الجمهورية الجزائرية الديمقراطية الشعبية

République Algérienne démocratique et populaire

وزارة التعليم العالي و البحث العلمي

Ministère de l'enseignement supérieur et de la recherche scientifique

جامعة سعد دحلب البليدة

Université SAAD DAHLAB de BLIDA

كلية التكنولوجيا

Faculté de Technologie

قسم الالكترونيك

Mémoire de Master

Filière Électronique

Spécialité Électronique des Systèmes Embarqués

présenté par

TALBI HADJER

&

KHALDI NACHIDA

Étude comparative des méthodes de Deep Learning pour la classification et la segmentation de la rétinopathie diabétique

Encadré par : Mme. BOUGHERIRA HAMIDA

Année Universitaire 2021-2022

Remerciements

Nous tenons tout d'abord à rendre grâce à DIEU tout puissant qui nous

permet de mener à bien toute chose.

Nous voudrions adresser toute nos gratitudes à l'encadreur de

Ce mémoire Madame BOUGHERIRA HAMIDA , pour sa

patience, sa disponibilité et surtout ses judicieux conseils .

Nous remercions madame NACEUR DJAMILA ,notre chef de

spécialité en master électronique des systèmes embarquées pour ses encouragements.

Nous remercions les membres du jury d'avoir pris la peine de lire et de juger ce travail.

Sans oublier tous les professeurs et les étudiants du notre promo 2022.

ملخص:

الهدف من هذا المشروع هو تنفيذ نظام للمساعدة في التشخيص الطبي من خلال الكشف عن اعتلال الشبكية السكري ، وتقييم مخاطر مهاجمة الأمراض والوقاية منها.تتطلب الاختلافات وتعقيد البيانات من تصوير الشبكية التعلم من الأمثلة ، وهذا واحد من الاهتمامات الرئيسية لمجال التعلم العميق لوصف وتمثيل الآفات في البيانات.

Résumé :

L'objectif de ce projet est d'implémenter un système d'aide au diagnostic médical par la détection de la rétinopathie diabétique, l'évaluation des risques d'atteinte de maladies et à les prévenir.Les variations et la complexité des données d'imagerie rétiniennes,nécessitent un apprentissage à partir d'exemples, et cela constitue un des intérêts clés du domaine de l'apprentissage profond pour décrire et représenter les lésions dans les données.

Mots clés : rétinopathie diabétique, Deep Learning, Google Colab, jupyter, Googlenet, VGGnet, Alexnet,U-net, intelligence artificielle, imagerie médicale.

Abstract :

The objective of this project is to implement a system to assist in medical diagnosis by detecting diabetic retinopathy, evaluating the risks of attacking diseases and preventing them. The variations and complexity of data from Retinal imaging, requires learning from examples, and this is one of the key interests of the field of deep learning to describe and represent lesions in data.

Keywords: diabetic retinopathy, deep learning, Google Colab, jupyter, Googlenet, VGGnet, Alexnet, U-net, intelligence artificial, medical imaging.

Liste des acronymes

RD : rétinopathie diabétique

IA : Intelligence Artificielle

CNN : convolutional neural network

RNA : réseau de neurones artificiel

FC : fully connected

VGG : Visual geometry group

CONV : convolution

GPU : graphics processing unit (unité de traitement graphique)

LR : Learning Rate

Adam : adaptive moment estimation

VAL : validation

RAM : Random Access memory

ReLU : rectified linear unit

U-NET : convolutional networks for biomedical image segmentation

Table des matières

Liste des figures

Liste de tableaux

Introduction générale

L'imagerie médicale est devenue indispensable dans le diagnostic et la thérapie des maladies oculaires avec l'avènement des systèmes rétiniens numérisés, il est possible d'analyser de manière automatique les images du fond d'œil,en particulier les images rétiniennes en couleur,car leur acquisition est simple peu coûteuseet non invasive.

L'intelligence artificielle permet d'exploiter les données plus efficacement, ses algorithmespour l'imagerie médicale.

Le dépistage de la rétinopathie diabétique et des pathologies ophtalmologiques en rapport avec le diabète est un bon exemple de ce que peut apporter cette technologie à la médecine.

Les techniques d'apprentissage automatique constituent des solutions pour développer des outils permettant d'aider les médecins à diagnostiquer, à prédire le risque d'atteinte de maladies et à les prévenir.

Ce travail de recherche se focalise sur l'analyse et le traitement des images du fond d'oeil,dans le but de proposer un système d'aide au diagnostic de la rétinopathie diabétique.

Ce système,basé sur l'apprentissage profond,permettra de réduire le temps du diagnostic et d'aider l'ophtalmologue dans sa prise de décision.

Dans ce projet, nous créons, sur la plateforme Colab et à l'aide du langage de programmation Python et ses bibliothèques (notamment Tensorflow et Keras), et bien beaucoup d'autres outils, un système de Deep Learning basé sur les réseaux de neurones àl'aide des différenets architectures pour la classification des images de larétinopathie diabétique.

Ainsi, nous utiliserons le masque U-net pour la détection et l'illustration des régions d'hémorragie.

Ce mémoire est composé de 3 chapitres:

Le premier chapitre : Généralités sur la rétinopathie diabétique et le Deep Learning,

Le deuxième chapitre décrit la conception de plusieurs architectures prédéfinies, Alexnet, Vggnet, et Googlenet et d'une nouvelle architecture pour la classification, et de deux architectures U-net1, et U-net2, pour la segmentation.

Le troisième chapitre montre l'Implémentation des différentes architectures, et les résultats obtenus.

Chapitre I

Généralité Sur La Rétinopathie Diabétique et Le Deep Learning

I.1 Introduction

Le but de notre projet étant la detéction de la rétinopathie diabètique par Deep Learning dans des images d'angiographie de la rétine. Nous donnons dans ce chapitre un aperçu sur la rétinopathie diabètique et les méthodes d'intelligence artificielle et traitement d'image, nous présentons les différentes types d'apprentissage et les méthodes utilisés pour classer et détecter la rétinopathie diabétique, y compris CNN et ses différents modèles, et la segmentation avec mask Unet.

I.2 Le Diabète

Le diabète est une maladie métabolique caractérisée par une élévation anormale chronique de la glycémie, définie par le taux de sucre dans le sang. Cette augmentation de la glycémie est causée par un dysfonctionnement de la sécrétion ou de l'action de l'insuline, une hormone fabriquée par le pancréas. Elle peut provoquer à plus ou moins long terme des lésions de différents organes, comme les yeux, les reins, les nerfs et les vaisseaux sanguins. Près de 90% des diabétiques vivent pendant des années avec cette maladie sans le savoir car le diabète ne provoque en général pas de manifestations au début de son évolution. Il existe essentiellement deux types de diabète. **[1]**

I.2.1 Le diabète de type 1

Ce type de diabète apparaît généralement chez les personnes de moins de 20 ans. Il touche environ 10 % des personnes vivant avec le diabète. Le diabète de type 1 était autrefois appelé diabète insuline dépendant ou diabète juvénile.

I.2.2 Le diabète de type 2

Le diabète de type 2 est la forme la plus fréquente de diabète, avec 90 % des cas. Il se manifeste généralement à l'âge adulte, chez les individus de 40 ans et plus.

Dans le diabète de type 2, deux phénomènes sont généralement présents :

-une résistance du corps à l'action de l'insuline;

-une diminution de la production d'insuline. [3]

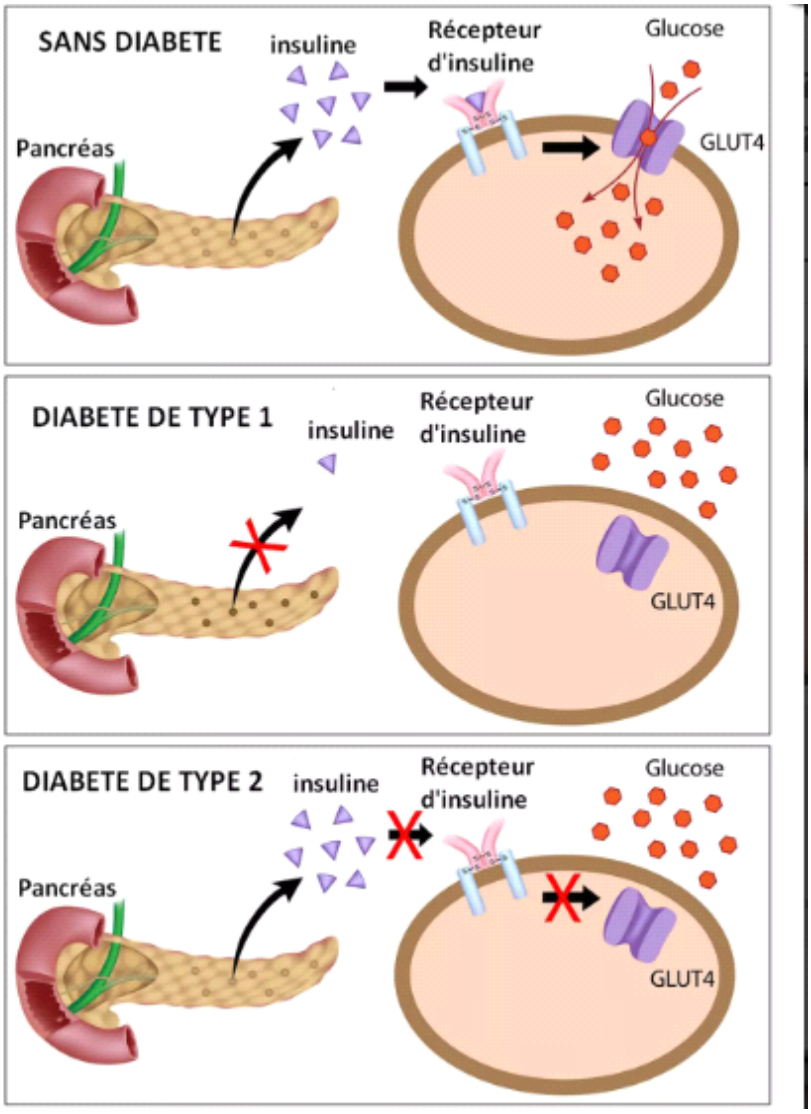


Figure I.1 : Les types du diabète

I.2.3 Les complications de diabète

Les complications du diabète sont nombreuses et peuvent être sévères.

-La rétinopathie diabétique responsable à terme d'une cécité).

-Les complications cardio-vasculaires.

-La néphropathie diabétique aboutissant à l'insuffisance rénale, la neuropathie diabétique.

-Les infections, les ulcères de pied et de jambe.

On peut aussi observer des complications aiguës comme le coma hypoglycémique ou acidocétosique.

I.3 La rétinopathie diabétique

Est une complication fréquente du diabète sucré qui compromet le fonctionnement de la rétine. C'est une pathologie qui apparaît lorsque les vaisseaux sanguins de la rétine se détériorent. Ces vaisseaux altérés peuvent se dilater, provoquer une fuite de liquide (plasma, lipides et / ou sang) et peuvent même s'obstruer, laissant une partie de la rétine sans circulation sanguine. Tous ces phénomènes qui surviennent en raison du diabète peuvent provoquer des lésions progressives des structures du globe oculaire, conduisant à une diminution sévère de la vision et même, sans traitement approprié, à la cécité. Même à des stades très avancés, la rétinopathie diabétique ne provoque pas toujours une gêne visuelle. Par conséquent, il est recommandé aux patients diabétiques de se soumettre à des contrôles ophtalmologiques réguliers.

Figure I.2: Patient atteint d'une Rétinopathie Diabétique

I.3.1 Les symptômes

La rétinopathie diabétique ne donne souvent aucun signe d'alerte dans les premiers stades. Dans la plupart des cas, elle ne provoque pas de symptômes tant que les lésions oculaires ne sont pas graves.

La diminution lente et progressive de la vision chez une personne diabétique se traduit généralement par la présence de liquide accumulé dans la partie centrale de la rétine (œdème maculaire). D'autres fois, la maladie débute par une hémorragie intraoculaire aiguë, le premier symptôme étant l'apparition soudaine et très alarmante de tâches qui obscurcissent partiellement ou totalement la vision. La rétinopathie diabétique peut être présente, même à des stades très avancés, et n'entraîne aucun type d'inconfort visuel.

Les symptômes qui apparaissent généralement sont:

•Vision trouble.

•Mouches volantes.

•Perte de vision lente au fil du temps.

•Perte soudaine de la vision.

Une détection précoce et un traitement à temps améliorent considérablement le pronostic visuel de la maladie et peuvent empêcher la progression vers la cécité. [4]

I.3.2 Anatomie Macroscopique de la rétine

L'œil humain permet de distinguer les formes et les couleurs. La science qui étudie l'œil s'appelle l'ophtalmologie. L'un des grands défis de la technologie sera de fabriquer des yeux électroniques, capables d'égaler voire de dépasser les aptitudes des yeux du monde vivant pour, par exemple, remplacer l'œil d'une personne accidentée. [28]

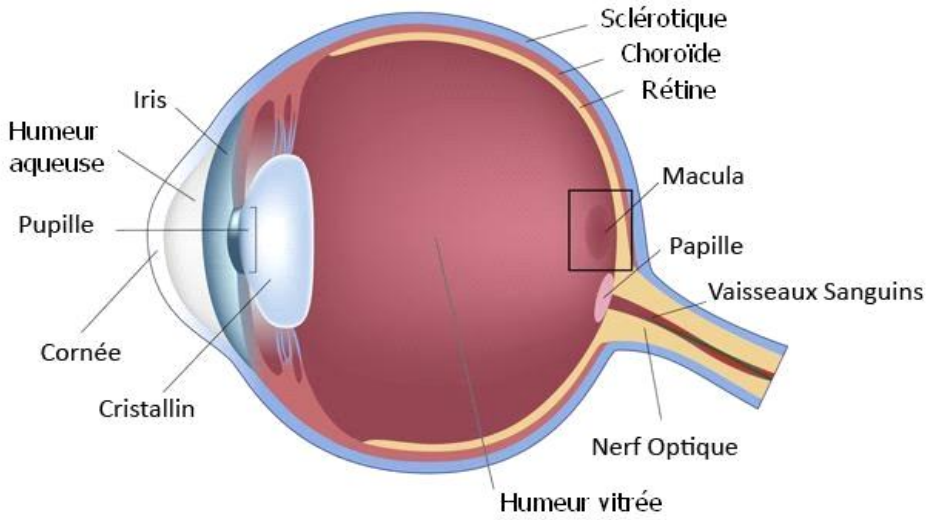


Figure I.3 : Coupe 3D montrant les différents constituants du globe oculaire

-L'illustration de figure I.3 montre la structure de l'œil . Sa forme sphérique est constituée de 3 enveloppes dites Tuniques qui baignent dans un liquide

I.3.3 Zones particulières de la Rétine

I.3.3.1 La macula

La macula est la zone centrale de la rétine ou du fond d'œil et constitue le point où l'acuité visuelle est à son maximum. Une macula intacte permet une vision centrale nette, de distinguer les détails, de lire et de reconnaître, par exemple, les visages des gens. [5]

I.3.3.2 La fovéa

La fovéa, la zone centrale de la macula, est la zone de la rétine où la vision des détails est la plus précise. Elle est située dans l'axe visuel de l'œil.

I.3.3.3 La pupille

Région d'émergence du nerf optique, et dépourvue de photorécepteurs, cette partie

représente le diaphragme de l'œil, avec la capacité de se contracter ou se dilater selon

l'intensité lumineuse.

I.3.4 Examen de Dépistage

On appelle test de dépistage tout examen destiné à déceler la présence d'une pathologie ou d'une substance dans l'organisme.En médecine, le dépistage consiste à rechercher une ou plusieurs maladies ou anomalies dites « à haut risque » chez des individus d'une population déterminée. Ces investigations peuvent ou non être suivies d'une consultation médicale, d'un examen clinique et d'un traitement. De plus, le dépistage est une procédure initiale (test ou examen) visant à détecter les signes ou symptômes qui caractérisent une ou plusieurs maladies, syndromes ou affections. On procède à des examens ophtalmologiques comme:

-Test du réflexe photomoteur.

-Examen des milieux oculaires.

-Examen du tonus oculaire.

-Examens complémentaires.

-Examens spécifiques.

-Examen du fond d'œil.

-Le principal contrôle effectué par les ophtalmologistes reste le classique "examen de fond

d'oeil" obtenu par dilatation de la pupille afin d'observer la rétine (Figure I.4).

I.3.5 Fond d'œil et angiographie

-Le fond d'œil est un examen ophtalmologique , qui examine les structures de l'œil généralement indiqué en cas de suspicion d'une affection de la rétine , le fond d'œil peut être effectué à tout âge de la petite enfance à l'âge avancé .

-Le fond d'oeil est réalisé par un ophtalmologiste consiste à examiner la partie centrale du fond d'œil. L'examen biomicroscopique du fond d'œil après dilatation pupillaire peut révéler une rétinopathie diabétique. Le fond d'œil doit être complété par des photographies du fond d'œil ou angiographie.

Figure I.4 : Examen Fond d’oeil

-Il existe une nouvelle méthode moins contraignante qui est le rétinographe non mydriatique. Il permet de récupérer une photographie numérique de l’oeil sans la dilatation de la pupille, et sans douleur, moins contraignante, et plus efficace que les tests de dépistages classiques, mais reste tout de même très peu utilisée. [31]

Dans notre projet ce sont cesphotographies ou angiographie de la rétine, que nous proposons de traiter par des méthodes de l’intilligence artificielle pour détecter la RD.

I.4 Imagerie médicale

L'imagerie médicale qui inclut les radiographies, les échographies et les IRM, nécessite généralement la flexibilité d'un œil humain pour détecter les anomalies. Cependant, l'analyse d'images basée sur l'apprentissage en profondeur peut aujourd'hui automatiser la recherche d'anomalies biologiques de manière fiable, reproductible et robuste.

Cette percée a révolutionné le rôle des radiologues modernes dans la mise à disposition de diagnostics assistés par ordinateur d'images médicales. Le processus est grandement simplifié grâce à deux outils dédiés intégrés. L'outil de localisation identifie les zones d'intérêt, telles que des organes spécifiques, même si l'arrière-plan est visuellement déroutant ou présente un faible contraste. Les outils de détection de défauts utilisent une série d'images d'entraînement pour développer un modèle de référence qui montre à quoi ressemble normalement cet organe et un type particulier d'anomalie. Cela permet aux anomalies qui se produisent d'être signalées comme des défauts qui s'écartent de la physiologie normale de la zone d'organe cible. [11]

I.5 La Relation entre deep learning et la rétinopathie diabétique

Les progrès de l'intelligence artificielle sont destinés à révolutionner le monde de la santé. Ici, nous présenterons un exemple d'application de l'apprentissage en profondeur aux images médicales.

"Deep Learning, l'une des approches les plus courantes de l'apprentissage automatique et de

l'intelligence artificielle pour améliorer la détection des lésions de la rétinopathie diabétique dans les images du fond d'œil. Il existe plusieurs techniques d'apprentissage en profondeur qui détectent automatiquement les maladies rétiniennes dans l'image du fond d'œil [12]

I.6 L'intelligence artificielle

C'est l'ensemble des théories et des techniques mises en ouvre en vue de réaliser des machines capables de simuler l'intelligence humaine.

I.7 Machine Learning (Apprentissage automatique)

I.7.1 Définition et principe

L'apprentissage automatique (en anglais : machine learning, litt. « apprentissage machine »), apprentissage artificiel ou apprentissage statistique est un champ d'étude de l'intelligence artificielle qui se fonde sur des approches mathématiques et statistiques pour donner aux ordinateurs la capacité d' apprendre à partir de données, c'est-à-dire d'améliorer leurs performances à résoudre des tâches sans être explicitement programmés pour chacune. Plus largement, il concerne la conception, l'analyse, l'optimisation, le développement et l'implémentation de telles méthodes.

L'apprentissage automatique comporte généralement deux phases. La première consiste à estimer un modèle à partir de données, appelées observations, qui sont disponibles et en nombre fini, lors de la phase de conception du système. La reconnaissance du modèle consiste à résoudre une tâche pratique, telle que traduire un discours, reconnaître une densité de probabilité, reconnaître la présence d'un chat dans une photographie ou participer à la conduite d'un véhicule autonome. Cette phase dite « d'apprentissage » ou « d'entraînement » est généralement réalisée préalablement à l'utilisation pratique du modèle. La seconde phase correspond à la mise en production : le modèle étant déterminé, de nouvelles données peuvent alors être soumises afin d'obtenir le résultat correspondant à la tâche souhaitée. En pratique, certains systèmes peuvent poursuivre leur apprentissage une fois en production, pour peu qu'ils aient un moyen d'obtenir un retour sur la qualité des résultats produits.

I.8 Apprentissage en profondeur (Deep Learning)

I.8.1 Définition

L'apprentissage en profondeur (Deep Learning : DL) est une branche de l'apprentissage automatique qui enseigne aux ordinateurs à faire ce qui vient naturellement aux humains : apprendre de l'expérience. Les algorithmes d'apprentissage automatique utilisent des méthodes de calcul pour "apprendre" des informations directement à partir de données sans s'appuyer sur une équation prédéterminée comme modèle. DL est une classe de techniques d'apprentissage automatique qui exploite de nombreuses couches de traitement d'informations non linéaires pour l'extraction et la transformation de fonctions supervisées ou non supervisées, et pour l'analyse et la classification de modèles.

Les caractéristiques et les concepts de niveau supérieur sont donc définis en termes de niveaux inférieurs, et une telle hiérarchie de caractéristiques est appelée architecture profonde. La plupart de ces modèles sont basés sur un apprentissage non supervisé des représentations. [7]

I.8.2 Concepts de Deep Learning

Les concepts de Deep Learning constituent une classe d'algorithmes d'apprentissage

automatique dont les caractéristiques sont les suivantes :

• Elles utilisent différentes couches d'unité de traitement non linéaire pour l'extraction et la transformation des caractéristiques. Chaque couche prend en entrée la sortie de la précédente. Les algorithmes peuvent être supervisés ou non supervisés et leurs applications comprennent la reconnaissance de modèles ou la classification statistique.

• Elles fonctionnent avec un apprentissage à plusieurs niveaux de détails ou de représentations des données. A travers les différentes couches on passe de paramètres de bas niveau à des paramètres de plus haut niveau.

• Ces différents niveaux correspondent à différents niveaux d'abstraction des données.

• Ce nouveau champ d'étude a pour objectif d'avancer davantage vers les capacités d'intelligence artificielle. Ses architectures permettent aujourd'hui de donner du sens à des données sous forme d'image, de son ou de texte.

Le système de Deep Learning est un système basé sur les réseaux de neurones artificiels, qui est constitué d'un ensemble de couches cachées. Le mot profond (apprentissage profond) vient de ce nombre des couches cachées .Le point différencie entre un perceptron classique, et un système de Deep Learning est que pour le premier les entrées des réseaux sont les caractéristiques de l'image, mais pour le deuxième ce sont les pixels bruts de l'image d'entrée. En effet, dans un système de Deep Learning chaque couche est considérée comme un niveau d'abstraction de l'image [29]

I.8.3 Les avantages de Deep Learning

Permis les avantages qui distinguent le Deep Learning :

• La robustesse pour comprendre et utiliser de nouvelles données.

• Il gère tout à un niveau d'abstraction beaucoup plus élevé que les réseaux de neurones standard.

• Il obtient ses résultats plus rapidement. Il apprend au fil du temps plutôt qu'en un éclair.

• Il peut gérer de grandes quantités de données pour de petits réseaux avec un coût

d'apprentissage bien moindre. [8]

I.8.4 Quelles différences entre intelligence artificielle, machine learning et deep learning

Pour mieux comprendre la différence d'approche entre ces trois technologies, voyons comment elles doivent procéder pour apprendre à un ordinateur à reconnaître la présence de chat sur des images :

L'IA exige qu'un programmeur écrive tout le code nécessaire à l'ordinateur pour reconnaître un chat présent sur une image. Le programmeur crée le modèle d'apprentissage.

Le machine learning exige que des programmeurs apprennent au système à quoi ressemble un chat en lui montrant différentes images et en corrigeant son analyse jusqu'à ce que celle-ci soit correcte (ou plus précise). On parle d'apprentissage supervisé puisque l'intervention humaine est nécessaire.

Le deep learning divise la tâche de reconnaissance des caractéristiques du chat en plusieurs couches : une couche de l'algorithme apprend à reconnaître les yeux, une autre les oreilles, une troisième la silhouette générale, etc. Une fois connectées, ces différentes couches possèdent une certaine capacité de reconnaissance des chats afin de reconnaître l'animal sur chaque nouvelle image soumise. [9]

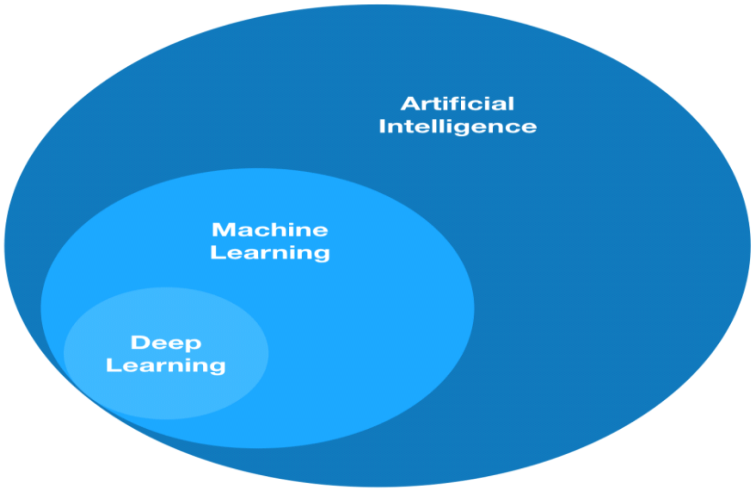


Figure I.5 : La relation Entre L'IA et ML et le Deep Learning

I.9 Les réseaux de neurones

I.9.1 Définition

Un réseau de neurones est un modèle informatique dont la structure en couches est similaire à la structure en réseau des neurones du cerveau, avec des couches de nœuds connectés. Un réseau de neurones peut apprendre à partir de données ; il peut ainsi être entraîné à reconnaître des tendances, classer des données et prévoir des événements à venir.

Un réseau de neurones décompose vos données en couches d'abstraction. Il peut être entraîné sur de nombreux exemples en vue de reconnaître des modèles au niveau de la parole ou des images, par exemple, à l'instar du cerveau humain. Son comportement est défini par la façon dont ses éléments individuels sont reliés et par la solidité (ou poids) de ces liaisons. Ces poids sont automatiquement ajustés au cours de l'entraînement selon une règle d'apprentissage spécifiée jusqu'à ce que le réseau de neurones exécute correctement la tâche souhaitée.

I.9.2 Fonctionnement

Un réseau de neurones combine plusieurs couches de traitement, utilisant des éléments simples fonctionnant en parallèle et inspirés du système nerveux biologique. Il se compose d'une couche d'entrée, d'une ou de plusieurs couches masquées et d'une couche de sortie. Les couches sont interconnectées par des nœuds, ou neurones, chaque couche utilisant la sortie de la couche précédente en guise d'entrée.

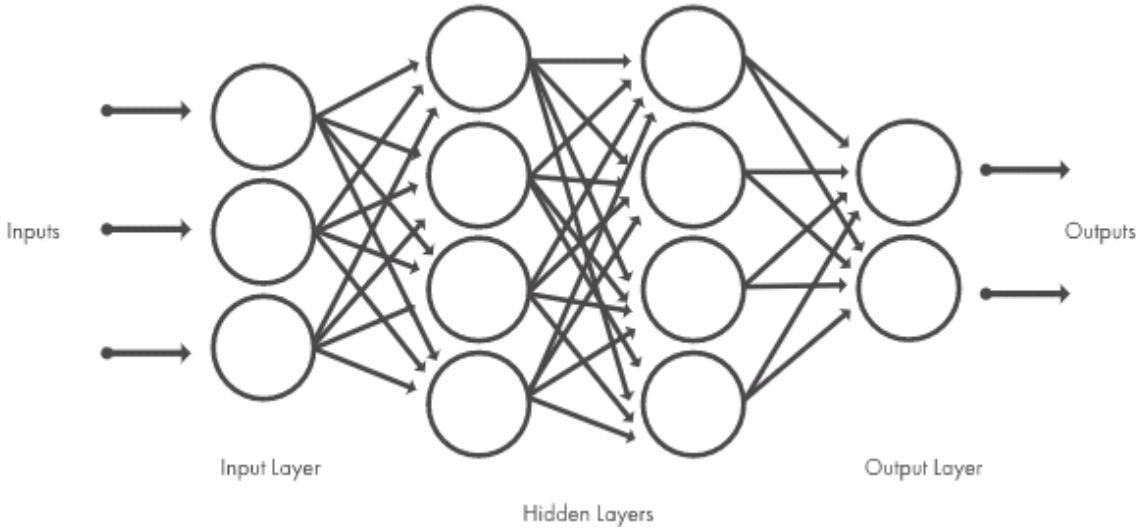


Figure I.6 : Architecture type d'un réseau de neurones

I.9.3 Techniques utilisées par les réseaux de neurones (Méthode d'apprentissage)

Les techniques d'apprentissage automatique couramment utilisées dans la conception d'applications de réseaux neuronaux comprennent Méthodes d'apprentissage : classification, régression, reconnaissance de formes et clustering.

I.9.3.1 Apprentissage supervisé

-La majorité des experts machine learning utilisent un apprentissage supervisé.

L'apprentissage supervisé consiste en des variables d'entrée **(x)** et une variable de sortie **(Y)** Vous utilisez un algorithme pour apprendre la fonction de mappage de l'entrée à la sortie

Y = f (X).

Le but est d'appréhender si bien la fonction de mappage que lorsque vous avez de nouvelles données d'entrée **(x)** , vous pouvez prévoir les variables de sortie **(Y)** pour ces données.

C'est ce qu'on appelle l'apprentissage supervisé, car le processus d'un algorithme tiré de l'ensemble de données, et peut être considéré comme un enseignant supervisant le processus d'apprentissage. Nous connaissons les réponses correctes, l'algorithme effectue des prédictions itératives sur les données d'apprentissage et est corrigé par l'enseignant. L'apprentissage s'arrête lorsque l'algorithme atteint un niveau de performance acceptable.

Les problèmes d'apprentissage supervisé peuvent être regroupés en problèmes de régression et de classification.

Classification: Un problème de classification survient lorsque la variable de sortie est une catégorie, telle que **«rouge»** ou **«bleu».**

***Régression*:** Un problème de régression se pose lorsque la variable de sortie est une valeur réelle, telle que **«dollars»** ou **«poids»** .

Certains types de problèmes fondés sur la classification et la régression incluant la prévision et la prévision de séries chronologiques.

L'apprentissage non supervisé consiste à ne disposer que des données d'entrée (X) et pas de variables de sortie correspondantes.

I.9.3.2 Apprentissage non supervisé

L'objectif de l'apprentissage non supervisé est de modéliser la structure ou la distribution sous-jacente dans les données afin d'en apprendre davantage sur les données.

Celles-ci sont appelées apprentissage non supervisé car, contrairement à l'apprentissage supervisé ci-dessus, il n'y a pas de réponse correcte ni d'enseignant. Les algorithmes sont laissés à leurs propres mécanismes pour découvrir et présenter la structure intéressante des données.

Les problèmes d'apprentissage non supervisés peuvent être regroupés en problèmes de clustering et d'association.

Association: Un problème d'apprentissage de règles d'association est l'endroit où vous souhaitez découvrir des règles décrivant une grande partie de vos données, telles que les acheteurs de **X** ont également tendance à acheter **Y.** [10]

I.10 Les réseaux de neurones convolutifs

I.10.1 Les Réseaux de neurones convolutifs

Le réseau neuronal convolutif est un type de réseau neuronal qui est le plus souvent appliqué aux problèmes de traitement d'images . Le fait qu'il soi utile dans les domaines à croissance rapide et l'une des principales raisons pour lesquelles ils sont si importants dans l'apprentissage en profondeur et l'intelligence artificielle aujourd'hui.

L'architecture d'un réseau neuronall convolutif se compose d'une succession de bloc de traitement pour extraire les caractéristiques discriminant la classe d'appartenance de l'image des autres.

Un bloc de traitement consiste en un ou plus

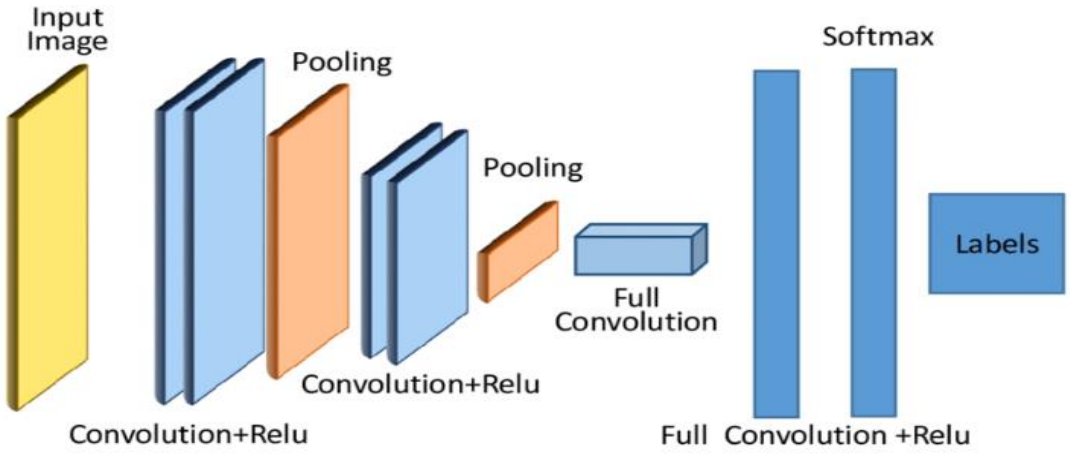


Figure I.7 : Architecture de CNN[13]

I.10.2 Les différentes couches de convolution

●Couche convolutive

Couche de convolution (CONV) : Le rôle de cette première couche est d'analyser l'image fournie en entrée et de détecter la présence de certaines caractéristiques. Obtenir un ensemble de cartes d'entités dans la sortie de cette couche.

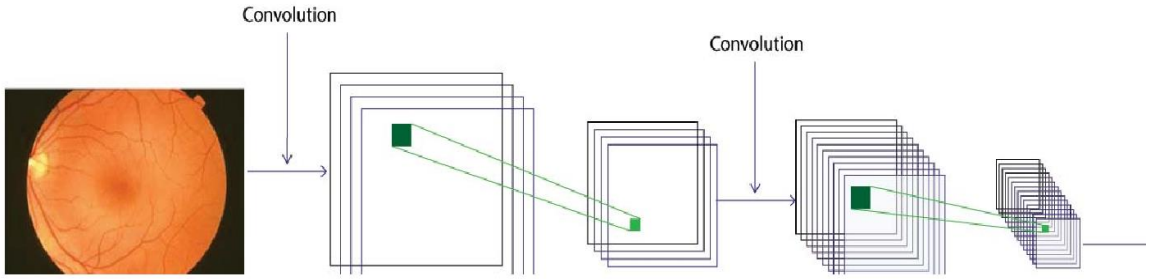


Figure I.8: Couche de convolution [14]

●Couche de Pooling

Est une opération qui est généralement appliquée entre deux couches de convolution. Il prend comme entrée une carte de caractéristiques formée par la sortie de la couche de convolution et sa tâche est de réduire la taille de l'image tout en préservant les propriétés les plus importantes. Certains des plus couramment utilisés sont le MaxPooling, ou Average Pooling , dont le fonctionnement consiste à maintenir la valeur moyenne dans la fenêtre de filtre à chaque étape. [15]

●La Fonction d'activation

La fonction d'activation sert avant tout **à** modifier de manière non-linéaire les données. Cette non-linéarité permet de modifier spatialement leur représentation**.**

La fonction ReLU engendre la suppression des valeurs négatives d'une carte d'activation en les remplaçant par zéro. L'avantage de la fonction ReLU est qu'elle offre la meilleure rapidité d'entraînement sans nuire à la précision de la généralisation du modèle. [16]

ReLU (*Rectified Linear Units*) désigne la fonction réelle non-linéaire définie par ReLU(x)=max(0,x)ReLU(x)=max(0,x).

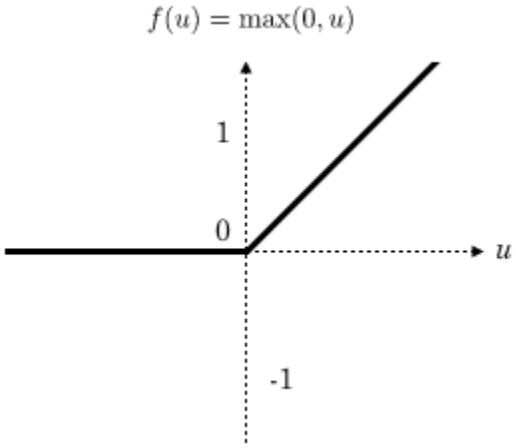


Figure I.9 : Allure de la fonction ReLU [30]

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.

●Maxpooling

Maxpooling est un processus de discrétisation basé sur des échantillons. Son but est de sous-échantillonner la représentation d'entrée (image, matrice de sortie de la couche cachée, etc.) en réduisant les dimensions. De plus, il est intéressés à réduire l'effort de calcul en réduisant le nombre de paramètres à apprendre et en fournissant une invariance avec de petits décalages (si de petits décalages ne changent pas la valeur maximale de la zone balayée, alors le maximum de chaque zone restent le même), de sorte que la matrice nouvellement créée reste la même. [17]

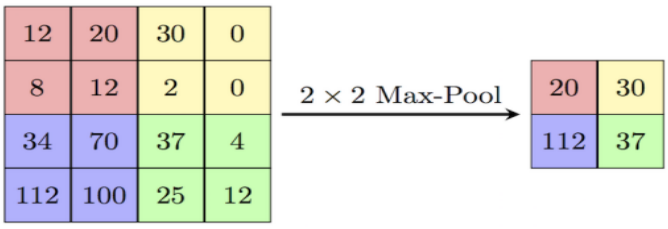


Figure I.10 : La couche de pooling

●La couche entièrement connectée

La couche entièrement connectée (Fully Connected) la fin d'un réseau neuronal convolutif CNN. Elle rejoint les séquences répétées des couches de convolution et de Pooling. Toutes les caractéristiques et tous les éléments des couches en amont sont liés à chaque caractéristique de sortie. [17]

I.10.3 Architectures CNN populaires

I.10.3.1 Le modèle VGGnet

VGG signifie (Visual Geometry Group) ; il s'agit d'une architecture standard de réseau de neurones à convolution profonde (CNN) à plusieurs couches. Le "profond" fait référence au nombre de couches avec VGG-16 ou VGG-19 composé de 16 et 19 couches convolutionnelles.

L'architecture VGG est à la base de modèles révolutionnaires de reconnaissance d'objets. Développé en tant que réseau neuronal profond, VGGNet va également au-delà des limites de base de nombreuses tâches et ensembles de données au-delà d'ImageNet. De plus, il reste l'une des architectures de reconnaissance d'images les plus populaires.[18]

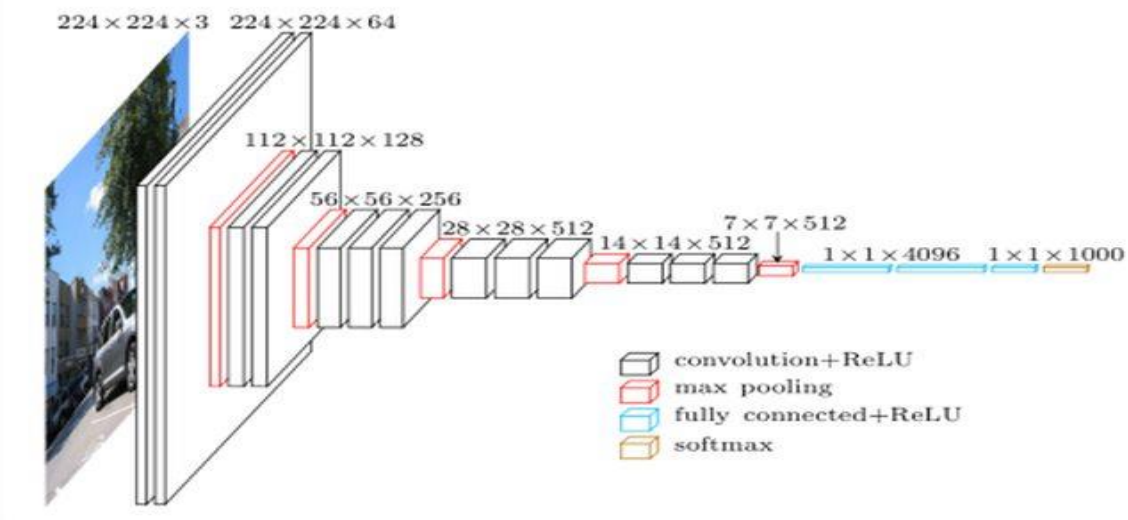


Figure I.11 : Architecture VGGnet [19]

I.10.3.2 Le Modèle AlexNet

AlexNet est une architecture de réseau neuronal convolutif classique. Il se compose de convolutions, d'une mise en commun maximale et de couches denses comme éléments de base. Des convolutions groupées sont utilisées pour ajuster le modèle sur deux GPU[20].

Alexnet elle est plus profonde avec plus de filtres par couche. Il se compose de huit couches : cinq couches convolutives (certaines d'entre elles sont suivies de couches de regroupement maximal), deux couches cachées entièrement connectées et une couche de sortie entièrement connectée. La formation de ce réseau est également réalisable sur plusieurs GPU. [21]

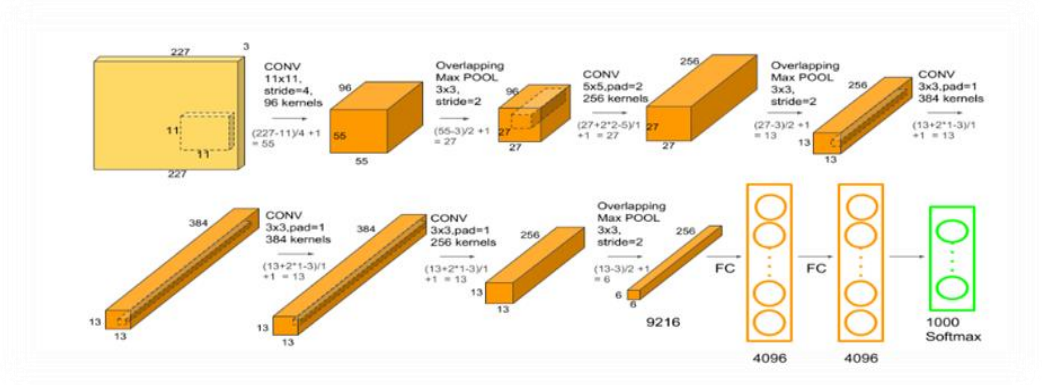


Figure I.12 : Architecture du Alexnet [22]

I.10.3.3 Le modèle GoogleNet

GoogLeNet est un réseau de neurones à convolution profonde à 22 couches qui est une variante du réseau Inception, un réseau de neurones à convolution profonde développé par des chercheurs de Google. L'architecture GoogLeNet présentée dans le ImageNet Large-Scale Visual Recognition Challenge 2014 (ILS RC14) a résolu des tâches de vision par ordinateur telles que la classification d'images et la détection d'objets. [23]

L'architecture GoogLeNet est très différente des architectures de pointe précédentes telles qu'AlexNet et ZF-Net. Il utilise de nombreux types de méthodes telles que la convolution 1×1 et la mise en commun des moyennes globales qui lui permettent de créer une architecture plus profonde[24].

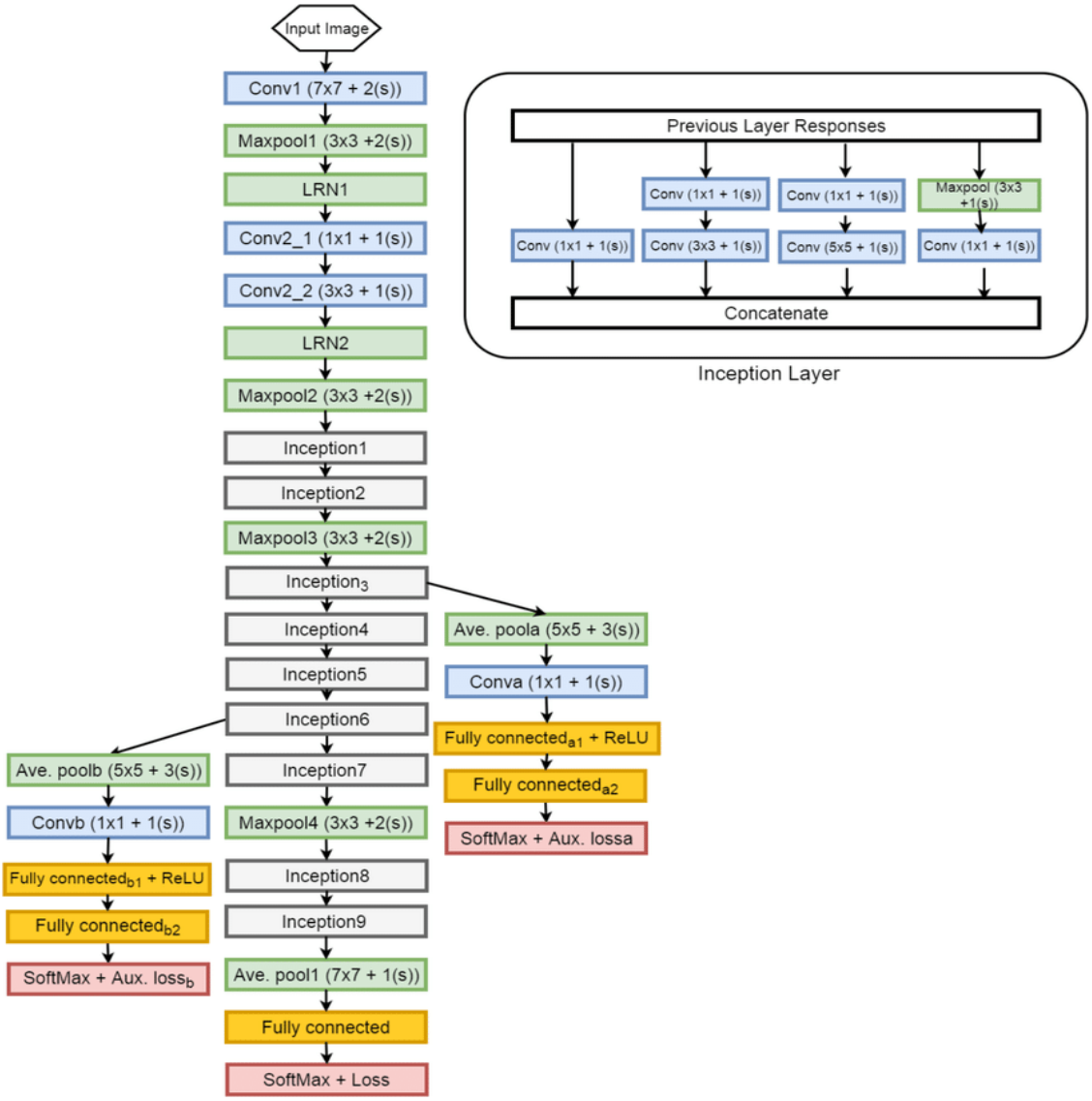


Figure I.13 : Architecture du GoogleNet [25]

I.11 Segmentation d'image

La segmentation d'image est un processus de vision par ordinateur dans lequel une image est segmentée en différents segments qui représentent différentes classes d'images.

I.11.1 Segmentation sémantique

Il s'agit de reconnaître des objets dans une image et de les regrouper selon des catégories définies. Par exemple, dans une scène de rue, vous dessinez des limites et étiquetez des éléments tels que des personnes, des voitures, des vélos, des feux de circulation, des trottoirs, des passages à niveau et des voies. [26]

I.11.2 Segmentation d'instance

Cela va encore plus loin dans la segmentation sémantique et inclut la reconnaissance d'objets dans des catégories définies. Par exemple, dans la même scène de rue, dessiner une bordure distincte pour chaque catégorie (personnes (adultes, enfants), voitures (voitures, bus, motos, etc.)) les étiqueter clairement. [26]

I.12 U-NET

UNet est une architecture de segmentation sémantique qui a eu un impact significatif sur le secteur biomédical car elle contribue à la segmentation profonde des images. Unet à été Développé en 2015 par Olaf Ronneberger, Philip Fischer et Thomas Brox de l'Université de Fribourg. [27]

UNet a considérablement amélioré le fonctionnement des réseaux de neurones convolutifs.

UNet était le meilleur à l'époque, toujours à la pointe de la technologie et le plus couramment utilisé pour la sémantique des tâches de segmentation. Un petit nombre d'exemples de formation sont nécessaires, contrairement aux réseaux convolutionnels traditionnels avec des milliers d'exemples de formation annotés. [27]

Sur le plan architectural, comme son nom l'indique, le modèle UNet ressemble à un "U". Il se compose de deux voies, une voie de sous-échantillonnage et une voie de sur-échantillonnage.[27]

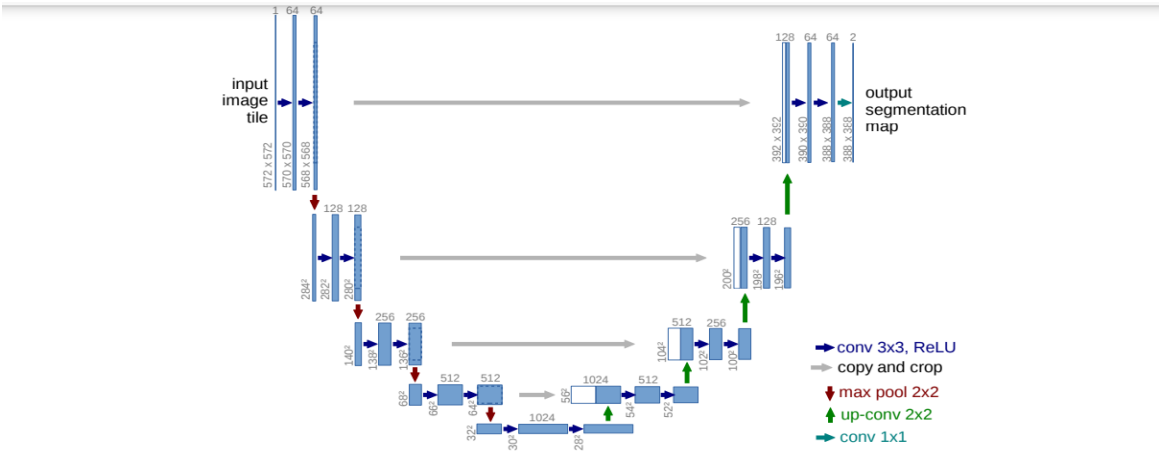


Figure I.14 : Représentation de Unet [27]

I.12.1 Les deux chemins de Unet

I.12.1.1 1er chemin

Le chemin de sous-échantillonnage est principalement destiné à l'extraction de caractéristiques. Le chemin de sous-échantillonnage a quatre blocs de convolution, et chaque bloc a deux couches de convolution. H Rembourrage 3x3 suivi de ReLU (Unité Linéaire Rectifiée), opérateur de mise en commun 2 foulées 2x2Max. Le nombre de filtres double pour chaque bloc de convolution

consécutif. Au départ, il y a 64 filtres. Chaque filtre contribue à augmenter la profondeur. Le cinquième bloc de convolution n'a pas d'opérateur de regroupement maximum, il participe simplement au chemin de sur-échantillonnage. [27]

I.12.1.2 2ème chemin

Le chemin de sur-échantillonnage est utilisé pour trouver l'objet. Le chemin de sur-échantillonnage comporte quatre blocs de convolution avec deux couches de convolution 2x2. Symétrique au chemin de sous-échantillonnage. Chaque fois que vous le pliez, la résolution augmente et la profondeur devient moins profonde. Le nombre de chaînes de fonctionnalités continuera à être divisé par deux. La couche de convolution est suivie d'un filtre 3x3 activé par ReLU. La couche de convolution finale est 1x1 et l'attribution d'entités aux classes appropriées aidera à une localisation précise. [27]

I.13 Conclusion

Dans ce chapitre, nous avons décrit différentes modèles de CNN qu'il est possible d'utiliser pour la détection de rétinopathie diabétique à partir d'images de fond d'œil .

Dans le chapitre suivant nous montrons l'implémentation de plusieurs architectures existantes, d'une architecture que nous avons concue nous-mêmes pour la classification et de l'architecture U-net pour la détection.

Chapitre II

Conception d’un système de classification pour la détection de la rétinopathie diabétique

II.1 Introduction

Dans ce chapitre nous présentons différents modèles CNN de réseau de neurones profonds, à savoir Alexnet, VGGnet, Googlenet, et le réseau CNN que nous avons défini et créé.Nous décrivons les paramètres nécessaires pour leur entrainement (ou apprentissage), pour l'obtention d'un système intelligent capable de fournir des résultats de détection de RD précis,exploitables par les médecins.

II.2 Principe de détection et de classification de la RD

Le diagnostic de la RD comporte deux aspect : la classification et la détection.

1/-Nous aurons classé les images en deux classes, malade du classe (oui) et non malade du classe (non).

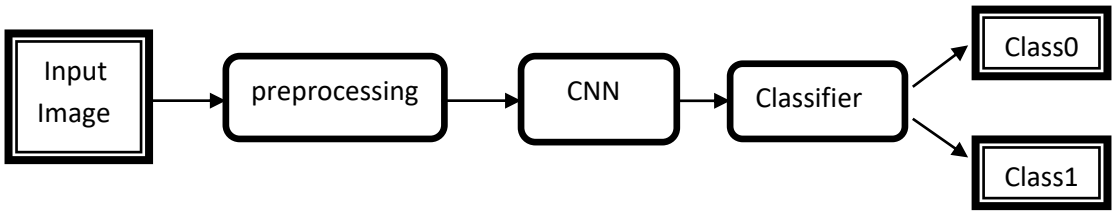


Figure II.1 : Structure de programmation du la classification des images

2/-Nous aurons détecté la zone d'hémorragie par masque U-net.

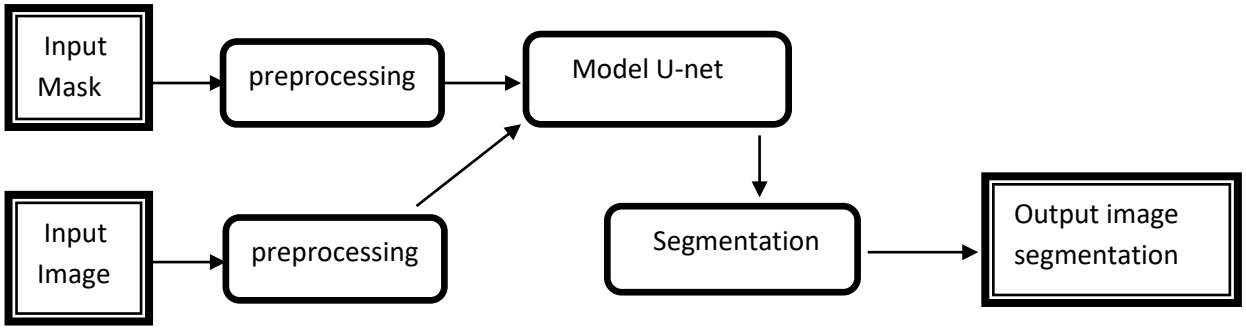


Figure II.2 : Structure de programmation du lasegmentation des images

II.3 Bases de donnés pour la classification des images

Notre travail consiste a créer un système de classification de la rétinopathie diabétique. Pour cela nous allons utilisé un ensemble de données publiés dans le site web kaggle et contenant un ensemble de 3648 images constitué des images saines (pas de RD) et d'images malade (contient la RD). On a 1800 images à étiquette "non" qui signifie : (pas de RD), et 1849 images à étiquette "oui" (elle a la RD).

II.4 Base de données pour la segmentation et détéction de pathologie

La base de données se compose de 89 images de fond d'œil en couleur .Les images ont été capturées à l'aide de la même caméra numérique à fond de vision à champ de vision de 50 degrés avec différents paramètres d'imagerie. Les données correspondent à une bonne situation pratique, où les images sont comparables, et peuvent être utilisées pour évaluer les performances générales des méthodes de diagnostic. Cet ensemble de données est appelé « images du fond d'œil de niveau d'étalonnage 1 ». [32]

II.5 Classification des images

La classification des images est un processus utilisé pour extraire une classe d'informations d'une image. C'est-à-dire le processus de classification des images en fonction de leur contenu visuel. La classification d'images Tensorflow est appelée processus de vision par ordinateur.

Le système commence d'abord par obtenir l'ensemble des données, effectue un prétraitement sur l'ensemble des données, divise l'ensemble des données, nous alimentons les modèles de CNN avec l'ensemble des données fractionné (voir section II.4.1.2), à la fin nous aurons un modèle qui peut classer de nouvelles images de rétinopathie diabétique.

II.5.1 Prétraitement

Le prétraitement doit être effectué avant de fournir les données au modèle CNN pour la formation. Nous apportons une modifications aux données : Redimensionner.

II.5.1.1 Redimensionnement des images

La taille des images du jeu de données d'origine était (1500x1152). Nous avons redimensionné les images à (200x200), où les deux premières valeurs font référence à la largeur et à la hauteur de l'image.

II.5.1.2 Division du jeu de donnés

Dans l'apprentissage en profondeur, le fractionnement d'un ensemble de données consiste à diviser l'ensemble de données en sous-ensemble de formation et sous-ensemble de test. Notre jeu de données est divisé en 2 sous-ensembles avec 80 % d'images par classe pour formation, 20 % pour les tests.

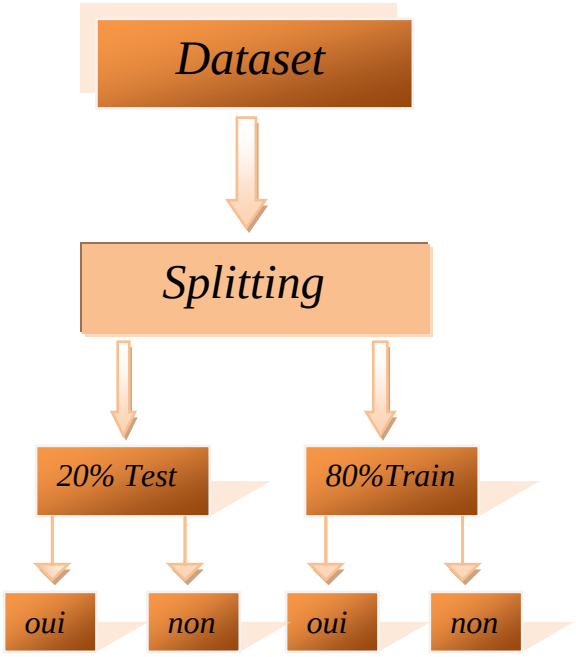


Figure II.3 : Structure de division de données par la fonction Split

II.6 Les architectures adoptées

Nous avons construit des architectures de CNN existantes (Alexnet,Googlnet et VGGnet) et nous avons créé notre propre CNN. Nous décrivons les différentes architectures dans ce qui suit.

II.6.1 Le modèle Alexnet

Alexnet est une architecture profonde. Les auteurs ont introduit un rembourrage pour que la taille des cartes de caractéristiques ne soit considérablement réduite.

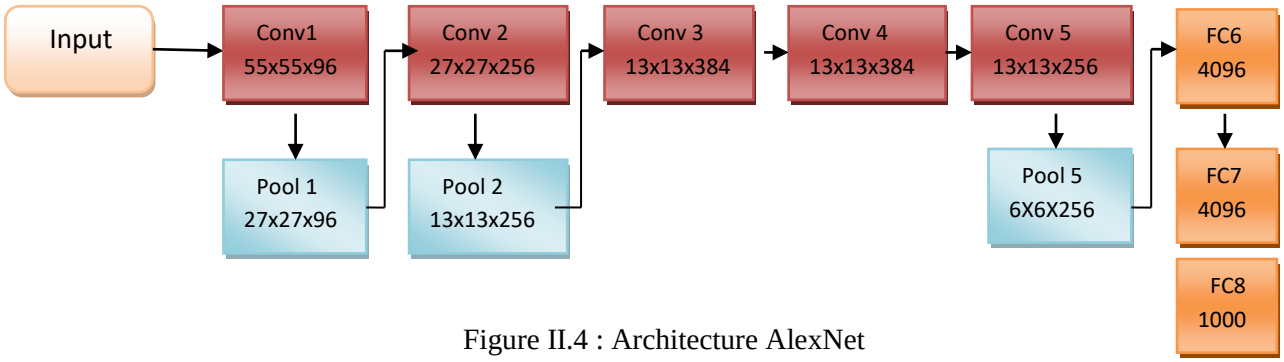


Figure II.4 : Architecture AlexNet

Couche de convolution et regroupement maximal : l'entrée de ce modèle sont les images de taille 200x200x3.

Ensuite nous appliquons la première couche de convolution avec 96 filtres de taille 11 x 11 avec foulée 4. La fonction d'activation utilisé dans cette couche et relu. La carte des caractéristiques de sortie est 55×55×96. En même temps le nombre de filtre devient le canal dans la carte des caractéristiques de sortie .

Ensuite nous avons la première couche de Maxpooling de taille 3x3 et foulée 2. Après nous obtenons la carte des caractéristiques résultante avec la taille27×27×96.

Après cela on applique la seconde opération de convolution, cette fois la taille du filtre est réduite à 5 x 5 et nous avons 256 filtres de ce type, la foulée est 1 et le rembourrage 2. La fonction d'activation utilisé est à nouveau relu.

Maintenant la taille de sortie que nous obtenons est 27 x 27 x 256, de nouveau nous appliquons une couche de regroupement maximum de taille 3 x 3 avec foulée 2. La carte des caractéristiques résultante à la forme 13 x 13 x 256. Nous appliquons maintenant la troisième opération de convolution avec 384 filtres de tailles 3x3 de foulées 1 et aussi le rembourrage 1. Encore une fois la fonction d'activation utilisée est relu, la carte des caractéristiques en sortie à la forme 13 x 13 x 384.

On a alors la quatrième opération de convolution avec 384 filtres de taille 3 x 3. La foulée avec le rembourrage est 1, en même temps que la fonction d'activation utilisée est relue, maintenant la taille de sortie reste inchangée, en d'autres termes 13 x 13 x 384. Après cela nous avons la couche de convolution finale de taille 3 x 3 avec 256 filtres de ce type la foulée et le rembourrage sont réglés sur 1 et la fonction de déclenchement est relu, la carte des caractéristiques résultant à la forme 13 x 13 x 256.

On remarque que l'architecture jusqu'à présent : le nombre de filtres augmente au fur et à mesure que nous approfondissons, pour cela extrait plus de fonctionnalités à mesure que nous passons à l'architecture, en même temps la taille de filtre diminue.

Ce que signifie que le filtre initial était plus gros et comme nous allons la taille de filtre diminue , résultant en une diminution de la forme de la carte des caractéristiques, ensuite nous appliquons la troisième couche de regroupement maximum de taille 3 x 3 foulées 2. D'où la carte des caractéristiques de la forme 6x6x 256.

Des couches absolument connectées et abandonnées : après cela nous avons notre première couche d'abandon, le taux de désabonnement est fixé à 0,5 ensuite nous avons la première couche absolument connectée avec une fonction d'activation réticente, la taille de sortie et 4096, vient ensuite une autre couche de désabonnement avec le taux de désabonnement réglé sur 0,5. Ceci suivi d'une deuxième couche absolument liée à 4096 neurones et activation relu.

En conclusion, nous avons la dernière couche ou couche de sortie absolument connectée avec 1000 neurones, depuis que nous avons 2 classes dans l'ensemble de données, la fonction d'activation utilisé dans cette couche est softmax. [34]

Couche	Filtres/neurones	Dimension du Filtre (taille)	Foulée	Rembourrage	Taille de carte Des caractéristiques	Fonction d'activation
Input	---	---	---	---	27 x 27 x 3	---
Conv 1	96	11x11	4	---	55 x 55 x 96	ReLU
Max Pool 1	---	3x3	2	---	27 x27 x 96	---
Conv 2	256	5x5	1	2	27 x 27 x 256	ReLU
Max Pool 2	---	3x3	2	---	13 x 13 x 256	---
Conv 3	384	3x3	1	1	13 x 13 x 384	ReLU
Conv 4	384	3x3	1	1	13 x 13 x 384	ReLU
Conv 5	256	3x3	1	1	13 x 13 x 256	ReLU
Max Pool 3	---	3x3	2	---	6 x 6 x 256	---
Dropout 1	Rate=0.5	---	---	---	6 x 6 x 256	---

Tableau I.1 :Les paramètres d'architecture Alexnet [34]

II.6.2 Le modèle Googlenet

L'architecture GoogleNet comprend 22 couches, dont 27 couches de regroupement. Il y a 9 modules de démarrage empilés linéairement au total. Les extrémités des modules de démarrage sont connectées à la couche de regroupement des moyennes globales. L'architecture complète de GoogleNet est montrée dans la figure II.

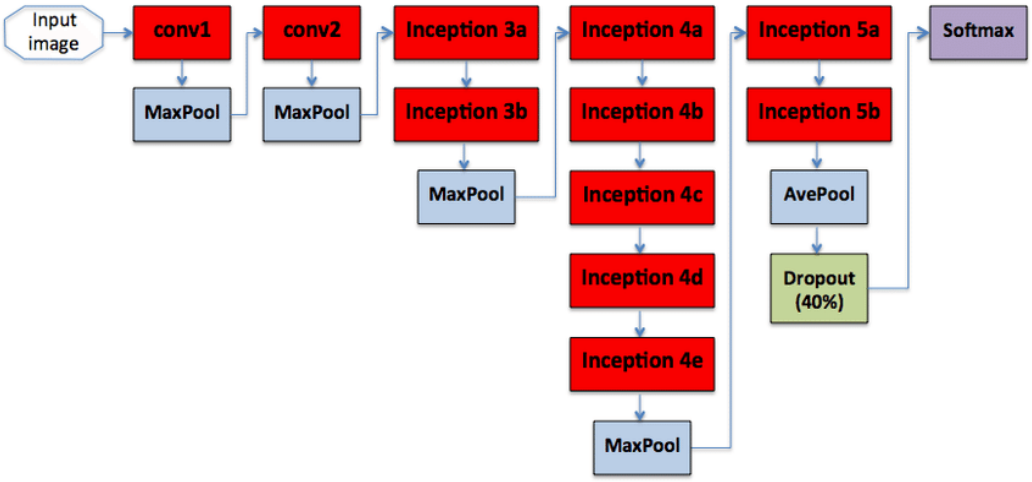


Figure II.5 : Architecture de GoogleNet[35]

La première couche de conv utilise un filtre de taille de 7x7.

L'utilité de cette couche est de réduire l'image d'entrée sans perdre d'informations spatiales, la deuxième couche de convolution à une profondeur de deux et exploite le bloc de conv 1x1, qui a pour effet de réduire la dimensionnalité. La réduction de la dimensionnalité via le bloc de conversion 1x1 permet de réduire la charge de calcul en réduisant le nombre d'opérations des couches. Il existe deux couches de mise en commun maximale entre certains modules de démarrage. L'utilité de ces couches de regroupement maximal est de sous-échantillonner l'entrée au fur et à mesure qu'elle est transmise via le réseau. [36]

II.6.3 Le modèle VGGnet

VGGNet est une architecture de réseau neuronal convolutif proposée par Karen Simonyan et Andrew Zisserman de l'Université d'Oxford en 2014. [37]

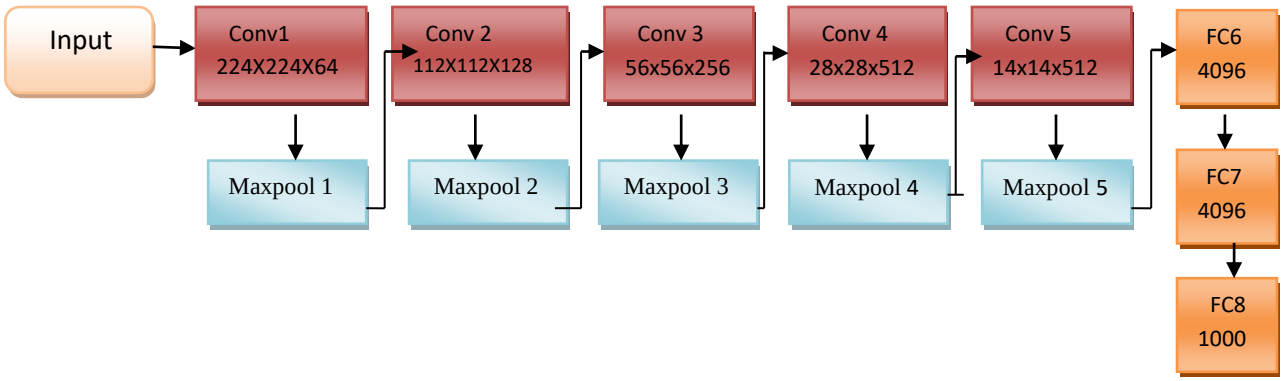


Figure II.6 : Architecture VGGnet

Cette architecture travaille comme suit :

Les deux premières couches sont des couches convolutionnelles avec 3*3 filtres, et utilisent 64 filtres de taille 3*3 avec une foulée de 1. Après cela, la couche de regroupement a été utilisée avec un pool maximum de taille 2 * 2 et une foulée 2. Ceci est suivi de 2 autres couches de convolution avec 128 filtres et une couche de regroupement. Trois autres couches de convolution sont ajoutées

avec 256 filtres de taille 3*3 et unecouche de regroupement. Trois couches de convolutions sont ajoutées avec nombre de 512 filtres de taille 3*3 et une couche de regroupement ,et aussi trois autres couches avec 512 filtres de taille 3*3 sont suivis avec couche de regroupement .Toutes les couches de regroupement précédentes sont utilisées avec foulée de 2 .

Après la couche de regroupement finale, le volume 7*7*512 est aplati dans la couche entièrement connectée (FC) avec 4096 canaux et une sortie softmax de 2 classes. [37]

II.6.4 Le modèle CNN(l'architecture que nous avons proposée figure II.8)

-Le premier modèle que nous présentons dans la figure II.8 est composé de trois couches de convolution et deux couches de maxpooling .

-L'image en entrée est de taille 200*200, l'image passe d'abord à la première couche de convolution,cette couche est composée de 32 filtres de taille 3*3, chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU cette fonction force les neurones à retourner des valeurs positives, après cette convolution 32 cartes des caractéristiques de taille 198*198 seront créés.

-Ensuite on applique le Maxpool aux les 32 cartes de caractéristiques qui sont obtenues pour réduire la taille de l'image et la quantité de paramètres de calcul. À la sortie de cette couche, nous aurons 32cartes des caractéristiques de taille 99*99.

On répète le même travail avec la deuxième et la troisième couche de convolution sans faire le maxpooling dans cette dernière couche qui est composée de 64 filtres, la fonction d'activation ReLU est appliquée toujours sur cette convolution. Nous aurons 64 cartes des caractéristiques de taille 48*48. Le vecteur de caractéristiques issu des convolutions a une dimension de 135424.

Dans la dernière couche nous appliquons une couche dense avec fonction d'activation sigmoïde qui classifier en binaire .

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 198, 198, 32)      896

 max_pooling2d (MaxPooling2D  (None, 99, 99, 32)       0
 )

 conv2d_1 (Conv2D)           (None, 97, 97, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 48, 48, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 46, 46, 64)        36928
```

Figure II.7 : Configuration du modèle

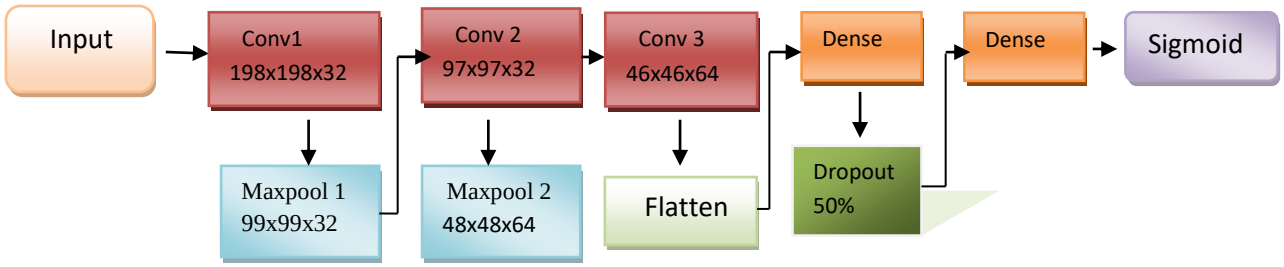


Figure II.8 : Notre model proposé

II.7 Architecture U-net pour la détection

Les architectures précédentes sont utilisées pour la classification des images. U-net crée un masque ou la zones atteintes par la Rd sont montrées.

Le réseau a une base importante qui ressemble à

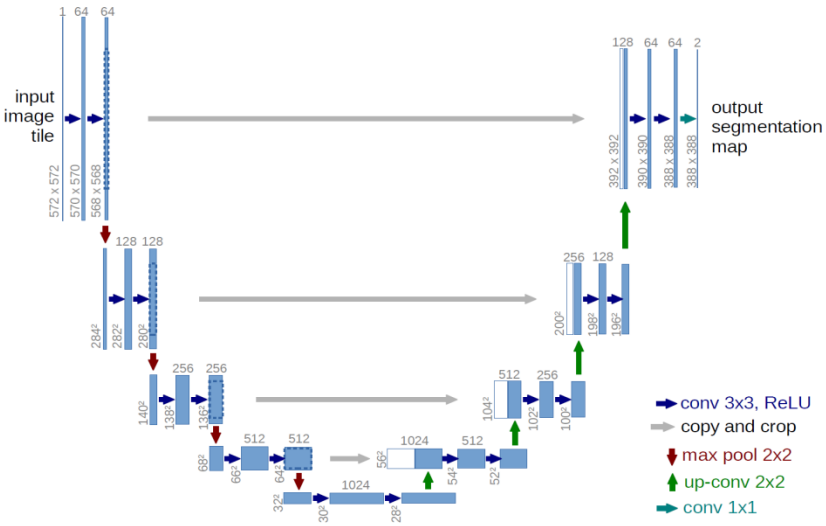


Figure II.9 : Architecture U-net

À première vue, il a une forme en « U ». L'architecture est symétrique et se compose de deux parties principales - la partie gauche est appelée chemin de contraction (aussi encodeur),qui est constitué par le processus général de convolution; la partie droite est le chemin expansif (décodeur), qui est constitué de couches convolutives 2D transposées (vous pouvez le penser comme une technique de suréchantillonnage pour l'instant).

1-Chemin de passation :

```
conv_layer1 -> conv_layer2 -> max_pooling -> dropout(optional)
```

Figure II.10 : Chemin de passation

Notez que chaque processus constitue deux couches convolutives

Dans le premier processus les deux couches de convolution est composée de 128 filtres de taille 3*3, Chacune de nos couches de convolution est suivie d'une fonction d'activation ReLU et de foulée de 1. A cause du problème de rembourrage l'implémentation utilise le padding = « same »),ces couches est séparées par une couche de batchnormalisation Cette approche conduit à des taux d'apprentissage plus rapides puisque la normalisation garantit qu'il n'y a pas de valeur d'activation trop élevée ou trop faible, tout en permettant à chaque couche d'apprendre indépendamment des autres. on applique une couche max pool de dimension de 2x2 et foulée de 2. On répète la même chose avec les autres processus avec 256 filtres, 512 filtres, 1024 et 2048 filtres.

-La convolution transposée est une technique de suréchantillonnage qui élargit la taille des images. Fondamentalement, il fait un peu de rembourrage sur l'image d'origine suivie d'une opération de convolution.

Après la convolution transposée, l'image est redimensionnée de 8x8 → 128x128, puis, cette image est concaténée avec l'image correspondante du chemin de contraction et fait une image de taille 128x128. La raison est de combiner les informations des couches précédentes afin d'obtenir une prédiction plus précise.

La dernière couche est une couche de convolution avec 3 filtres de taille 3x3 (notez qu'il n'y a pas de couche dense dans l'ensemble du réseau) contient aussi une fonction de sigmoïde et de et foulée de 1.

II.8 Paramètres d'entraînement

II.8.1 La fonction Relu

on a utilisé la fonction relu dans les couches intermédiaires, elle permet de passer les valeurs supérieure à 0 dans les couches suivantes du réseau de neurones.

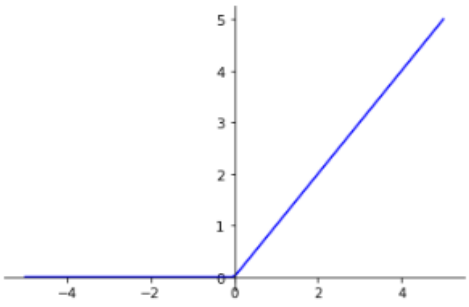


Figure II.11 : La Représentation de la Fonction Relu

II.8.2 La fonction softmax

la fonction softmax est ajoutée à la fin de la sortie car c'est l'endroit où les nœuds se rencontrent finalement et ainsi ils peuvent être classés. La fonction Softmax est utilisée dans les algorithmes de classification où il est nécessaire d'obtenir une probabilité ou une distribution de probabilité en sortie. [39]

Utilisation de fonction softmax

- La fonction Softmax est utilisée pour convertir la sortie numérique en valeurs dans la plage [0, 1]
- La fonction Softmax est utilisée dans les méthodes de classification multi-classe telles que les réseaux de neurones, la régression logistique multinomiale, la LDA multi-classe, le classificateur Naïve Bayes.
- La fonction Softmax est utilisée comme fonction d'activation dans la dernière couche de l'algorithme du réseau neuronal[40].

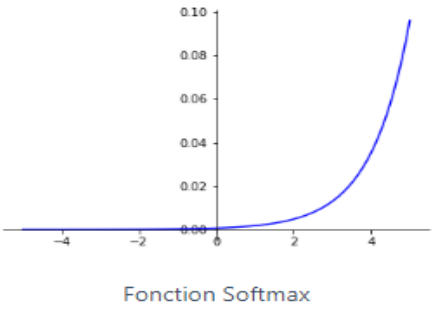


Figure II.12: La Représentation de la Fonction Softmax. [41]

II.8.3 La Fonction sigmoïde

La sigmoïde est une fonction non linéaire, on a utilisé la fonction sigmoïde en la classification binaire ,l'avantage principale de cette fonction sera toujours comprise entre 0 et 1.

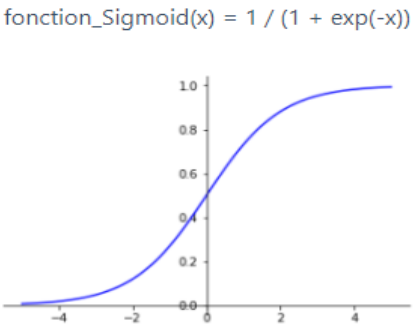


Figure II.13 : La Représentation de la fonction sigmoïde. [41]

II.8.4 Flatten

Comme le nom de cette étape l'indique, nous allons littéralement aplatir notre carte de caractéristiques regroupées dans une colonne comme dans la figure II.14.Nous l'avons fait pour alimenter ces données dans un réseau neuronal artificiel.

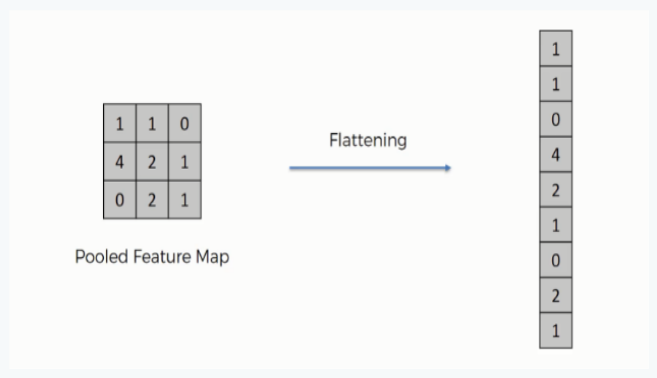


Figure II.14 : Représentation de l'aplanissement [42]

II.8.5 La couche dense

La couche dense est une couche de réseau de neurones profondément connectée. Autrement dit, chaque neurone de la couche dense reçoit une entrée de tous les neurones de la couche précédente. On a Utilisé des couche denses pour modifier les dimensions du vecteur, et cette couches effectuent une multiplication matrice-vecteur. [43]

II.8.6 Optimisation Adam

L'optimisation d'Adam est une méthode de descente de gradient stochastique basée sur l'estimation adaptative des moments de premier et de second ordre. Selon Kingma et al., 2014, la méthode est "efficace en termes de calcul, nécessite peu de mémoire, est invariante à la mise à l'échelle diagonale des gradients et convient bien aux problèmes volumineux en termes de données/paramètres". [44]

II.8.7 Epoch

Epoch signifie former un réseau de neurones en utilisant toutes les données de formation en un seul cycle. Epoch utilise toutes les données une seule fois. La passe avant et la passe arrière sont combinées et comptées comme une seule passe.

II.8.8 Dropout

Le Dropout est une technique permettant de réduire l'overfitting lors de l'entraînement du modèle. Le terme " Dropout " fait référence à la suppression de neurones dans les couches d'un modèle de Deep learning. [45]

II.8.9 batch size

La taille du lot est le nombre de valeurs de données d'entrée que vous entrez une fois dans le formulaire. Il est très important lors de la formation et secondaire lors des tests.

II.8.10 La fonction du perte(loss)

La fonction de perte est une fonction qui évalue la différence entre la prédiction du réseau de neurones et la valeur réelle des observations utilisées lors de l'apprentissage. Plus le résultat de cette fonction est petit, meilleur sera le fonctionnement du réseau de neurones.

II.8.10 Accuracy

La précision est une mesure qui décrit généralement les performances du modèle dans toutes les classes.

La précision est calculée comme le rapport entre le nombre d'échantillons *positifs* correctement classés et le nombre total d'échantillons classés comme *positifs* (correctement ou incorrectement). La précision mesure l'exactitude du modèle à classer un échantillon comme positif. [46]

La précision est une mesure permettant d'évaluer les modèles de classification. De manière informelle, la précision est la fraction des prédictions que notre modèle a eu raison.

II.8.11 Input shape

C'est une image dont les deux premières valeurs font référence à la largeur et la hauteur de l'image, et la troisième fait référence aux canaux de l'image.

II.8.12 Padding

l'un des paramètres des calques, lorsque cette valeur est définie sur "identique", les éléments de bordure de la forme ne sera pas jeté.

II.8.13 Shuffle

lorsque cette valeur est définie sur "True", l'ensemble de données sera mélangé avant l'entraînement.

II.8.14 Class weight

La façon dont le CNN accorde de l'importance aux classes, il est utilisé lorsque l'ensemble de données est déséquilibré.

II.8.15 Step per epoch

Le nombre d'itérations du lot avant qu'une époque d'apprentissage ne soit terminé, il est généralement calculé en divisant le nombre d'échantillons en formation un ensemble sur la taille du lot.

II.8.16 Matrice de confusion

C’est un tableau qui permet d’évaluer les performances d’un modèle de classification et donne le résumé de résultats de prédiction tel que les vraies négatives et les vraies positives ,en contre partie les fausses négatives et les fausses positives .

II.9 Environnement de programmation autour de Python

Python est un langage de programmation puissant et facile à apprendre. Il a une structure de données de haut niveau et fournit une approche simple mais efficace de la programmation orientée objet. Sa syntaxe raffinée, son typage dynamique et son interopérabilité font de Python un langage idéal pour la création de scripts et

le développement rapide d'applications dans de nombreux domaines et sur la plupart des plates-formes.

python c'est une langage de programmation à niveau haut ,innové par Guido Van Rossum en 1989.Python est le langage qui connait la croissance la plus rapide et le plus largement utilisé.

Les Avantages de langage python

-Open source et gratuit.

- Simple et facile à apprendre.

-Langage puissant.

-Multi plateforme : Windows, Mac Os, Linux , Ios , Android , Raspberry Pi Os..

II.9.1 les bibliothèques de python

II.9.1.1 Tensorflow

Tensorflow est une bibliothèque open source pour le calcul numérique et l'apprentissage automatique à grande échelle. Tensorflow regroupe une multitude de modèles et d'algorithmes d'apprentissage automatique et d'apprentissage en profondeur (alias réseaux neuronaux) et les rend utiles au moyen d'une métaphore commune. Il utilise Python pour fournir une API frontale pratique pour créer des applications avec le framework [47]

Tensorflow simplifie l'envoi de nouveaux calculs et examens, tout en conservant une conception de serveur et des API similaires. Tensorflow Serving est fourni hors du boîtier en se joignant aux modèles Tensorflow, mais peut être facilement atteint pour servir différents types de modèles.

II.9.1.1.1 TFlearn

TFlearn est une bibliothèque d'apprentissage profond modulaire construite sur Tensorflow. Il a été conçu pour fournir une API de niveau supérieur à Tensorflow afin de faciliter et d'accélérer les expérimentations. On a utilisé TFlearn par exemple pour redimensionner l'image.

II.9.1.2 Keras

Keras est une API d'apprentissage en profondeur écrite en Python, s'exécutant sur la plate-forme d'apprentissage automatique Tensorflow . Il a été développé dans le but de permettre une expérimentation rapide.

Keras a été créé pour être convivial, modulaire, facile à étendre et fonctionner avec Python. Les couches neuronales, les fonctions de coût, les optimiseurs, les schémas d'initialisation, les fonctions d'activation et les schémas de régularisation sont tous des modules autonomes que vous pouvez combiner pour créer de nouveaux modèles. De nouveaux modules sont simples à ajouter, comme de nouvelles classes et fonctions. Les modèles sont définis dans le code Python, et non dans des fichiers de configuration de modèle séparés.

II.9.1.3 Open cv

Open CV est une bibliothèque gratuite de vision industrielle multiplateforme **(**versions existantes pour GNU / Linux, Mac OS X, Windows et Android) qui était à l'origine développé par Intel et utilisé dans d'innombrables applications des systèmes de sécurité avec détection de mouvement aux applications de contrôle de processus où la reconnaissance d'objets est requise. En effet, sa publication est donnée sous licence BSD, ce qui lui permet d'être utilisé librement à des fins commerciales et de recherche dans les conditions qui y sont exprimées. [49]

II.9.1.4 Matplotlib

Matplotlib est une bibliothèque Python open source**,** initialement développée par le neurobiologiste John Hunter en 2002. L'objectif était de visualiser les signaux électriques du cerveau de personnes épileptiques. Pour y parvenir, il souhaitait répliquer les fonctionnalités de création graphique de MATLAB avec Python.

Il est par exemple possible de créer des tracés, des histogrammes, des diagrammes à barre et tous types de graphiques à l'aide de quelques lignes de code. Il s'agit d'un outil très complet, permettant de générer des visualisations de données très détaillées. [50]

II.9.1.5 Numpy

Numpy est une bibliothèque *numérique* apportant le support efficace de larges tableaux multidimensionnels, et de routines mathématiques de haut niveau (fonctions spéciales, algèbre linéaire, statistiques, etc..).

II.9.1.6 Pandas

Pandas est un outil d'analyse et de manipulation de données open source rapide, puissant, flexible et facile à utiliser, construit sur le langage de programmation Python.

II.9.1.7 Tqdm

Tqdm est une bibliothèque en Python utilisée pour créer des indicateurs de progression ou des barres de progression. tqdm tire son nom du nom arabe ***taqaddum*** qui signifie « progrès ». L'implémentation de tqdm peut se faire sans effort dans nos boucles, fonctions ou même Pandas. [51]

II.10 Jupyter

JupyterLab est le dernier environnement de développement interactif basé sur le Web pour les blocs-notes, le code et les données. Son interface flexible permet aux utilisateurs de configurer et d'organiser des flux de travail en science des données, en informatique scientifique, en journalisme informatique et en apprentissage automatique. Une conception modulaire invite les extensions à étendre et enrichir les fonctionnalités.

Jupyter Notebook est l'application Web originale pour créer et partager des documents informatiques. Il offre une expérience simple, rationalisée et centrée sur les documents. [52]

Il est simple, permet de traiter directement le programme ,si on écrivons quelques lignes de code nous pouvons les exécuter sans avoir à exécuter tout le programme, c'est une fonctionnalité de python que jupyter a utilisé pour faciliter la manipulation ,les explications et le travail avec le big data.Après l'installation de Anaconda ,Jupyter s’ouvre via un raccourci ou à la ligne de commande

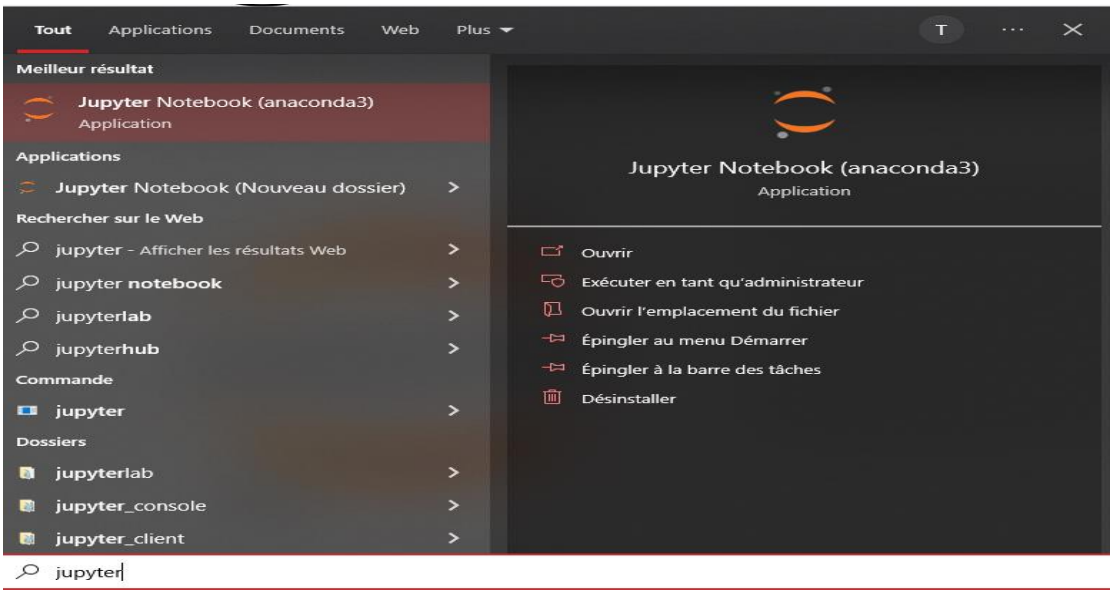


Figure II.15 : Ouverture de jupyter

Jupyter s'ouvre dans votre navigateur par défaut

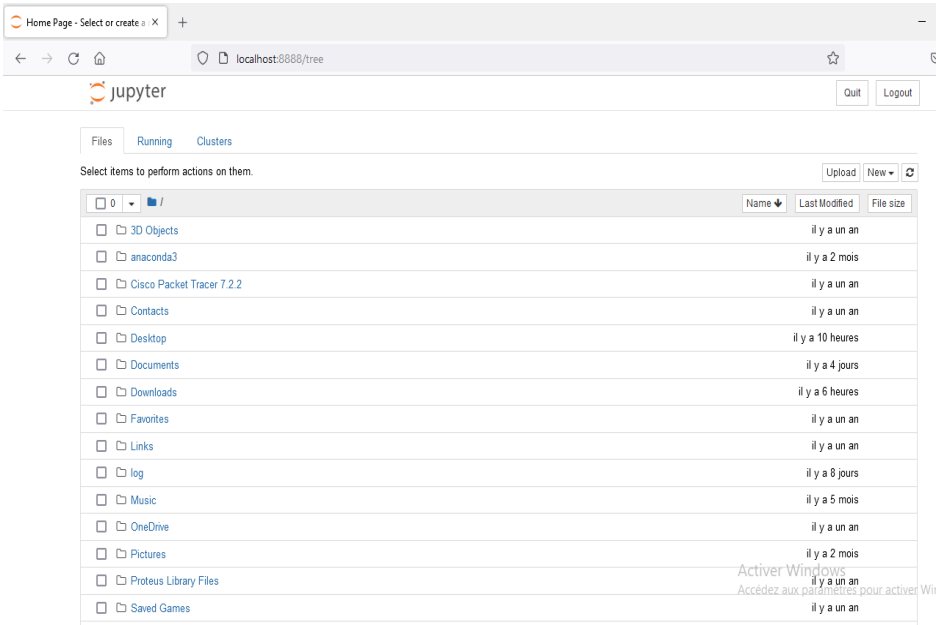


Figure II.16 : Navigature jupyter

Grâce au bouton New vous pouvez créer un nouveaunotebook

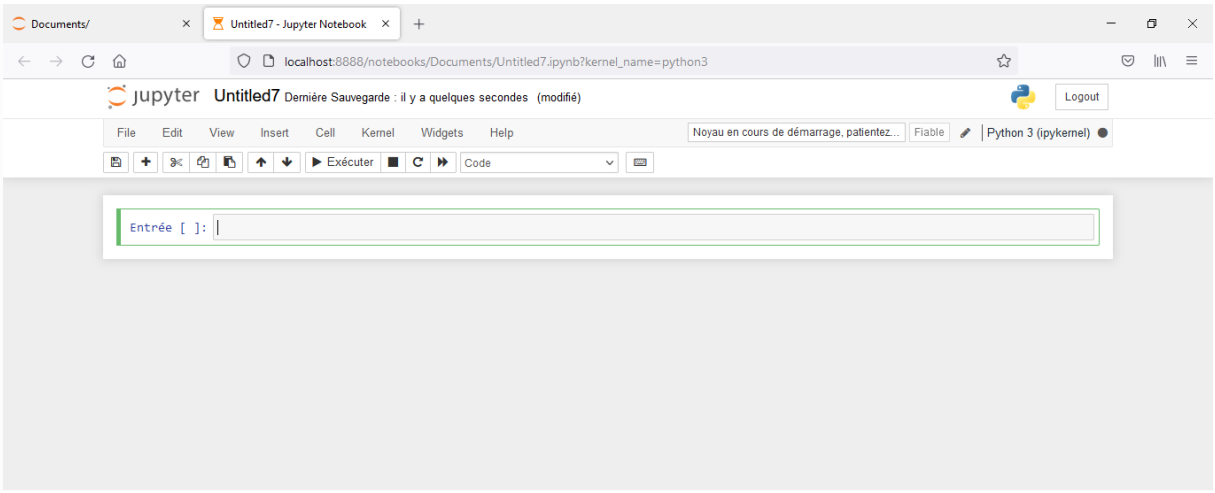


Figure II.17 : Création de nouveau notebook

II.11 Environnement Google Colab pour le Deep learning

II.11.1 Définition

C'est un document exécutable qui nous permet d'écrire ,exécuter et partager des codes sur Google drive .Un document notebook est composé de cellules ,chacune desquelles peut contenir un code, du texte,des images . Colab utilise toutes les fonctionnalité de python.

Google Colab est basé sur Jupyter Notebook et destiné à la formation et à la recherche dans l'apprentissage automatique. Cette plateforme permet d'entraîner des modèles de Machine Learning directement dans le cloud. Sans donc avoir besoin d'installer quoi que ce soit sur notre ordinateur à l'exception d'un navigateur.

II.11.2 Utilisation de Google colab

●Pour utiliser Google colab il suffit d'aller sur Google drive ensuite cliquer sur nouveau ensuite sur « plus » et choisissez « Colaboratory ».

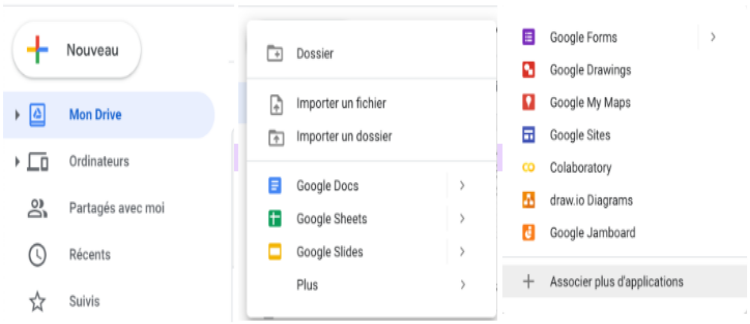


Figure II.18 : Ouverture de google colab

●Interface de Google colab

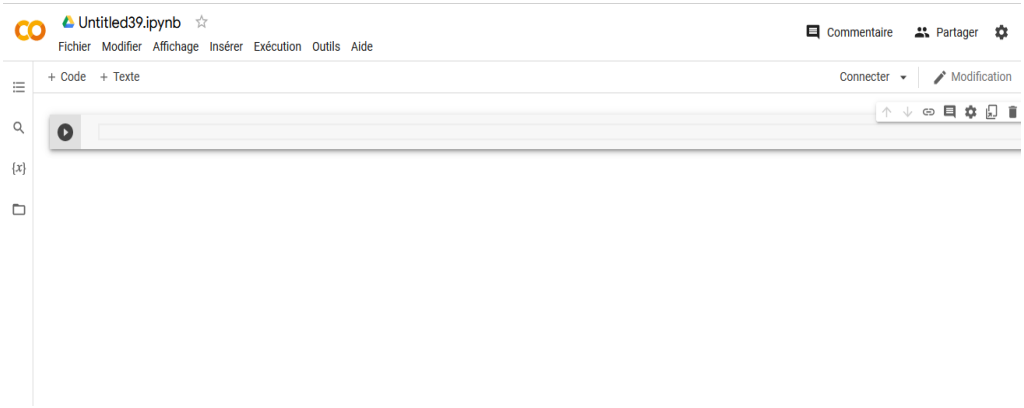


Figure II.19 : Interface de google colab

●On commence par installer les bibliothèque que'on utilise dans notre code

Exemple TFlearn

Figure II.20 : Installation des bibliothèque

●Importation des Fichiers

Avant l'importation d'une fichier il suffit de connecter colab avec googl.drive comme le montre dans la figure ci-dessous :

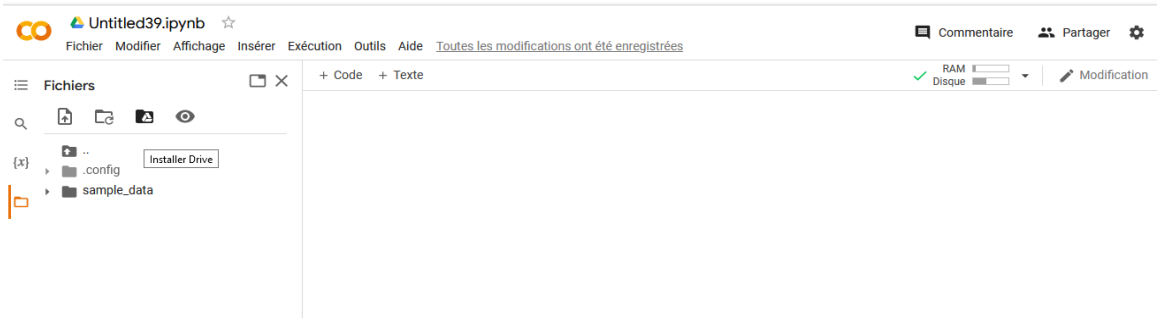


Figure II.21 : Connections du Google colab et Google drive

●Importation des fichiers depuis l'ordinateur

```
from google.colab import files
uploaded = files.upload()
```

Figure II.22: Importation des fichiers depuis l'ordinateur

Files.upload renvoie un dictionnaire des fichiers qui ont été importés. La clé du dictionnaire est le nom du fichier importé, la valeur correspond aux données des fichiers importés.

●Téléchargement des fichiers

```
from google.colab import files
with open('example.txt', 'w') as f:
  f.write('exemple de contenu')
files.download('example.txt')
```

Figure II.23 : Téléchargement des fichiers

●Utilisation des fichiers depuis google drive

Après avoir lié colab avec google.drive, nous importons les fichiers que nous voulons utiliser :

Figure II.24 : Importation des fichiers

II.12 Conclusion

Dans ce chapitre, nous avons présenté les réseaux de neurones convolutionnels.

Ces réseaux sont capables d'extraire des caractéristiques d'images présentées en entrée et de classifier ces caractéristique, ils implémentent aussi l'idée de partage des poids qui permettant de réduire beaucoup de nombre de paramètres libres de l'architecture . Nous avons introduit ces réseaux de neurones convolutionnelles en présentant les différents types de couches utilisées (la couche convolutionnelle,la couche de pooling),Nous avons parlé aussi sur le s méthodes de régularisation (dropout) utilisées pour éviter le problème de sur apprentissage. Dans le chapitre suivant nous allons implémenter les architectures décrite y compris celle que nuis avons créé et ensuite interpréter les résultats obtenus dans la phase d'apprentissage et de test, et les discuter et critiquer

Chapitre III

Implémentation
Et
résultats

III.1 Introduction

Ce dernier chapitre est consacré à la mise en oeuvre de notre proposition pour améliorer la classification et la détection de la rétinopathie à partir d'images médicales . Dans un premier temps nous présenterons l'implémentation des méthodes théoriques présentées dans les chapitres précédents ,nous détaillerons chacune de ces différentes architectures pour la classification y compris l'architecture que nous avons proposé pour faire une étude comparative. Ensuite nous exposerons deux modèles de segmentation et l'ensemble des étapes d'implémentation U-net.

III.2 Classification

Nous utilisons les trois architectures **: Alexnet , Googlenet , VGGNet** ,ces architectures ne diffèrent que par la cellule d'architecture. Le but de trouver quelle architecture donne le meilleurs résultats de classification de la RD.

A fin de pouvoir comparer leurs performances nous les entrainons avec la même base de données et elles sont appelés avec le même programme principale.

III.2.1 Le programme principal

III.2.1.1 Importation des bibliothèques

La première cellule contient l'importation des bibliothèque que nous utilisons dans notre programme .

```
import tensorflow as tf
import cv2
import numpy as np
import os
from random import shuffle
from tqdm import tqdm
```

Figure III.1 Extraire de programme pour l'importation des bibliothèques

Nous importons ces bibliothèques nécessaires pour faciliter notre travail en traitement d'images, les calculs numériques, les tableaux, créer des compteurs de progression.

III.2.1.2 Description de la base de données

Pour l'apprentissage nous utilisons la base de données de site web Kaggle.

Pour le test nous utilisons la base de données donnée par les ophtalmologistes de l'hôpital Frantz-Fanon de Blida. Chaque image caractérisée par le nom du patient, les images d'entraînement est 3616 et de test 114 images .

Nous importons les dossiers :

```
TRAIN_DIR = '/content/drive/MyDrive/Train'
TEST_DIR = '/content/drive/MyDrive/Test'
IMG_SIZE = 200
LR = 1e-3
MODEL_NAME = 'ouivsnon'
```

Figure III.2 : Importation des dossiers « entraînement » et « Test »

Nous affichons le nombre d'images dans le dossier d'entraînement:

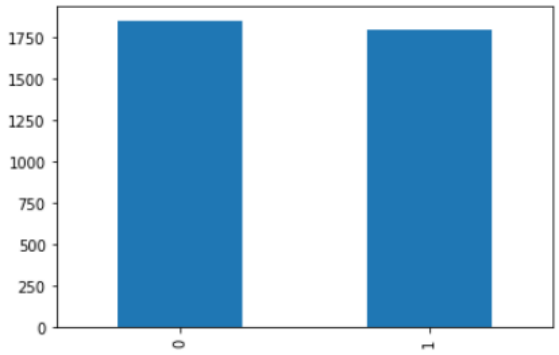


Figure III.3 Affichage le nombre d'images dans le dossier d'entraînement

III.2.1.3 Configuration des fonctions

III.2.1.3.1 Configuration de fonction « train »

Nous créons la fonction qui contient les différents opérations de traitement d'image (l'ouverture de fichier d'entraînement **os.path.join** ,le nom d'étiquette **label_img** ,la lecture des images **cv2.imread** en niveau de gris , redimensionner la taille des images **cv2.resize** ..)

```
def create_train_data():
    training_data = []
    for img in tqdm(os.listdir(TRAIN_DIR)):
        label = label_img(img)
        path = os.path.join(TRAIN_DIR,img)
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        training_data.append([np.array(img),np.array(label)])
    shuffle(training_data)
    np.save('train_data.npy', training_data)
    return training_data
```

Figure III.4 : Fonction d'entraînement (Train)

III.2.1.3.2 Configuration de fonction « test »

Avec les mêmes étapes précédentes

```
def process_test_data():
    testing_data = []
    for img in tqdm(os.listdir(TEST_DIR)):
        path = os.path.join(TEST_DIR,img)
        img_num = img.split('.')[0]
        img = cv2.imread(path,cv2.IMREAD_GRAYSCALE)
        img = cv2.resize(img, (IMG_SIZE,IMG_SIZE))
        testing_data.append([np.array(img), img_num])

    shuffle(testing_data)
    np.save('test_data.npy', testing_data)
    return testing_data
```

Figure III.5 : Fonction test

III.2.1.4 Chargement des dossiers

Nous passons à l'exécution du chargement des dossiers d'entraînement et de test pour afficher le nombre d'images :

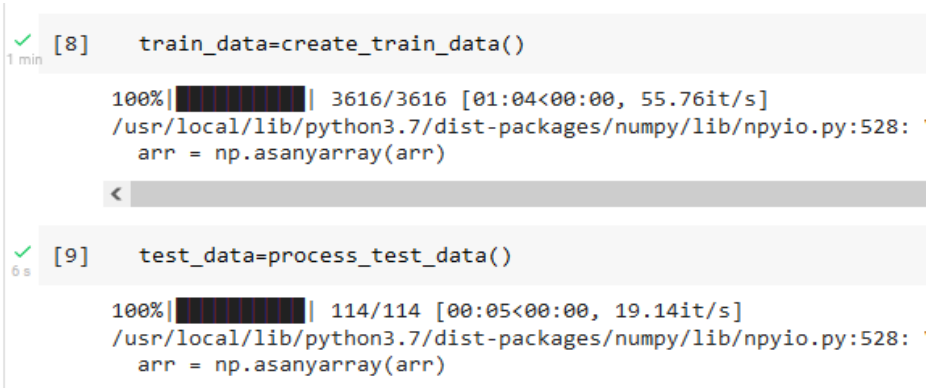


Figure III.6 : Chargement des dossiers

III.2.1.5 Implémentation et entraînement du réseau et entrainement

III.2.1.5.1 Entraînement des modèles

•Nous utilisons **« model.fit »** pour former nos modèles d'apprentissage en profondeur (Alexnet,VGGnet,Googlenet). Il nécessite un générateur pour les données de l'entraînement.

•Nous utilisons 500 échantillons pour **« Step_per_epoch »** qui représente le nombre total d'étapes (lots d'échantillons).

•Nous utilisons **« Epoch »** pour séparer la formation en phases distinctes, ce qui est utile pour la journalisation et l'évaluation périodique. Nous utilisons respectivement 50,80,100 nombre d'époques pour obtenir une accuracy parfait pour nos modèles .

III.2.1.5.2 Alexnet

Alexnet se compose de cinq couches convolutives, trois couches de maxpool et trois couches entièrement connectées

i/-Architecture :

```
from __future__ import division, print_function,absolute_import
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
from tflearn.layers.normalization import local_response_normalization

convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')

network = conv_2d(convnet, 96, 11, strides=4, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = conv_2d(network, 256, 5, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = conv_2d(network, 384, 3, activation='relu')
network = conv_2d(network, 384, 3, activation='relu')
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 3, strides=2)
network = local_response_normalization(network)
network = fully_connected(network, 4096, activation='tanh')
network = dropout(network, 0.5)
network = fully_connected(network, 4096, activation='tanh')
network = dropout(network, 0.5)
network = fully_connected(network, 2, activation='softmax')
convnet = regression(network, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')
model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Figure III.7 : Architecture Alexnet

ii/ Entraînement et Résultats pour Alexnet

L'entraînement du notre modèle pour le nombre **d'epoch=50** :

a/-

```
model.fit({'input': X}, {'targets': Y}, n_epoch=50, validation_set=({'input': test_x}, {'targets': test_y}),
snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 2299  | total loss: 11.13881 | time: 4.453s
| Adam | epoch: 050 | loss: 11.13881 - acc: 0.5162 -- iter: 2880/2888
Training Step: 2300  | total loss: 11.28415 | time: 5.554s
| Adam | epoch: 050 | loss: 11.28415 - acc: 0.5099 | val_loss: 11.60781 - val_acc: 0.4959 -- iter: 2888/2888
--
```

b/- Epoch=80

```
model.fit({'input': X}, {'targets': Y}, n_epoch=80, validation_set=({'input': test_x}, {'targets': test_y}),
snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 3599  | total loss: 0.85728 | time: 4.377s
| Adam | epoch: 080 | loss: 0.85728 - acc: 0.4903 -- iter: 2816/2880
Training Step: 3600  | total loss: 0.84580 | time: 5.478s
| Adam | epoch: 080 | loss: 0.84580 - acc: 0.4944 | val_loss: 0.69994 - val_acc: 0.4959 -- iter: 2880/2880
--
```

c/-Epochs =100 :

```
[12]  model.fit({'input': X}, {'targets': Y}, n_epoch=100, validation_set=({'input': test_x}, {'targets': test_y}),
      snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

Training Step: 4499  | total loss: 11.62664 | time: 4.535s
| Adam | epoch: 100 | loss: 11.62664 - acc: 0.4951 -- iter: 2816/2880
Training Step: 4500  | total loss: 11.65125 | time: 5.641s
| Adam | epoch: 100 | loss: 11.65125 - acc: 0.4940 | val_loss: 12.24039 - val_acc: 0.4684 -- iter: 2880/2880
--
```

Figure III.8 : Résultats d'apprentissag de Alexnet pour a=50,b=80,c=100

D'après ces résultats, nous observons que la précision d'entraînement est en train de diminuer avec le nombre d' epochs ,tandis que la perte d'entraînement augmente avec le nombre d'épochs.

III.2.1.5.3 Googlenet

GoogLeNet est un réseau de neurones à convolution profonde à 22 couches qui est une variante du réseau Inception.

i/-Architecture

```
from __future__ import division, print_function, absolute_import

import tflearn
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.conv import conv_2d, max_pool_2d, avg_pool_2d
from tflearn.layers.normalization import local_response_normalization
from tflearn.layers.merge_ops import merge
from tflearn.layers.estimator import regression
```

```
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')
conv1_7_7 = conv_2d(convnet, 64, 7, strides=2, activation='relu', name='conv1_7_7_s2')
pool1_3_3 = max_pool_2d(conv1_7_7, 3, strides=2)
pool1_3_3 = local_response_normalization(pool1_3_3)
conv2_3_3_reduce = conv_2d(pool1_3_3, 64, 1, activation='relu', name='conv2_3_3_reduce')
conv2_3_3 = conv_2d(conv2_3_3_reduce, 192, 3, activation='relu', name='conv2_3_3')
conv2_3_3 = local_response_normalization(conv2_3_3)
pool2_3_3 = max_pool_2d(conv2_3_3, kernel_size=3, strides=2, name='pool2_3_3_s2')
```

```
# 3a
inception_3a_1_1 = conv_2d(pool2_3_3, 64, 1, activation='relu', name='inception_3a_1_1')
inception_3a_3_3_reduce = conv_2d(pool2_3_3, 96, 1, activation='relu', name='inception_3a_3_3_reduce')
inception_3a_3_3 = conv_2d(inception_3a_3_3_reduce, 128, filter_size=3,  activation='relu', name='inception_3a_3_3')
inception_3a_5_5_reduce = conv_2d(pool2_3_3, 16, filter_size=1, activation='relu', name='inception_3a_5_5_reduce')
inception_3a_5_5 = conv_2d(inception_3a_5_5_reduce, 32, filter_size=5, activation='relu', name='inception_3a_5_5')
inception_3a_pool = max_pool_2d(pool2_3_3, kernel_size=3, strides=1, name='inception_3a_pool')
inception_3a_pool_1_1 = conv_2d(inception_3a_pool, 32, filter_size=1, activation='relu', name='inception_3a_pool_1_1')
inception_3a_output = merge([inception_3a_1_1, inception_3a_3_3, inception_3a_5_5, inception_3a_pool_1_1], mode='concat', axis=3)
```

```
# 3b
inception_3b_1_1 = conv_2d(inception_3a_output, 128, filter_size=1, activation='relu', name='inception_3b_1_1')
inception_3b_3_3_reduce = conv_2d(inception_3a_output, 128, filter_size=1, activation='relu', name='inception_3b_3_3_reduce')
inception_3b_3_3 = conv_2d(inception_3b_3_3_reduce, 192, filter_size=3, activation='relu', name='inception_3b_3_3')
inception_3b_5_5_reduce = conv_2d(inception_3a_output, 32, filter_size=1, activation='relu', name='inception_3b_5_5_reduce')
inception_3b_5_5 = conv_2d(inception_3b_5_5_reduce, 96, filter_size=5,  name='inception_3b_5_5')
inception_3b_pool = max_pool_2d(inception_3a_output, kernel_size=3, strides=1,  name='inception_3b_pool')
inception_3b_pool_1_1 = conv_2d(inception_3b_pool, 64, filter_size=1, activation='relu', name='inception_3b_pool_1_1')
inception_3b_output = merge([inception_3b_1_1, inception_3b_3_3, inception_3b_5_5, inception_3b_pool_1_1], mode='concat', axis=3, name='incept:
pool3_3_3 = max_pool_2d(inception_3b_output, kernel_size=3, strides=2, name='pool3_3_3')

# 4a
inception_4a_1_1 = conv_2d(pool3_3_3, 192, filter_size=1, activation='relu', name='inception_4a_1_1')
inception_4a_3_3_reduce = conv_2d(pool3_3_3, 96, filter_size=1, activation='relu', name='inception_4a_3_3_reduce')
inception_4a_3_3 = conv_2d(inception_4a_3_3_reduce, 208, filter_size=3,  activation='relu', name='inception_4a_3_3')
inception_4a_5_5_reduce = conv_2d(pool3_3_3, 16, filter_size=1, activation='relu', name='inception_4a_5_5_reduce')
inception_4a_5_5 = conv_2d(inception_4a_5_5_reduce, 48, filter_size=5,  activation='relu', name='inception_4a_5_5')
inception_4a_pool = max_pool_2d(pool3_3_3, kernel_size=3, strides=1,  name='inception_4a_pool')
inception_4a_pool_1_1 = conv_2d(inception_4a_pool, 64, filter_size=1, activation='relu', name='inception_4a_pool_1_1')
inception_4a_output = merge([inception_4a_1_1, inception_4a_3_3, inception_4a_5_5, inception_4a_pool_1_1], mode='concat', axis=3, name='incept:
```

```
# 4b
inception_4b_1_1 = conv_2d(inception_4a_output, 160, filter_size=1, activation='relu', name='inception_4a_1_1')
inception_4b_3_3_reduce = conv_2d(inception_4a_output, 112, filter_size=1, activation='relu', name='inception_4b_3_3_reduce')
inception_4b_3_3 = conv_2d(inception_4b_3_3_reduce, 224, filter_size=3, activation='relu', name='inception_4b_3_3')
inception_4b_5_5_reduce = conv_2d(inception_4a_output, 24, filter_size=1, activation='relu', name='inception_4b_5_5_reduce')
inception_4b_5_5 = conv_2d(inception_4b_5_5_reduce, 64, filter_size=5,  activation='relu', name='inception_4b_5_5')
inception_4b_pool = max_pool_2d(inception_4a_output, kernel_size=3, strides=1,  name='inception_4b_pool')
inception_4b_pool_1_1 = conv_2d(inception_4b_pool, 64, filter_size=1, activation='relu', name='inception_4b_pool_1_1')
inception_4b_output = merge([inception_4b_1_1, inception_4b_3_3, inception_4b_5_5, inception_4b_pool_1_1], mode='concat', axis=3, name='incept

# 4c
inception_4c_1_1 = conv_2d(inception_4b_output, 128, filter_size=1, activation='relu', name='inception_4c_1_1')
inception_4c_3_3_reduce = conv_2d(inception_4b_output, 128, filter_size=1, activation='relu', name='inception_4c_3_3_reduce')
inception_4c_3_3 = conv_2d(inception_4c_3_3_reduce, 256,  filter_size=3, activation='relu', name='inception_4c_3_3')
inception_4c_5_5_reduce = conv_2d(inception_4b_output, 24, filter_size=1, activation='relu', name='inception_4c_5_5_reduce')
inception_4c_5_5 = conv_2d(inception_4c_5_5_reduce, 64,  filter_size=5, activation='relu', name='inception_4c_5_5')
inception_4c_pool = max_pool_2d(inception_4b_output, kernel_size=3, strides=1)
inception_4c_pool_1_1 = conv_2d(inception_4c_pool, 64, filter_size=1, activation='relu', name='inception_4c_pool_1_1')
inception_4c_output = merge([inception_4c_1_1, inception_4c_3_3, inception_4c_5_5, inception_4c_pool_1_1], mode='concat', axis=3, name='incept
```

```
# 4d
inception_4d_1_1 = conv_2d(inception_4c_output, 112, filter_size=1, activation='relu', name='inception_4d_1_1')
inception_4d_3_3_reduce = conv_2d(inception_4c_output, 144, filter_size=1, activation='relu', name='inception_4d_3_3_reduce')
inception_4d_3_3 = conv_2d(inception_4d_3_3_reduce, 288, filter_size=3, activation='relu', name='inception_4d_3_3')
inception_4d_5_5_reduce = conv_2d(inception_4c_output, 32, filter_size=1, activation='relu', name='inception_4d_5_5_reduce')
inception_4d_5_5 = conv_2d(inception_4d_5_5_reduce, 64, filter_size=5,  activation='relu', name='inception_4d_5_5')
inception_4d_pool = max_pool_2d(inception_4c_output, kernel_size=3, strides=1,  name='inception_4d_pool')
inception_4d_pool_1_1 = conv_2d(inception_4d_pool, 64, filter_size=1, activation='relu', name='inception_4d_pool_1_1')
inception_4d_output = merge([inception_4d_1_1, inception_4d_3_3, inception_4d_5_5, inception_4d_pool_1_1], mode='concat', axis=3, name='incept

# 4e
inception_4e_1_1 = conv_2d(inception_4d_output, 256, filter_size=1, activation='relu', name='inception_4e_1_1')
inception_4e_3_3_reduce = conv_2d(inception_4d_output, 160, filter_size=1, activation='relu', name='inception_4e_3_3_reduce')
inception_4e_3_3 = conv_2d(inception_4e_3_3_reduce, 320, filter_size=3, activation='relu', name='inception_4e_3_3')
inception_4e_5_5_reduce = conv_2d(inception_4d_output, 32, filter_size=1, activation='relu', name='inception_4e_5_5_reduce')
inception_4e_5_5 = conv_2d(inception_4e_5_5_reduce, 128,  filter_size=5, activation='relu', name='inception_4e_5_5')
inception_4e_pool = max_pool_2d(inception_4d_output, kernel_size=3, strides=1,  name='inception_4e_pool')
inception_4e_pool_1_1 = conv_2d(inception_4e_pool, 128, filter_size=1, activation='relu', name='inception_4e_pool_1_1')
inception_4e_output = merge([inception_4e_1_1, inception_4e_3_3, inception_4e_5_5, inception_4e_pool_1_1], axis=3, mode='concat')
pool4_3_3 = max_pool_2d(inception_4e_output, kernel_size=3, strides=2, name='pool_3_3')

# 5a
inception_5a_1_1 = conv_2d(pool4_3_3, 256, filter_size=1, activation='relu', name='inception_5a_1_1')
inception_5a_3_3_reduce = conv_2d(pool4_3_3, 160, filter_size=1, activation='relu', name='inception_5a_3_3_reduce')
inception_5a_3_3 = conv_2d(inception_5a_3_3_reduce, 320, filter_size=3, activation='relu', name='inception_5a_3_3')
inception_5a_5_5_reduce = conv_2d(pool4_3_3, 32, filter_size=1, activation='relu', name='inception_5a_5_5_reduce')
inception_5a_5_5 = conv_2d(inception_5a_5_5_reduce, 128, filter_size=5,  activation='relu', name='inception_5a_5_5')
inception_5a_pool = max_pool_2d(pool4_3_3, kernel_size=3, strides=1,  name='inception_5a_pool')
inception_5a_pool_1_1 = conv_2d(inception_5a_pool, 128, filter_size=1, activation='relu', name='inception_5a_pool_1_1')
inception_5a_output = merge([inception_5a_1_1, inception_5a_3_3, inception_5a_5_5, inception_5a_pool_1_1], axis=3, mode='concat')

# 5b
inception_5b_1_1 = conv_2d(inception_5a_output, 384, filter_size=1, activation='relu', name='inception_5b_1_1')
inception_5b_3_3_reduce = conv_2d(inception_5a_output, 192, filter_size=1, activation='relu', name='inception_5b_3_3_reduce')
inception_5b_3_3 = conv_2d(inception_5b_3_3_reduce, 384,  filter_size=3, activation='relu', name='inception_5b_3_3')
inception_5b_5_5_reduce = conv_2d(inception_5a_output, 48, filter_size=1, activation='relu', name='inception_5b_5_5_reduce')
inception_5b_5_5 = conv_2d(inception_5b_5_5_reduce, 128, filter_size=5, activation='relu', name='inception_5b_5_5')
inception_5b_pool = max_pool_2d(inception_5a_output, kernel_size=3, strides=1,  name='inception_5b_pool')
inception_5b_pool_1_1 = conv_2d(inception_5b_pool, 128, filter_size=1, activation='relu', name='inception_5b_pool_1_1')
inception_5b_output = merge([inception_5b_1_1, inception_5b_3_3, inception_5b_5_5, inception_5b_pool_1_1], axis=3, mode='concat')
pool5_7_7 = avg_pool_2d(inception_5b_output, kernel_size=7, strides=1)
pool5_7_7 = dropout(pool5_7_7, 0.4)
```



```
# fc
loss = fully_connected(pool5_7_7, 2, activation='softmax')


convnet = regression(loss, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Figure III.9: Architecture du Googlenet

ii/-Entraînement et résultats pour Googlenet:

a/-Pour epoch=50 :

```
[13]  model.fit({'input': X}, {'targets': Y}, n_epoch=50, validation_set=({'input': test_x}, {'targets': test_y}),
      snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 2299  | total loss: 11.50043 | time: 17.257s
| Adam | epoch: 050 | loss: 11.50043 - acc: 0.5005 -- iter: 2880/2888
Training Step: 2300  | total loss: 11.42972 | time: 19.245s
| Adam | epoch: 050 | loss: 11.42972 - acc: 0.5036 | val_loss: 11.48130 - val_acc: 0.5014 -- iter: 2888/2888
--
```

b/-Epochs=80 :

```
model.fit({'input': X}, {'targets': Y}, n_epoch=80, validation_set=({'input': test_x}, {'targets': test_y}),
snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 3599  | total loss: 0.10585 | time: 17.419s
| Adam | epoch: 080 | loss: 0.10585 - acc: 0.9550 -- iter: 2816/2880
Training Step: 3600  | total loss: 0.10175 | time: 19.392s
| Adam | epoch: 080 | loss: 0.10175 - acc: 0.9579 | val_loss: 0.28318 - val_acc: 0.9162 -- iter: 2880/2880
--
```

c/-Epochs=100 :

```
model.fit({'input': X}, {'targets': Y}, n_epoch=100, validation_set=({'input': test_x}, {'targets': test_y}),
snapshot_step=500, show_metric=True, run_id=MODEL_NAME)
```

```
Training Step: 4499  | total loss: 0.15650 | time: 17.403s
| Adam | epoch: 100 | loss: 0.15650 - acc: 0.9254 -- iter: 2816/2880
Training Step: 4500  | total loss: 0.16026 | time: 19.383s
| Adam | epoch: 100 | loss: 0.16026 - acc: 0.9204 | val_loss: 0.28428 - val_acc: 0.8997 -- iter: 2880/2880
--
```

Figure III.10: Résultats d'apprentissage de Googlenet avec a=50,b=80,c=100

D'après ces résultats nous observons que la précision d'entraînement est en train de d'augmenter avec le nombre d'epochs,tandis que la perte d'entraînement augmente avec le nombre d'epochs.

III.2.1.5.4 VGGnet

VGGnet se compose de cinq couches convolutives, cinq couches de maxpool et trois couches entièrement connectées.

i/-Architecture

```
from __future__  import division, print_function,absolute_import
import tflearn
from tflearn.layers.conv import conv_2d, max_pool_2d
from tflearn.layers.core import input_data, dropout, fully_connected
from tflearn.layers.estimator import regression
```

```
convnet = input_data(shape=[None, IMG_SIZE, IMG_SIZE, 1], name='input')
network = conv_2d(convnet, 64, 3, activation='relu')
network = conv_2d(network, 64, 3, activation='relu')
network = max_pool_2d(network, 2, strides=2)

network = conv_2d(network, 128, 3, activation='relu')
network = conv_2d(network, 128, 3, activation='relu')
network = max_pool_2d(network, 2, strides=2)

network = conv_2d(network, 256, 3, activation='relu')
network = conv_2d(network, 256, 3, activation='relu')
network = conv_2d(network, 256, 3, activation='relu')
network = max_pool_2d(network, 2, strides=2)

network = conv_2d(network, 512, 3, activation='relu')
network = conv_2d(network, 512, 3, activation='relu')
network = conv_2d(network, 512, 3, activation='relu')
network = max_pool_2d(network, 2, strides=2)

network = conv_2d(network, 512, 3, activation='relu')
network = conv_2d(network, 512, 3, activation='relu')
network = conv_2d(network, 512, 3, activation='relu')
network = max_pool_2d(network, 2, strides=2)
```

```
network = fully_connected(network, 4096, activation='relu')
network = dropout(network, 0.5)
network = fully_connected(network, 4096, activation='relu')
network = dropout(network, 0.5)
network=fully_connected(network,2,activation='softmax')

convnet = regression(network, optimizer='adam', learning_rate=LR, loss='categorical_crossentropy', name='targets')

model = tflearn.DNN(convnet, tensorboard_dir='log')
```

Figure III.11 : Architecture de VGGnet

ii/- Entraînement et résultats pour VGGnet

a/-Epochs=50

```
✓ [14]   model.fit({'input': X}, {'targets': Y}, n_epoch=50, validation_set=({'input': test_x}, {'targets': test_y}),
5 min       snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

    Training Step: 2299  | total loss: 11.23159 | time: 4.530s
    | Adam | epoch: 050 | loss: 11.23159 - acc: 0.5122 -- iter: 2880/2921
    Training Step: 2300  | total loss: 11.43961 | time: 5.637s
    | Adam | epoch: 050 | loss: 11.43961 - acc: 0.5032 | val_loss: 11.00686 - val_acc: 0.5220 -- iter: 2921/2921
    --
```


b/-Epoch =80

```
model.fit({'input': X}, {'targets': Y}, n_epoch=80, validation_set=({'input': test_x}, {'targets': test_y}),
snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

Training Step: 3599  | total loss: 11.37829 | time: 31.866s
| Adam | epoch: 080 | loss: 11.37829 - acc: 0.5058 -- iter: 2816/2880
Training Step: 3600  | total loss: 11.17589 | time: 35.603s
| Adam | epoch: 080 | loss: 11.17589 - acc: 0.5146 | val_loss: 11.38641 - val_acc: 0.5055 -- iter: 2880/2880
--
```

c/-Epoch =100

```
[12]  model.fit({'input': X}, {'targets': Y}, n_epoch=100, validation_set=({'input': test_x}, {'targets': test_y }),
      snapshot_step=500, show_metric=True, run_id=MODEL_NAME)

Training Step: 4499  | total loss: 11.59544 | time: 34.498s
| Adam | epoch: 100 | loss: 11.59544 - acc: 0.4964 -- iter: 2816/2880
Training Step: 4500  | total loss: 11.73110 | time: 38.696s
| Adam | epoch: 100 | loss: 11.73110 - acc: 0.4905 | val_loss: 11.86084 - val_acc: 0.4849 -- iter: 2880/2880
--
```

Figure III.12 : Résultats d'apprentissage de VGGnet avec a=50,b=80,c=100

D'après ces résultats nous pouvons observer que la précision d'entraînement est en train d'augmenter et le la perte diminue avec l'augmentation de nombre d'epoch.

III.2.1.6 Etude comparative entre les trois architectures

Architecture	Couche de convolution	Couche de Max Pooling	Fully connected	Epoch 50		Temps d'exécution (s)	Epoch 80		Temps d'exécution (s)	Epoch 100		Temps d'exécution (s)
				Val-acc (%)	Val-loss		Val-acc(%)	Val-loss		Val-acc (%)	Val-loss	
Alexnet	5	3	3	0.49	11.60	5.554	0.49	0.69	5.478	0.46	12.24	5.641
Google-net	2	4	/	0.50	11.48	19.245	0.91	0.28	19.392	0.89	0.28	19.383
VGGnet	5	5	3	0.52	11.06	5637	0.50	11.38	35.603	0.48	11.86	38.696

Tableau III.1 :Etude comparative entre les trois architectures

III.2.1.7 Discussion du tableau III.1

Après l'analyse des résultats obtenus, nous faison les remarques suivantes en analysant le tableau III.1 comparant les résultats des architectures **Alexnet** , **VGGnet** et **Googlenet.**

●Avec l'augmentation du nombre d'époch nous observons que le la précision des modèles **Alexnet** et **VGGnet** est diminués et que la valeur de perte augment.

●Pour **VGGnet** on observe :

1. Longue durée d'entraînement **35.603 s**

2. Modèle lourd

3. Coût de calcul important

●L'architecture **Alexnet** donne une performance très faible à cause d'hyper-paramètres, dont la valeur est utilisée pour contrôler le processus d'apprentissage. Ce problème a été résolu par **VGGnet** en remplaçant les grands filtres de la taille d'un noyau (11 et 5 dans la première et la deuxième couche de convolution, respectivement) par plusieurs filtres de la taille d'un noyau 3×3 l'un après l'autre.

●Finalement nous obsevons que la précision de l'apprentissage augmente avec le nombre d'époques ,ceci reflète qu'à chaque époque le modèle apprend plus ou acquiert plus d'informations.De même, l'erreur d'apprentissage diminue avec le nombre d'époques.

-Nous constatons donc que dans notre travail, l'architecture **Googlenet** est meilleure que l'architecture **Alexnet** et l'architecture **VGGnet**. Donc, plus l'architecture est profonde, plus le résultat est meilleur et précis.

III.2.2:Le programme principal pour notre modèle

III.2.2.1 Importation des bibliothèques

Tout d'abord, nous importons les bibliothèques nécessaires :

```
import numpy as np
import matplotlib.pyplot as plt
import glob
import cv2
from keras.models import Model, Sequential
from keras.layers import Dense,Dropout, Flatten, Conv2D, MaxPooling2D
from keras.callbacks import EarlyStopping
import os
import seaborn as sns
```

Figure III.13 : Importation des bibliothèques notre modèle proposé

III.2.2.2 Division de la base de donnée

Nous divisons la base des donnés en sous-ensembles aléatoires d'entraînement et de test.

Voici quelques paramètres importants que nous devons utiliser:

-test_size: Nous l'avons divisé pour le train (0,8) et un ensemble de test (0,2) aléatoire.

-random: Pour nous obtenons toujours le même résultat la première fois que nous faisons le split. Ceci est utile pour obtenir des résultats reproductibles .

```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(dataset, labels_encoded, test_size = 0.20, random_state = 7)
x_train, x_test = x_train / 255.0, x_test / 255.0
```

Figure III.14 : Division de la base de données

III.2.2.3 Importation de la base de données

Nous générons une liste de tous les fichiers (et répertoires) dans un dossier.Pour l'utiliser, il suffit de passer le répertoire comme argument ,Pour suivre nous chargeons les fichiers d'exemple dans un seul répertoire et passons le chemin d'accès au dossier files dans l'argument de la fonction Glob :

- Pour tout renvoyer dans un répertoire,nous utilisons l'astérisque (*)

-nous utilisons resize() dans OpenCV pour redimensionner l'image vers le haut / bas à la taille dont nous avons besoin qui y elle de 200x200.

-finalement nous affichons le nombre d'étiquettes Oui/Non.

```
SIZE = 200

dataset = []
label = []

for directory_path in glob.glob ("/content/drive/MyDrive/PFE/base des donnes/*"):
    Label = directory_path.split("/")[-1]
    Label = Label.split(".")[0]
    #print(Label)
    for img_path in glob.glob(os.path.join(directory_path)):
        img = cv2.imread(img_path, cv2.IMREAD_COLOR)
        img = cv2.resize(img,(200,200))
        img = cv2.cvtColor(img, cv2.COLOR_RGB2BGR)
        dataset.append(img)
        label.append(Label)

print('The count of element: oui is ', label.count("oui"))
print('The count of element: non is ', label.count("non"))
```

```
The count of element: oui is  1849
The count of element: non is  1800
```

Figure III.15 : Pré-traitement des images (redimensionnement=200x200)

III.2.2.4 Importer un encodeur d'étiquettes

Nous utilisons l'encodage d'étiquettes pour faire référence à la conversion des étiquettes en une forme numérique afin de les convertir en une forme lisible . Les algorithmes d'apprentissage automatique peuvent alors décider d'une meilleure manière d'utiliser ces étiquettes. Il s'agit d'une étape importante du prétraitement pour l'ensemble des données structurées dans l'apprentissage supervisé , ces étapes :

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit(label)
labels_encoded = le.transform(label)
print(labels_encoded)
```

```
3649
3649
(3649, 200, 200, 3)
(3649,)
[1 1 1 ... 0 0 0]
```

Figure III.16 : Encodeur d'étiquettes

III.2.2.5 Implémentation du réseau

```
INPUT_SHAPE = (SIZE, SIZE, 3)
# Create an Instance of Early Stopping Callback
early_stopping_callback = EarlyStopping(monitor = 'val_accuracy', patience =15, mode ='max',restore_best_weights =True,verbose=1 )

model = Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', input_shape=INPUT_SHAPE))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D((2, 2)))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Flatten())
model.summary()  #pour afficher les diff layers dans notre modele, on peut supprimer cette ligne sans aucun probleme
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

Figure III.17 : Implémentation de réseau

III.2.2.6 Entraînement du réseau

Nous compilons le modèle avec Keras qui fournit une méthode compile() . L'argument que nous utilisons les méthodes compile() sont les suivant:

-Optimizer : nous optimisons les poids d'entrée en comparant la prédiction et la fonction de perte

-Metrics: nous utilisons la métrique pour évaluer la performance de notre modèle. Il est similaire à la fonction de perte.

```
#Do not use softmax for binary classification

model.compile(loss='binary_crossentropy',
              optimizer='rmsprop',              #also try adam
              metrics=['accuracy'])

history = model.fit(x_train,
                        y_train,
                        batch_size = 64,
                        callbacks = [early_stopping_callback],
                        epochs = 80,
                        validation_data=(x_test,y_test),
                        shuffle = False
                    )
```

Figure III.18 : Compilation et entraînement

-Nous exécutons notre algorithme d'apprentissage avec le **modèle.fit**

-Nous choisissons l'époque de modèle **epochs=80** avec la valeur de perte la plus faible, et nous utilisons Early Stopping qui est L'arrêt précoce consiste essentiellement à arrêter l'entraînement une fois que notre perte commence à augmenter (ou en d'autres termes, la précision de la validation commence à diminuer) l'arrêt d'entraînement est **epoch=22** .

III.2.2.6 Résultats obtenus pour le modèle

D'après cette résultats nous observons que la précision d'entraînement est en train d'augmenter avec le nombre d'époch ,tandis que la perte d'entraînement diminue avec le nombre d'époch.

a/-**Epoch=1/80**

```
Epoch 1/80
46/46 [==============================] - 202s 4s/step - loss: 0.6592 - accuracy: 0.8116 - val_loss: 0.2983 - val_accuracy: 0.9192
Epoch 2/80
```

b/-Epoch=12/80

```
Epoch 12/80
46/46 [==============================] - 193s 4s/step - loss: 0.1784 - accuracy: 0.9490 - val_loss: 0.2547 - val_accuracy: 0.9260
```

c/-**Epoch=22/80**

```
Epoch 21/80
22/46 [=============>................] - ETA: 1:38 - loss: 0.1459 - accuracy: 0.9517
```

Figure III.19: Résultats d'apprentissage du modèle avec a=1/80,b=12/80,c=22/80

III.2.2.7 Analyse et discussion de résultat

-Dans cette section, nous évaluons la performance de notre méthode proposée. La performance de classification de chaque epoch est évaluée par le taux de succès (%) qui est calculé par la division de nombre d'images correctement classés sur le nombre total d'images de test. **le taux de succès = 95%.**

III.2.2.8 Matrice de confusion

-Pour faire un résumé des résultats de prédictions sur la classification .Nous affichons deux matrices de confusion , Ces matrices permet de comprendre de quelle façon le modèle de classification est confus lorsqu'il effectue des prédictions.À partir des résultats escomptés et des prédictions, la matrice indique le nombre de prédictions correctes pour la classe OUI et la classe NON. nous comparons par cette matrice l'étiquette (figure III.20)

-Il y a l'indication correcte d'une prédiction positive comme » vraie positive » (true positive) qui est **341 images** et d'une prédiction négative comme » vraie négative » (true negative**) 350 images**, ou une prédiction positive incorrecte comme » fausse positive » (false positive)**26 images** et une prédiction négative incorrecte comme » fausse négative » (false negative)**13 images**.

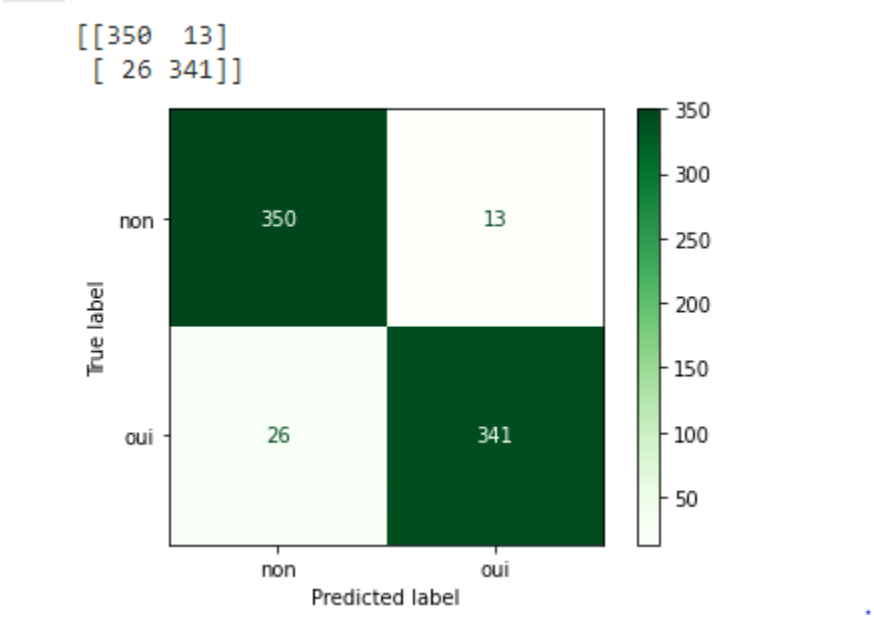


Figure III.20 : Matrice de confusion

-Pour faciliter la lecture des résultats et de les rendre mieux compréhensibles on a affiché d'autre matrice qui indique le pourcentage de ces images par rapport le nombre d'images de test .Nous concluons que les performances des modèles de classification sont très élevées.

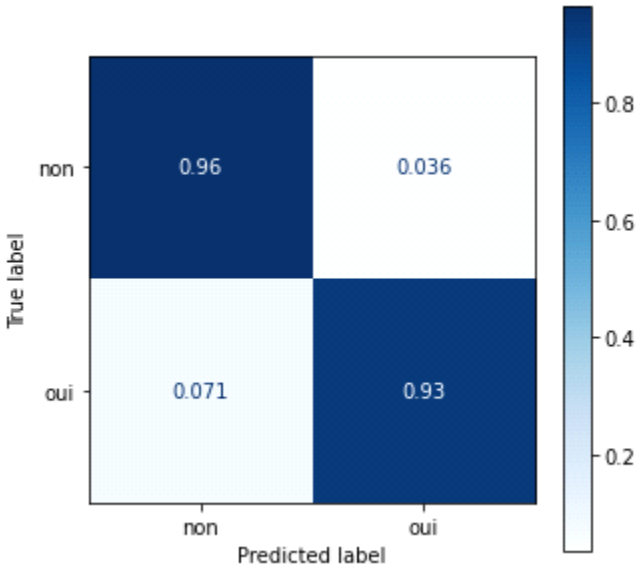


Figure III.21 Matrice du confusion de pourcentage

-La figure III.22 représentent l'évolution de l'erreur de classification (Malade/non Malade) pour 30 époques.

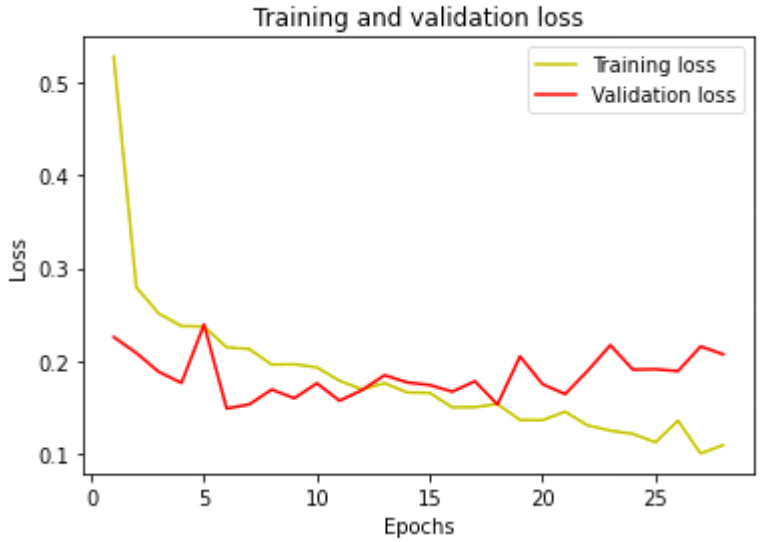


Figure III.22: Évolution de la courbe des pertes du modèle proposé

- Nous trouvons qu' il existe clairement une corrélation entre la perte d'entraînement et la perte de validation. Ils semblent tous deux réduire et rester à une valeur constante. Cela signifie que le modèle est bien formé et qu'il est tout aussi bon sur les données d'entraînement que sur les données.

-La figures III.23 représentent l'évolution de la précision de classification (Malade/non Malade) pour 30 époques.

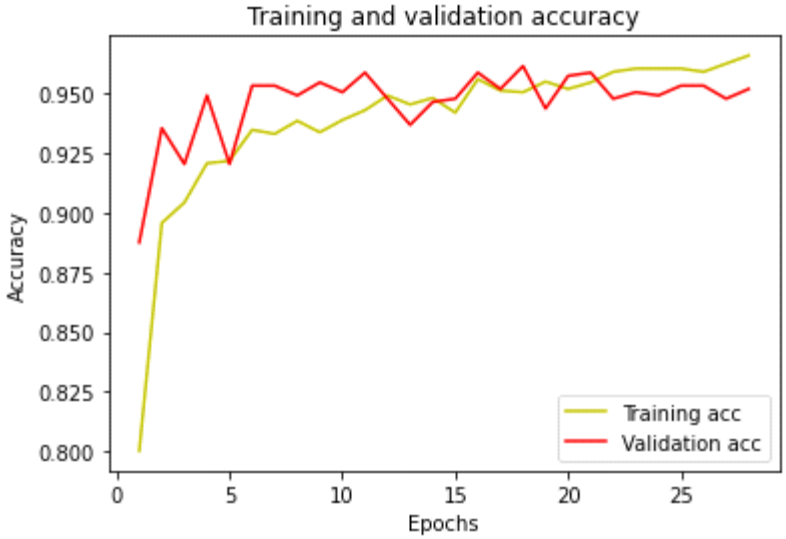


Figure III.23 : Évolution de la courbe des précision du modèle proposé

-Nous remarquons que la précision d'entraînement et de validation sont corrélées.

-Ces résultats montrent que les courbes de l'entraînement et de validation diminuent toutes les deux pour la perte (Loss) et elles augmentent pour la précision (Accuracy). Elles évoluent vers le bon sens et tendent vers de meilleurs résultats, alors les performances du modèle augmentent avec le temps jusqu'à la stabilité , ce qui signifie que le modèle s'améliore avec l'expérience .

III.2.3 Comparaison entre Googlenet (la meilleure architecture parmis les architectures précédentes)

			Epochs 80	
	Couche de convolutions	Couche de max pooling	Val-acc	Val-loss
Modèle googlenet	2	4	0.91	0.28
Modèle proposé	2	2	0.95	0.20

Tableau III.2 :Comparaison entre Googlenet et le modèle proposé

III.2.3.1 Discussion du tableau III.2

Nous comparons les résultats dans l'epoch 80 ou les modèles sont plus performants nous en déduisons que les meilleurs résultats sont donnés par le modèle que nous avons proposé, la précision était très proche entre les deux malgré les différents modèles .

III.3 La segmentation

●La Segmentation des images médicales joue un rôle majeur dans le traitement d'images et pour l'aide au diagnostic surtout dans le cas des maladies rétiniennes dont la RD.Dans ce travail nous présentons une technique de segmentation des images d'hémorragies à partir de l'image et son masque. Nous segmentons les images avec deux modèles U-Net 1 et U_net 2. Exemple d'image avec son mask:

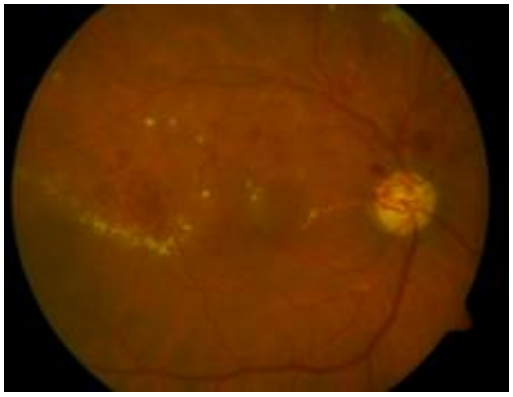


Figure III.24 : Image

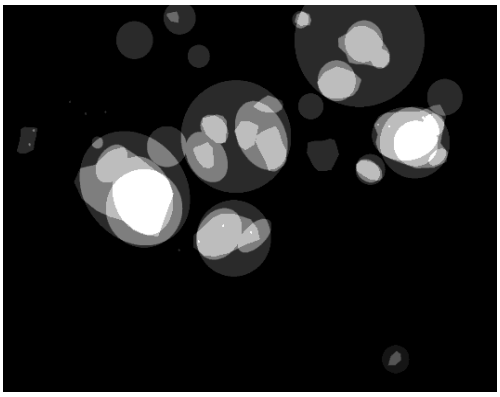


Figure III.25: Mask

III.3.1 Implémentation du masque U-Net 1 pour la détection

III.3.1.1 Bibliothèques et Importation de fonctions

-Nous importons la bibliothèque keras avec le package tensorflow pour préparer l'environnement et d'autres bibliothèques nécessaires .

```
#Step 1: Load libraries for the U-net Model
import numpy as np
import os
import skimage.io as io
import skimage.transform as trans
import numpy as np
from tensorflow.keras.models import *
from tensorflow.keras.layers import *
from tensorflow.keras.optimizers import *
from tensorflow.keras.callbacks import ModelCheckpoint, LearningRateScheduler
from tensorflow.keras import backend as keras
#from tensorflow import keras
import tensorflow as tf
```

Figure III.26 : Importation des bibliothèques pour le modèle U-net 1

III.3.1.2 Importation de la base de données

```
import os
!ls os.chdir('/content/diaretdb1_v_1_1/resources/')
```

Figure III.27: Importation de la base de données pour la segmentation

III.3.1.3 Division de la base de données

-Dans le but de la détection de la rétinopathie diabitique plus précisément la pathologie hemorrhage ,nous faisons une segmentation sur la base de données « diaretdb1_v_1_1 » de la plateforme « Kaggle », Cette base des données contient 89 images du fond d'œil et 89 masques de chaque image au format PNG et de taille 1500 x 1152.

III.3.1.4 Architecture de Modèl U_net 1

```
#Step : Define the U-net model with Depth=4
def unet(pretrained_weights = None,input_size = (256,256,1)):
    inputs = tf.keras.Input(shape=input_size)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(inputs)
    conv1 = BatchNormalization()(conv1)
    conv1 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv1)
    conv1 = BatchNormalization()(conv1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(conv1)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool1)
    conv2 = BatchNormalization()(conv2)
    conv2 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv2)
    conv2 = BatchNormalization()(conv2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(conv2)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool2)
    conv3 = BatchNormalization()(conv3)
    conv3 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv3)
    conv3 = BatchNormalization()(conv3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(conv3)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool3)
    conv4 = BatchNormalization()(conv4)
    conv4 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv4)
    conv4 = BatchNormalization()(conv4)
    drop4 = Dropout(0.5)(conv4, training=True)
    pool4 = MaxPooling2D(pool_size=(2, 2))(drop4)

    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(pool4)
    conv5 = BatchNormalization()(conv5)
    conv5 = Conv2D(1024, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv5)
    conv5 = BatchNormalization()(conv5)
```

```
    up6 = Conv2D(512, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(drop5))
    merge6 = concatenate([drop4,up6], axis = 3)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge6)
    conv6 = Conv2D(512, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv6)


    up7 = Conv2D(256, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv6))
    merge7 = concatenate([conv3,up7], axis = 3)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge7)
    conv7 = Conv2D(256, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv7)


    up8 = Conv2D(128, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv7))
    merge8 = concatenate([conv2,up8], axis = 3)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge8)
    conv8 = Conv2D(128, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv8)


    up9 = Conv2D(64, 2, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(UpSampling2D(size = (2,2))(conv8))
    merge9 = concatenate([conv1,up9], axis = 3)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(merge9)
    conv9 = Conv2D(64, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)
    conv9 = Conv2D(2, 3, activation = 'relu', padding = 'same', kernel_initializer = 'he_normal')(conv9)

    conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)

    model = tf.keras.Model(inputs = inputs, outputs = conv10)
```

```
conv10 = Conv2D(1, 1, activation = 'sigmoid')(conv9)

model = tf.keras.Model(inputs = inputs, outputs = conv10)

model.compile(optimizer = Adam(lr = 0.0001), loss = dice_coef_loss, metrics = dice_coef)

if(pretrained_weights):
  model=keras.models.load_model(pretrained_weights)

return model
```

Figure III.28 : Architecture de modèle U-net 1

III.3.1.5 Entraînement du modèle

```
opt = Adam(lr=learning_rate, beta_1=0.9, beta_2=0.999, epsilon=None, amsgrad=False)
model.compile(optimizer=opt, loss=bce_dice_loss, metrics=[iou, dice_coef])

callbacks = [EarlyStopping(monitor='val_loss', restore_best_weights=True, patience=15)]

history = model.fit(train_gen,
                    steps_per_epoch=len(df_train) / BATCH_SIZE,
                    epochs=EPOCHS,
                    callbacks=callbacks)
validation_data = val_gen,
validation_steps=len(df_val) / BATCH_SIZE
EarlyStopping(monitor='val_loss', restore_best_weights=True, patience=15)
```

Figure III.29: Entraînement de modèle U-net1

•Nous utilisons **« compile »** pour définit la fonction de perte ,l'optimiseur et les métriques.

•Nous utilisons l'algorithme d'optimisation **Adam** pour mettre à jour les poids de réseau itératifs qui sont basés sur les données d'entraînement .

•Nous utilisons **« loss »** fonction de perte pour la catégorisation d'une étiquette.

•Nous utilisons **« Metric »** pour évaluer les performances de ce modèle avec les paramètres suivants :

-IOU (Intersection-Over-Union) :métrique d'évaluation courante pour la segmentation sémantique des images.

-Dice-coeff(coefficient de similarité des dés) : est un indice de chevauchement spatial et une représente la proportion d'accord spécifique . La valeur d'un DSC va de :

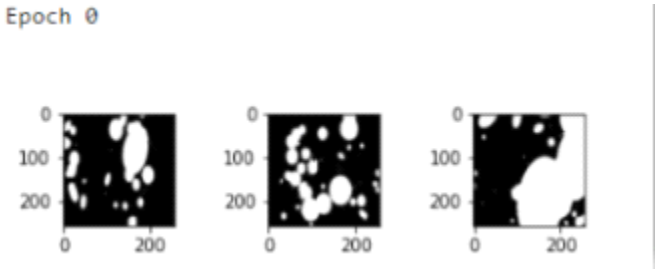
0: indiquant l'absence de chevauchement spatial entre deux ensembles de résultats de segmentation binaire,

à 1: indiquant un chevauchement complet.

III.3.1.6 Résultats obtenus pour le modèle

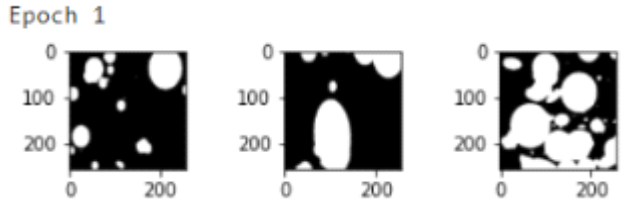
Nous appliquons le modèle sur nos images , nous avons obtenu les résultats suivants :

a/-Epoche=0

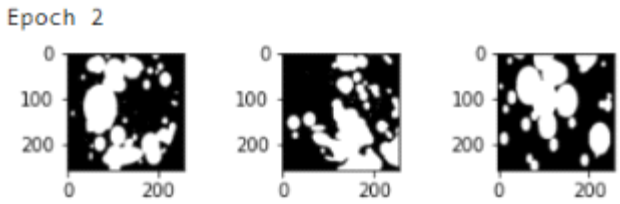


-Nous affichons des masques de certaines images de test pour quatre epochs et observons la performance de notre modèle.

b/-Epoche=1



c/-Epoche=2



d/-Epoche =3

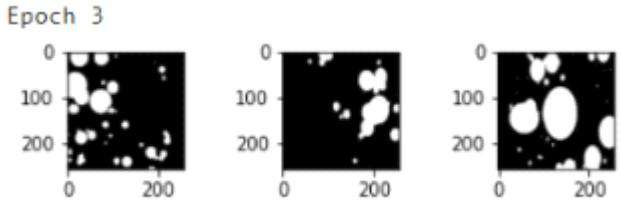


Figure III.30: Résultats pour a=0,b=1,c=2,d=3

-Cette méthode nous a donné les résultats satisfaisants en nous permettant d'extraire la pathologie en blanc et le reste en noir pour chaque image de test .Nous avons crée automatiquement le masque . La précision etait trés élevée qui est **0.965.**

-Afin de calculer la précision de la segmentation entre les masqus des images test et les masques de segmentation , nous avons calculé le coefficient de Dice. Le coefficient de Dice (le coefficient de similarité) : **Coeff de Dice 2VP / 2VP +FP+VN** Indique la corrélation positive entre deux images, il varie entre 0 et 1, où 1 signifie la plus grande similitude entre la prédiction et la vérité. **Coeff de Dice=0.99734** .

-Ainsi les résultats obtenus sont satisfaisants et encourageants, et peuvent être améliorés en utilisant une base de données plus grande.

III.3.2 Implémentation de l'architecture U-Net 2 pour la segmentation

Tout d'abord on va importer les bibliothèques nécessaire pour travailler sur le modèle Unet 2 :

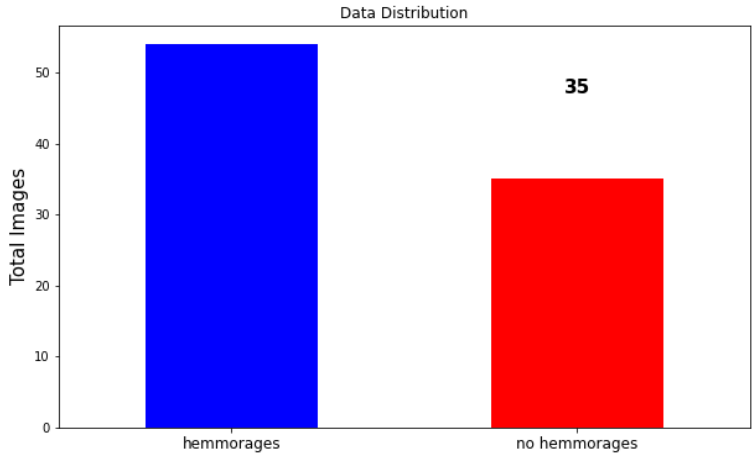
III.3.2.1 Importation des bibliothèques pour le modèle U-net 2

```
import os
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
from mpl_toolkits.axes_grid1 import ImageGrid
import cv2
from glob import glob
import tensorflow as tf
from tensorflow.keras import Input
from tensorflow.keras.models import Model, load_model, save_model
from tensorflow.keras.layers import Input, Activation, BatchNormalization, Dropout, Lambda, Conv2D
from tensorflow.keras.layers import Conv2DTranspose, MaxPooling2D, concatenate, AveragePooling2D, Dense, Flatten
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint, ReduceLROnPlateau
from tensorflow.keras import backend as K
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Figure III.31: Importation des bibliothèques de modèle U-net 2

III.3.2.2 Division de la base de données:

Cette figure représente le nombres d'images contenant une hémorragie et le nombre des images de rétinopathie saine.



FigureIII.32 : Représentation du nombres d'images

Les images et leurs masks :

Nouv affichons des exemples d'images et leur masques respectivement .

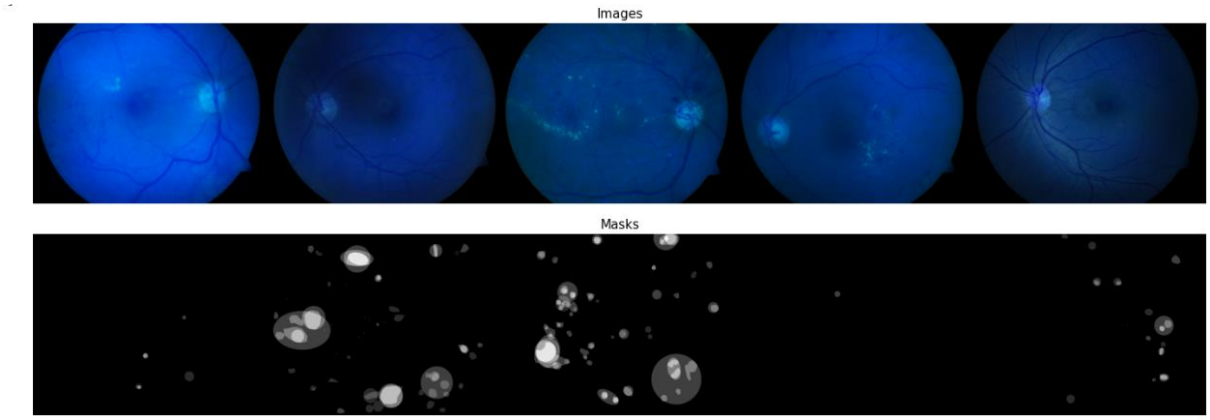


Figure III.33 : Représentation des images et leurs mask

III.3.2.3 Architecture de model U-net 2:

```
def unet(input_size=(256,256,3)):
    inputs = Input(input_size)

    conv1 = Conv2D(64, (3, 3), padding='same')(inputs)
    bn1 = Activation('relu')(conv1)
    conv1 = Conv2D(64, (3, 3), padding='same')(bn1)
    bn1 = BatchNormalization(axis=3)(conv1)
    bn1 = Activation('relu')(bn1)
    pool1 = MaxPooling2D(pool_size=(2, 2))(bn1)

    conv2 = Conv2D(128, (3, 3), padding='same')(pool1)
    bn2 = Activation('relu')(conv2)
    conv2 = Conv2D(128, (3, 3), padding='same')(bn2)
    bn2 = BatchNormalization(axis=3)(conv2)
    bn2 = Activation('relu')(bn2)
    pool2 = MaxPooling2D(pool_size=(2, 2))(bn2)

    conv3 = Conv2D(256, (3, 3), padding='same')(pool2)
    bn3 = Activation('relu')(conv3)
    conv3 = Conv2D(256, (3, 3), padding='same')(bn3)
    bn3 = BatchNormalization(axis=3)(conv3)
    bn3 = Activation('relu')(bn3)
    pool3 = MaxPooling2D(pool_size=(2, 2))(bn3)

       conv4 = Conv2D(512, (3, 3), padding='same')(pool3)
       bn4 = Activation('relu')(conv4)
       conv4 = Conv2D(512, (3, 3), padding='same')(bn4)
       bn4 = BatchNormalization(axis=3)(conv4)
       bn4 = Activation('relu')(bn4)
       pool4 = MaxPooling2D(pool_size=(2, 2))(bn4)

       conv5 = Conv2D(1024, (3, 3), padding='same')(pool4)
       bn5 = Activation('relu')(conv5)
       conv5 = Conv2D(1024, (3, 3), padding='same')(bn5)
       bn5 = BatchNormalization(axis=3)(conv5)
       bn5 = Activation('relu')(bn5)

       up6 = concatenate([Conv2DTranspose(512, (2, 2), strides=(2, 2), padding='same')(bn5), conv4], axis=3)
       conv6 = Conv2D(512, (3, 3), padding='same')(up6)
       bn6 = Activation('relu')(conv6)
       conv6 = Conv2D(512, (3, 3), padding='same')(bn6)
       bn6 = BatchNormalization(axis=3)(conv6)
       bn6 = Activation('relu')(bn6)
```

```
up7 = concatenate([Conv2DTranspose(256, (2, 2), strides=(2, 2), padding='same')(bn6), conv3], axis=3)
conv7 = Conv2D(256, (3, 3), padding='same')(up7)
bn7 = Activation('relu')(conv7)
conv7 = Conv2D(256, (3, 3), padding='same')(bn7)
bn7 = BatchNormalization(axis=3)(conv7)
bn7 = Activation('relu')(bn7)

up8 = concatenate([Conv2DTranspose(128, (2, 2), strides=(2, 2), padding='same')(bn7), conv2], axis=3)
conv8 = Conv2D(128, (3, 3), padding='same')(up8)
bn8 = Activation('relu')(conv8)
conv8 = Conv2D(128, (3, 3), padding='same')(bn8)
bn8 = BatchNormalization(axis=3)(conv8)
bn8 = Activation('relu')(bn8)

up9 = concatenate([Conv2DTranspose(64, (2, 2), strides=(2, 2), padding='same')(bn8), conv1], axis=3)
conv9 = Conv2D(64, (3, 3), padding='same')(up9)
bn9 = Activation('relu')(conv9)
conv9 = Conv2D(64, (3, 3), padding='same')(bn9)
bn9 = BatchNormalization(axis=3)(conv9)
bn9 = Activation('relu')(bn9)

conv10 = Conv2D(1, (1, 1), activation='sigmoid')(bn9)

return Model(inputs=[inputs], outputs=[conv10])
```

Figure III.34 : Architecture du modèle U-net 2

III.3.2.4 Entrainement du modèle

```
opt = Adam(lr=learning_rate, beta_1=0.9, beta_2=0.999, epsilon=None, amsgrad=False)
model.compile(optimizer=opt, loss=bce_dice_loss, metrics=[iou, dice_coef])

callbacks = [EarlyStopping(monitor='val_loss', restore_best_weights=True, patience=15)]

history = model.fit(train_gen,
                    steps_per_epoch=len(df_train) / BATCH_SIZE,
                    epochs=EPOCHS,
                    callbacks=callbacks)
validation_data = val_gen,
validation_steps=len(df_val) / BATCH_SIZE
EarlyStopping(monitor='val_loss', restore_best_weights=True, patience=15)
```

Figure III.35 : Compilation et entraînement de U-net 2

III.3.2.5 Résultats de notre entrainement

```
Found 14 validated image filenames.
Found 14 validated image filenames.
/usr/local/lib/python3.7/dist-packages/tensorflow/python/util/dispatch.py:1082:
  return dispatch_target(*args, **kwargs)
      6/Unknown - 106s 18s/step - loss: 1.6744 - iou: 0.0065 - dice_coef: 0.0129
```

Figure III.36 : Résultats de l'entraînement

D'après les résultats obtenus nous observons que les valeurs du IOU et Dice coeff sont trés faibles. (IOU et Dice coeff << 1)

Résultats du test d'image avec la prédiction du masque

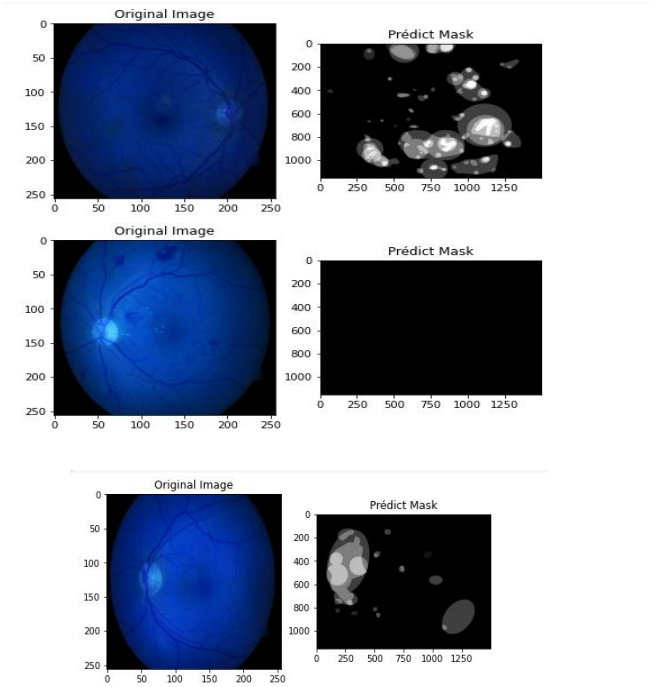


Figure III.37 : Résultats de test

III.3.2.6 Discussion

U-Net 2 produit plus d'erreurs de classification ainsi qu'une plus petite proportion de prédictions vraies dans l'ensemble de données du test . Nous trouvons que la prédiction du masque ne détecte pas la zone d'hémorragie sur les images malades, contrairement au image saine dans plusieurs prédictions .Cela signifie que ce modèle a une mauvaise performance.

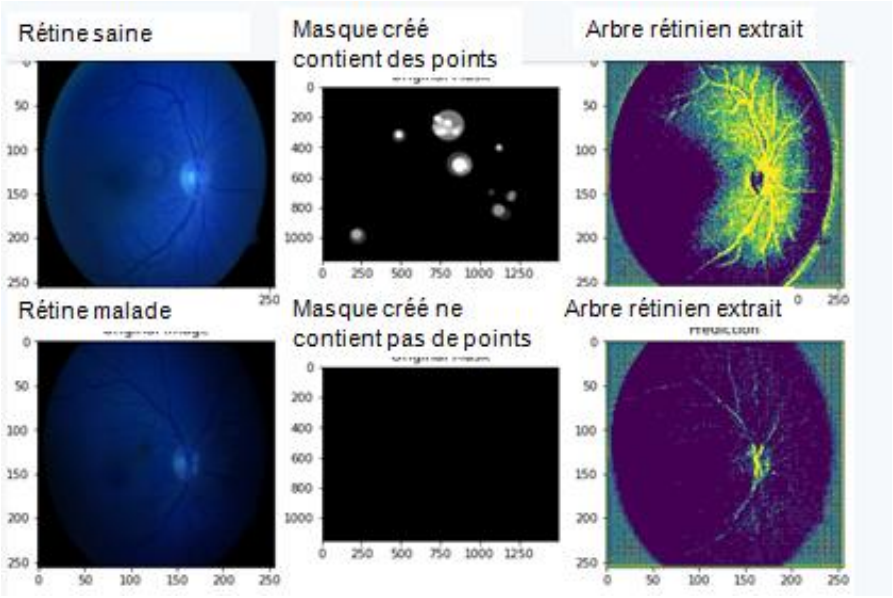


Figure III.38 : a- image d'origine saine, b-masque mauvaise détection de la pathologie , c- bonne extraction de l'arbre rétinien

Par comparaison la base de données que nous avons utilisé et les résultats obtenus nous observons que le test affiche la prédiction du masque qui détecte l'arbre rétinien mais ce n'est pas notre but de l'entrainement , donc nous constatons que ce modèle ne répond pas à notre but

et aux résultats que nous voulions. Nous constatons que la cause de ces mauvais résultats

est probablement la base de données très petite

III.4 Conclusion:

-Dans le contexte de l'imagerie médicale ,le but d'un système de classification et segmentation automatique des images pathologique sont en effet multiples: aide au diagnostic, contrôler l'action d'une thérapie, développement des systèmes médicaux...etc

-Dans ce chapitre, nous avons présenté les différentes architectures développées pour la classification et segmentation de la rétinopathie diabétique et faire une comparaison entre ces architectures.

Conclusion générale

La rétinopathie diabétique regroupe l'ensemble des maladies de la rétine dues à la détérioration desvaisseaux rétiniens par le diabète.

Dans ce projet, nous avons traité une problématique majeure qui vise à détecter laprésence d'unerétinopathie et détecterla zone atteinte sur des clichés rétiniens (Fond d'oeil).

L'apprentissage en profondeur est actuellement utilisé pour l'analyse des images médicales de la rétineet l'extraction de caractéristiques . Nous avons discuté des notions fondamentales des réseaux deneurones convolutionnels en particulier et introduit ces réseaux de neurones en présentant les différentstypes de couches utilisées dans la classification dans le but d'élaborer un système capable d'affecterune classe automatiquement à une image.

Pour la classification, nous avons implémenté des architectures de deep CNN, prédéfinies, Alexnet, Vggnet, et Googlenet, puis nous les avons entrainés sur la base de données de 3500 images étiquetées, de rétines saines et malades. Nous les avons testés sur des images d'une autre base de données non étiquetées..Nous avons fait une étude comparative entre les différentesarchitectures utilisées. Seul Googlenet a donné des résultats acceptables Il s'avère que la sélection de fonctionnalités d'algorithmes d'apprentissage enprofondeur tels que CNN affecte considérablement les performances du système.

Nous avons proposé une nouvelle architecture, de réseau CNN, validée après plusieurs essais de configurations des différents paramètres. Nous l'avons entrainée et obtenu de très bons résultats de classification avec une excellente précision.

Dans le but de la détection de pathologie sur la RD nous avons implémenté deux différents modèles :

- U-net1 qui a permis de générer des masques de zones rétiniennes malades, avec une très bonne précision

- et U-net2, qui a extrait l'arbre rétinien, à notre surprise, alors que, vu que l'étiquetage de la base de données consistait en des masques contenant les zones malades, nous aurions dû obtenir des masques. Comme perspective, nous proposons donc de comprendre cette architecture, et de l'exploiter pour une détection optimale des rétinopathie diabétique.

	Références
[1]	« Définition, symptômes du diabète » Dr Anne-Christine Della Valle https://sante.journaldesfemmes.fr/fiches-maladies/2499922-diabete-type-1-2-definition-insipide-causes-symptomes-traitement-taux-normal-journee-mondiale/ « consulté le 25 mai 2022»
[2]	« Les types de diabète » https://www.diabete.qc.ca/fr/comprendre-le-diabete/tout-sur-le-diabete/types-de-diabete/le-diabete-de-type-1/ « consulté le 29mai 2022 »
[3]	« Les types de diabète » https://www.diabete.qc.ca/fr/comprendre-le-diabete/tout-sur-le-diabete/types-de-diabete/le-diabete-de-type-1/ « consulté le 29mai 2022 »
[4]	« Les symptômes de la rétinopathie diabétique » https://www.barraquer.com/fr/pathologie/retinopathie-diabetique « consulté le 30mai 2022 »
[5]	« Zones particulières de la Rétine » https://www.barraquer.com/fr/pathologie/oedeme-maculaire#:~:text=La%20macula%20est%20la%20zone%20centrale%20de%20la,de%20reconna%C3%AEtre%2C%20par%20exemple%2C%20les%20visages%20des%20gens « consulté le 30mai 2022 »
[6]	« Intelligence artificielle » https://fr.wikipedia.org/wiki/Intelligence_artificielle#D%C3%A9finition_2 « consulté le 22mai 2022 »
[7]	ALOUACHE Rayhana, CHIA Radhia, ÉVALUATION DE LA PERFOR_MANCE D'UN CAPTEUR LOGICIEL EN UTILISANT L'APPRENTISSAGE EN PROFONDEUR, Mémoire de Master, UNIVERSITÉ MOHAMED BOUDIAF - M'SILA,2019
[8]	« Les avantages de Deep Learning » https ://mc.ai/32-advantages-and-disadvantages-of-deep-learning/ « consulté le 22 mai 2022 »
[9]	« Le machine learning et le deep learning » https://www.talend.com/fr/resources/ai-vs-machine-learning-vs-deep-learning/:https « consulté le 22 mai 2022 »

[10]	« Apprentissage non supervisé » https://waytolearnx.com/2018/11/difference-entre-apprentissage-supervise-et-non-supervise.html#:~:text=L%E2%80%99apprentissage%20supervis%C3%A9%20est%20la%20technique%20permettant%20d%E2%80%99accomplir%20une,les%20caract%C3%A9ristiques%20de%20la%20population%20d%E2%80%99entr%C3%A9e%20par%20lui-m%C3%AAme « consulté le 22 mai2022 »
[11]	« Imagerie médiacle »JORG VANDENHIRTZ | 04-01-2019 https://www.cognex.com/fr-fr/blogs/deep-learning/why-medical-imaging-is-the-next-frontier-for-deep-learning « consulté le 5 juin 2022 »
[12]	« La Relation entre deep learning et la rétinopathie diabétique »Rétinopathie diabétique : l'intelligence artificielle au service d'une détection précoce | Le Quotidien du Médecin (lequotidiendumedecin.fr) « CONSULTE LE 14 JUIN 2022 »
[13]	A Multimodal Memes Classification: A Survey and Open Research Issues - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/A-generic-CNN-Architecture_fig1_344294512 « consulté le 14 Jun 2022 »
[14]	« Figure couche du convolutionMobeen-ur-Rehman , Sharzil Haris Khan, Zeeshan Abbas , S. Danois Rizvi [https://d3i71xaburhd42.cloudfront.net/9b744cc61a8d215ece2734217705f3ef00841022/3-Figure1-1.png] « consulté le 1 juin »
[15]	« couche de pooling »Gary B 25 juin 2020 [https://datascientest.com/convolutional-neural-network] « consulté le 1 Jun 2022 »
[16]	« La fonction Relu » https://deep-neural-networks.com/cnn/« consulté le 5 juin »
[17]	« Maxpoolinhg » Gray B 25 juin 2020 https://datascientest.com/convolutional-neural-network « consulté le 1 Jun 2022 »
[18]	« VGGnet »https://viso.ai/deep-learning/vgg-very-deep-convolutional-networks/ « consulté le 20 mai 2022 »
[19]	« Figure d'architecture VGGnet »https://www.datacorner.fr/vgg-transfer-learning/ « consulté le 20 mai 2022 »
[20]	« Définition de Alexnet »https://paperswithcode.com/method/alexnet « consulté le 22 mai 2022 »
[21]	« Architectire Alexnet »https://medium.com/mlearning-ai/alexnet-and-image-classification-8cd8511548b4 « consulté le 22 mai 2022 »

[22]	« Figure d'architecture Alexnet »ImageNet Classification with Deep Convolutional Neural Networks by Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton, « consulté le 22 mai2022 »
[23]	« Architecture Googlenet »https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765 « consulté le 18 mai 2022 »
[24]	« Architecture Googlenet »https://www.geeksforgeeks.org/understanding-googlenet-model-cnn-architecture/ « consulté le 18 mai 2022 »
[25]	Transfer learning based classification of optical coherence tomography images with diabetic macular edema and dry age-related macular degeneration - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/GoogLeNet-with-filter-dimensions-and-illustration-of-inception-layer-The-LeNet_fig2_312075562 [consumté le 14 Jun, 2022]
[26]	« Segmentation sémantique » Innodata.com « consulté le 10 juin 2022 »
[27]	« U-net »Anubhav Tewari Architecture U-Net (opengenus.org) « consulté le 10 juin 2022 »
[28]	« Anatomie Macroscopique de la rétine »https://fr.wikipedia.org/wiki/%C5%92il_humain#Anatomie_et_physiologie_de_l'%C5%93il_humain « consulté le 30 mai 2022 »
[29]	Daha Boubaker Medileh Mounir, Un miroir intelligent interactif basé sur Rasp_berry Pi, Mémoire de Master, UNIVERSITÉ ECHAHID 24-09-2018
[30]	« Figure du l'allure du fonction relu »Découvrez les différentes couches d'un CNN - Classez et segmentez des données visuelles - OpenClassrooms « consulté le 1 juin 2022 »
[31]	« Fond d'œil » https://www.sante- centre.fr/portail_v1/gallery_files/site/133/3687/3741.pdf. « consulté le 10 juin »
[32]	Kauppi, T.Kalesnykiene, V.Kamarainen, J.-K.Lensu, L.Sorri, I. Raninen A. Voutilainen R., Uusitalo, H., Kälviäinen, H., Pietilä, J., *DIARETDB1 diabetic retinopathy database and evaluation protocol,* In Proc of the 11th Conf. on Medical Image Understanding and Analysis (Aberystwyth, Pays de Galles, 2007). DIARETDB1 - BASE DE DONNÉES STANDARD SUR LA RÉTINOPATHIE DIABÉTIQUE (lut.fi) « consulté le 14 juin 2022 »
[33]	Content Based Image Retrieval Approach using Deep Learning - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/ALexNet-Architecture-21_fig1_337787638 [consulté Jul, 2022]
[34]	« Les paramètres de Alexnet »Architecture d'Alexnet | Introduction à l'architecture Alexnet | Haut-parleur de données (datapeaker.com) « consulté le 50juin 2022 »
[35]	« Figure d'architecture Googlenet »Comparing Local Descriptors and Bags of Visual

	Words to Deep Convolutional Neural Networks for Plant Recognition - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/The-illustration-of-the-GoogleNet-architecture-Szegedy-et-al-2015-All-convolutional_fig1_314119821 « consulté le 20 juin 2022 »
[36]	« Architecture Googlenet » Lac de Richmond 2020 https://towardsdatascience.com/deep-learning-googlenet-explained-de8861c82765 « consulté le 20 juin 2022 »
[37]	« VGGnet »Prabin Népal 2020 https://medium.com/analytics-vidhya/vggnet-architecture-explained-e5c7318aa5b6 « consulté le 5 juin 2022 »
[38]	Automatic localization of casting defects with convolutional neural networks - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Fig-A1-The-standard-VGG-16-network-architecture-as-proposed-in-32-Note-that-only_fig3_32251243 [consulté le 16 Jun, 2022]
[39]	« La fonction softmax »https://vitalflux.com/what-softmax-function-why-needed-machine-learning/ « consulté le 5 juin 2022 »
[40]	« Utilisation du fonction softmax »https://vitalflux.com/what-softmax-function-why-needed-machine-learning/ Ajitesh Kumar « consulté le 5 juin 2022 »
[41]	« La Représentation de la Fonction Softmax. »https://www.inside-machinelearning.com/fonction-dactivation-comment-ca-marche-une-explication-simple/ « consulté le 5 juin 2022 »
[42]	« Représentation de l'aplanissement »SuperDataScience Aug 2018 Team https://www.superdatascience.com/blogs/convolutional-neural-networks-cnn-step-3-flattening « consulté le 8 juin 2022 »
[43]	« La couche dense » Palais sharma octobre 2020 https://machinelearningknowledge.ai/keras-dense-layer-explained-for-beginners/ « consulté le 15 juin 2022 »
[44]	« Optimizer Adam » https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam « consulté le 10 juin 2022 »
[45]	« Dropout »https://inside-machinelearning.com/le-dropout-cest-quoi-deep-learning-explication-rapide/#:~:text=Le%20Dropout%20est%20une%20technique,un%20mod%C3%A8le%20de%20Deep%20Learning.&text=Le%20choix%20des%20neurones%20%C3%A0%20d%C3%A9sactiver%20est%20al%C3%A9atoire. « Consulté le 10 juin 2022 »
[46]	« La précision » Ahmed Fawzy Gad 2020 https://blog.paperspace.com/deep-learning-metrics-precision-recall-accuracy/ « consulté le 10 juin 2022 »
[47]	« Tensorflow »https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html « consulté le 28 mai 2022
[48]	*Martin Heller is a contributing editor and reviewer for InfoWorld. Formerly a web and Windows programming consultant, he developed databases, software, and websites from 1986 to 2010. More recently, he has served as VP of technology and education at Alpha Software and chairman and CEO at Tubifi.*Martin Heller . *(1986 to 2010)s a contributing editor and reviewer for InfoWorld*

[49]	« Open cv »https://blog.desdelinux.net/fr/opencv-una-biblioteca-para-el-reconocimiento-de-objetos-en-imagenes-y-camaras/?utm_source=dlvr.it&utm_medium=facebook « consulté le 20 juin 2022 »
[50]	« Matplotlib »https://datascientest.com/matplotlib-tout-savoir « consulté le 22 juin 2022 »
[51]	« Tqdm »Rahul Shah 22-may_2021 https://www.analyticsvidhya.com/blog/2021/05/how-to-use-progress-bars-in-python/ « consulté le 20 juin 2022 »
[52]	« Jupyter »https://jupyter.org/ « consulté le 14 juin 2022 »