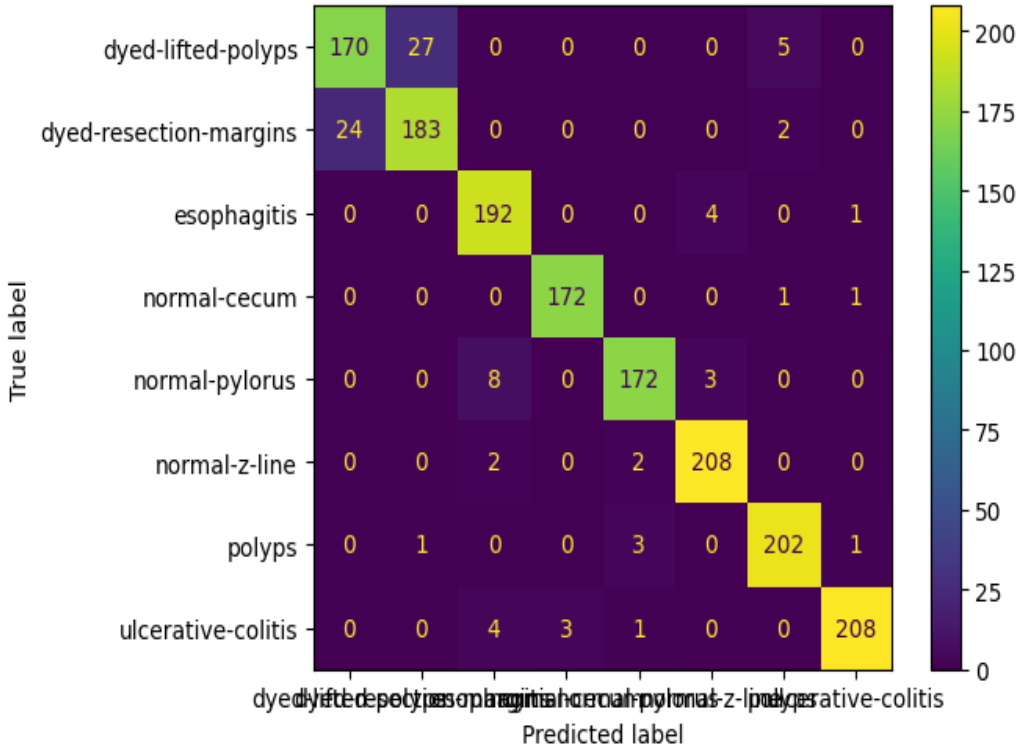


Figure 52 - Matrice de confusion pour notre modèle avec RandomForest.

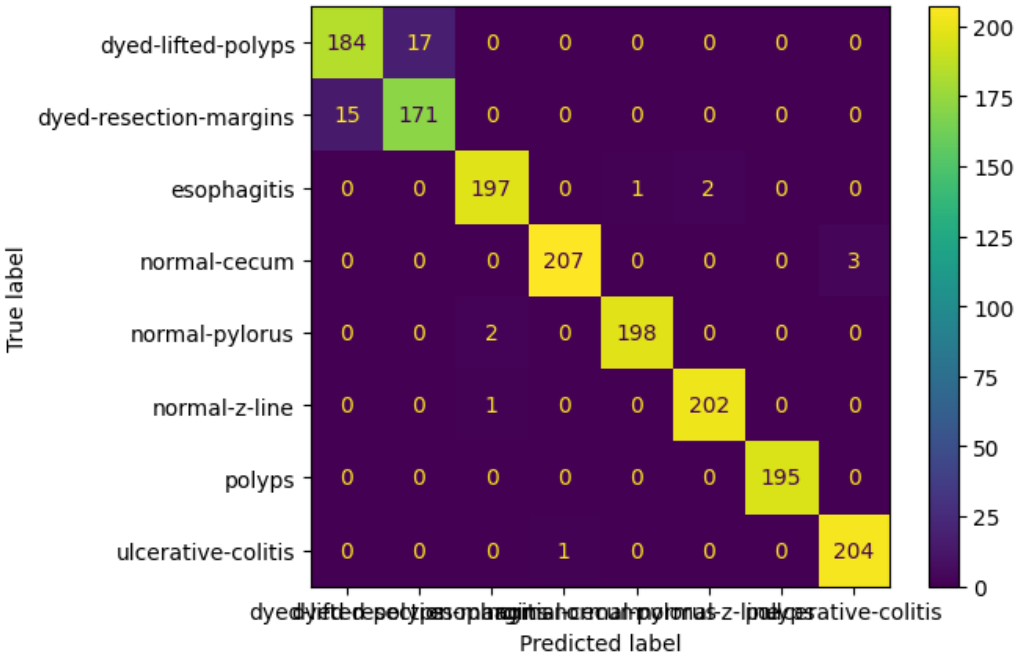


Figure 53 - Matrice de confusion pour notre modèle avec MLP.

4.3.1.1 Discussion

Pour tous les modèles, nous observons que les performances sont meilleures avec le classificateur RandomForest et MLP. L'accuracy, le F-Score, le rappel et la précision sont tous plus élevés avec MLP par rapport à RandomForest et KNN. La matrice de confusion pour MLP montre une meilleure capacité à classifier correctement les exemples dans chaque classe par rapport à RandomForest et KNN.

Notre modèle combiné, qui fusionne les caractéristiques extraites des cinq modèles pré-entraînés. Ce modèle montre une nette amélioration des performances par rapport aux autres modèles. Les métriques d'évaluation sont significativement plus élevées pour les trois classificateurs, comme le montre la Figure 54, avec une précision globale atteignant 97,37% avec MLP. La matrice de confusion pour notre modèle avec MLP montre une excellente capacité de classification, avec un nombre élevé d'exemples correctement classés dans chaque classe.

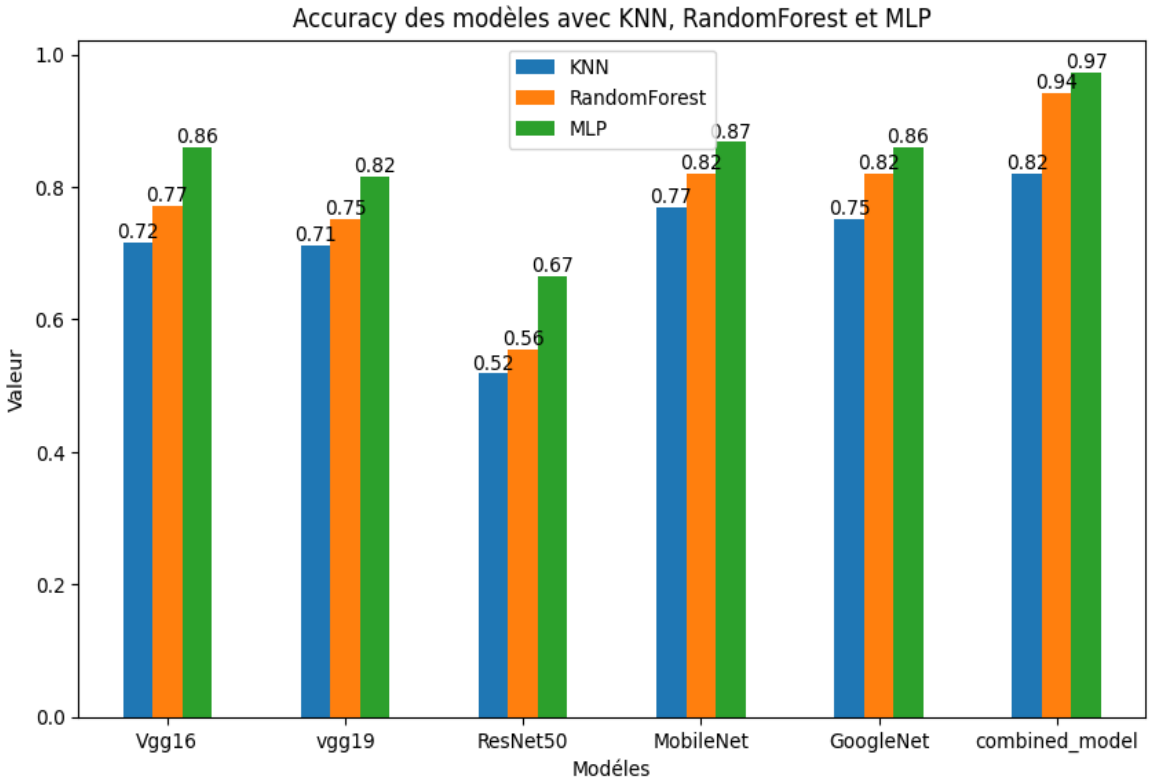


Figure 54 - Graphique à barres de KNN, RandomForest et MLP pour tous les modèles.

4.3.2 Résultats avec optimisation

4.3.2.1 K plus proches voisins (KNN)

À ce stade de notre étude, nous avons appliqué les algorithmes d'optimisation AO, CGO, GA et PSO sur les modèles en utilisant le classificateur KNN comme algorithme de base avec 50 itérations et une population de 10 individus.

En appliquant les différents algorithmes d'optimisation, nous avons évalué les performances de chaque modèle en utilisant différentes mesures telles que Fitness, Accuracy, F-score, Rappel, Précision, Taille de Chromosome et Temps d'exécution. Les résultats sont présentés ci-dessous :

- **VGG16**

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG16 avec AO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,714317	0,7175	0,71394	0,730527	0,7175	2040	43,8586
min	0,709778	0,71125	0,707488	0,723106	0,71125	1955	19,72207
var	6,53E-07	3,14E-06	3,16E-06	4,08E-06	3,14E-06	688,7445	39,72129
std	0,000808	0,001772	0,001777	0,002021	0,001772	26,24394	6,302483

Tableau 9 - Les résultats obtenus sur VGG16 avec AO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG16 avec CGO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,746351	0,72	0,717689	0,734829	0,72	2068	111,0606
min	0,726416	0,710625	0,707269	0,724401	0,710625	745	60,63979
var	1,98E-05	4,66E-06	5,1E-06	6,05E-06	4,66E-06	70548,84	118,669
std	0,004455	0,002159	0,002258	0,00246	0,002159	265,6103	10,89353

Tableau 10 - Les résultats obtenus sur VGG16 avec CGO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG16 avec GA pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,74189	0,7275	0,724465	0,73981	0,7275	2072	22,04579
min	0,722141	0,71125	0,707043	0,722774	0,71125	2045	18,61287
var	2,32E-05	8,17E-06	1,03E-05	1,06E-05	8,17E-06	37,65878	0,792534
std	0,004821	0,002858	0,003215	0,00326	0,002858	6,136675	0,890244

Tableau 11 - Les résultats obtenus sur VGG16 avec GA pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG16 avec PSO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,733927	0,72125	0,717669	0,733333	0,72125	2033	21,19734
min	0,730845	0,710625	0,707023	0,721422	0,710625	2020	18,6256
var	1,69E-06	2E-05	1,98E-05	2,38E-05	2E-05	29,03061	0,685524
std	0,001301	0,00447	0,004449	0,004878	0,00447	5,388006	0,827964

Tableau 12 - Les résultats obtenus sur VGG16 avec PSO pour KNN.

Les résultats globaux montrent que les algorithmes CGO et GA obtiennent les meilleures performances en termes de fitness et de taux de reconnaissance pour le modèle VGG16 avec KNN. L'algorithme AO obtient des résultats stables mais légèrement inférieurs, tandis que le PSO affiche des performances légèrement plus faibles.

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 50 itérations pour le modèle VGG16 avec KNN sur les différents algorithmes d'optimisations :

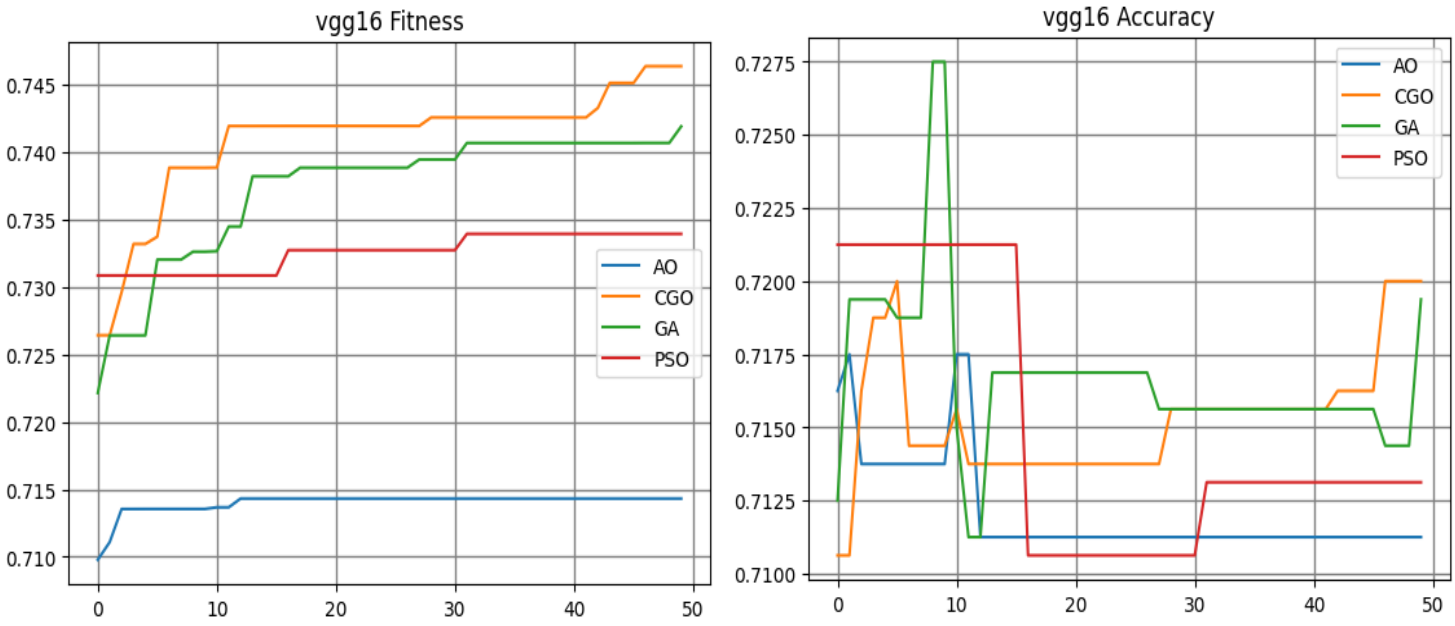


Figure 55 - Graphiques du Fitness et Accuracy pour le modèle VGG16 avec KNN.

- **VGG19**

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG19 avec AO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,719094	0,7	0,696106	0,710667	0,7	2026	40,24809
min	0,715023	0,685625	0,681487	0,697737	0,685625	1550	17,95021
var	4,79E-07	2,63E-05	2,61E-05	2,16E-05	2,63E-05	33491,76	37,72975
std	0,000692	0,005132	0,005105	0,004653	0,005132	183,0076	6,142455

Tableau 13 - Les résultats obtenus sur VGG19 avec AO pour KNN

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG19 avec CGO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,707442	0,691875	0,68775	0,705954	0,691875	2760	126,0011
min	0,704209	0,679375	0,674749	0,691269	0,679375	2040	99,77643
var	6,44E-07	7,47E-06	8,89E-06	1,09E-05	7,47E-06	25999,52	33,15063
std	0,000802	0,002732	0,002982	0,003309	0,002732	161,2437	5,757658

Tableau 14 - Les résultats obtenus sur VGG19 avec CGO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG19 avec GA pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,712246	0,69625	0,69306	0,707394	0,69625	2095	22,47771
min	0,699871	0,69	0,685536	0,69998	0,69	2043	18,30553
var	1,75E-05	5,85E-06	7,75E-06	7,62E-06	5,85E-06	547,2347	1,103595
std	0,004178	0,002418	0,002784	0,00276	0,002418	23,39305	1,050521

Tableau 15- Les résultats obtenus sur VGG19 avec GA pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle VGG19 avec PSO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,707937	0,694375	0,690693	0,705947	0,694375	2066	22,53953
min	0,701091	0,684375	0,679511	0,694493	0,684375	2021	18,51781
var	9,34E-06	3,49E-06	4,23E-06	4,75E-06	3,49E-06	80,31388	1,398672
std	0,003055	0,001869	0,002056	0,002179	0,001869	8,961801	1,182655

Tableau 16- Les résultats obtenus sur VGG19 avec PSO pour KNN.

Les résultats globaux montrent que les algorithmes AO et CGO obtiennent les meilleures performances en termes de fitness et de taux de reconnaissance pour le modèle VGG19 avec KNN, tandis que le PSO et GA affiche des performances légèrement plus faibles.

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 50 itérations pour le modèle VGG19 avec KNN sur les différents algorithmes d'optimisations :

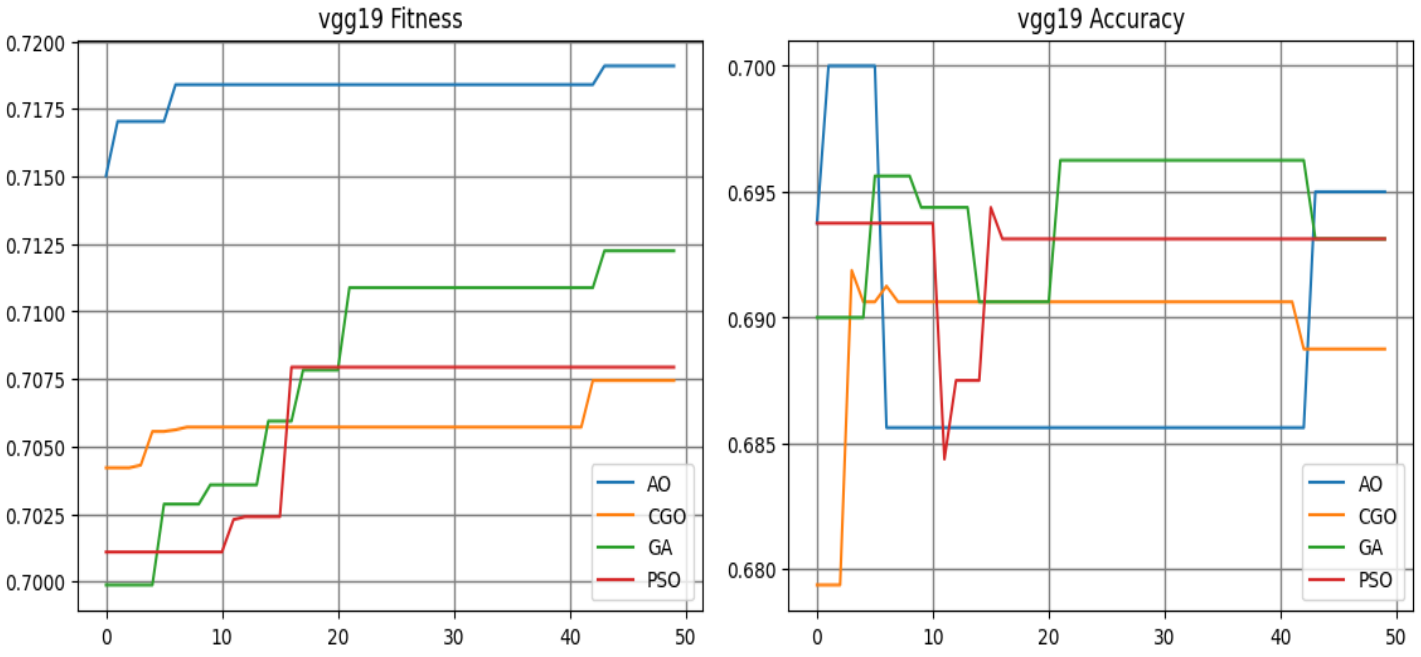


Figure 56- Graphiques du Fitness et Accuracy pour le modèle VGG19 avec KNN.

- **ResNet50**

Le tableau ci-dessous présent les résultats obtenus sur le modèle ResNet50 avec AO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,527327	0,51125	0,50265	0,518956	0,51125	1004	20,57555
min	0,527327	0,51125	0,50265	0,518956	0,51125	1004	8,586693
var	0	0	1,26E-32	0	0	0	8,710761
std	0	0	1,12E-16	0	0	0	2,9514

Tableau 17 -Les résultats obtenus sur ResNet50 avec AO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle ResNet50 avec CGO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,527054	0,501875	0,492856	0,515072	0,501875	1949	77,3458
min	0,50741	0,48875	0,479236	0,499647	0,48875	1028	54,82305
var	1,8E-05	7,22E-06	7,84E-06	9,53E-06	7,22E-06	31603,96	24,92393
std	0,004243	0,002686	0,002799	0,003087	0,002686	177,775	4,992387

Tableau 18 - Les résultats obtenus sur ResNet50 avec CGO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle ResNet50 avec GA pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,530211	0,5125	0,503722	0,522586	0,5125	1052	12,43251
min	0,516117	0,490625	0,480683	0,505087	0,490625	1019	8,505359
var	1,88E-05	8,3E-05	8,85E-05	5,15E-05	8,3E-05	131,7698	0,740507
std	0,004332	0,00911	0,009406	0,007174	0,00911	11,4791	0,860527

Tableau 19 - Les résultats obtenus sur ResNet50 avec GA pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle ResNet50 avec PSO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,52284	0,509375	0,497997	0,513126	0,509375	1048	12,22483
min	0,508838	0,485	0,473164	0,489893	0,485	989	8,3487
var	2,75E-05	3,64E-05	4,17E-05	4,28E-05	3,64E-05	553,1547	0,625386
std	0,005244	0,006032	0,006458	0,006541	0,006032	23,51924	0,790814

Tableau 20 - Les résultats obtenus sur ResNet50 avec PSO pour KNN.

Les résultats globaux montrent que les algorithmes AO et GA obtiennent les meilleures performances en termes de fitness et de taux de reconnaissance pour le modèle ResNet50 avec KNN, tandis que le PSO affiche des performances légèrement plus faibles.

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 50 itérations pour le modèle ResNet50 avec KNN sur les différents algorithmes d'optimisations :

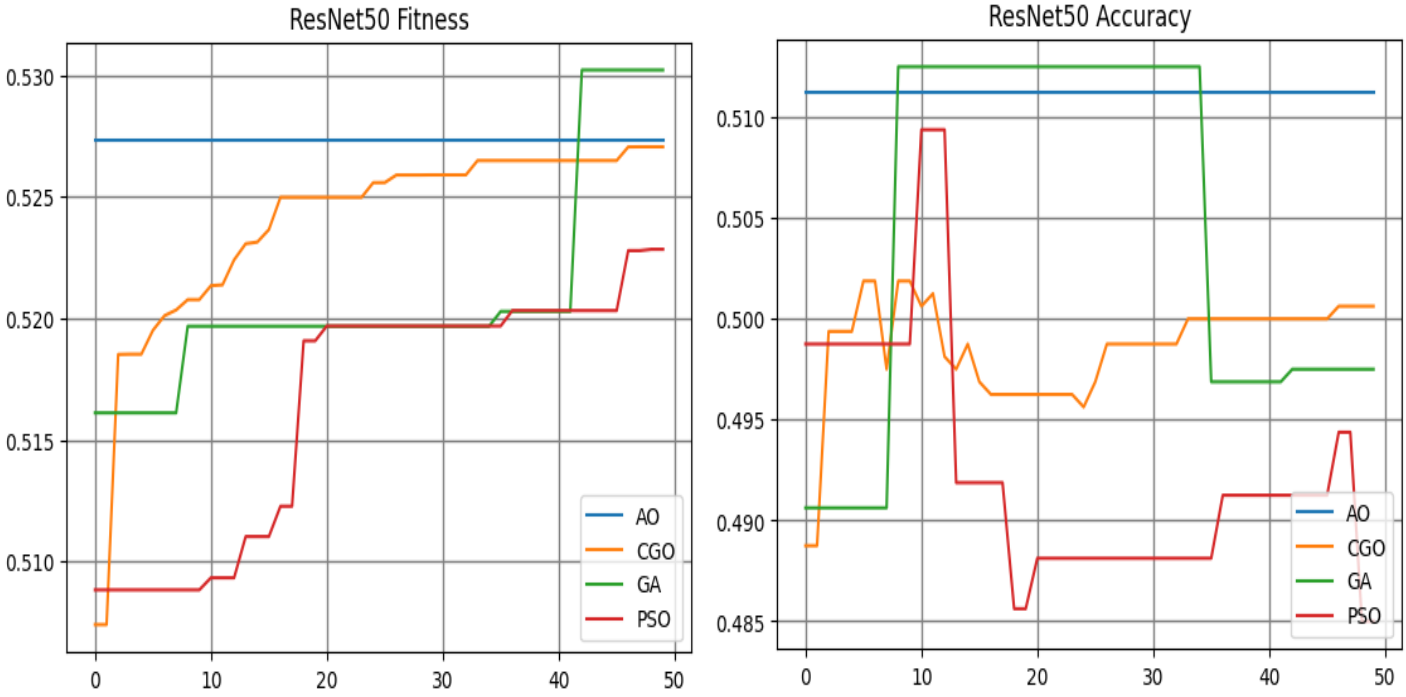


Figure 57 - Graphiques du Fitness et Accuracy pour le modèle ResNet50 avec KNN.

- **MobileNet**

Le tableau ci-dessous présent les résultats obtenus sur le modèle MobileNet avec AO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,774975	0,768125	0,765302	0,780037	0,768125	369	10,73735
min	0,771092	0,7625	0,759322	0,773152	0,7625	352	3,283933
var	1,79E-06	8,02E-07	9,29E-07	1,33E-06	8,02E-07	32,88816	4,956287
std	0,001337	0,000895	0,000964	0,001151	0,000895	5,73482	2,226272

Tableau 21 -Les résultats obtenus sur MobileNet avec AO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle MobileNet avec CGO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,785723	0,76875	0,764935	0,778089	0,76875	701	31,26316
min	0,780304	0,759375	0,755549	0,770875	0,759375	610	23,93276
var	7,32E-07	3,02E-06	2,96E-06	1,47E-06	3,02E-06	278,48	2,265038
std	0,000856	0,001737	0,001721	0,001213	0,001737	16,68772	1,505004

Tableau 22 - Les résultats obtenus sur MobileNet avec CGO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle MobileNet avec GA pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
Max	0,783497	0,763125	0,759393	0,775587	0,763125	509	7,360998
Min	0,780214	0,758125	0,753935	0,767497	0,758125	490	4,424972
Var	6,18E-07	1,99E-06	2,16E-06	4,84E-06	1,99E-06	20,1649	0,843815
Std	0,000786	0,001411	0,001471	0,002201	0,001411	4,490534	0,918594

Tableau 23 - Les résultats obtenus sur MobileNet avec GA pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle MobileNet avec PSO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
Max	0,782849	0,76875	0,765039	0,780702	0,76875	527	8,473994
Min	0,774426	0,75375	0,74953	0,76577	0,75375	469	4,423835
Var	9,04E-06	1,99E-05	2,16E-05	1,91E-05	1,99E-05	230,2551	0,995194
Std	0,003006	0,004465	0,004644	0,004375	0,004465	15,17416	0,997594

Tableau 24 - Les résultats obtenus sur MobileNet avec PSO pour KNN.

Les résultats globaux montrent que l'algorithme CGO obtienne les meilleures performances en termes de fitness et l'algorithme AO obtienne les meilleures performances en termes de taux de reconnaissance pour le modèle MobileNet avec KNN.

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 50 itérations pour le modèle MobileNet avec KNN sur les différents algorithmes d'optimisations :

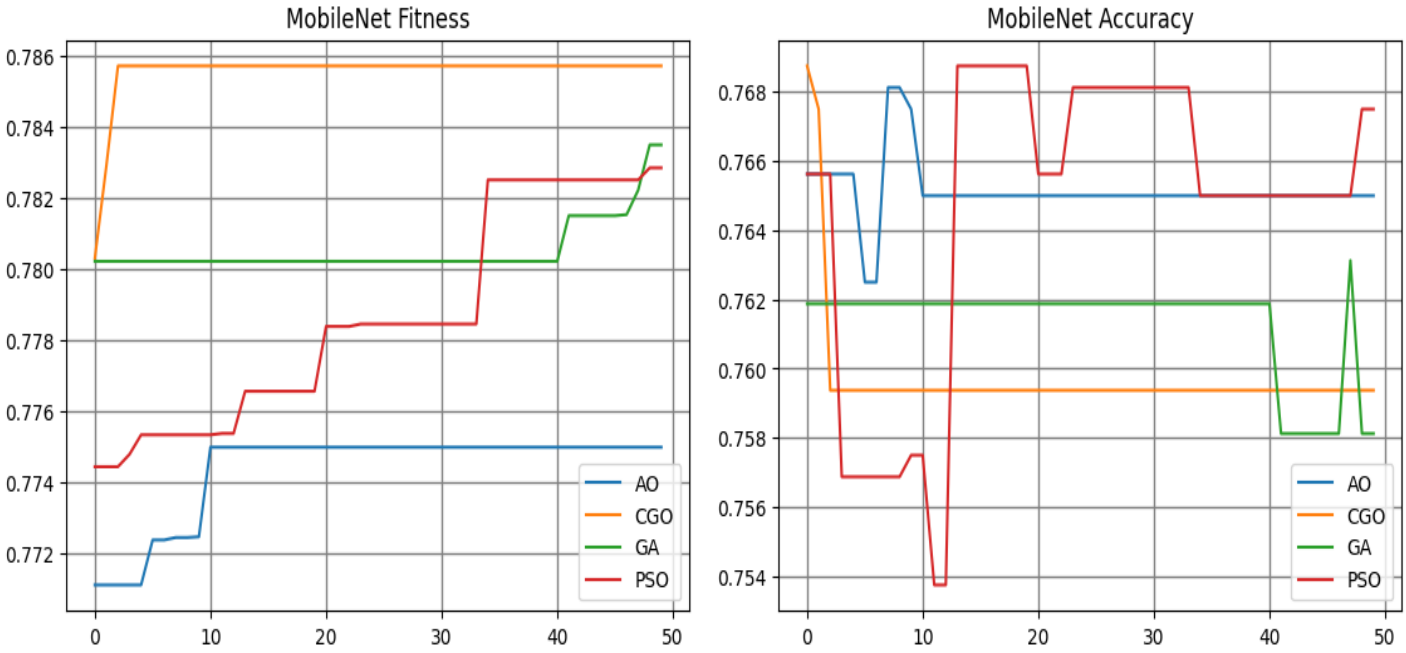


Figure 58 - Graphiques du Fitness et Accuracy pour le modèle MobileNet avec KNN.

- **GoogleNet**

Le tableau ci-dessous présent les résultats obtenus sur le modèle GoogleNet avec AO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,781784	0,7675	0,763983	0,776364	0,7675	1123	21,21013
min	0,774253	0,7525	0,749492	0,762138	0,7525	956	8,522704
var	4,32E-06	1,52E-05	1,45E-05	1,45E-05	1,52E-05	1406,858	12,62018
std	0,002078	0,003899	0,003801	0,003803	0,003899	37,5081	3,552489

Tableau 25 -Les résultats obtenus sur GoogleNet avec AO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle GoogleNet avec CGO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,793325	0,77625	0,773915	0,786864	0,77625	1332	61,58894
min	0,782569	0,765	0,761896	0,77274	0,765	1043	47,60023
var	7,29E-06	2,24E-05	2,47E-05	3,63E-05	2,24E-05	9828,483	10,53124
std	0,0027	0,004738	0,004972	0,006029	0,004738	99,13871	3,245187

Tableau 26 - Les résultats obtenus sur GoogleNet avec CGO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle GoogleNet avec GA pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,791314	0,76375	0,759472	0,772753	0,76375	1055	17,79997
min	0,779168	0,75625	0,752949	0,764552	0,75625	1002	8,740847
var	1,63E-05	2,15E-06	1,78E-06	3,33E-06	2,15E-06	89,59184	5,097324
std	0,00404	0,001465	0,001333	0,001823	0,001465	9,465296	2,257725

Tableau 27 - Les résultats obtenus sur GoogleNet avec GA pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle GoogleNet avec PSO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,789088	0,76875	0,7659	0,777465	0,76875	1052	11,19191
min	0,772274	0,76125	0,758039	0,769828	0,76125	998	8,659481
var	1,62E-05	3,29E-06	3,52E-06	4,21E-06	3,29E-06	283,6882	0,879286
std	0,004031	0,001814	0,001877	0,002051	0,001814	16,84304	0,937702

Tableau 28 - Les résultats obtenus sur GoogleNet avec PSO pour KNN.

Les résultats globaux montrent que l'algorithme CGO obtienne les meilleures performances en termes de fitness et de taux de reconnaissance pour le modèle GoogleNet avec KNN.

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 50 itérations pour le modèle GoogleNet avec KNN sur les différents algorithmes d'optimisations :

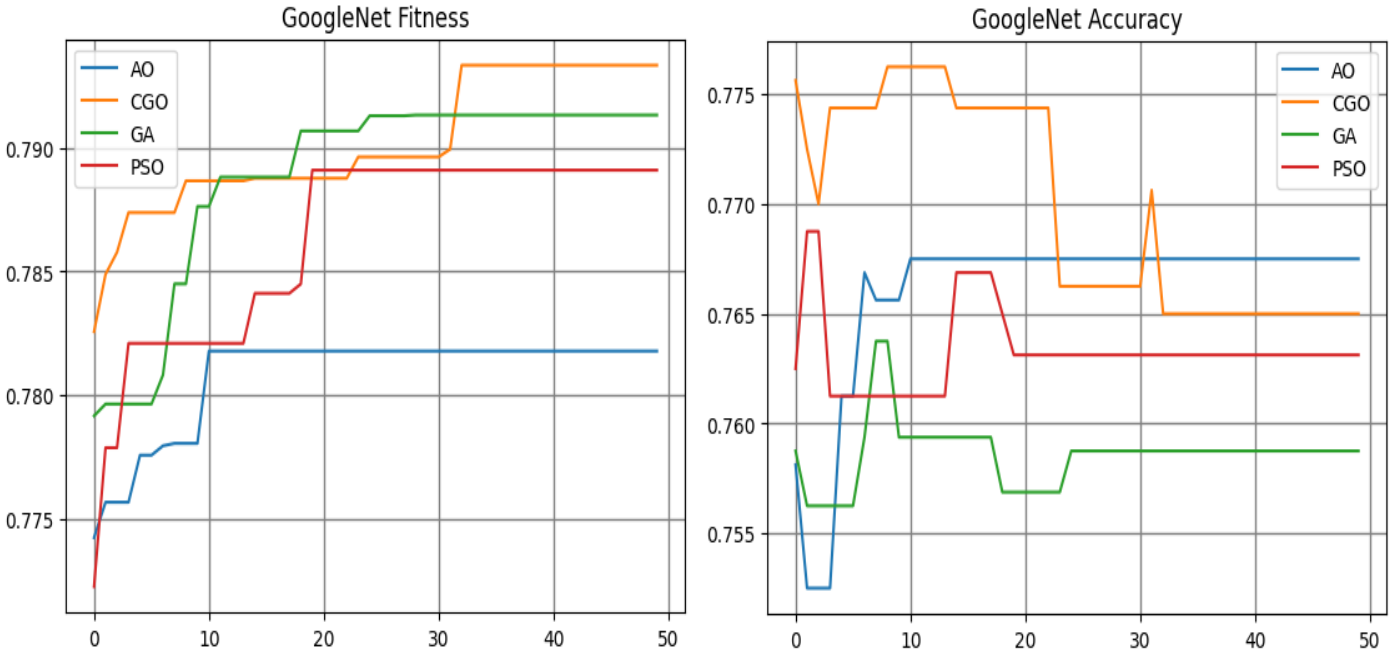


Figure 59 - Graphiques du Fitness et Accuracy pour le modèle GoogleNet avec KNN.

- **Modèle proposé**

Le tableau ci-dessous présent les résultats obtenus sur le modèle combiné avec AO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,835758	0,83875	0,835992	0,845711	0,83875	7101	137,4742
min	0,823027	0,82625	0,822602	0,833416	0,82625	5983	64,76964
var	1,91E-05	1,74E-05	2,02E-05	1,51E-05	1,74E-05	181624,1	243,5616
std	0,004366	0,004176	0,004497	0,003892	0,004176	426,1738	15,60646

Tableau 29 -Les résultats obtenus sur le modèle combiné avec AO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle combiné avec CGO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,830173	0,83375	0,829435	0,841092	0,83375	7947	419,9912
min	0,822379	0,825625	0,821364	0,831823	0,825625	5188	300,3575
var	4,36E-06	5,19E-06	5,69E-06	6,85E-06	5,19E-06	149171,5	453,9864
std	0,002088	0,002278	0,002385	0,002617	0,002278	386,2273	21,30696

Tableau 30 - Les résultats obtenus sur le modèle combiné avec CGO pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle combiné avec GA pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,838458	0,841875	0,838148	0,849174	0,841875	6761	68,4241
min	0,821044	0,824375	0,819248	0,830263	0,824375	6544	61,82263
var	4,5E-05	4,6E-05	5,02E-05	4,75E-05	4,6E-05	3089,675	1,748022
std	0,006705	0,006781	0,007088	0,006895	0,006781	55,58485	1,322128

Tableau 31 - Les résultats obtenus sur le modèle combiné avec GA pour KNN.

Le tableau ci-dessous présent les résultats obtenus sur le modèle combiné avec PSO pour KNN sur 50 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,829204	0,8325	0,82878	0,838878	0,8325	6670	65,06313
min	0,81928	0,8225	0,818589	0,82859	0,8225	6600	62,11131
var	9,09E-06	9,21E-06	9,4E-06	8,97E-06	9,21E-06	274,2596	0,749091
std	0,003014	0,003035	0,003065	0,002995	0,003035	16,56078	0,8655

Tableau 32 - Les résultats obtenus sur le modèle combiné avec PSO pour KNN.

Après avoir évalué plusieurs modèles pré-entraînés en utilisant différentes méthodes d'optimisation, nous avons constaté que le modèle GoogleNet avec la classification KNN, associé à l'algorithme d'optimisation CGO, a produit les meilleurs résultats en termes de performance. Ce modèle a atteint un taux de reconnaissance impressionnante de 79,33%, démontrant sa capacité à capturer et à exploiter efficacement les caractéristiques discriminantes des données. En revanche, le modèle ResNet, malgré son potentiel, a montré un mauvais taux de reconnaissance de seulement 51,52 % lorsqu'il était associé à l'algorithme d'optimisation PSO. Ces résultats soulignent l'importance de choisir judicieusement à la fois le modèle pré-entraîné et l'algorithme d'optimisation pour maximiser les performances de classification.

Notre modèle développé a dépassé tous les modèles pré-entraînés évalués en termes de performance, en atteignant un taux de reconnaissance parfait de 83,84 %. Avec l'utilisation de la classification KNN en combinaison avec l'algorithme d'optimisation GA.

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 50 itérations pour notre modèle combiné avec KNN sur les différents algorithmes d’optimisations :

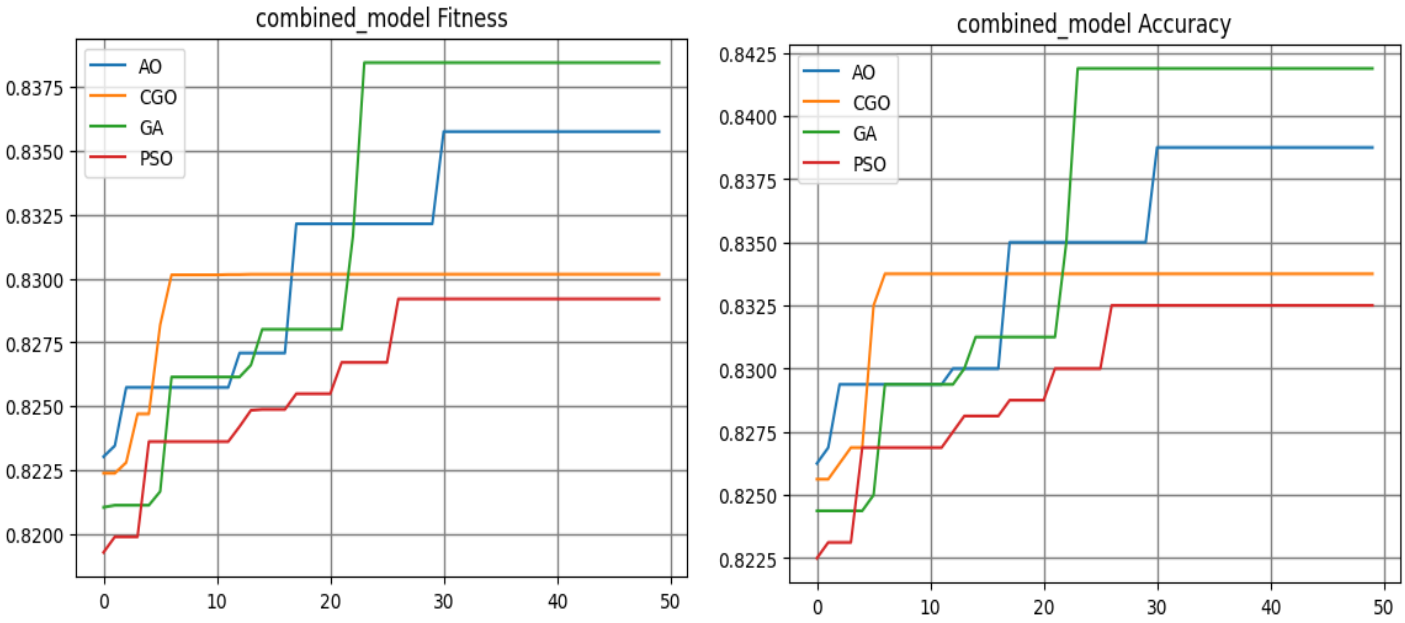


Figure 60 - Graphiques du Fitness et Accuracy pour le modèle combiné avec KNN.

4.3.2.2 RandomForest

À ce stade de notre étude, nous avons appliqué les algorithmes d'optimisation AO, CGO, GA et PSO sur notre modèle combiné en utilisant le classificateur RandomForest comme algorithme de base avec 10 itérations et une population de 10 individus.

Les résultats sont présentés ci-dessous :

- **Aquila Optimizer (AO)**

Le tableau ci-dessous présent les résultats obtenus sur notre modèle avec AO pour RandomForest sur dix itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,943024	0,931875	0,931842	0,931956	0,931875	13288	636,3093
min	0,935552	0,914375	0,914021	0,913863	0,914375	9935	493,4584
var	1,3E-05	2,67E-05	2,73E-05	2,75E-05	2,67E-05	2623275	2986,647
std	0,00361	0,005169	0,005223	0,005248	0,005169	1619,653	54,65022

Tableau 33 – Les résultats obtenus sur notre modèle avec AO pour RandomForest.

- **Chaos Game Optimization (CGO)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec CGO pour RandomForest sur dix itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,937943	0,925	0,924798	0,924876	0,925	9287	2282,392
min	0,935867	0,9125	0,912266	0,912162	0,9125	8757	1912,903
var	5,87E-07	1,71E-05	1,77E-05	1,88E-05	1,71E-05	49937,78	11904,01
std	0,000766	0,004136	0,004203	0,004338	0,004136	223,4676	109,1055

Tableau 34 - Les résultats obtenus sur notre modèle avec CGO Pour RandomForest.

- **Les algorithmes génétiques (GA)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec GA pour RandomForest sur dix itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,942966	0,93125	0,930966	0,930922	0,93125	6724	659,5302
min	0,938728	0,915625	0,915237	0,914976	0,915625	6599	637,43
var	2,94E-06	2,07E-05	2,09E-05	2,11E-05	2,07E-05	2168,711	55,48097
std	0,001715	0,004555	0,004569	0,004594	0,004555	46,56942	7,448555

Tableau 35 - Les résultats obtenus sur notre modèle avec GA pour RandomForest.

- **L'optimisation par essaim particulaire (PSO)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec PSO pour RandomForest sur dix itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
max	0,940552	0,925	0,924842	0,924832	0,925	6643	676,8672
min	0,938716	0,915	0,914872	0,915218	0,915	6595	631,5754
var	2,07E-07	9,24E-06	9,32E-06	9E-06	9,24E-06	249,0667	240,5296
std	0,000455	0,00304	0,003053	0,003	0,00304	15,78185	15,50902

Tableau 36 - Les résultats obtenus sur notre modèle avec PSO pour RandomForest.

- **Discussion**

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 10 itérations pour notre modèle combiné avec RandomForest sur les différents algorithmes d'optimisations :

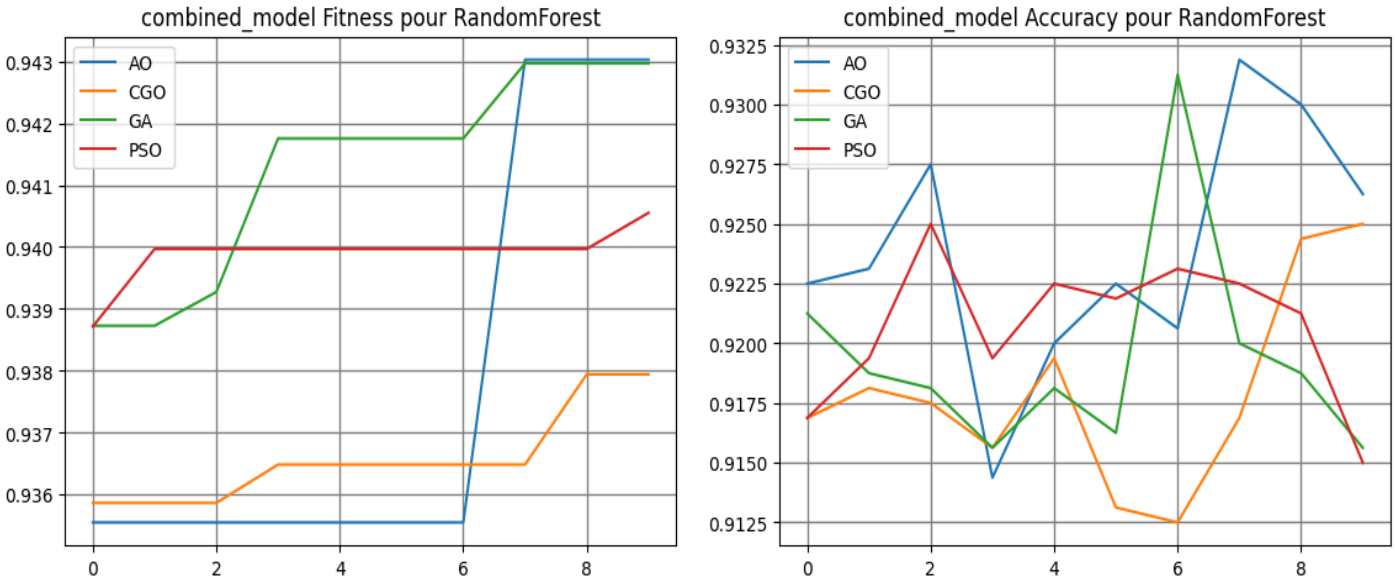


Figure 61 - Graphiques du Fitness et Accuracy pour le modèle combiné avec RandomForest.

Lors de notre étude comparative des algorithmes d'optimisation appliqués à la classification RandomForest, nous avons observé des performances différentes pour chaque algorithme. L'algorithme génétique (GA) s'est révélé être le plus performant en termes de fitness, atteignant une valeur remarquable de 94,29%.

Ensuite, l'algorithme PSO a obtenu la deuxième meilleure fitness avec un score de 94,05%. En revanche, l'algorithme AO a généré des solutions plus complexes, avec un chromosome de taille 9935, tandis que les autres algorithmes ont généré des solutions moins complexes, avec des tailles de chromosome inférieures à 10 000.

En ce qui concerne le temps d'exécution, l'algorithme AO s'est avéré être le plus rapide, avec un temps moyen de 600,7799275 secondes. L'algorithme GA a suivi de près avec un temps moyen de 647,0533216 secondes.

En revanche, algorithme d'optimisation CGO malgré son potentiel, a montré un mauvaise taux de reconnaissance de seulement 93,58 % par rapport tous les algorithmes d'optimisation Cependant, les algorithmes CGO et PSO ont nécessité plus de temps d'exécution, avec des valeurs moyennes de 2201,053709 secondes et 634,5547064 secondes respectivement.

Ces résultats mettent en évidence les différences de performances et de complexité des algorithmes d'optimisation dans le contexte de la classification RandomForest. Bien que l'algorithme génétique ait obtenu les meilleures performances en termes de fitness, il est essentiel de prendre en compte le temps d'exécution et la complexité des solutions générées lors du choix de l'algorithme le plus adapté à une tâche spécifique.

4.3.2.3 Le perceptron multicouche (MLP)

À ce stade de notre étude, nous avons appliqué les algorithmes d'optimisation AO, CGO, GA et PSO sur notre modèle combiné en utilisant le classificateur MLP comme algorithme de base avec 5 itérations et une population de 10 individus. Les résultats sont présentés ci-dessous :

- **Aquila Optimizer (AO)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec AO pour MLP sur 5 itérations:

	Fitness	**Taux**	**F-score**	**Rappel**	**Précision**	**Taille de Chrom**	**Temps (s)**
max	0,9746	0,970625	0,97061	0,970784	0,970625	6608	4244,845
min	0,970897	0,968125	0,968134	0,968222	0,968125	5646	2625,678
var	3,2E-06	1,64E-06	1,61E-06	1,65E-06	1,64E-06	162775,2	488006,1
std	0,001789	0,001281	0,00127	0,001283	0,001281	403,4541	698,5743

Tableau 37 - Les résultats obtenus sur notre modèle avec AO pour MLP.

- **Chaos Game Optimization (CGO)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec CGO pour MLP sur 5 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
Max	0,971757	0,978125	0,978125	0,978158	0,978125	8754	8083,153
Min	0,970235	0,975	0,975006	0,975196	0,975	6665	6434,52
Var	6,95E-07	2,93E-06	2,92E-06	2,63E-06	2,93E-06	1309176	459841,6
Std	0,000834	0,001712	0,001708	0,001622	0,001712	1144,192	678,1162

Tableau 38 - Les résultats obtenus sur notre modèle avec CGO pour MLP.

- **Les algorithmes génétiques (GA)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec GA pour MLP sur 5 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
Max	0,974558	0,97125	0,971166	0,971163	0,97125	6676	2111,056
Min	0,97149	0,96875	0,968719	0,968715	0,96875	6581	1976,694
Var	1,64E-06	1,64E-06	1,58E-06	1,61E-06	1,64E-06	2292,7	3131,593
Std	0,001279	0,001281	0,001258	0,001268	0,001281	47,88215	55,96064

Tableau 39 - Les résultats obtenus sur notre modèle avec GA pour MLP.

- **L'optimisation par essaim particulaire (PSO)**

Le tableau ci-dessous présente les résultats obtenus sur notre modèle avec PSO pour MLP sur5 itérations:

	Fitness	Taux	F-score	Rappel	Précision	Taille de Chrom	Temps (s)
Max	0,975876	0,970625	0,970598	0,970604	0,970625	6614	2876,014
Min	0,97213	0,968125	0,967997	0,968146	0,968125	6569	2453,755
Var	2,81E-06	1,25E-06	1,35E-06	1,21E-06	1,25E-06	405	25451,69
Std	0,001675	0,001118	0,001163	0,001099	0,001118	20,12461	159,5359

Tableau 40 - Les résultats obtenus sur notre modèle avec PSO pour MLP.

- **Discussion**

Les graphiques ci-dessous présentent les résultats du Fitness et de taux de reconnaissance obtenus sur 5 itérations pour notre modèle combiné avec MLP sur les différents algorithmes d'optimisations :

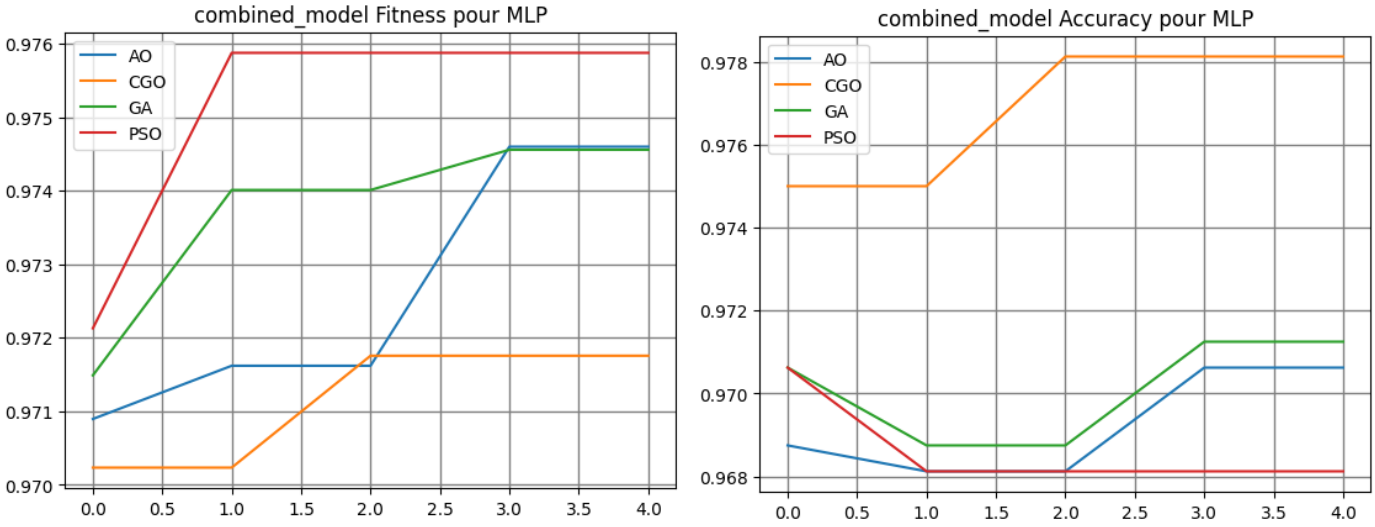


Figure 62 - Graphiques du Fitness et Accuracy pour le modèle combiné avec MLP.

Les algorithmes génétiques (GA) et l'optimisation par essaim particulaire (PSO) ont obtenu des résultats légèrement supérieurs en termes de fitness (97,46% et 97,59% respectivement) par rapport à Aquila Optimizer (AO) et Chaos Game Optimization (CGO).

Cela indique que GA et PSO ont réussi à trouver des solutions plus précises et discriminantes pour la tâche de classification.

Les résultats des algorithmes génétiques (GA) montrent une stabilité dans les performances au fil des itérations, avec une amélioration continue mais modérée. Cela suggère une exploration progressive et régulière de l'espace de recherche.

L'optimisation par essaim particulaire (PSO) présente également une stabilité dans les performances, avec des améliorations graduelles et similaires à celles de GA. Aquila Optimizer (AO) et Chaos Game Optimization (CGO) montrent des fluctuations dans les performances d'une itération à l'autre. Cela peut indiquer une exploration plus chaotique de

l'espace de recherche. Les temps d'exécution varient pour chaque algorithme, mais en général, Aquila Optimizer (AO) et les algorithmes génétiques (GA) semblent être plus rapides par rapport à Chaos Game Optimization (CGO) et l'optimisation par essaim particulaire (PSO).

4.3.3 Synthèse

Les algorithmes d'optimisation jouent un rôle crucial dans le processus de sélection des attributs pour minimiser le nombre de paramètres. Ces algorithmes permettent d'identifier les attributs les plus pertinents et significatifs pour améliorer les performances des modèles de classification. Les résultats obtenus dans notre étude démontrent l'importance de l'utilisation de ces algorithmes. Pour l'algorithme KNN, nous avons observé des pourcentages de diminution significatifs lorsque l'optimisation est appliquée. Avec l'algorithme d'optimisation (AO), nous avons obtenu une diminution de 54.01%, tandis que CGO, GA et PSO ont conduit à des diminutions respectives de 47.62%, 50.02% et 50.32%. De même, pour l'algorithme RandomForest, les pourcentages de diminution obtenus étaient de 25.27% avec AO, 30.10% avec CGO, 49.38% avec GA et 50.02% avec PSO. En ce qui concerne l'algorithme MLP, nous avons constaté des améliorations significatives grâce à l'utilisation des algorithmes d'optimisation. Les pourcentages de diminution obtenus étaient de 56.37% avec AO, 34.11% avec CGO, 49.71% avec GA et 50.57% avec PSO.

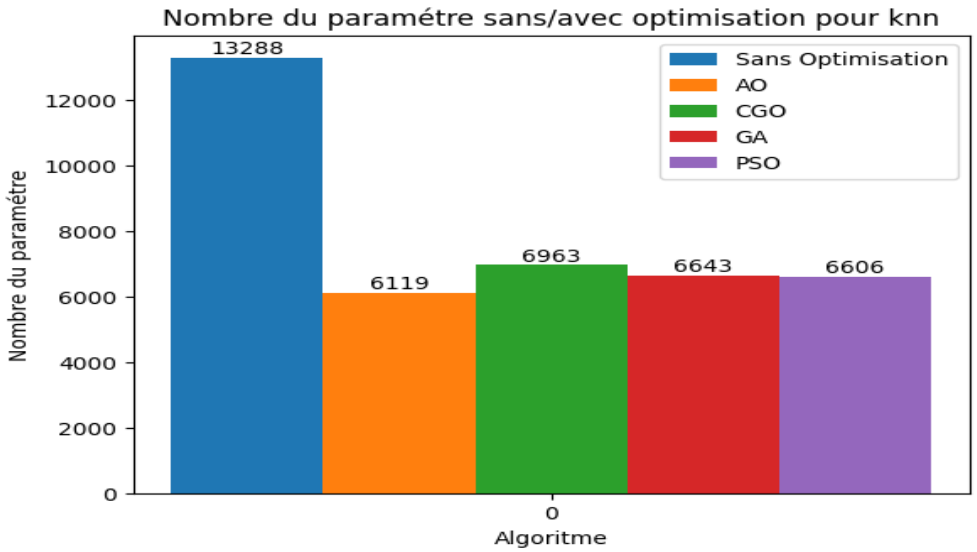


Figure 63 – Nombre du paramètre sans/avec optimisation pour KNN.

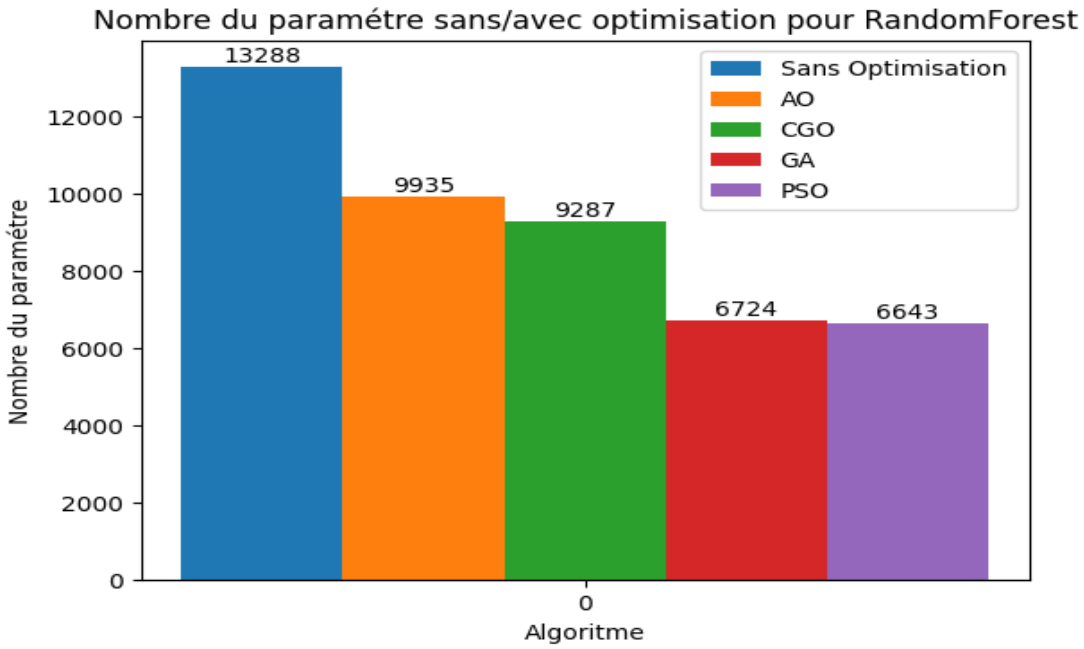


Figure 64 - Nombre du paramètre sans/avec optimisation pour RandomForest.

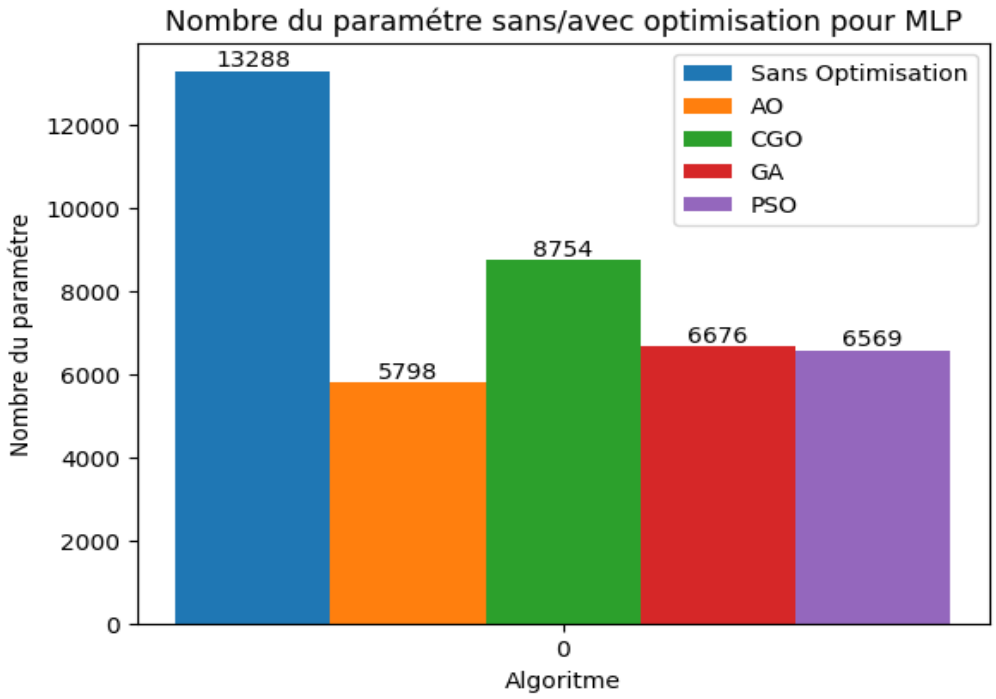


Figure 65 - Nombre du paramètre sans/avec optimisation pour MLP.

Dans l'état de l'art nous avons présenté des études récentes utilisant des méthodes CNN pour classifications et segmentations des maladies Gastro-intestinale. En comparant ces travaux avec notre étude, il est évident que notre modèle combiné a obtenu des performances exceptionnelles avec un taux de reconnaissance de 97,59% comme illustré sur le tableau 41. Cela démontre l'efficacité de notre approche pour la base de données KVASIR V2.

Etude	**Base de données**	**Taux de reconnaissance**
Taruna Agrawal et al, 2017	KVASIR V2	82.6%
Yogapriya Jaganathan et al, Septembre 2021	KVASIR V2	96.33%
Zenebe Markos Lonseko et al, Novembre 2021	KVASIR V2	94.33%
Notre Méthode, Juin 2023	KVASIR V2	97,59%

Tableau 41 - Comparaison des travaux avec notre étude.

4.4 Application

Ci-dessous, nous présentons nos interfaces d'application Web « AutoGastro » dans le but de permettre aux médecins de connaître le type de maladie gastro-intestinale en fonction de l'image importée par endoscopie.

Figure 66 - Le logo d’application Web.

- **La page d’accueil**

La page d’accueil fournit des informations générales de notre travail. Cette page fournit des hyperliens vers les sections et les interfaces que constitue notre application Web. Les sections que constitue notre application Web illustré dans la figure 65.

Figure 67 – La page d'accueil de l'application.

- **L'interface de détection**

Le but de cette interface est de prédire le type de maladie gastro-intestinale, pour cela il doit sélectionner une image de l'intérieur d'un organe digestif supérieur.

Figure 68 - L'interface de détection.

Une fois l'image est importer il doit cliquer sur le bouton « Predict » pour que l'image sera récupérée et entrainées par notre modèle et le résultat de prédiction sera affichée :

Figure 69 – Résultats de prédiction.

- **La section A Propos**

Cette section permet de fournir des informations générales sur nos services.

Figure 70 – La section A Propos.

- **La section Données**

Cette section permet de fournir des données générales sur les cas et les décès par cancer gastro-intestinale dans le monde et en Algérie.

Figure 71 – La section Données.

- **La section Endoscopie**

Cette section présente une vidéo explicative sur le travail d'endoscopie

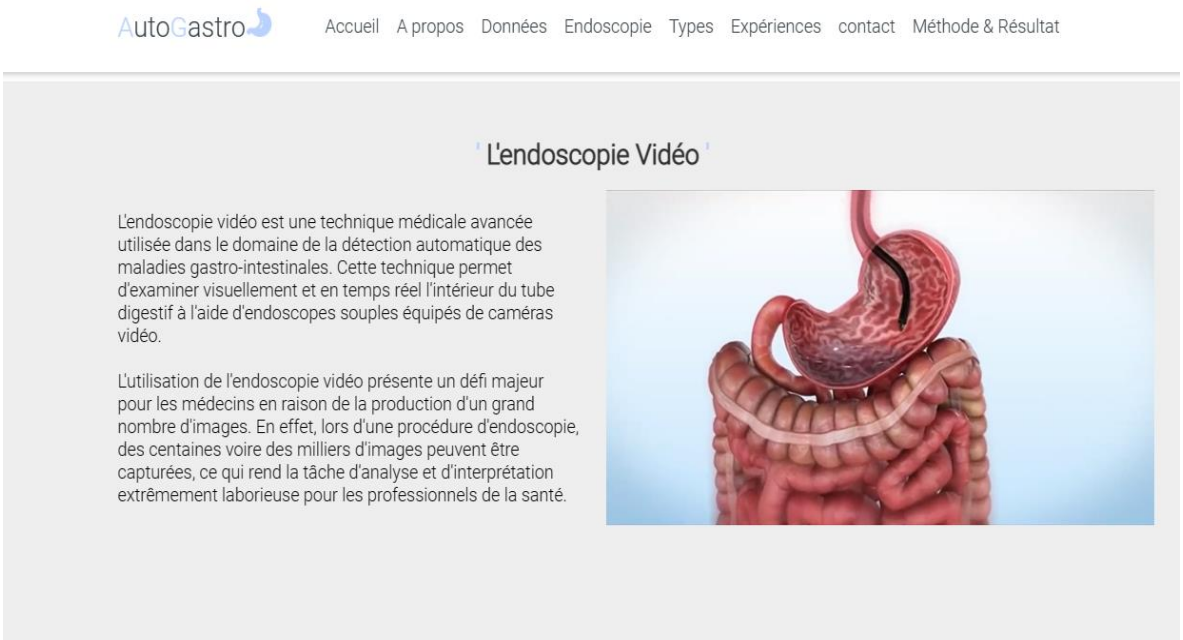


Figure 72 – La section Endoscopie.

- **La section Types**

Cette section permet d'afficher des informations sur les types des maladies gastro-intestinales.

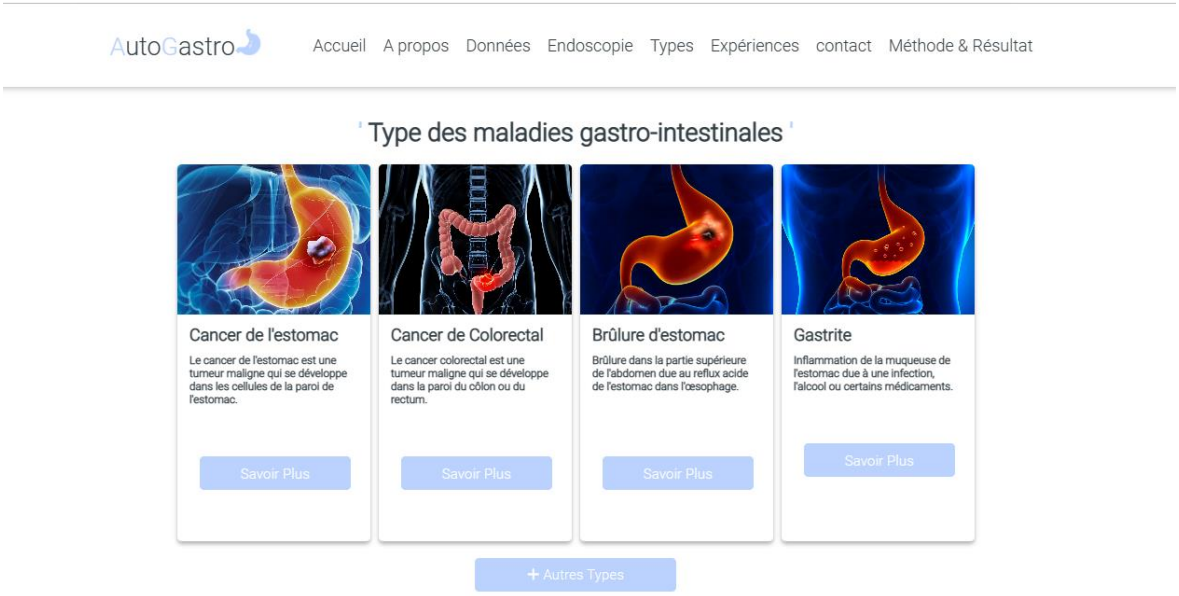


Figure 73 – La section Types.

Et voici quand cliquer sur le bouton « Savoir plus» :

Figure 74 – Interface sur Cancer de l'estomac.

Figure 75 – Interface sur les symptômes de Cancer de l'estomac.

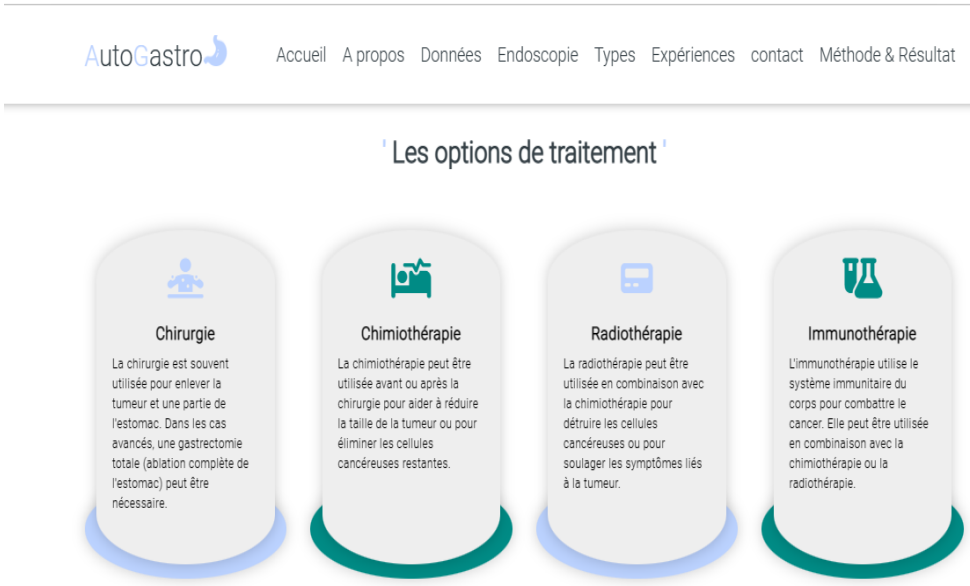


Figure 76 - Interface sur le traitement de Cancer de l'estomac.

Et quand cliquer sur le bouton « Autres Types » :

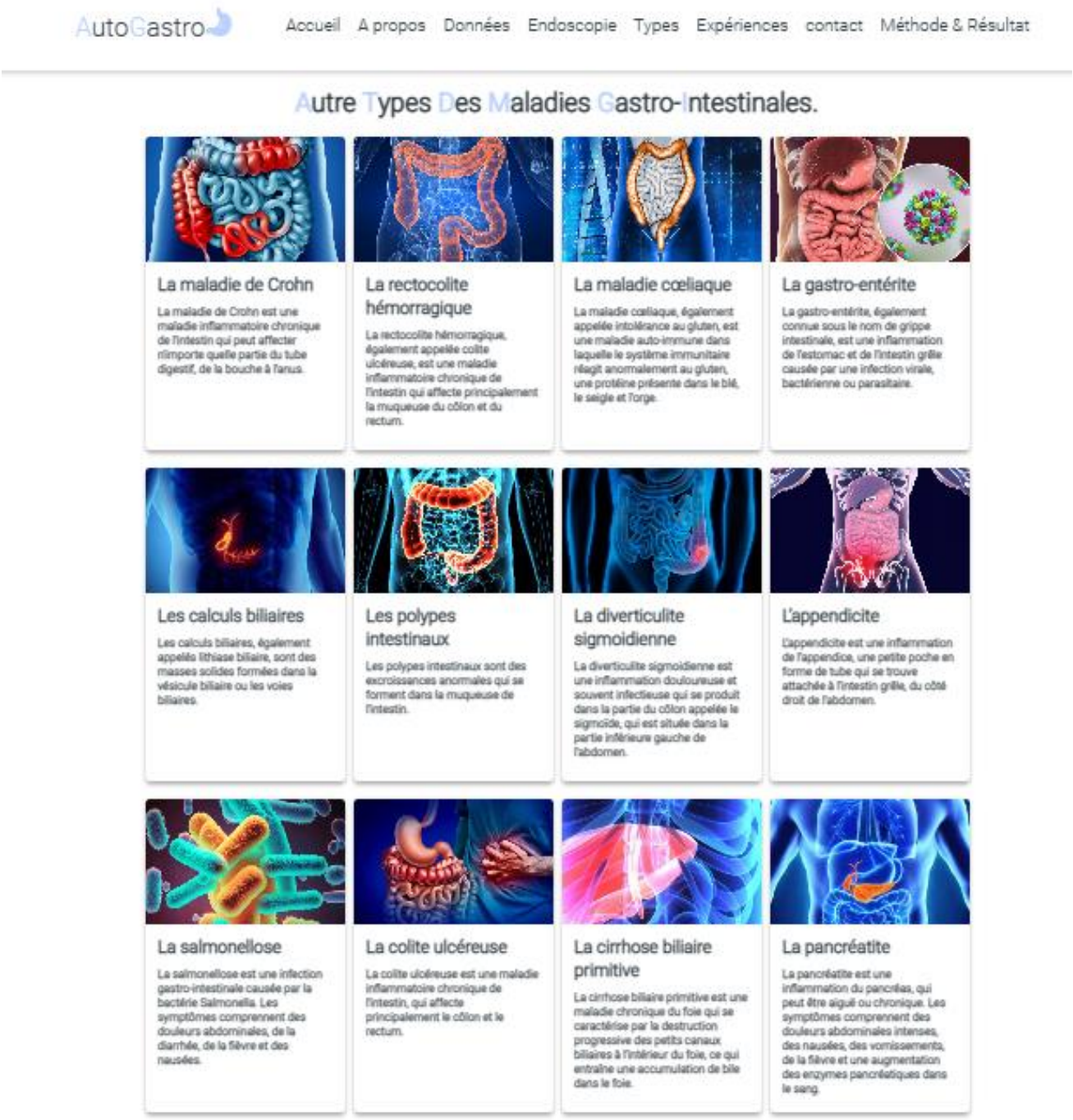


Figure 77 – Interface Autres types des maladies gastro-intestinales.

- **La section Expériences**

Cette section permet d'afficher les expériences déjà réalisé dans ce domaine avec différents base de données et différents méthodes.

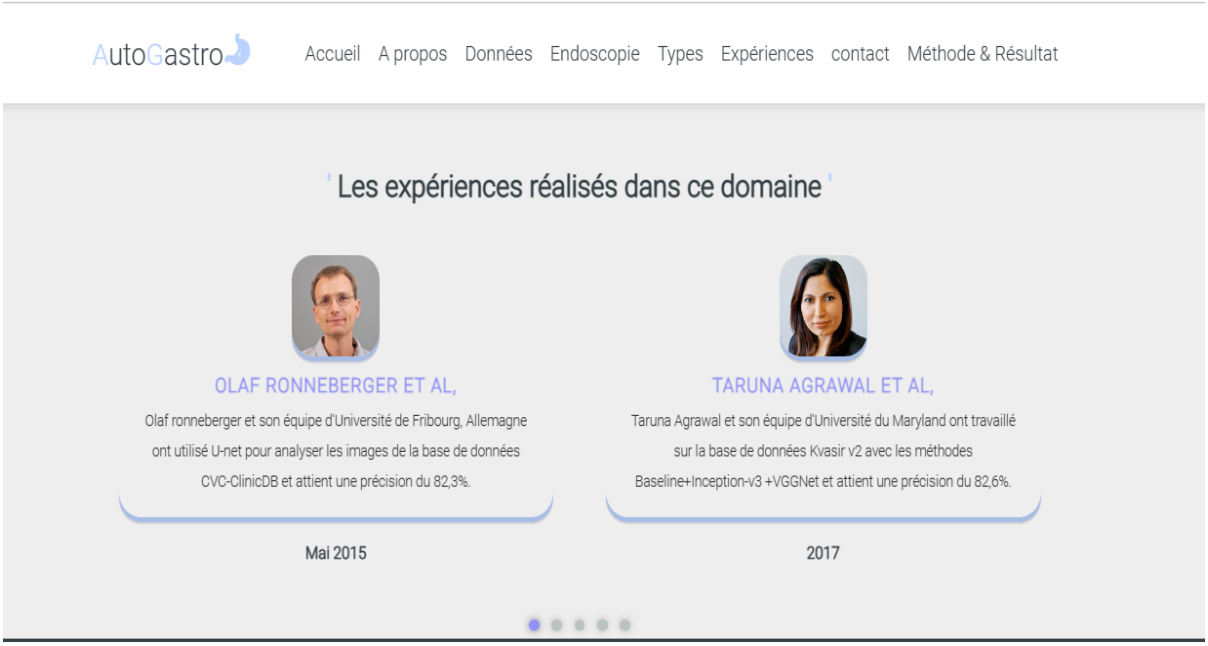


Figure 78 – La section Expériences.

- **La section Contact**

Cette interface aide les utilisateurs à nous contacter pour prendre un rendez-vous.

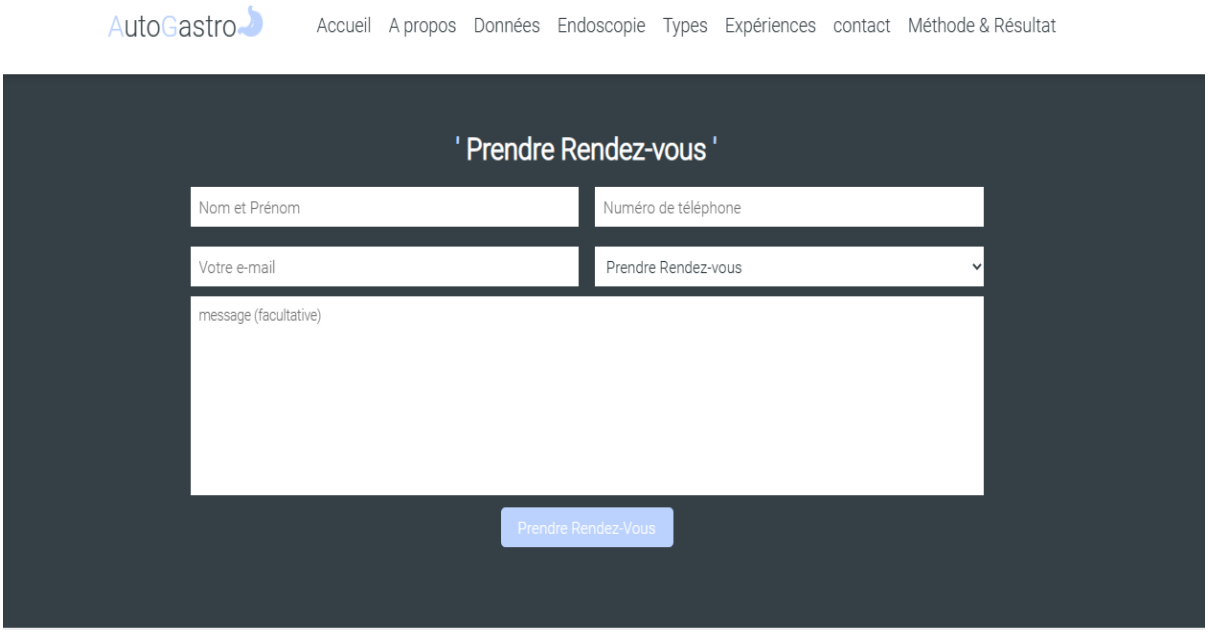


Figure 79 – La section Contact.

- **La section Méthodes et Résultats**

Cette interface permet d'accéder aux méthodes, résultats sans optimisation et avec optimisation.

Figure 80 – La section Méthodes et Résultats.

Quand cliquer sur le bouton « Savoir Plus » pour Les méthodes:

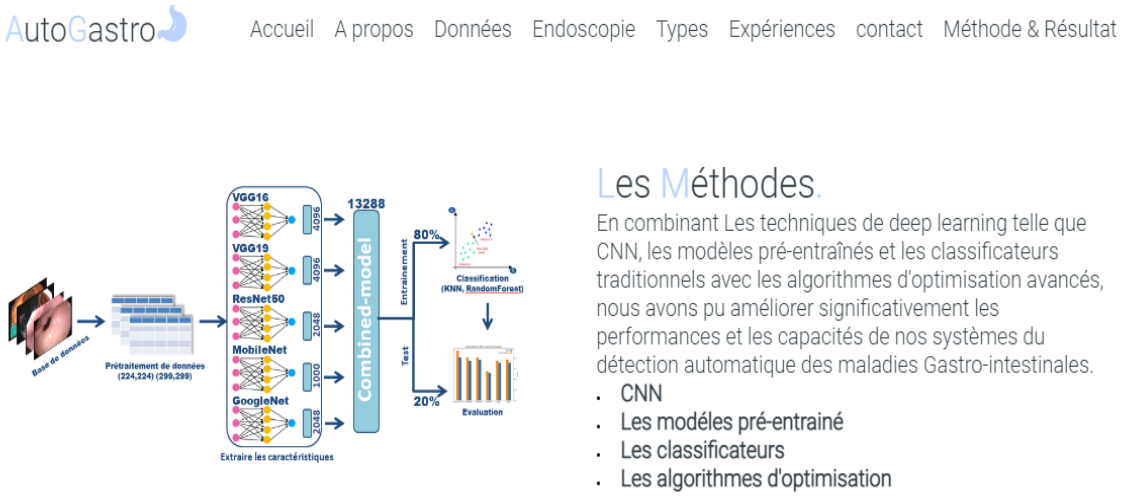


Figure 81 - Interface Les méthodes.

Quand cliquer sur le bouton « Savoir Plus » pour Résultats Sans Optimisation:

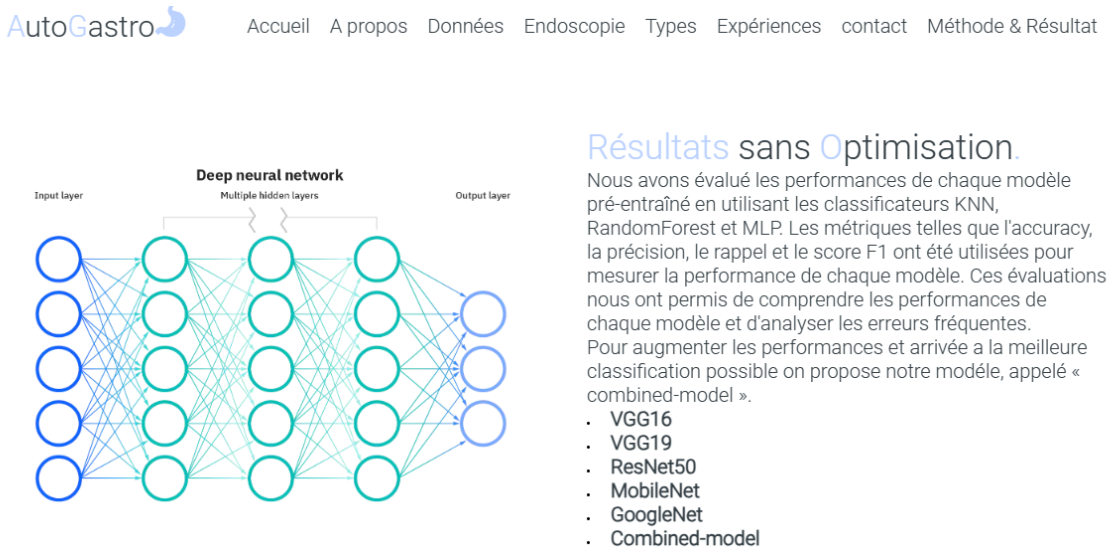


Figure 82 – Interface Résultats Sans Optimisation.

Quand cliquer sur le bouton « Savoir Plus » pour Résultats Avec Optimisation:

Figure 83 - Interface Résultats Avec Optimisation.

4.5 Conclusion

En conclusion, notre étude a porté sur l'implémentation de différents algorithmes d'optimisation tels qu'Aquila Optimizer (AO), Chaos Game Optimization (CGO), algorithme génétique (GA) et optimisation par essaim particulaire (PSO), associés à des classificateurs tels que KNN, RandomForest et MLP. Nous avons également évalué plusieurs modèles pré-entraînés couramment utilisés.

Par ailleurs, notre modèle proposé, appelé "combined-model", a surpassé tous les modèles pré-entraînés évalués en termes de performance, en atteignant un taux de reconnaissance remarquable de 97,59% et un pourcentage de diminution du nombre des caractéristiques de 50.57% avec le classificateurs MLP et l'algorithme PSO.

Enfin, nous avons présenté notre application, qui utilise l'apprentissage profond optimisé. Nous avons expliqué son fonctionnement et mis en avant ses fonctionnalités clés, démontrant ainsi l'application pratique de notre approche dans un contexte réel.

Conclusion Générale

Ce projet de détection automatique des maladies gastro-intestinales représente une avancée significative. Les méthodes développées visent à fournir un diagnostic rapide et précis des troubles gastro-intestinaux courants tels que la gastrite, les brûlures d'estomac, le cancer colorectal et d'estomac. L'utilisation de techniques d'imagerie médicale, notamment les bases de données telles que KVASIR V2, HYPERKVASIR, KVASIR-INSTRUMENT et CVC-CLINICDB, combinée aux algorithmes d'apprentissage profond, a démontré des performances prometteuses.

L'apprentissage profond, en particulier les réseaux de neurones convolutionnels tels que VGG16, VGG19, ResNet50, MobileNet et GoogleNet, associés à des classificateurs tels que KNN, RandomForest et MLP, joue un rôle clé dans la détection automatique des maladies gastro-intestinales. Notre modèle innovant, le "combined-model", qui combine les caractéristiques extraites de ces différents modèles pré-entraînés, a permis d'obtenir une classification optimale et des performances exceptionnelles.

De plus, l'utilisation d'algorithmes d'optimisation tels qu'Aquila Optimizer (AO), Chaos Game Optimization (CGO), l'optimisation par essaim particulaire (PSO) et les algorithmes génétiques pour la sélection d'attributs a apporté des avantages significatifs en termes de performance et de précision. Notamment, notre modèle combiné a obtenu des performances exceptionnelles avec un taux de reconnaissance de 97,59% et un pourcentage de diminution du nombre des caractéristiques de 50.57% en utilisant l'optimisation par essaim particulaire (PSO) et le classificateur MLP.

Comparé aux travaux existants dans la littérature, notre modèle combiné a démontré la meilleure performance, ce qui souligne son potentiel pour améliorer le diagnostic des maladies gastro-intestinales.

Pour les travaux futurs, il est essentiel de continuer à améliorer la précision de la détection en intégrant notre système dans un environnement médical réel.

Cela permettra de bénéficier aux patients et aux professionnels de la santé en fournissant des outils précis et fiables pour le diagnostic précoce et le traitement des troubles gastro-intestinaux. Il est également important de poursuivre la recherche et le développement de nouvelles méthodes et techniques afin d'élargir la portée de la détection automatique des maladies gastro-intestinales et d'améliorer encore les résultats obtenus.

Bibliographie

[1] Drossman, D.A., Chang, L., Chey, W.D., et al., "Functional Gastrointestinal Disorders: History, Pathophysiology, Clinical Features and Rome IV", Gastroenterology, vol. 150, no. 6, pp. 1262-1279.e2, mai 2016.

[2] Merck Sharp & Dohme Corp., "Présentation de l'appareil digestif", Manuel MSD (version française en ligne).

[3] Halvorsen, P.L., "Kvasir v2 - A Gastrointestinal Tract Dataset", Kaggle (version en ligne), dernière mise à jour le 23 avril 2021, disponible sur : https://www.kaggle.com/datasets/plhalvorsen/kvasir-v2-a-gastrointestinal-tract-dataset

[4] Agrawal, T., Gupta, R., Sahu, S., Wilson, C.E., "SCL-UMD at the Medico Task-MediaEval 2017: Transfer learning based Classification of Medical Images", Proceedings of the MediaEval 2017 Workshop, Dublin, Ireland, 13-15 septembre 2017.

[5] Yogapriya, J., Chandran, V., Sumithra, M.G., Anitha, P., Jenopaul, P., Dhas, C.S.G., "Gastrointestinal Tract Disease Classification from Wireless Endoscopy Images Using Pretrained Deep Learning Model", Journal of Healthcare Engineering, vol. 2021, article ID 5940433, 2021.

[6] Lonseko, Z.M., Adjei, P.E., Du, W., Luo, C., Hu, D., Zhu, L., Gan, T., Rao, N., "Gastrointestinal Disease Classification in Endoscopic Images Using Attention-Guided Convolutional Neural Networks", IEEE Journal of Biomedical and Health Informatics, November 2021, doi: 10.1109/JBHI.2021.3110809.

[7] Stensland, H.K., Reisaeter, A.V., Gladhaug, I.P., Johnsen, G., Eide, H.A., Bozorgi, M., Berstad, P., "Hyper-Kvasir: A Comprehensive Multi-Class Image and Video Dataset for Gastrointestinal Endoscopy", arXiv preprint arXiv:1904.00987, 2019.

[8] Haile, M.B., Salau, A.O., Enyew, B., Belay, A.J., Jin, Z., "Detection and classification of gastrointestinal disease using convolutional neural network and SVM", International Journal of Medical Informatics, Volume 159, June 2022, 104643, ISSN 1386-5056, https://doi.org/10.1016/j.ijmedinf.2022.104643.

[9] Raju, A.S.N., Jayavel, K., Rajalakshmi, T., "Dexterous Identification of Carcinoma through ColoRectalCADx with Dichotomous Fusion CNN and UNet Semantic Segmentation", Journal of Medical Systems, Volume 44, Issue 9, August 2020, 152, ISSN 0148-5598, https://doi.org/10.1007/s10916-020-01668-y.

[10] Kvåle, K., Gilja, O.H., Gregersen, H. et al. "KVASIR-Instrument dataset: A dataset of endoscopic video recordings for the development, validation and benchmarking of instruments for automatic detection of gastrointestinal lesions", Data in Brief, Volume 26, February 2019, 104340, ISSN 2352-3409, https://doi.org/10.1016/j.dib.2019.104340.

[11] Debesh Jha, Sharib Ali, Krister Emanuelsen, Steven A. Hicks, Vajira Thambawita, Enrique Garcia-Ceja, Michael A. Riegler, Thomas de Lange, Peter T. Schmidt, Håvard D. Johansen, Dag Johansen, Pål Halvorsen || Diagnostic and Therapeutic Tool Segmentation Dataset in Gastrointestinal Endoscopy, 2020.

[12] Taira Watanabe , Kensuke Tanioka , Satoru Hiwa , Tomoyuki Hiroyasu||Performance Comparison of Deep Learning Architectures for Artifact Removal in Gastrointestinal Endoscopic Imaging, Janvier 2022

[13] https://paperswithcode.com/dataset/cvc-clinicdb

[14] Olaf Ronneberger, Philipp Fischer, Thomas Brox|| Computer Science Department and BIOSS Centre for Biological Signalling Studies,Mai 2015

[15] Jha, D., Riegler, M. A., Johansen, D., Halvorsen, P., & Johansen, H. D. (2020). DoubleU-Net: A Deep Convolutional Neural Network for Medical Image Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (pp. 828-829).

[16] Qi Chang, Danish Ahmad, Jennifer Toth, Rebecca Bascom, and William E. Higginsa || ESFPNet: efficient deep learning architecture for real-time lesion segmentation in autofluorescence bronchoscopic video,Décembre 2022

[17] Céline Deluzarche. Définition | Deep Learning - Apprentissage profond | Futura

Tech, https://www.futura-sciences.com/tech/definitions/intelligence-artificielle-deep-learning-17262/

[18] 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/

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

[20] What are Convolutional Neural Networks ? en-us. url : https://www.ibm.com/topics/convolutional-neural-networks

[21] Houacine Noura, Khelifa Nadia, Classification des textures par les réseaux de neurones convolutif, thèse MASTER ACADEMIQUE, Domaine : Sciences et Technologies, Filière : Génie électrique, Spécialité : Commande des systèmes, FACULTE DU GENIE ELECTRIQUE ET D'INFORMATIQUE DEPARTEMENT D'AUTOMATIQUE, UNIVERSITE MOULOUD MAMMERI DE TIZI-OUZOU, 27/09/2018.

[22] Mokri Mohammed Zakaria, Classification des images avec les réseaux de neurones convolutionnels, Mémoire de fin d'études Pour l'obtention du diplôme de Master en informatique, OPTION : Modèle Intelligent et Décision, Faculté Des Sciences, Département D'informatique, Université Abou Bakr Belkaid Tlemcen, 03/07/2017.

[23] Simonyan, K., & Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In Proceedings of the International Conference on Learning Representations (ICLR), San Diego, CA, USA.

[24] MathWorks. "VGG19." MATLAB Deep Learning Toolbox Documentation, https://in.mathworks.com/help/deeplearning/ref/vgg19.html.

[25] GeeksforGeeks. (n.d.). Residual Networks (ResNet) – Deep Learning. https://www.geeksforgeeks.org/residual-networks-resnet-deep-learning/

[26] Chrzastowski-Wachtel, P., & Wachtel, A. (2019). A Comparative Study of Deep Learning Architectures for Image Classification. IEEE Access, 7, 22721-22732. https://doi.org/10.1109/ACCESS.2019.2898972

[27] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto et H. Adam, «Mobilenets: Efficient convolutional neural networks for mobile vision applications,» arXiv preprint arXiv:1704.04861, 2017.

[28] Thirumuruganathan, S. (2010). A detailed introduction to k-nearest neighbor (KNN) algorithm. Retrieved from https://saravananthirumuruganathan.wordpress.com/2010/05/17/a-detailed-introduction-to-k-nearest-neighbor-knn-algorithm/

[29] Descôteaux, s. les règles d'association maximale au service de l'interprétation des résultats de la classification. thèse de maitrise en informatique, université du québec à trois-rivières, trois-rivières, 2014: p. 174.

[30] java T point.Random Forest Algorithm.[en ligne].Disponible sur : https ://www.javatpoint.com/machine-learning-random-forest-algorithm

[31] Holland, J. H. (1973). Genetic Algorithms and the Optimal Allocation of Trials.SIAM Journal on Computing, 2(2) :88–105.

[32] KENNEDY J., EBERHART R., “Particle Swarm Optimization,” Proceedings of the IEEE International Joint Conference on Neural Networks, IEEE Press, vol. 8, no. 3, pp. 1943–1948. 1995.

[33] COOREN Y., Perfectionnement d'un algorithme adaptatif d'Optimisation par Essaim Particulaire. Applications en génie médical et en électronique. Thèse de Doctorat, Université de Paris 12 Val de Marne, France. 2008.

[34] GHERBOUDJ A., Méthodes de résolution de problèmes difficiles académiques. Thèse de Doctorat, Université de Constantine 2, Algérie, 2013.

[35] Abualigah, L., Yousri, D., Elaziz, M.A., Ewees, A.A., A. Al-qaness, M.A., Gandomi, A.H., "Aquila Optimizer: A novel meta-heuristic optimization Algorithm", Computers & Industrial Engineering, vol. 159, p. 107250, octobre 2021. https://doi.org/10.1016/j.cie.2021.107250

[36] Talatahari, S., Azizi, M., "Chaos Game Optimization: a novel metaheuristic algorithm", Artificial Intelligence Review, vol. 53, pp. 2851–2876, 2020. https://doi.org/10.1007/s10462-020-09867-w

[37] Welcome to Colaboratory! (2019, July 15). [Online]. Available: https://colab.research.google.com/notebooks/welcome.ipynb#scrollTo=-Rh3-Vt9Nev9.

[38] Crochet-Damais, A. (2019, February 1). "Visual Studio Code, l'IDE open source que les développeur s'arrachent." Le Journal du Net. Retrieved May 15, 2023, from https://www.journaldunet.com/web-tech/developpeur/1204957-visual-studio-code-l-ide-open-source-que-les-devs-s-arrachent/

[39] Teradata. (n.d.). What is Python? Retrieved from https://www.teradata.com/Glossary/What-is-Python

[40] Mozilla. (n.d.). Web technologies. Retrieved from https://developer.mozilla.org/en-US/docs/Web

[41] Gavin Hackeling. «Mastering Machine Learning with scikit-learn Second Edition ». Packt Publishing Ltd, Livery Place, UK, Juin 2017.