N°
Ordre...../F.S.S.A/UAMOB/2022

REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE AKLI MOHAND OULHADJ-BOUIRA

Faculté des Sciences et des Sciences Appliquées
Département : **Génie Electrique**

Mémoire de fin d'étude

Présenté par :

NOURI Abdelmalek
MANSOURI Yanis

En vue de l'obtention du diplôme de **Master** en **:**

Filière : Télécommunications
Spécialité : Système des Télécommunications

Thème :

Segmentation de lésion dérmoscopiques avec les réseaux de neurones convolutifs (Deep learning)

Devant le jury composé de :

A. KABLA	MCB	UAMOB	Président
R. KASMI	MCA	UAMOB	Encadreur
S. MEDJEDOUB	MAA	UAMOB	Examinateur

Année Universitaire 2021/2022

Remerciement

Nous remercions notre encadreur Mr R. Kasmi pour sa patiencetout au long de ce travail, qui a consacré tous son temps à suivre de près comme de loin l'évolution de ce projet, à orienter les différentes étapes et à pallier toutes les difficultés auxquelles nous avons eu pendant la réalisation de ce travail.

Nous tenons à remercier le chef de département du Génie Electriqueet tous les enseignant qui ont contribué par leurs savoirs et leurs encouragements au long de notre parcours

Dédicaces

Je dédie ce modeste travail à toutes celles et à vous ceux qui m'ont aidé et soutenu durant ce long parcours

A mon frère, à toute ma famille, à mes amis, à mon entourage et à tout l'équipe pédagogique qui m'a donné la science et le savoir

Une dédicace spéciale à mon encadrant qui m'a informé, guidé et orienté

J'apporte toute ma gratitude et ma reconnaissance à mes parents, qui tout au long de ma carrière d'étudiant, m'ont doté de leur soutien, leur aide logistique, leur soutien et surtout leur amour

Merci d'être toujours là pour moi.

Yanis

Dédicaces

A mes chers parents, pour tous leurs sacrifices, leur amour, leur soutien et leurs prières tout au long de mes études,

A toute ma famille pour leur soutien tout au long de mon parcours universitaire,

Abdelmalek

Sommaire

Liste des acronymes et abréviations

ACS : American Cancer Society.

CNN : Réseau neurone convolutif.

GVF : Gradient Vectorial Flux.

GAC : Géodisic Actif Contour.

CV : Corps Vertébreaux.

DL : Deep Learning.

ML :Maching Learning.

TL: Apprentissage par Transfert.

DRL: Apprentissage par Renforcement Profond.

RNN : Réseaux de Neurones Récurrents.

ReLu : Rectified Linear Unit.

FC : Fully Connected.

VGG : Visual Geometry Group.

ISIC: Inetrnational Skin Imaging Collaboration.

ISDIS: Inetrnational Society for Digital Imaging of the Skin.

API: Application Program Interface.

RVB :Rouge Vert Bleu .

GPU: Graphics Processing Unit.

TP: True Positive.

TN: True Negative.

FP: False Positive.

FN: False Negative.

Liste de tableaux

Liste de figures

Introduction Générale

Le mélanome malin, appelé ici « mélanome », est un type de cancer qui prend naissance dans les cellules pigmentaires (mélanocytes) de la peau. Il provoque globalement plus de décès que tout autre type de cancer de la peau. Selon l'American Cancer Society (ACS), environ 99 780 (57 180 hommes et 42 600 femmes) nouveaux cas de mélanomes ont été diagnostiqués en 2022 (1). Pour la même année, ACS a également estimé à environ 7 650 décès (5 080 hommes et 2 570 femmes) (1). L'incidence du mélanome augmente chaque année et beaucoup de patients seront guéris si le mélanome est détecté à un stade le plus précoce. Un certain nombre d'études qui utilisent déférentes techniques sont menées dans le monde pour la détection précoce du mélanome.

Comme l'équipement et les ressources humaines professionnelles ne sont généralement pas disponibles pour chaque patient à tester, un système automatisé de diagnostic assisté par ordinateur est nécessaire pour déterminer les lésions cutanées telles que le mélanome. Ces systèmes bien entrainés assurent l'interprétation correcte des images dermoscopiques et améliorent l'objectivité de l'interprétation visuelle des images dermoscopiques.

Les techniques d'apprentissage en profondeur ont récemment été étendues à la segmentation des lésions cutanées. Dans le but d'améliorer l'évolutivité de l'expertise diagnostique, des réseaux de neurones à convolution profonde CNN ont été utilisés pour localiser les cancers de la peau dans des images avec une haute précision (2).

Pour la segmentation des lésions cutanées dermoscopiques, plusieurs algorithmes ont été développés. Les images dermoscopiques peuvent être classées plus précisément si les lésions cutanées sont segmentées. La segmentation des lésions est une tâche difficile en raison de leur faible contraste à leur état précoce.

Dans notre projet, nous sommes intéressés aux méthodes d’apprentissage profond pour segmenter les lésions de la peau notamment en étudiant et implémentant la méthode U-Net. Afin d’évaluer ses performances en termes de précision de segmentation, les résultats obtenus par U-Net sont comparés aux résultats obtenus avec d’autres méthodes.

Le travail présenté dans ce mémoire est divisé en trois chapitres comme suit : dans le premier chapitre nous avons évoqué les déférentes techniques de segmentation d’images. Le deuxième chapitre présente les méthodes d’apprentissage profond. Le troisième chapitre explique l’implémentation de la segmentation des lésions de la peau et les résultats obtenus en les comparant avec d’autre méthodes. Le mémoire se termine par une conclusion générale.

Chapitre I : Généralités sur la segmentation d'images

1.1 Introduction

La segmentation est une des étapes fondamentale de l'analyse d'images qui conditionne la qualité des mesures effectuées ultérieurement. C'est généralement une première étape d'un traitement plus complexe comme la reconnaissance de formes. De ce fait, ce chapitre sera consacré aux concepts de la segmentation sémantique où le terme « segmentation d'images » sera défini dans un premier temps, ses différents types seront ensuite cités et différentes applications seront évoquées. Une conclusion clôtura ce chapitre.

1.2 Définition de la segmentation d'images

La segmentation permet de cerner les formes des objets sur lesquels doit porter l'analyse, de délimiter les régions d'intérêts et de les extraire du fond et de sélectionner ainsi les zones d'intérêts. D'une façon formelle, la segmentation est un traitement de bas niveau qui consiste à créer une partition de l'image *I* en sous-ensembles *Ri*, appelés régions telles que l'ensemble des régions recouvre toute l'image. Une région est un ensemble de pixels ayant des propriétés communes qui les différencient des pixels des régions voisines (3).

Figure 1 Segmentation d'une image (par région)

1.3 Les types de segmentation d'images

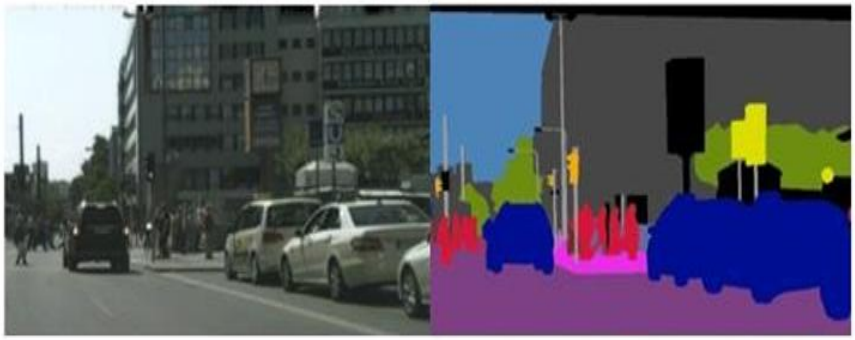
Les types de segmentation varient selon le besoin, qu'elle soit capable de distinguer les différents objets d'image ou bien de distinguer les différentes instances selon l'application.

Il existe trois principaux types de segmentation : la segmentation d'instance, la segmentation

sémantique et la segmentation panoptique. La figure 4 montre les trois types de segmentation.

1.3.1 Segmentation sémantique

La segmentation sémantique est une classification aux niveaux pixels où chaque pixel sera classé dans une classe qui correspond à un type de segment. Par exemple, dans l'image de la figure 2 où toutes les voitures sont considérées comme appartiennes à une classe sans se soucier de type de voiture ou de leurs couleurs.



(a) (b)

Figure 2 : Segmentation sémantique. (a) image originale, (b) segmentation sémantique

1.3.2 Segmentation d'instance

La segmentation par instance est un type de segmentation de l'image. Elle est essentielle pour la détection des objets, qui contrairement à la segmentation sémantique, elle permet d'identifier chaque objet dans l'image. S'il y a trois voitures dans une image, la segmentation sémantique classera chaque voiture comme une classe comme le montre la figure 3.

(a) (b)

Figure 3 : Segmentation d'instance. (a) image originale, (b) segmentation sémantique

1.3.3 Segmentation panoptique

La segmentation panoptique est une combinaison entre la segmentation sémantique et la segmentation d'instance. Les pixels reçoivent une étiquette de classe et que toutes les instances

d'objet soient segmentées de manière unique (4). Tous les pixels de l'image seront classés comme appartenant à une catégorie, mais également il sera identifié à quelle instance de cette classe ils appartiennent, d'une manière plus simple chaque pixel sera associé à deux valeurs : sa classe et le numéro d'instance, contrairement à la segmentation d'instances la segmentation panoptique s'intéresse à toutes les régions, pas seulement les régions dont on estime avoir des instances. La figure 4 (d) illustre les différents types de segmentation.

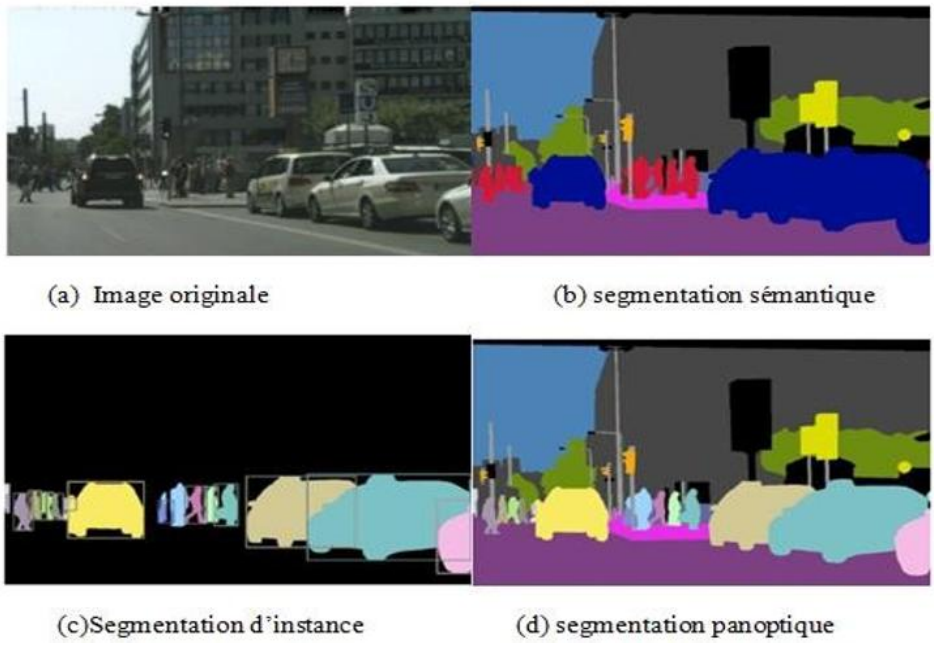


Figure 4:Types de segmentations

1.4 Applications de la segmentation dans le monde réel

Plusieurs domaines reposent sur les segmentations sémantiques telles que l'imagerie médicale, l'analyse des images satellitaires, la perception de conduite autonome ainsi que la détection des objets. Cette technique est utilisée dans les cabines d'essayage en ligne, le maquillage virtuel et la reconnaissance d'images de vente au détail. Ces applications nécessitent souvent l'utilisation d'images de segmentation intelligentes. La figure suivante montre quelques images médicales segmentées.

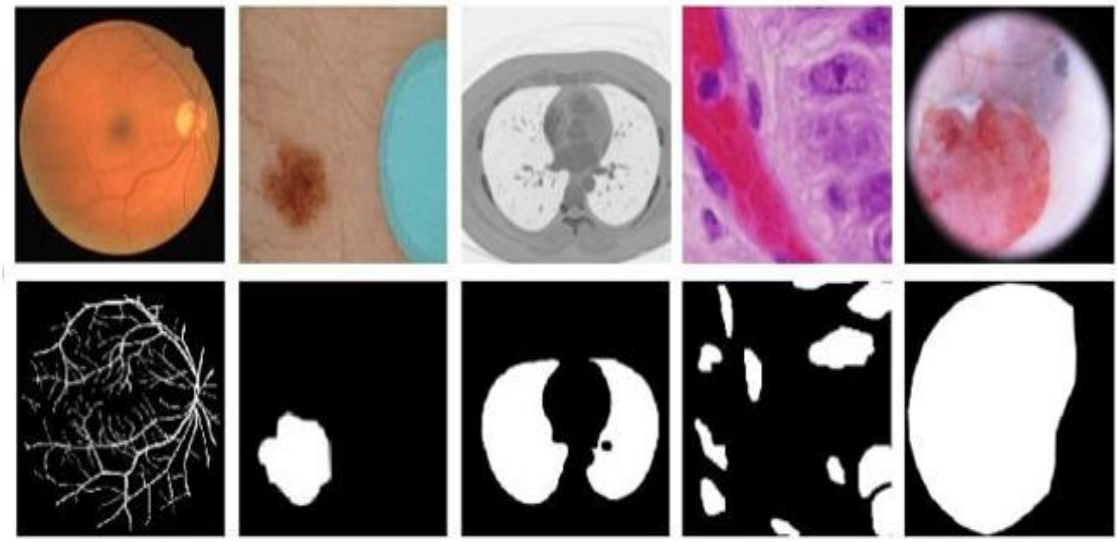


Figure 5 : Segmentation sémantique de différentes images médicales. (Haut) images originale (bas) la segmentation de l'image

1.5 Les méthodes de la segmentation

Plusieurs algorithmes de segmentation ont été proposés pour délimiter les frontières des lésions dans les images. Nous limiterons à citer quelques méthodes de segmentation : détection de contour, contour actif, seuillage, segmentation basée sur les régions, et celles basées sur l'apprentissage profond.

1.5.1 Méthodes basées sur la détection des contours :

Les changements d'intensité des pixels dans l'image peuvent être détectés avec l'amplitude du gradient. Selon Les techniques Sonka et Al (5), sont des exemples courants de détecteurs de bord qui conduisent à la segmentation d'image. Ces détecteurs de contours présentent plusieurs problèmes tel que : la détection d'un bord où il n'y a pas de frontière réelle, la non-détection d'un bord dans une image mal contrastée, la grande sensibilité au bruit de l'image**.**

1.5.2 Méthodes reposent sur les techniques des contours actifs

Les contours actifs (6), représentent une alternative intéressante pour segmenter les lésions de forme complexe et de bordure irrégulière**.** Plusieurs algorithmes basés sur les contours actifs ont été utilisés pour segmenter les images des lésions cutanées (7) (8). Cela consiste, à déplacer les courbes initiales vers les bords des objets d'intérêt à travers une déformation appropriée. Un modèle déformable peut être classé comme paramétrique ou

géométrique, selon la technique utilisée pour suivre le mouvement de la courbe. Les modèles paramétriques incluent les modèles traditionnels de contours actifs : Snakes (6)**.** Typiquement, dans ces modèles, la déformation de la courbe est guidée par l'estimation d'énergie véhiculée par l'évolution de la courbe, dans lesquelles une énergie interne détermine le niveau de lissage par la définition de l'élasticité et de la rigidité de la courbe. En d'autres termes, il contrôle le degré de retrait ou d'expansion de la courbe du modèle afin d'éviter les déformations excessives. L'énergie externe est également incluse dans les modèles, qui a pour fonction de conduire la courbe à la frontière. Cette énergie peut être définie par l'utilisateur ou par un processus automatique. Cependant, ces modèles ont certaines limites : très sensibles aux paramètres d'initialisations de la courbe, l'initialisation de la courbe se fait au niveau des bords de l'objet d'intérêt, ces modèles estiment mal les critères d'arrêt de l'évolution de la courbe au niveau des contours de l'objet d'intérêt, dépendent généralement du niveau dégradation de l'image. Cela peut entrainer une mauvaise localisation des contours lorsque la valeur du dégradé n'est pas suffisamment élevée**.**

Afin de pallier aux limitations du modèle Snake, Le flux vectoriel de gradient (GVF) (9) et le Géodisic Actif Contour (GAC) (10), ont été introduit.

1.5.3 Méthodes reposent sur des techniques de seuillage

La technique de seuillage implique la sélection d'une ou plusieurs valeurs de seuil pour séparer les régions d'image selon les valeurs du niveau de gris. Parmi les différentes techniques proposées dans la littérature pour définir la valeur de seuil, on peut citer la méthode d'Otsu (11). Cette méthode est basée sur l'histogramme normalisé de l'image.

L'histogramme construit, afin de définir la valeur de seuil optimale K, qui sépare les pixels de l'image en deux classes homogènes (c_0, c_1). Otsu a pour but de minimiser les variances intra-classes et de maximiser les variances interclasses.

1.5.4 Méthodes basées sur les régions

Les méthodes basées sur les régions (12) utilisent l'algorithme de croissance de région et les opérations de fractionnement et de fusion.

1.5.5 Méthodes basées sur l'apprentissage profond

Les approches précédemment citées présentent des limitations majeures qui ne favorisent pas leur déploiement en clinique pour effectuer des tâches critiques. Si une approche est performante, elle nécessite un temps non négligeable d'exécution ou une interaction humaine. Si l'approche est

automatique et ne nécessite pas un mécanisme d'apprentissage, elle est très sensible au bruit dans l'image.

Le succès des méthodes d'apprentissage profond à réaliser la classification d'images a étendu leur utilisation afin de résoudre des tâches plus complexes dont la segmentation sémantique en l'interprétant comme un problème soit de régression ou de discrimination. La localisation est traitée comme un problème de régression résolu par un réseau de neurones profond. Ce dernier est entraîné à calculer la distance entre un voxel et le centre de tous les CVs (corps vertébraux) de la même façon que cela a été fait par Chen et al (13). La segmentation se fait via le recalage d'un modèle statistique. Contrairement à C.hen et al (13), les résultats ont montré que l'utilisation des réseaux de neurones a permis une plus grande robustesse à détecter les CVs sans se restreindre à la tranche médiane du volume. Toutefois, les résultats qualitatifs dévoilent une sur-segmentation où certaines parties des CVs sont considérées comme étant l'arrière-plan de l'image.

1.6 Démoscopie

La dermoscopie est une technique utilisée par les dermatologues pour diagnostiquer les lésions de la peau. La technique nécessite l'utilisation d'un outil appelé dermatoscope (ou dermoscope), composé d'une lentille grossissante et d'une lumière polarisée qui permet de mieux visualiser à l'intérieur du derme (14). Le dermatoscope peut être relié à un ordinateur, et enregistrer les lésions.

Figure 6: (a) Dermoscope. (b) Image dermoscopique des tumeurs de la peau (b)

1.6.1 Lésions malignes et Bénignes

Le terme « tumeur » désignait autrefois toute augmentation de volume localisé déformant un organe ou une partie du corps. Elle réunissait des lésions différentes.

1.6.2 Lésions Bégnines

Les lésions bénignes ne sont pas cancéreuses, c'est-à-dire qu'elles ont un développement généralement limité et n'envahissent pas les organes voisins. Les lésions bénignes ne mettent pas la vie en danger.

1.6.3 Lésions malignes

Contrairement aux tumeurs bénignes, les tumeurs malignes aboutissent à la mort du patient s'elle n'est pas traitée. Le tableau 1 montre les différences entre les lésions malignes et bénignes.

Tableau 1 :différences entre tumeur (malignes/bénignes)

Tumeurs bénignes	**Tumeurs malignes**
Contours réguliers	Contours irréguliers
Couleur homogène	Multicolores
Symétrique	Non Symétrique
Taille fixe	Se développe

1.7 L'utilité de la segmentation dans l'identification du cancer de la peau

En dermatologie, le diagnostic repose le plus souvent sur la délimitation des frontières des lésions pigmentées.

Il est donc important d'effectuer une segmentation précise pour délimiter efficacement les bords des lésions. Cela signifie que l'image doit être segmentée en deux régions comme lésion et peau normale.

Considérer la tumeur comme un seul objectif dans l'image et employer une méthode dédiée complètement à cette tâche. La segmentation aide à détecter et à diagnostiquer le cancer de la peau à un stade précoce. Les expériences effectuées dans montrent qu'en générale (15), une classification correcte dépend fortement la précision de segmentation. En effet, une lésion mal détectée conduira à un diagnostic mal établi (16)**.** Donc, la segmentation doit être faite avec le plus grand soin et avec précision afin que la frontière de la lésion soit estimée. La segmentation d'images a donc un rôle essentiel dans l'efficacité de la détection du cancer de la peau (17)**.** Des études antérieures ont montré que les méthodes de calcul sur la segmentation d'image peuvent fournir des résultats appropriés pour l'identification des lésions cancéreuses sur les images (18). Souvent, les images analysées sont prétraitées pour l'amélioration de l'image et la suppression des artefacts, de sorte que des segmentations plus robustes peuvent être atteintes**.**

1.8 Les causes du cancer de la peau

Les facteurs de risques les plus communs pour le cancer de la peau sont les suivants (19) :

- Exposition à la lumière ultraviolette

- Antécédents de cancer de la peau
- Exposition à des rayonnements ionisants

1.9 Conclusion

Ce chapitre a mis l'accent sur les concepts de la segmentation tout en montrant son importante place dans plusieurs applications. La réalisation de cette tâche nécessite des outils comme le Deep Learning et en particulier sur les réseaux neuronaux convolutifs (CNN). De ce fait, le chapitre suivant se portera sur ces axes.

Chapitre II : Deep learning

2.1 Introduction

Au cours des dernières années, le paradigme informatique d'apprentissage profond, en anglais Deep Learning (DL), qui est un domaine de l'apprentissage automatique ou Maching Learning (ML) (20). L'apprentissage profond est largement utilisé et a montré son efficacité sur plusieurs tâches cognitives complexes (21). L'un des avantages du DL est sa capacité d'apprendre à partir d'énormes quantités de données.

Ce chapitre, souligne l'importance du DL, présente les types de techniques et de réseaux DL. Il présente ensuite les réseaux de neurones convolutifs (CNN) qui sont le type de réseau DL le plus utilisé.

2.2 Approche Deep Learning

Récemment, l'apprentissage automatique est devenu très répandu dans la recherche et a été intégré dans une variété d'applications, y compris l'exploration de texte (22), la détection de spam (23), la recommandation vidéo (24) et la segmentation des d'images médicales (25).

Le Deep Learning fait partie du domaine de recherche du Machine Learning qui il est même un domaine inclus dans l'intelligence artificiel comme illustrée sur la figure 7. Le DL ne nécessite aucune règle conçue par l'homme pour fonctionner ; il utilise plutôt une grande quantité de données pour l'apprentissage.

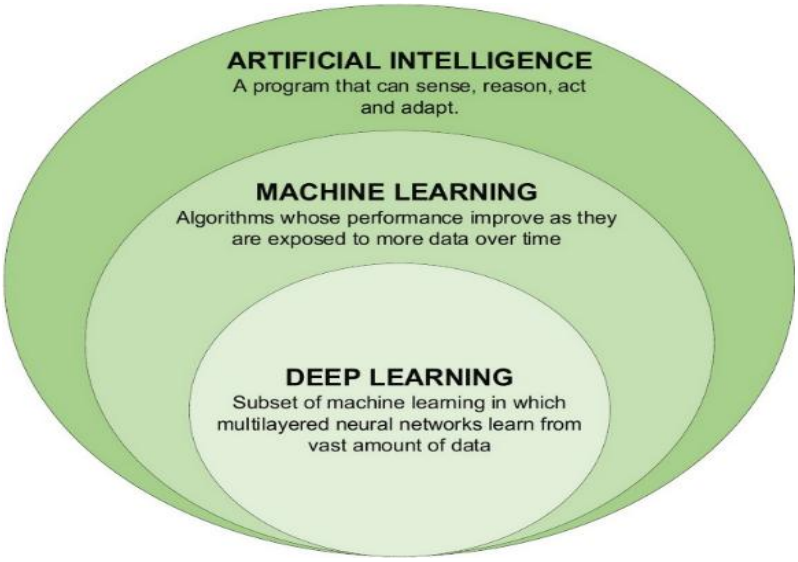


Figure 7: Domaine de L'IA et ses sous domaines.

Le DL est conçu à l'aide de nombreuses couches de neurones artificiels, dont chacune extrait et interprète des données qui leur ont été transmises (26) (27). Le DL a la capacité d'automatiser

l'apprentissage d'ensembles de caractéristiques pour plusieurs tâches (26) (28). Il traite directement l'image brute. Cependant, la plupart des approches d'apprentissage en profondeur ont permis un redimensionnement de l'image en raison de nombre de calcule à exécuter. Alors que certaines techniques ont donné une normalisation de l'intensité et une amélioration du contraste. En conséquence, le DL a une précision de classification plus élevée car il peut éviter les erreurs associées à un vecteur de caractéristiques erronées ou à une segmentation imprécise (29). Les approches basées sur DL ont modifié l'orientation de la recherche des techniques de traitement d'image traditionnelles pour l'ingénierie des caractéristiques à la conception d'architecture de réseaux pour obtenir des résultats optimaux. Les réseaux DL ont généralement plusieurs couches cachées, ce qui signifie que davantage d'opérations mathématiques sont effectuées par rapport aux approches basées sur ML et que les modèles nécessitent donc plus de calculs.

2.3 Différence entre la segmentation classique et la segmentation par Deep learning

Comme il a été démonté par l'expérience, le souillage est l'une des techniques les plus appropriées pour la segmentation d'images (30) mais cette technique reste incapable pour les images médicales compliquées. Les images médicales ont une distribution d'image complexe et une intensité similaire qui rendent la segmentation une tache inappropriée pour le souillage.

Pour surpasser ce problème, des techniques comme les réseaux de neurones peuvent être utilisées. La segmentation non linéaire des bordures des réseaux de neurones aide mieux pour la segmentation des images médicales. Plus précisément, l'architecture profonde de réseaux de neurones se comporte plus intelligemment avec l'apprentissage adaptatif, le parallélisme et la robustesse face au bruit. La disposition spatiale des unités de réseaux de neurones convolutives CNN sont des caractéristiques principales conçues pour fonctionner avec des images. L'exigence de connaissances préalables ou d'efforts humains supplémentaires pour l'extraction de caractéristiques n'est pas requise pour la segmentation à l'aide de CNN, car ce NN fonctionne à la fois pour l'extraction de caractéristiques et la prédiction de manière intégrée (31)

2.4 Contexte de Deep Learning

L'intelligence artificielle est utile dans de nombreuses situations et est rivalise avec l'expertise humaine dans certains cas. Le DL peut être une solution aux problèmes suivants :

1. Cas où les experts humains ne sont pas disponibles.
2. Cas où les humains sont incapables d'expliquer les décisions prises en utilisant leur expertise (compréhension du langage, décisions médicales et reconnaissance de la parole).

3. Cas où la solution du problème est mise à jour au fil du temps (prévision de prix, préférence de stock, prévision météo et suivi).
4. Cas où les solutions nécessitent une adaptation en fonction des cas particuliers (personnalisation, biométrie).
5. Cas où la taille du problème est extrêmement importante et dépasse nos capacités de raisonnement (analyse des sentiments, correspondance des publicités, classement des pages Web de calcul).

2.5 Caractéristiques de Deep Learning

Le Deep Learning est caractérisé (26) par :

1. Approche d'apprentissage général : étant donné que DL a la capacité de fonctionner dans presque tous les domaines d'application, il est parfois appelé apprentissage universel.
2. Robustesse : les modèles de DL ont la capacité de traiter les erreurs survenues au moment de l'exécution et également, ils peuvent traiter les entrées et les paramètres erronés. Une segmentation d'image robuste sera celle qui a la capacité de traiter des images bruitées et dont les résultats de segmentation contiennent le moins de bruit possible.
3. Généralisation : différents types de données ou différentes applications peuvent utiliser la même technique DL, une approche fréquemment appelée apprentissage par transfert (TL), utile dans les problèmes où les données sont insuffisantes.

2.6 Classification des approches DL

Les techniques DL sont classés en quatre grandes catégories : supervisées, non supervisées, partiellement supervisées (semi-supervisées) et L'apprentissage par renforcement profond (28).

2.6.1 Apprentissage supervisé en profondeur

Cette technique traite des données étiquetées. Le réseau estime le résultat y^=f(x_t), puis calcule l'erreur par rapport à la sortie souhaitée. Ensuite, les poids du réseau sont mis à jour à plusieurs reprises pour obtenir une estimation proche ou égale au résultat souhaité (6). L'apprentissage supervisé nécessite une énorme quantité de données d'entrainement pour classer les données de test, ce qui est un processus rentable mais consume du temps.

2.6.2 Apprentissage profond non supervisé

L'apprentissage non supervisé ne nécessite aucune donnée étiquetée, qui regroupe les données en fonction de la similitude des points de données en utilisant soit l'approche de regroupement, soit

l'approche du maximum de vraisemblance. Principal inconvénient de cette approche, elle ne peut pas regrouper avec précision des données inconnues.

2.6.3 Apprentissage profond semi-supervisé

L'apprentissage est considéré comme semi-supervisé lorsque l'ensemble de données est partiellement étiqueté (10).

2.6.4 Apprentissage par renforcement profond DRL

L'apprentissage par renforcement fonctionne sur l'interaction avec l'environnement, tandis que l'apprentissage supervisé fonctionne sur des exemples de données fournis. Cette technique a été développée en 2013 avec Google Deep Mind (32). Cette technique désigne l'ensemble des méthodes qui permettent à un agent d'apprendre à choisir quelle action prendre, et ceci de manière autonome. Dans un environnement donné, il apprend en recevant des récompenses ou des pénalités en fonction de ses actions. Au travers de son expérience, l'agent cherche à trouver la stratégie décisionnelle optimale qui puisse lui permettre de maximiser les récompenses accumulées au cours du temps (33). L'apprentissage par renforcement a comme but d'associer les observations reçues à des retours évaluatifs fournis par l'environnement et non pas les labels.

2.7 Types de réseaux DL

Des propositions des architectures DL sont en constante évolution, dans cette section les plus connus sont abordés. Il s'agit notamment des Réseaux de Neurones Récurrents (RNN) et les Réseaux de Neurones Convolutifs (CNN).

2.7.1 Réseaux de neurones récurrents

Le RNN utilise des données séquentielles dans le réseau et répète la même tâche pour tous les éléments et peut prendre en compte les états passés des neurones. Ils sont adaptés aux applications qui traitent les séquences temporelles (34). Pour une séquence d'entrée donnée, un diagramme RNN est illustré à la figure 8.

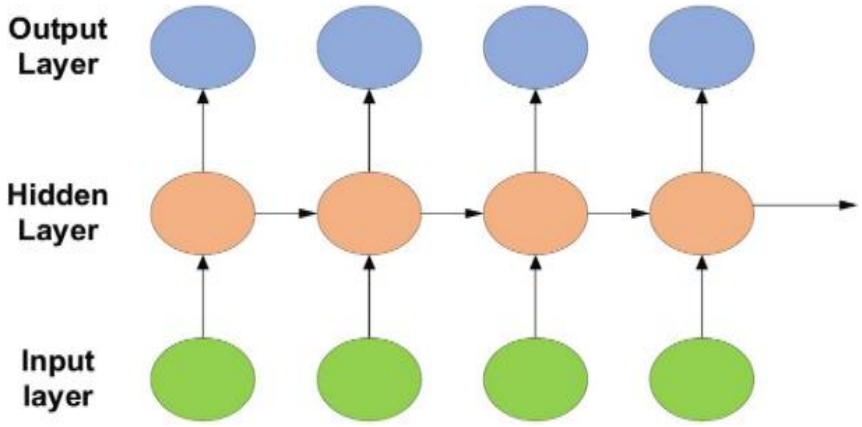


Figure 8: Diagramme RNN déplié typique

2.7.2 Réseaux de neurones convolutifs

Le CNN (Convulutional Neural Network) est l'algorithme de DL le plus utilisé (35). Le principal avantage de CNN est qu'il identifie automatiquement les caractéristiques pertinentes sans aucune supervision humaine (36). Les CNN ont été largement appliqués dans une gamme de domaines différents, y compris la vision par ordinateur (37), le traitement de la parole (38), la reconnaissance faciale (18) etc. Goodfellow et al. (39) ont identifié trois avantages clés du CNN : l'interaction clairsemée, le partage de paramètres et la représentation équivariante.

L'interaction clairsemée ou les poids clairsemés sont implémentés en utilisant des noyaux ou un détecteur de caractéristiques plus petits que l'image d'entrée. Si nous avons une image d'entrée de la taille 256 par 256, il devient difficile de détecter les bords de l'image qui peuvent n'occuper qu'un plus petit sous-ensemble de pixels dans l'image. Si nous utilisons des détecteurs de caractéristiques plus petits, nous pouvons facilement identifier les bords en nous concentrant sur l'identification des caractéristiques locales (39).

Le partage de paramètres est utilisé pour contrôler le nombre de paramètres ou de pondérations utilisés dans CNN. Dans les réseaux de neurones traditionnels, chaque poids est utilisé exactement une fois, mais dans CNN, nous supposons que si le détecteur de caractéristiques est utile pour calculer une position spatiale, il peut être utilisé pour calculer une position spatiale différente. Comme nous partageons des paramètres à travers le CNN, cela réduit le nombre de paramètres à apprendre et réduit également les besoins de calcul (39).

Représentation équivariante signifie que la détection d'objet est invariante aux changements d'éclairage, au changement de position, mais que la représentation interne est équivariante à ces changements (39).

L'architecture CNN se compose de nombreuses couches et chaque couche se compose principalement de trois opérations ou sous-couches : convolution, suivi d'une fonction d'activation ReLu (Rectified Linear Unit) et d'un sous-échantillonnage de regroupement (Max Pooling), tandis que les couches de sorties sont des couches complètement connectées FC (Fully connected). Un exemple d'architecture CNN pour la classification d'images est illustré sur la figure 9.

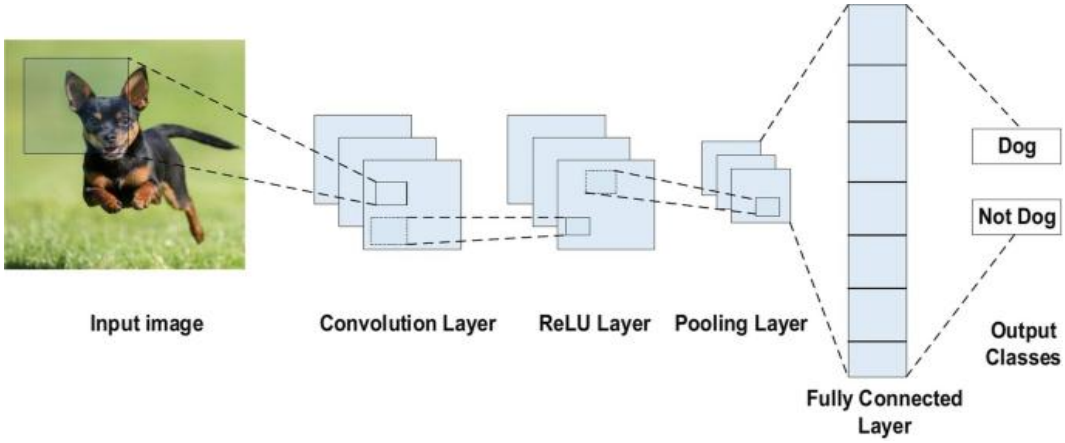


Figure 9:Exemple d'architecture CNN pour la classification d'images

Chaque couche de l'architecture CNN, y compris sa fonction, est décrite en détail en se basant sur la revue de Couchot et al. (40).

2.7.2.1 Couche de convolution

Cette couche est la première couche utilisée pour extraire les différentes caractéristiques des images d'entrée. Dans cette couche, l'opération mathématique de convolution est effectuée entre l'image d'entrée et un filtre d'une taille MxN, on l'appelle noyau ou masque de convolution (41).

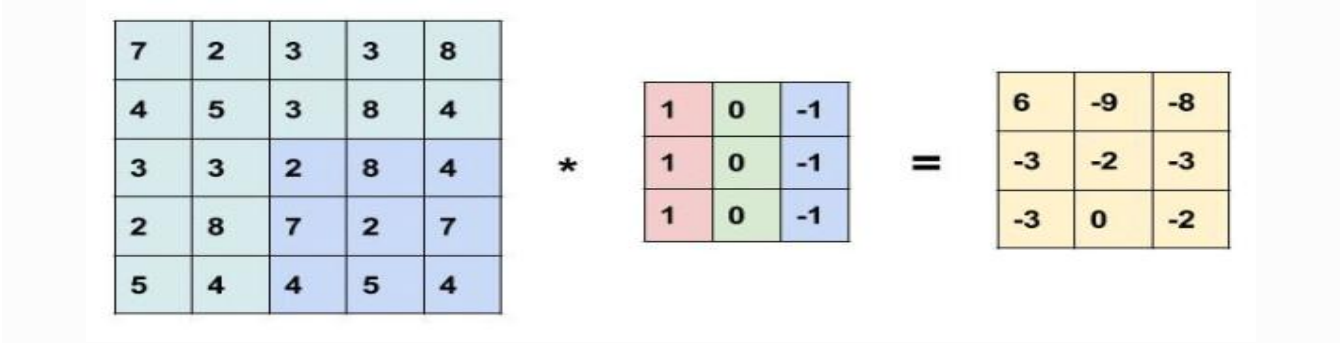


Figure 10 : Example de calcul de convolution 17

Formellement, le produit de convolution est une opération entre deux images en niveau de gris f et g, notée f∗g définie par :

$$(f * g)(x, y) = \sum_i \sum_j f(i, j) g(x - i, y - j) \qquad (1)$$

La sortie est appelée la carte des fonctionnalités qui nous donne des informations sur l'image telles que les pièces et les bords. Plus tard, cette carte de caractéristiques est transmise à d'autres couches pour apprendre plusieurs autres caractéristiques de l'image d'entrée.

2.7.2.2 Couche de regroupement : Max Pooling

Dans la plupart des cas, une couche convolutive est suivie d'une couche de regroupement. L'objectif principal de cette couche est de réduire la taille de la carte d'entités convoluées afin de réduire les coûts de calcul. Selon la méthode utilisée, il existe plusieurs types d'opérations de regroupement comme Max Pooling, le plus grand élément est extrait de la carte de caractéristiques. Average Pooling calcule la moyenne des éléments dans une section d'image de taille prédéfinie.

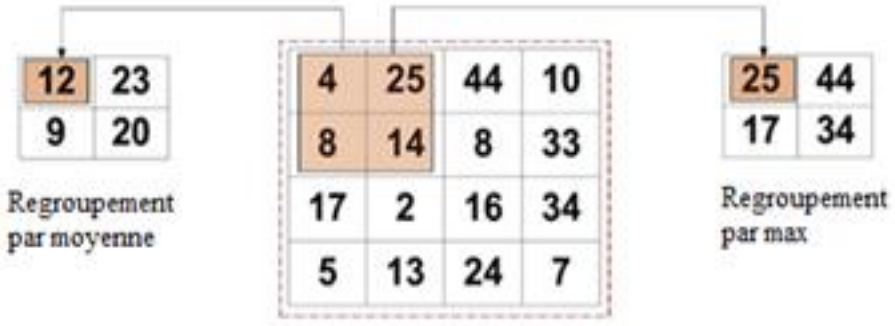


Figure 11 Types d'opérations de regroupement

2.7.2.3 Fonctions d'activation

La fonction d'activation est une fonction mathématique appliquée à la sortie d'un neurone, comme son nom l'indique, elle permet d'activer, de faire passer, bloquer ou modifier l'information sous certaine condition gérée par une fonction.

La fonction d'activation ajoute la non-linéarité au réseau. Il existe plusieurs fonctions d'activation couramment utilisées telles que les fonctions ReLU, Softmax, TanH et Sigmoid. Chacune de ces usages spécifiques. Pour un modèle CNN de classification binaire, les fonctions sigmoïdes et softmax sont préférées et pour une classification multi-classes, nous utilisons généralement softmax.

- Sigmoïde : Est une fonction d'activation qui permet de limiter la sortie d'un neurone entre 0 et 1. La courbe (figure 12) de la fonction sigmoïde est représentée mathématiquement par Eq. 2.

$$f(x)_{sigm} = \frac{1}{1+e^{-x}} \qquad (2)$$

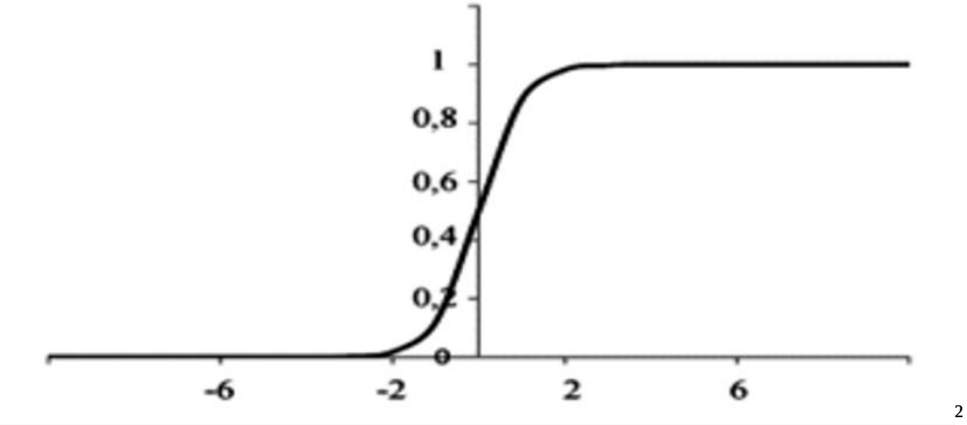


Figure 12 Fonction Sigmoïde

• Tanh: Est une fonction d'activation qui permet de limiter la sortie d'un neurone entre -1 et 1. La courbe (figure 13) de la fonction Tanh est représentée mathématiquement par Eq 3.

$$f(x)_{tanh} = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}} \qquad (3)$$

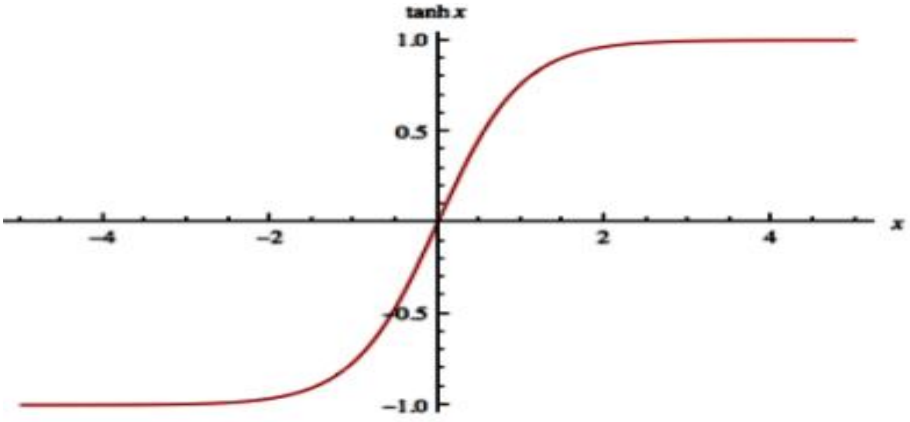


Figure 13 Fonction Tanh

- ReLU (Rectified Linear Unit): La fonction la plus couramment utilisée dans le contexte CNN. Il convertit les valeurs entières de l'entrée en nombres positifs. Une charge de calcul plus faible est le

principal avantage de ReLU par rapport aux autres. Sa représentation mathématique est dans l'Eq. 4.

$$f(x)_{ReLU} = max(0, x) \qquad (4)$$

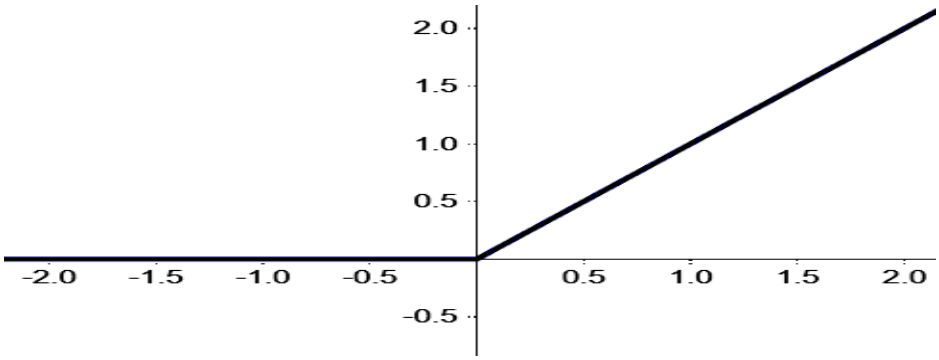


Figure 14 Fonction ReLU

2.7.2.4 Couche entièrement connectée

La couche entièrement connectée est similaire à la façon dont les neurones sont disposés dans un réseau de neurones traditionnel. Par conséquent, chaque nœud dans une couche entièrement connectée est directement connecté à chaque nœud à la fois dans la couche précédente et dans la couche suivante, comme illustré à la figure 15 (40).

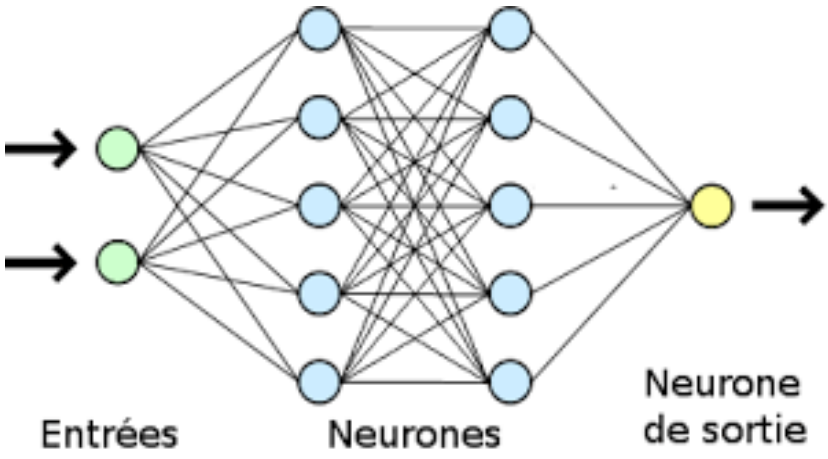


Figure 15 Couche entièrement connectée.

2.8 Quelques Architectures CNN

L'architecture CNN pour la segmentation utilise des modèles d'encodeur et de décodeur. Les codeurs sont utilisés pour coder l'entrée dans une représentation qui peut être envoyée via le réseau,

puis les décodeurs sont utilisés pour décoder la représentation en retour. Les codeurs peuvent être des réseaux de neurones convolutifs et les décodeurs peuvent être basés sur les réseaux de neurones dé-convolutifs ou transposés dans le but de créer une carte de segmentation. La figure 16 représente l'architecture CNN de base pour la segmentation d'images (36):

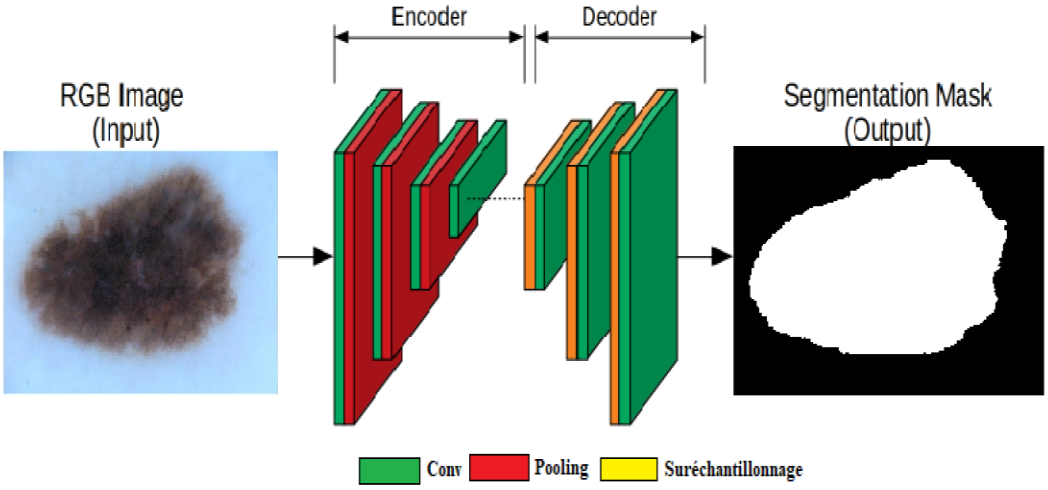


Figure 16 Architecture CNN de base pour la segmentation d'images

2.8.1 VGG

Après que CNN ait été déterminé étant comme efficace dans le domaine de la segmentation d'images, un principe de conception simple et efficace pour CNN a été proposé par Simonyan et Zisserman et qui s'appelait Visual Geometry Group (VGG) (41). Ce modèle multicouche comporte 16 couches pour VGG-16 et 19 couches pour VGG-19 et il utilise des filtres de petite taille 3×3 qui permettent d'améliorer les performances de CNN. En diminuant le nombre de paramètres, un avantage supplémentaire de réduction des complications de calcul a été obtenu en utilisant des filtres de petite taille. Ces résultats ont établi une nouvelle tendance de recherche pour travailler avec des filtres de petite taille dans CNN. L'architecture de VGG se compose de (41):

1. Entrée : Le VGG prend une taille d'entrée d'image de 224×224. Pour le concours ImageNet, les créateurs du modèle ont recadré le patch central 224×224 dans chaque image pour conserver la cohérence de la taille d'entrée de l'image.
2. Couches convolutifs : Les couches convolutifs de VGG tirent parti d'un champ récepteur minimal, c'est-à-dire 3×3, la plus petite taille possible qui capture toujours haut/bas et gauche/droite. De plus, il existe également des filtres de convolution 1×1 agissant comme une

transformation linéaire de l'entrée. Vient ensuite une unité Relu, qui est une énorme innovation pour réduire le temps de traitement. Le stride de convolution est fixée à 1 pixel pour conserver la résolution spatiale conservée après la convolution (le stride est le nombre de décalages de pixels sur la matrice d'entrée).

3. Couches cachées : Toutes les couches cachées du réseau VGG utilisent ReLU. VGG n'utilise généralement pas la normalisation de la réponse locale car elle augmente la consommation de mémoire et le temps d'entrainement. De plus, il n'apporte aucune amélioration à la précision globale.
4. Couches entièrement connectées : Le VGG a trois couches entièrement connectées. Sur les trois couches, les deux premières sur 4096 canaux chacune et la troisième sur 1000 canaux.

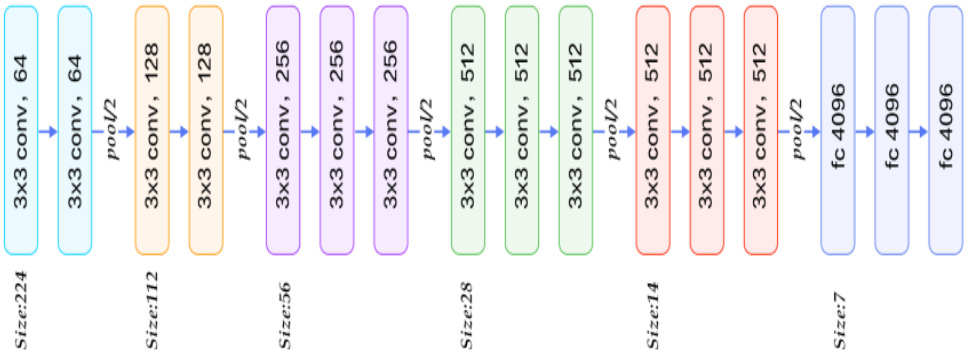


Figure 17 Architecture de VGG

2.8.2 Inception

Le réseau InceptionV3 (42) a été une étape importante dans le développement des classificateurs CNN. Avant sa création, les CNN les plus populaires empilaient simplement des couches de convolution de plus en plus profondes, dans l'espoir d'obtenir de meilleures performances.

Le réseau Inception, en revanche, était complexe (fortement conçu). Il a utilisé de nombreuses astuces pour améliorer les performances ; à la fois en termes de rapidité et de précision. Son évolution constante a conduit à la création de plusieurs versions du réseau. Les versions populaires sont les suivantes : Inception v1, Inception v2, Inception v3.

Comme expliqué dans (43), l'objectif principal d'Inception V1 est de trouver une architecture de réseau neuronal profond efficace. Cette architecture permet de détecter l'emplacement de l'information dans l'image en utilisant des filtres de différentes tailles. Inception V1 ou bien GoogLeNet dispose de 9

modules de démarrage de empilés linéairement. Elle a 22 couches de profondeur (27 y compris les couches de regroupement). Elle utilise la mise en commun des moyennes globales à la fin du dernier module de démarrage.

C. Szegedy et al. (42) ont présenté Inception v2 et Inception v3 dans le même article. Les auteurs ont proposé un certain nombre de mises à jour qui ont augmenté la précision et réduit la complexité de calcul. Inception v2 a proposé de factoriser la convolution 5x5 en deux opérations de convolution 3x3 pour ne pas trop réduire les dimensions et par conséquent garder le max d'information de l'image. De plus, cette architecture factorise les convolutions de taille de filtre nxn en une combinaison de convolutions 1xn et nx1. Par exemple, une convolution 3x3 équivaut à effectuer d'abord une convolution 1x3, puis à effectuer une convolution 3x1 sur sa sortie. Ils ont trouvé que cette méthode était 33 % moins chère que la simple convolution 3x3 (42). Dans la version 3, les auteurs souhaitent explorer des moyens d'étendre les réseaux de manière à utiliser le calcul ajouté aussi efficacement que possible par des convolutions correctement factorisées et une régularisation agressive.

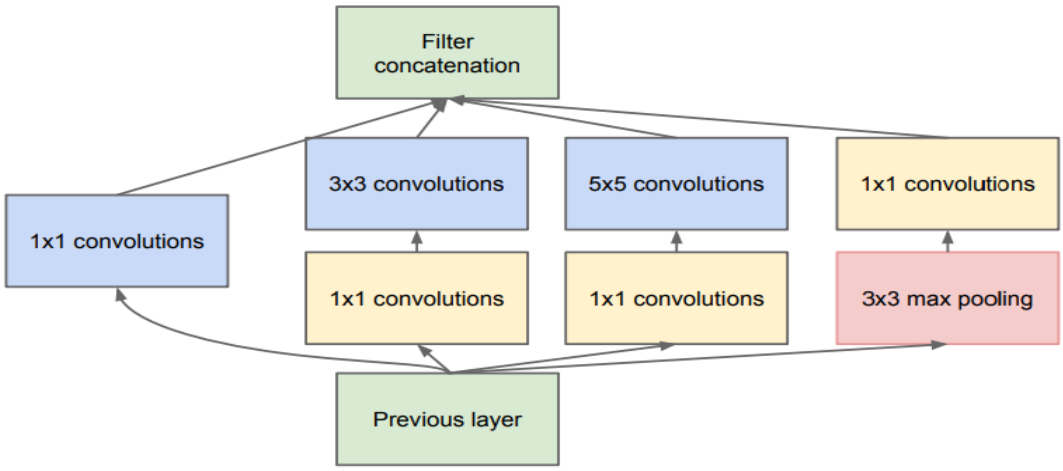


Figure 18 Module d'architecture d'Inception (42)

2.8.3 U-Net

U-Net a été développé par Olaf Ronneberger et al. (44) pour la segmentation des images biomédicales. L'architecture contient une partie contractante et une partie expansive.

La partie de contraction (également appelé encodeur) qui est utilisé pour capturer le contexte dans l'image. L'encodeur n'est qu'une pile traditionnelle de couches convolutionnelles et de mise en commun

maximale. La deuxième partie est le chemin d'expansion utilisé pour permettre une localisation précise à l'aide de convolutions transposées. La figure 19 montre l'architecture de UNet.

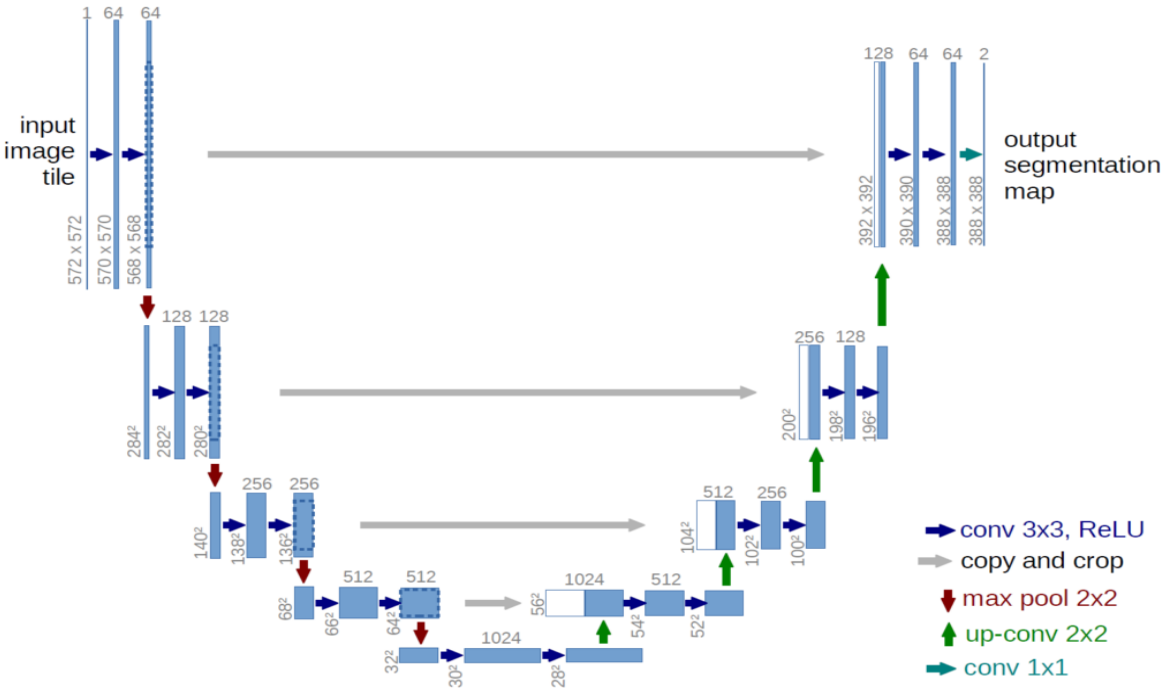


Figure 19 Architecture U-Net (44)

U-Net consiste de réseau codeur-décodeur en forme de U, qui se compose de quatre blocs codeurs et de quatre blocs décodeurs connectés via un pont. Le réseau d'encodeurs (chemin de contraction) réduit de moitié les dimensions spatiales et double le nombre de filtres (canaux de caractéristiques) à chaque bloc d'encodeur. De même, le réseau de décodeurs double les dimensions spatiales et réduit de moitié le nombre de canaux de caractéristiques.

La première partie (le codeur) consiste en l'application répétée de deux convolutions 3x3, chacune est suivie séquentiellement par une fonction de normalisation, une fonction d'activation unité linéaire rectifiée (ReLU) et d'une opération de mise en commun maximale (MaxPooling) 2x2 avec un pas de 2 pour le sous-échantillonnage.

Pour la deuxième partie (décodeur), chaque étape consiste en un sur-échantillonnage (convolution inverse) de la carte de caractéristiques et une concaténation suivie d'opérations de convolution. Dans le réseau de suréchantillonnage, les représentations d'images abstraites sont sur-

échantillonnées à l'aide de diverses techniques pour rendre leurs dimensions spatiales égales à l'image d'entrée. Les techniques de suréchantillonnage les plus utilisées sont :

- Voisins les plus proches : Dans les Voisins les plus proches, comme son nom l'indique, nous prenons une valeur de pixel d'entrée et la copie dans les K-Voisins les plus proches où K dépend de la sortie attendue.
- Interpolation bilinéaire : Dans l'interpolation bilinéaire, nous prenons la valeur des 4 pixels les plus proches du pixel d'entrée et effectuons une moyenne pondérée basée sur la distance des quatre cellules les plus proches en lissant la sortie.
- Bed Of Nails : Dans Bed of Nails, nous copions la valeur du pixel d'entrée à la position correspondante dans l'image de sortie et remplissons les zéros dans les positions restantes.
- Max-Unpooling : La couche Max-Pooling dans CNN prend le maximum parmi toutes les valeurs du noyau. Pour effectuer le max-unpooling, tout d'abord, l'index de la valeur maximale est enregistrée pour chaque couche de max-pooling pendant l'étape de codage. L'index enregistré est ensuite utilisé lors de l'étape de décodage où le pixel d'entrée est mappé sur l'index enregistré, remplissant les zéros partout ailleurs.

2.9 Conclusion

Dans ce chapitre, une vision générale sur l'apprentissage profond a été présentée. Cependant, nous avons fait un tour d'horizon détaillé sur les différents modèles basés sur les CNN, constituant l'état de l'art dans la segmentation sémantique. Dans le chapitre suivant nous allons présenter l'implémentation de la segmentation les lésions de la peau et les résultats obtenus en les comparant avec d'autre méthodes.

Chapitre III Résultats et discussion

3.1 Introduction

Dans ce dernier chapitre, nous abordons le problème de la segmentation des lésions cutanées à l'aide d'un réseau de neurones convolutifs basé sur l'architecture U-Net, expliqué dans le chapitre précédant, en utilisant les images dermoscopiques fournies par l'International Skin Imaging Collaboration (ISIC)

3.2 Cadre de travail

L'International Skin Imaging Collaboration (ISIC) est une conférence challenge visant à améliorer le diagnostic du mélanome, sponsorisé par l'International Society for Digital Imaging of the Skin (ISDIS). Les archives ISIC contiennent la plus grande collection publiquement disponible d'images dermoscopiques, de qualité contrôlée, de lésions cutanées. Actuellement, les archives ISIC contiennent plus de 20 000 images dermoscopiques, qui ont été collectées dans des centres cliniques de premier plan à l'échelle internationale et acquises à partir d'une variété d'appareils au sein de chaque centre. Une participation large et internationale à la contribution d'images est conçue pour assurer un échantillon représentatif cliniquement. Toutes les images entrantes dans les archives ISIC sont filtrées pour assurer la qualité. La plupart des images sont associées à des métadonnées cliniques, qui ont été vérifiées par des experts.

Dans notre projet nous avons focalisé sur la segmentation des lésions basée sur les données fournées dans le challenge ISIC 2018 (45).

3.3 Environnement de travail

3.3.1 Langage de programmation : Python

Python, le langage de programmation open source le plus utilisé. Il offre une fluidité pour le développeur car il libère les développeurs des contraintes de formes qui occupent leur temps avec les langages les plus anciens. Ce langage est très facile à apprendre et à maîtriser (46).

Il est très populaire dans le Deep learning grâce aux framework développés : Keras, Tensorflow et bien d'autres pour servir les recherches dans le domaine d'intelligence artificielle.

3.3.2 Keras et Tensorflow

Keras est une interface de programmation d'application (API) de réseau neuronal efficace de haut niveau écrite en Python. Cette bibliothèque de réseaux de neurones open source est conçue pour fournir une expérimentation rapide avec des réseaux de neurones profonds. Keras se concentre sur la modularité, la convivialité et l'évolutivité. Il ne gère pas les calculs de bas niveau ; au lieu de cela, il les transfère à une autre bibliothèque appelée Backend.

Keras a été adopté et intégré à TensorFlow. Les utilisateurs peuvent y accéder via le module tf.keras. Cependant, la bibliothèque Keras peut toujours fonctionner séparément et indépendamment.

TensorFlow est un cadre d'apprentissage en profondeur open source de bout en bout développé par Google et publié en 2015. Il est connu pour son support de documentation, ses options de production et de déploiement évolutives, ses multiples niveaux d'abstraction et sa prise en charge de différentes plates-formes.

TensorFlow est une bibliothèque mathématique symbolique utilisée pour les réseaux de neurones et convient le mieux à la programmation de flux de données dans une gamme de tâches. Il offre plusieurs niveaux d'abstraction pour la construction et l'entrainement de modèles.

Une entrée prometteuse et à croissance rapide dans le monde de l'apprentissage en profondeur, TensorFlow offre un écosystème flexible et complet de ressources communautaires, de bibliothèques et d'outils qui facilitent la création et le déploiement d'applications d'apprentissage automatique.

Comme mentionné précédemment, TensorFlow a adopté Keras, donc nous avons exploité cette intégration pour la conception et entrainement du modèle.

3.3.3 Configuration hardware

Pour l'environnement de développement nous avons utilisé la configuration hardware suivante :

- CPU Intel Core I77700HQ 2.8 GHz - 3.8 GHz
- RAM 16 Go
- Stockage 256 Go SSD - (M.2) et 1 To HDD SATA
- Processeur graphique
- NVIDIA GeForce GTX 1050 / Intel HD Graphics 630
- Mémoire vidéo 4 Go GDDR5 SDRAM

3.4 Ensemble des données

Nous avons utilisé 2594 images et 5 masques de réponse de vérité correspondant à chaque image pour entrainer le modèle et 1000 images pour tester la performance du modèle.

3.4.1 Données d'entrée

Les données d'entrée sont des images de lésions dermoscopiques au format JPEG. Toutes les images de lésions sont nommées selon le schéma ISIC_<image_id>.jpg, où <image_id> est un identifiant unique à 7 chiffres. Les balises EXIF dans les images ont été supprimées ; les balises EXIF restantes ne doivent pas être utilisées pour fournir des métadonnées précises.

Les images des lésions ont été acquises avec une variété de types de dermatoscopes, de tous les sites anatomiques (à l'exclusion des muqueuses et des ongles), à partir d'un échantillon historique de

patients présentés pour le dépistage du cancer de la peau, provenant de plusieurs institutions différentes. Chaque image de lésion contient exactement une lésion primaire ; d'autres marqueurs repèrent, des lésions secondaires plus petites ou d'autres régions pigmentées peuvent être négligés (47).

3.4.2 Données de réponse

Les données de réponse sont des images de masque binaire au format PNG, indiquant l'emplacement de la lésion cutanée primaire dans chaque image de lésion d'entrée.

Les images de masque sont nommées à l'aide du schéma ISIC_<image_id>_segmentation.png, où <image_id> correspond à l'image de lésion correspondante pour le masque.

Les images de masque doivent avoir exactement les mêmes dimensions que leur image de lésion correspondante. Les images de masque sont encodées sous forme de PNG 8 bits à canal unique (niveaux de gris) (pour fournir une compression sans perte), où chaque pixel est soit :

- 0 : représentant le fond de l'image, ou des zones en dehors de la lésion primaire
- 255 : représentant le premier plan de l'image, ou des zones à l'intérieur de la lésion primaire

Comme la lésion cutanée primaire est une seule région contiguë, les images de masque ne doivent également contenir qu'une seule région de premier plan contiguë, sans composants ni trous déconnectés. La région de premier plan peut être de n'importe quelle taille (y compris l'image entière) et peut être adjacente aux bordures de l'image.

3.4.3 Provenance des images de vérité

Les données de vérité de l'image du masque ont été générées à l'aide de plusieurs techniques, mais toutes les données ont été examinées et conservées par des dermatologues praticiens ayant une expertise en dermoscopie. Les segmentations de vérité ont été générées soit par :

- Algorithme entièrement automatisé, revu et accepté par un expert humain
- Algorithme de remplissage semi-automatisé, avec des paramètres choisis par un expert humain
- Traçage manuel de polygones par un expert humain

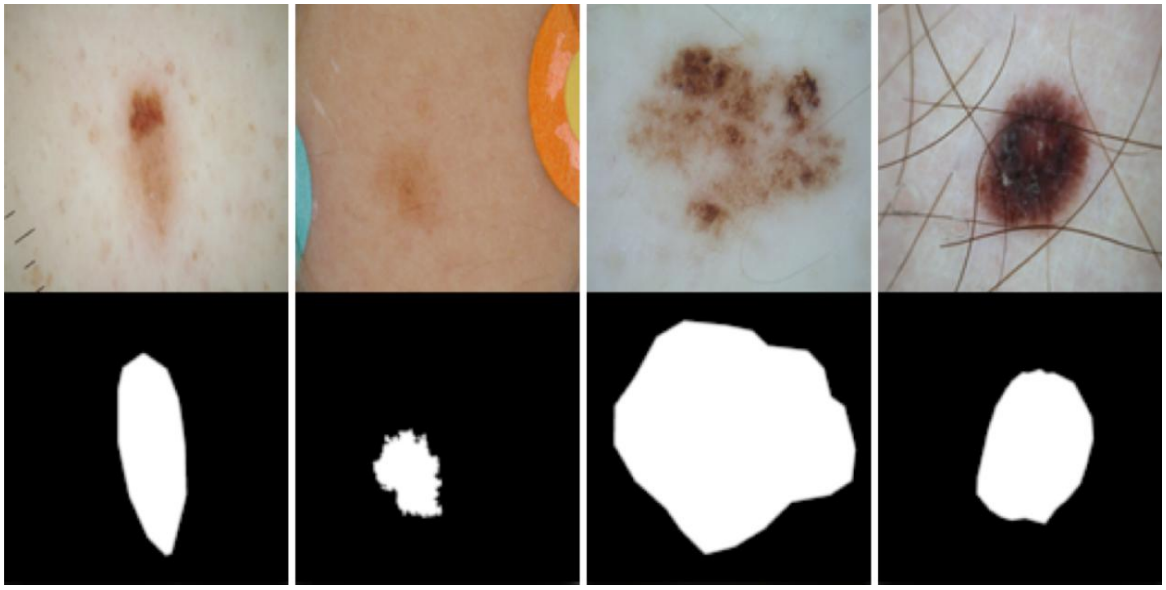


Figure 20 Exemple de données d'entrainement et de vérité

3.5 Etapes de la mise en œuvre de l'application

3.5.1 Préparation de données

Apres avoir télécharger les données et les mettre dans le stockage local, la première étape de développement de l'application consiste à créer un ensemble de fonctions qui permet de mettre les données prêtes à être utilisées pour entrainer le modèle.

La fonction **load_data** charge les **images** et **mask**. La fonction **glob** construit les liens aux fichiers images à télécharger, et pour assurer le tri des fichiers nous avons utilisé la fonction **sorted** et nous avons appliqué la même procédure pour les masques. Nous avons procédé à découper les données en données d'entrainement, données de validation et données de test afin d'assurer que notre modèle va être apte de prédire les résultats les plus justes possible.

- Données d'entraînement : C'est la partie des données les plus volumineuses en termes de données. En effet, c'est sur cet ensemble de données que le modèle va itérer durant la phase d'entraînement pour pouvoir s'approprier des paramètres, et les ajuster au mieux. Certaines règles préconisent qu'il soit composé de 80 % des données disponibles.
- Données de validation : On préconise d'avoir environ 10 % des données disponibles. Ce jeu sera appelé une seule fois, à la fin de chaque itération d'entraînement. Il va permettre d'évaluer les performances du modèle afin de l'ajuster.
- Données de test : Cet ensemble de données va avoir un rôle bien différent des autres, puisqu'il va évaluer le réseau sous sa forme finale. C'est pour cela qu'il doit être composé exclusivement

de nouveaux échantillons, encore jamais utilisé pour éviter de biaiser les résultats. Celui-ci peut encore être estimé de l'ordre de 10 % des données disponibles.

La prochaine étape de préparation de données consiste à lire les images en format RVB, à les redimensionner 256 x 256car les modèles s'entrainent plus rapidement sur les petites images (48) et à normaliser les pixels en divisant par 255 pour gagner encore en temps d'exécution.

Pour terminer l'étape de préparation, nous avons créé un pipeline de données d'entrée qui va faire passer les données au modèle par blocs. Cette technique permet de charger des données volumineuses toute en conservant de l'espace mémoire

Entrée [5]:
```
def load_data(dataset_path , split = 0.1):
    images = sorted(glob(os.path.join(dataset_path, "ISIC2018_Task1-2_Training_Input", "*.jpg")))
    masks  = sorted(glob(os.path.join(dataset_path, "ISIC2018_Task1_Training_GroundTruth", "*.png")))

    test_size = int(len(images) * split)

    train_x , valid_x = train_test_split(images, test_size = test_size, random_state = 42)
    train_y , valid_y = train_test_split(masks, test_size = test_size, random_state = 42 )

    train_x , test_x = train_test_split(images, test_size = test_size, random_state = 42)
    train_y , test_y = train_test_split(masks, test_size = test_size, random_state =42 )

    return (train_x,train_y), (valid_x,valid_y), (test_x,test_y)
```

Entrée [5]:
```
def load_data(dataset_path , split = 0.2):
    images = sorted(glob(os.path.join(dataset_path, "ISIC2018_Task1-2_Training_Input", "*.jpg")))
    masks  = sorted(glob(os.path.join(dataset_path, "ISIC2018_Task1_Training_GroundTruth", "*.png")))

    test_size = int(len(images) * split)

    train_x , valid_x = train_test_split(images, test_size = test_size, random_state = 42)
    train_y , valid_y = train_test_split(masks, test_size = test_size, random_state = 42 )

    train_x , test_x = train_test_split(images, test_size = test_size, random_state = 42)
    train_y , test_y = train_test_split(masks, test_size = test_size, random_state =42 )

    return (train_x,train_y), (valid_x,valid_y), (test_x,test_y)
```

Entrée [6]:
```
def read_image(path):
    path = path.decode()
    x = cv2.imread(path, cv2.IMREAD_COLOR)  ## (H, W, 3)
    x = cv2.resize(x, (W, H))
    x = x/255.0
    x = x.astype(np.float32)
    return x
```

Entrée [7]:
```
def read_mask(path):
    path = path.decode()
    x = cv2.imread(path, cv2.IMREAD_GRAYSCALE)  ## (H, W)
    x = cv2.resize(x, (W, H))
    x = x/255.0
    x = x.astype(np.float32)                    ## (256, 256)
    x = np.expand_dims(x, axis=-1)              ## (256, 256, 1)
    return x
```

Entrée [9]:
```
def tf_dataset(X, Y, batch):
    dataset = tf.data.Dataset.from_tensor_slices((X, Y))
    dataset = dataset.map(tf_parse)
    dataset = dataset.batch(batch)
    dataset = dataset.prefetch(10)
    return dataset
```

Figure 21 Ensemble de fonction pour la préparation des données

3.5.2 Conception du modèle

Comme nous avons mentionné précédemment, nous avons choisi d'utiliser l'architecture U-Net pour développer le modèle de réseau de neurones convolutifs. U-Net est une architecture de réseau codeur-décodeur en forme de U, qui se compose de quatre blocs codeurs et de quatre blocs décodeurs connectés via un pont. Le réseau d'encodeurs (chemin de contraction) réduit de moitié les dimensions spatiales et double le nombre de filtres (canaux de fonctions) à chaque bloc d'encodeur. De même, le réseau de décodeurs double les dimensions spatiales et réduit de moitié le nombre de canaux de fonctionnalités. L'approche de développement était basée sur le développement des fonctions chacune représente un bloc de l'architecture.

Chaque bloc codeur est constitué de deux convolutions 3×3, où chaque convolution est suivie d'une fonction d'activation ReLu. La fonction d'activation ReLU introduit une non-linéarité dans le réseau, ce qui aide à une meilleure généralisation des données d'apprentissage. La sortie du ReLU agit comme une connexion de saut pour le bloc décodeur correspondant. En ensuite une mise en commun maximale de 2 × 2, où les dimensions spatiales (hauteur et largeur) des cartes d'entités sont réduites de moitié. Cela réduisait le coût de calcul en diminuant le nombre de paramètres peuvent être entraînés (44).

Chaque bloc codeur est constitué de deux convolutions 3×3, où chaque convolution est suivie d'une fonction d'activation ReLU (Rectified Linear Unit). La fonction d'activation ReLU a introduit une non-linéarité dans le réseau, ce qui aide à une meilleure généralisation des données d'apprentissage. La sortie du ReLU agit comme une connexion de saut pour le bloc décodeur correspondant (44).

Les connexions de saut fournissent des informations supplémentaires qui vont être envoyer au décodeur afin de les utilise pour produire de meilleures caractéristiques sémantiques. En termes simples, nous pouvons dire que le saut de connexion aide à un meilleur flux de gradient lors de la rétropropagation, ce qui aide le réseau à apprendre une meilleure représentation (44).

Le pont relie le réseau d'encodeurs et de décodeurs. Il se compose de deux convolutions 3 × suivies 3, où chaque convolution est suivie d'une fonction d'activation ReLU.

Finalement le bloc décodeur commence par une convolution de transposition 2x2. Ensuite, il est concaténé avec la carte de caractéristiques de saut de connexion correspondante du bloc d'encodeur. Ces connexions de saut permettent de passer des fonctionnalités des couches précédentes qui sont parfois perdues en raison de la profondeur du réseau. Après cela, deux convolutions 3 × 3 sont suivies, où chaque convolution est suivie d'une fonction d'activation ReLU. La sortie du dernier décodeur passe

par une convolution 1x1 avec activation sigmoïde. La fonction d'activation sigmoïde donne le masque de segmentation représentant la classification pixel par pixel (44).

Entrée [11]:
```
def conv_block(inputs, num_filters):
    x = Conv2D(num_filters, 3, padding="same")(inputs)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    x = Conv2D(num_filters, 3, padding="same")(x)
    x = BatchNormalization()(x)
    x = Activation("relu")(x)

    return x
```

Entrée [12]:
```
def encoder_block(inputs, num_filters):
    x = conv_block(inputs, num_filters)
    p = MaxPool2D(2,2)(x)
    return x , p
```

Entrée [13]:
```
def decoder_block(inputs, skip_features, num_filters):
    "Déconvolution Aussi connu sous le nom de convolution de transposition , Généralement utilisé pour restaurer feature map
    x = Conv2DTranspose(num_filters, (2, 2), strides=2, padding="same")(inputs)
    x = Concatenate()([x, skip_features])
    x = conv_block(x, num_filters)
    return x
```

Entrée [14]:
```
def build_unet(input_shape):
    #create an input layer
    inputs = Input(input_shape)
    """ Encoder """
    # S1 act as skip connection, it will be passed to the decoder part as skip feature
    # P1 act as input for the next encoder
    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)
    s4, p4 = encoder_block(p3, 512)
    """ Bridge """
    # It take input as P4 and 1024 filters
    b1 = conv_block(p4, 1024)
    """ Decoder """
    # B1 act as input for the next encoder
    # S4 act as input for the next encoder
    d1 = decoder_block(b1, s4, 512)
    d2 = decoder_block(d1, s3, 256)
    d3 = decoder_block(d2, s2, 128)
    d4 = decoder_block(d3, s1, 64)
    """ Outputs """
    #create an input layer
    outputs = Conv2D(1, 1, padding="same", activation="sigmoid")(d4)
    """ Model """
    model = Model(inputs, outputs)
    return model
```

Figure 22 Conception du modèle

3.5.3 Entrainement du modèle

Les images d'entrée et leurs masques (vérité terrain) sont utilisés pour entraîner le réseau avec l'implémentation de gradient descendant stochastique (49). En raison des convolutions non rembourrées, l'image de sortie est plus petite que l'entrée d'une largeur de bordure constante. Pour optimiser les ressources hardware et utiliser au maximum la mémoire GPU, nous privilégions une taille de lot de 4 soit 219 images par lot. En conséquence, Le modèle est compilé avec l'algorithme optimiseur Adam et nous avons utilisé la fonction de perte d'entropie croisée binaire car il n'y a que deux classes avec un taux d'apprentissage de 0,001. Nous avons choisi de diminuer le taux d'apprentissage si la perte de validation ne s'améliore pas après 10 époques. Nous avons enregistré, au fur et au mesure, les poids uniquement s'il y a une amélioration de la perte de validation. Notez qu'il pourrait y avoir beaucoup de possibilités d'ajuster ces hyper paramètres et d'améliorer encore les performances du modèle.

Entrée [16]:
```
def build_unet(input_shape):
    #create an input layer
    inputs = Input(input_shape)
    """ Encoder """
    # S1 act as skip connection, it will be passed to the decoder part as skip feature
    # P1 act as input for the next encoder
    s1, p1 = encoder_block(inputs, 64)
    s2, p2 = encoder_block(p1, 128)
    s3, p3 = encoder_block(p2, 256)
    s4, p4 = encoder_block(p3, 512)
    """ Bridge """
    # It take input as P4 and 1024 filters
    b1 = conv_block(p4, 1024)
    """ Decoder """
    # B1 act as input for the next encoder
    # S4 act as input for the next encoder
    d1 = decoder_block(b1, s4, 512)
    d2 = decoder_block(d1, s3, 256)
    d3 = decoder_block(d2, s2, 128)
    d4 = decoder_block(d3, s1, 64)
    """ Outputs """
    #create an input layer
    outputs = Conv2D(1, 1, padding="same", activation="sigmoid")(d4)
    """ Model """
    model = Model(inputs, outputs)
    return model
```

Figure 23 Conception du modèle

Entrée [18]:

```
if __name__ == "__main__":
    """ Seeding """
    #We need to seeding the environment, to keep a good result
    np.random.seed(42)
    tf.random.set_seed(42)
    """ Folder for saving data """
    #We create a folder to save the output data and we named files
    #create_dir("files")
    """Hyperparepters"""
    #We create hyperparameters values: batch_size, learning rate, number of epoch, and choice where we will save
    #the file, and where we will save the metrics of the model: training loss, testing loss, and other metrics ….
    batch_size = 4
    lr = 1e-4 ## (0.0001)
    num_epoch = 20
    #model_path = "D:\Project\DL Project\Skin Segmentation\Dataset\Challenge 2018\files/model.h5"
    #csv_path = "D:\Project\DL Project\Skin Segmentation\Dataset\Challenge 2018\files/data.csv"
    model_path = "files/model_20.h5"
    csv_path = "files/data_20.csv"
    """ Dataset : 60/20/20 """
    #Now, we will load the data, we specify the path of the data and we will split the data: 60 for
    #training, 20 for validation, 20 for testing.
    dataset_path = "D:\Project\DL Project\Skin Segmentation\Dataset\Challenge 2018"
    (train_x, train_y), (valid_x, valid_y), (test_x, test_y) = load_data(dataset_path)
    #We check the length of train, valid and test
    print(f"Train: {len(train_x)} - {len(train_y)}")
    print(f"Valid: {len(valid_x)} - {len(valid_y)}")
    print(f"Test: {len(test_x)} - {len(test_y)}")
    #Now we will create the train and the valid datasets using the tf_dataset function
    train_dataset = tf_dataset(train_x, train_y, batch_size)
    valid_dataset = tf_dataset(valid_x, valid_y, batch_size)
    #After that we will calculate the batch size and the total number of image / batch
    #it will gave the number of images in each batch and it is named the train_steps
    #And the sane for valid
    train_steps = len(train_x)//batch_size
    valid_steps = len(valid_x)//batch_size
    #After that we need to check if this steps will left same images  If it is the case we should add one step
    if len(train_x) % batch_size != 0:
        train_steps += 1
    if len(valid_x) % batch_size != 0:
        valid_steps += 1
    """ Model """
    #Now we gonna create the model
    model_20 = build_unet((H, W, 3))
    metrics = [dice_coef, iou, Recall(), Precision()]
    #We build the Unet model and we compile it
    #We choice the inary cross entropy for the loss and for the optimizer we choice Adem optimizer with learning rate
    #of 0.001, and metrics as dice coef and IOU AND RECALL AND PRECISION
    model_20.compile(loss="binary_crossentropy", optimizer=Adam(lr), metrics=metrics)
    model_20.summary()
    #Now we will do the training part
    #We create a list of callbacks for the training:
    #    1.*Model checkpoint to save the model weighs parameters while training
    #    2.*Use lronplateau reducer it reduces the learning rate by the factor of 0.1 for 5 epochs
    #    3.*Csv logger log all the data will training
    #    4.*Tensor board for visualization
    #    5.*Early Stopping is basically using to reduce the overfitting, it is use to stop the training when specific
    #      | metrics are meet in this case the val_loss stop to be impoving or after 20 epochs

    callbacks = [
        ModelCheckpoint(model_path, verbose=1, save_best_only=True),
        ReduceLROnPlateau(monitor='val_loss', factor=0.1, patience=5, min_lr=1e-7, verbose=1),
        CSVLogger(csv_path),
        TensorBoard(),
        EarlyStopping(monitor='val_loss', patience=20, restore_best_weights=False)
    ]

    #It takes the train data and the number of epochs and the validation data and the step per epoch
    #it is equal to the training steps and the validation steps and callbacks

    #After that we gonna train our model
    #We will get the model.h5 that contain the weights and data file that contains the metrics for each epoch
    model_20.fit(
        train_dataset,
        epochs=num_epoch,
        validation_data=valid_dataset,
        steps_per_epoch=train_steps,
        validation_steps=valid_steps,
        callbacks=callbacks
    )
```

```
30 - precision: 0.9251
Epoch 18: val_loss did not improve from 0.17097
584/584 [==============================] - 14245s 24s/step - loss: 0.0948 - dice_coef: 0.8542 - iou: 0.7511 - reca
ll: 0.8930 - precision: 0.9251 - val_loss: 0.2074 - val_dice_coef: 0.7399 - val_iou: 0.5992 - val_recall: 0.7604 -
val_precision: 0.9070 - lr: 1.0000e-04
Epoch 19/20
584/584 [==============================] - ETA: 0s - loss: 0.0888 - dice_coef: 0.8624 - iou: 0.7631 - recall: 0.89
81 - precision: 0.9305
Epoch 19: val_loss did not improve from 0.17097
584/584 [==============================] - 12860s 22s/step - loss: 0.0888 - dice_coef: 0.8624 - iou: 0.7631 - reca
ll: 0.8981 - precision: 0.9305 - val_loss: 0.1992 - val_dice_coef: 0.7564 - val_iou: 0.6208 - val_recall: 0.8208 -
val_precision: 0.8665 - lr: 1.0000e-04
Epoch 20/20
584/584 [==============================] - ETA: 0s - loss: 0.0798 - dice_coef: 0.8746 - iou: 0.7808 - recall: 0.90
86 - precision: 0.9362
Epoch 20: val_loss did not improve from 0.17097
584/584 [==============================] - 20774s 36s/step - loss: 0.0798 - dice_coef: 0.8746 - iou: 0.7808 - reca
ll: 0.9086 - precision: 0.9362 - val_loss: 0.1743 - val_dice_coef: 0.7778 - val_iou: 0.6466 - val_recall: 0.8436 -
val_precision: 0.8782 - lr: 1.0000e-04
```

Figure 24 Procédure d'entrainement

L'archeticture concu est de la fome suivante :

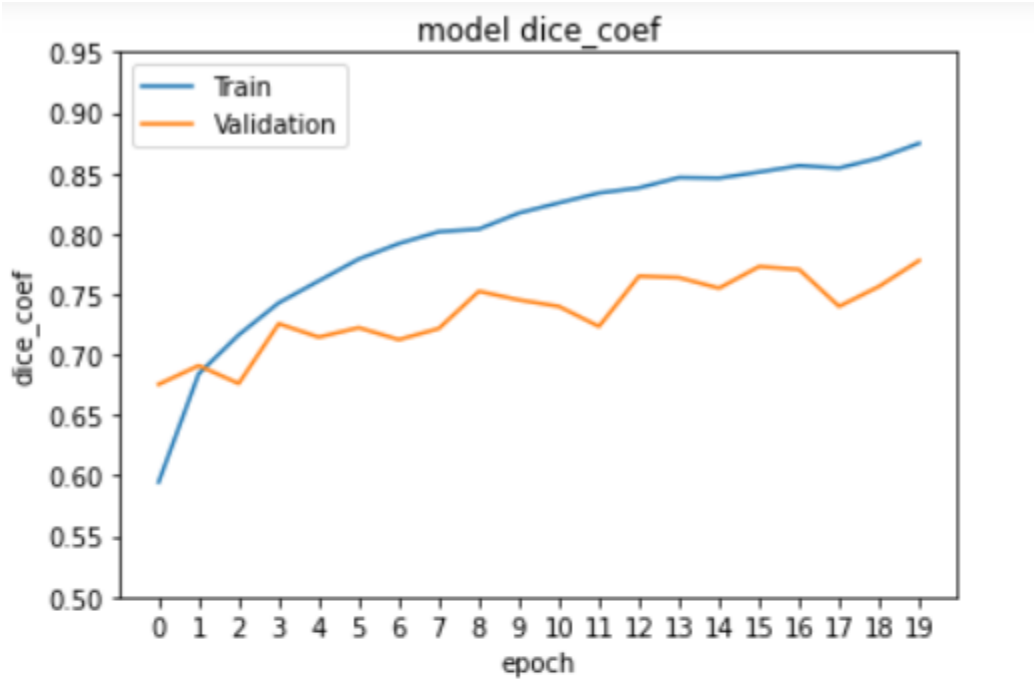
```
Entrée [19]: from tensorflow.keras.utils import CustomObjectScope
             from sklearn.metrics import accuracy_score, f1_score, jaccard_score, precision_score, recall_score
             from tqdm import tqdm
             with CustomObjectScope({'iou': iou, 'dice_coef': dice_coef}):
                 model20 = tf.keras.models.load_model("files/model_20.h5")
```

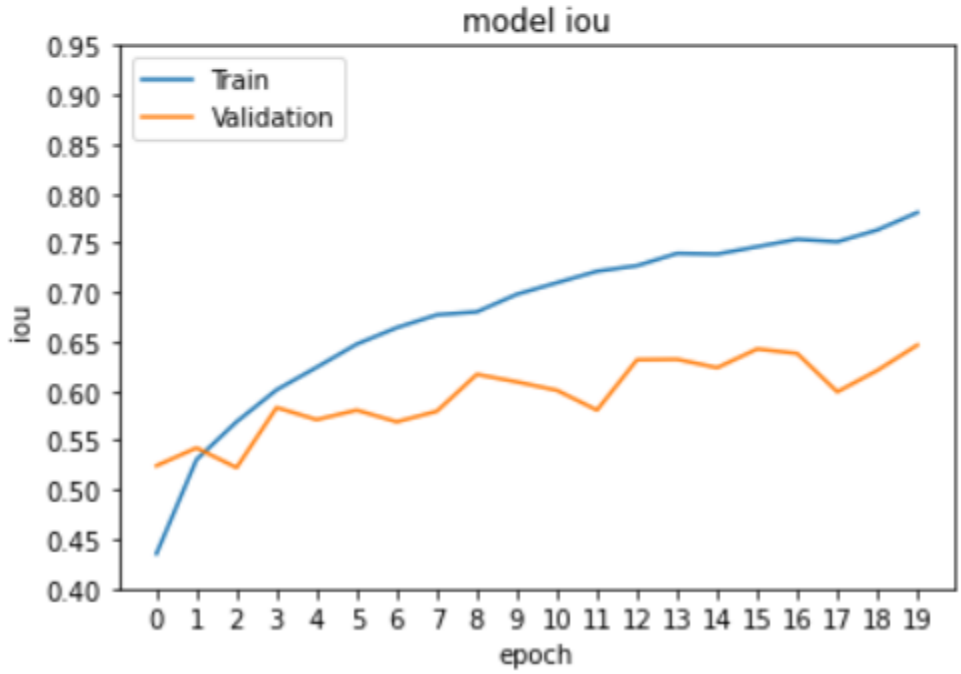
```
Entrée [22]: import visualkeras
             from PIL import ImageFont
             font = ImageFont.truetype("arial.ttf", 20)
             visualkeras.layered_view(model20, legend=True , font=font)
```

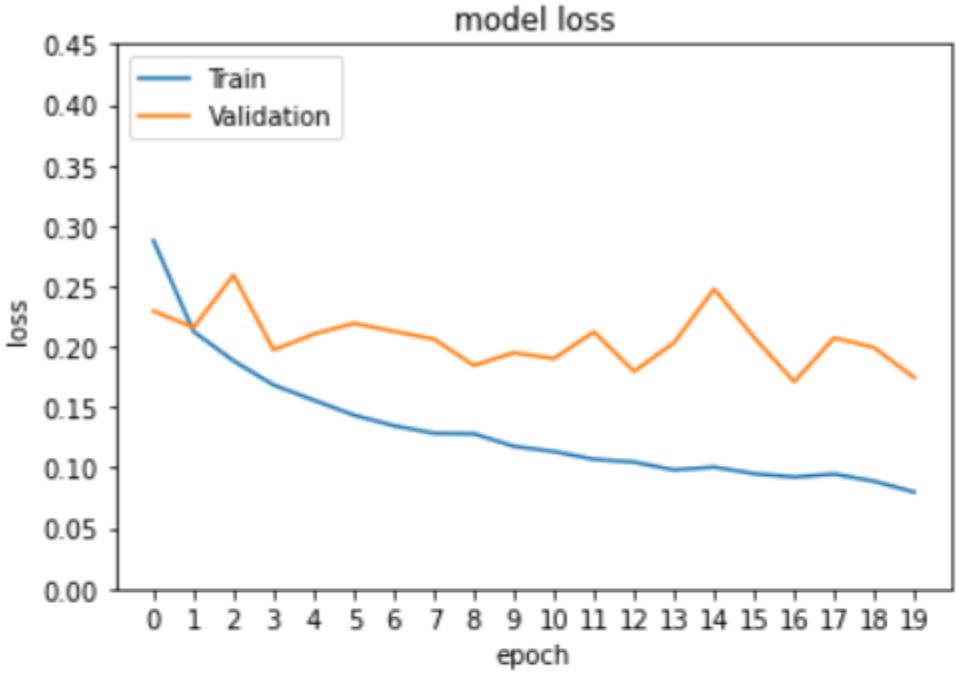
Out[22]:

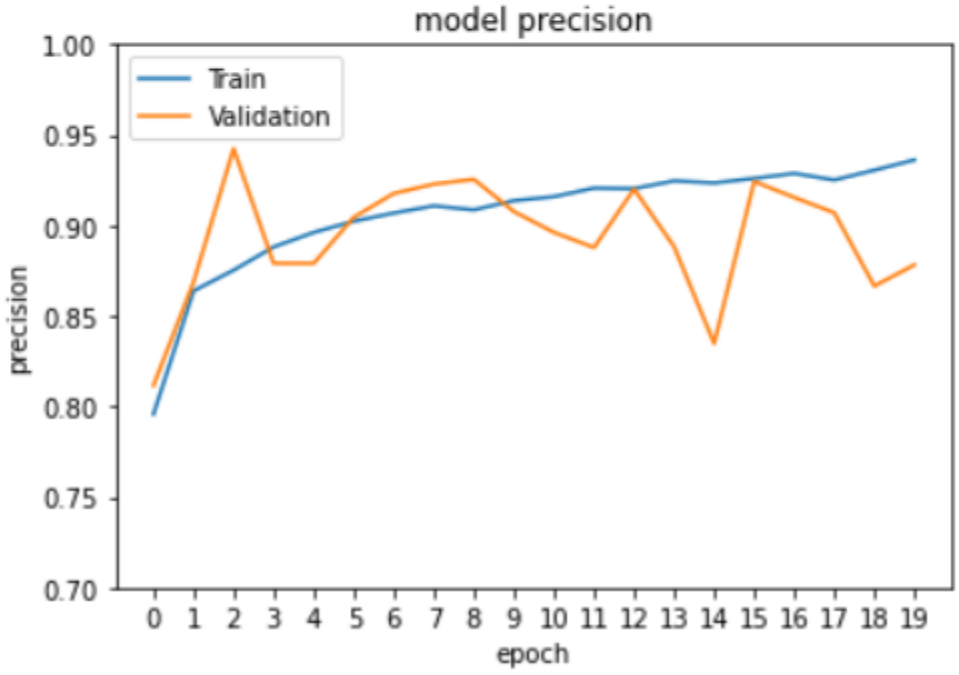
Figure 25 Architecture U-Net conçue

Apres avoir entrainer le modele 20 époques nous allons obtenu les métriques


model dice_coef
Train
Validation
dice_coef
epoch
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19


model iou
Train
Validation
iou
epoch
0.95
0.90
0.85
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
0.40
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19


model loss
Train
Validation
loss
epoch
0.45
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19


model precision
Train
Validation
precision
epoch
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19

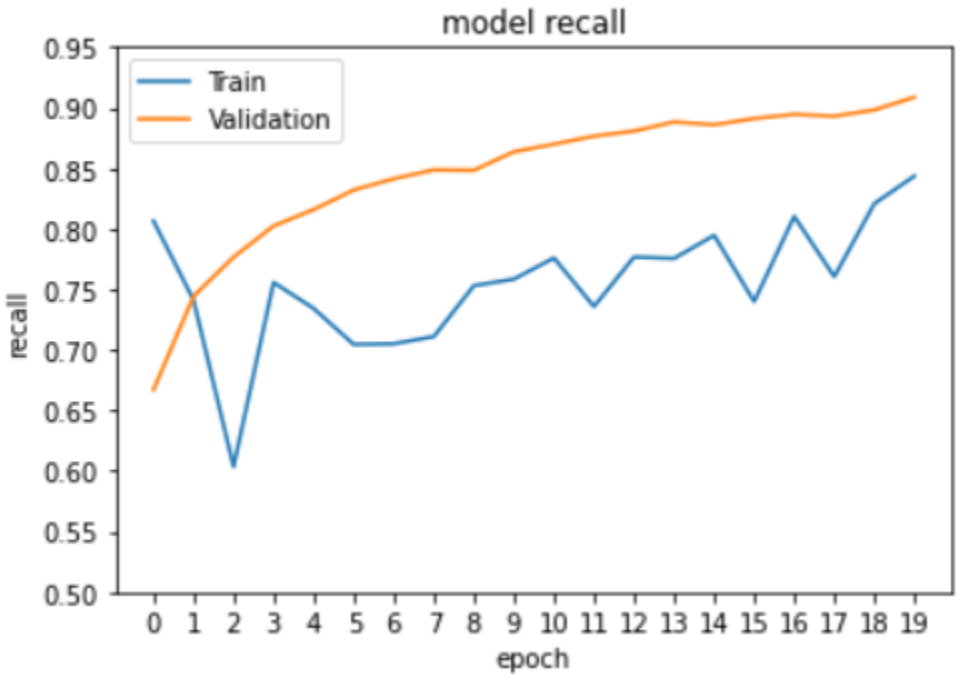


Figure 26 : Evaluation les métriques de performance durant les époques

3.5.4 Evaluation du modèle

Apres avoir terminé l'entrainement du modèle, nous avons testé les performances du modèle en utilisant des données de test jamais utilisées et nous avons stocké la segmentation des images prédites pour les comparer visuellement avec les masques.

Pour évaluer les performances du modèle nous avons utilisé les métriques suivantes :

1. Accuraccy : calcule le rapport entre les classes prédites correctes et le nombre total d'échantillons évalués.

$$Accuraccy = \frac{TP}{TP+TN+FP+FN} \qquad (5)$$

2. Précision : Utilisé pour calculer les modèles positifs qui sont correctement prédits par tous les modèles prédits dans une classe positive.

$$Precision = \frac{TP}{TP+FP} \qquad (6)$$

3. Recall : Utilisé pour calculer la fraction de modèle positif qui est correctement classés

$$Recall = \frac{TP}{TP+FN} \qquad (7)$$

4. F1-Score : calcule la moyenne harmonique entre les taux de rappel et de précision

$$F1 - Score = 2\frac{Precision*Recall}{Precision+Recall} \qquad (8)$$

5. Specificity : Utilisé pour calculer la fraction de motifs négatifs qui est correctement classés

$$Specificity = \frac{TN}{FP+TN} \quad (9)$$

6. Jaccard : Utilisé pour mesurer la similarité

$$Jaccard = Recall + Specificity - 1 \quad (10)$$

Comme indiqué dans les équations précédentes, TN et TP sont définis comme le nombre d'instances négatives et positives, respectivement, qui sont classées avec succès. De plus, FN et FP sont définis comme le nombre d'instances positives et négatives mal classées respectivement.

Entrée [19]:
```python
def save_result(ori_x, ori_y, y_pred, save_image_path):
    line = np.ones((H, 10, 3)) * 255
    ori_y = np.expand_dims(ori_y, axis=-1)  ## (256, 256, 1)
    ori_y = np.concatenate([ori_y, ori_y, ori_y], axis=-1) ## (256, 256, 3)
    y_pred = np.expand_dims(y_pred, axis=-1)  ## (256, 256, 1)
    y_pred = np.concatenate([y_pred, y_pred, y_pred], axis=-1) ## (256, 256, 3)
    cat_images = np.concatenate([ori_x, line, ori_y, line, y_pred*255], axis=1)
    cv2.imwrite(save_image_path, cat_images)
```

Entrée [31]:
```python
"Model 20"
if __name__ == "__main__":
    """ Seeding """
    #We need to seeding the environment, to keep a good result
    np.random.seed(42)
    tf.random.set_seed(42)
    """ Folder for saving results """
    #First we will create folder to save the predicted mask
    create_dir("results")
    """ Load the model """
    #Next we load the model from the model.h5
    with CustomObjectScope({'iou': iou, 'dice_coef': dice_coef}):
        model20 = tf.keras.models.load_model("files/model_20.h5")
    """ Load the test data """
    # After that we load the test data using two function that allow to read the data read_images_eval
    # and read_masks_eval
    dataset_path = "D:\Project\DL Project\Skin Segmentation\Dataset\Challenge 2018"
    (train_x, train_y), (valid_x, valid_y), (test_x, test_y) = load_data(dataset_path)
    # We start the prediction, and we creqte a loop to :
    # Extract the image name, Read the image and mask, Predicting the mask, Saving the predicted mask,
    # Flatten the array, and finally the Calculating metrics values
    SCORE = []
    for x, y in tqdm(zip(test_x, test_y), total=len(test_x)):
        """ Exctracting the image name """
        name = x.split("/")[-1]
        name2 = x.split("\\")[-1]
        """ Read the image and mask """
        ori_x, x = read_image_eval(x)
        ori_y, y = read_mask_eval(y)
        """ Predicting the mask """
        y_pred = model20.predict(x)[0] > 0.5
        y_pred = np.squeeze(y_pred, axis=-1)
        y_pred = y_pred.astype(np.int32)
        """ Saving the predicted mask """
        #save_image_path = f"/results/{name}.png"
        save_image_path = f"D:\\Project\\DL Project\\Skin Segmentation\\Dataset\\Challenge 2018\\output20\\{name2}"
        save_results(ori_x, ori_y, y_pred, save_image_path)
        """ Flatten the array """
        y = y.flatten()
        y_pred = y_pred.flatten()
        """ Calculating metrics values """
        acc_value = accuracy_score(y, y_pred)
        f1_value = f1_score(y, y_pred, labels=[0, 1], average="binary")
        jac_value = jaccard_score(y, y_pred, labels=[0, 1], average="binary")
        recall_value = recall_score(y, y_pred, labels=[0, 1], average="binary")
        precision_value = precision_score(y, y_pred, labels=[0, 1], average="binary")
        SCORE.append([name, acc_value, f1_value, jac_value, recall_value, precision_value])
    """ mean metrics values """
    score = [s[1:] for s in SCORE]
    score = np.mean(score, axis=0)
    print(f"Accuracy: {score[0]:0.5f}")
    print(f"F1: {score[1]:0.5f}")
    print(f"Jaccard: {score[2]:0.5f}")
    print(f"Recall: {score[3]:0.5f}")
    print(f"Precision: {score[4]:0.5f}")
    df = pd.DataFrame(SCORE, columns = ["Image Name", "Acc", "F1", "Jaccard", "Recall", "Precision"])
    df.to_csv("files/score_20.csv")
```

```
100%|██████████| 259/259 [03:46<00:00,  1.14it/s]

Accuracy: 0.93742
F1: 0.83538
Jaccard: 0.74595
Recall: 0.86283
Precision: 0.86749
```

Figure 27 Evaluation des performances du modèle et résultat de la perdition

3.6 Résultat et discussion

Dans la figure ci-dessous, on étale quelques prédictions obtenues après que le modèle a été entrainé durant 10 époques. Le modèle est assez précis pour la segmentation des lésions de différentes formes à partir des images dermoscopiques.

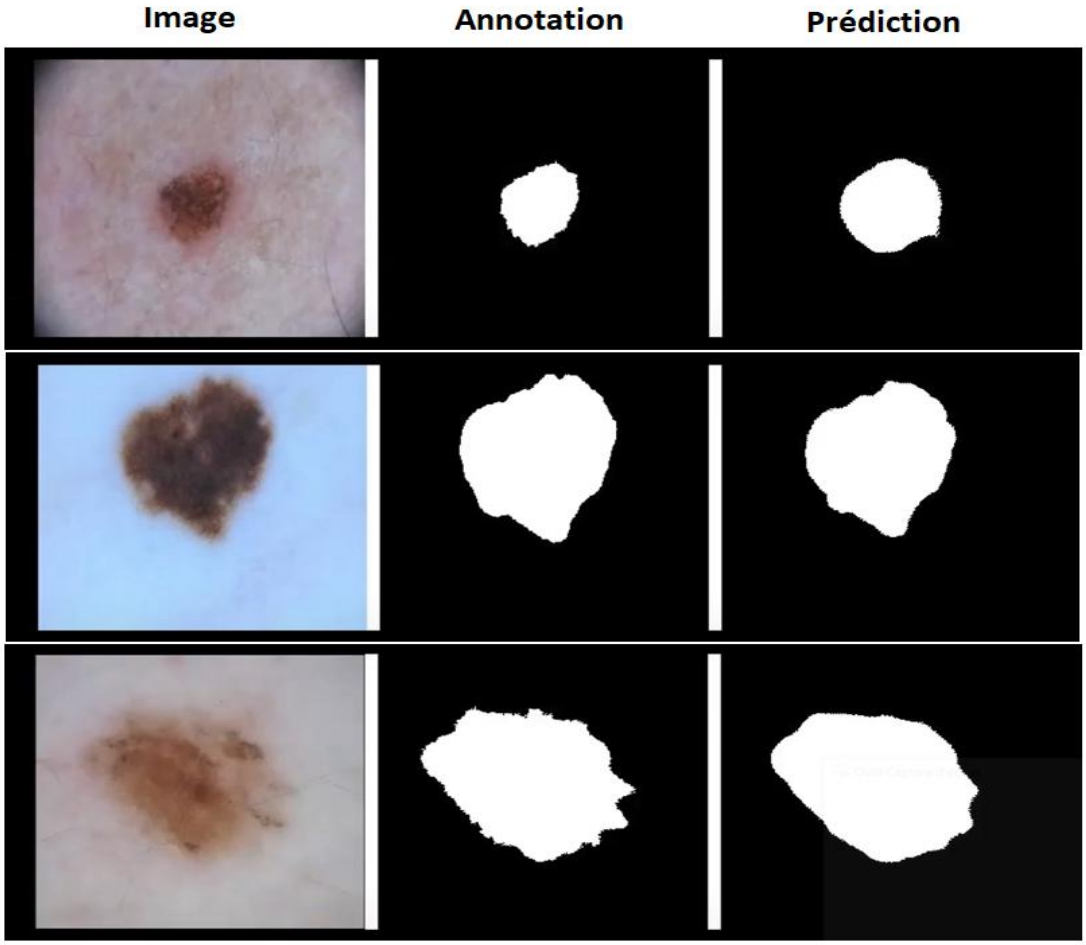


Figure 28 Exemple de segmentation des images de lésions dermoscopiques

Pour évaluer le modèle U-Net, on a choisi trois métriques jugées efficaces pour la tâche de segmentation sémantique à savoir : Accuracy, F1-Score et Précision.

Tableau 2 :Matrice de confusion pour le calcul des sources.

Prédiction	Vérité terrain		
		Lésion	No-Lésion
	No-Lésion	TP	FP
	Lésion	FN	TN

En utilisant la matrice de confusion (tableau 01) pour calculer les différents scores selon les formules de chaque métrique d'évaluation, le modèle entraîné de bout en bout sur des images de lésions dermoscopiques pour la segmentation de lésions a obtenu les résultats suivants :

Tableau 3 Résultat du modèle U-Net.

F1 Score (%)	Précision (%)	Accuracy (%)	Jaccard	Recall
83.54	86.75	93.74	74.60	86.28

D'après ces résultats, on voit clairement que le modèle proposé a atteint des taux très prometteurs en termes de segmentation. Visuellement, les meilleures segmentations sont très proches aux masques manuels des lésions.

Le model montre des limitations quand les images sont peu contrastées ou éparpillées, la figure 29 montre des images mal segmenté par le model U-Net. Nous avons constaté que U-Net est incapable de segmenté avec précision les limites de la lésion des cas complexes

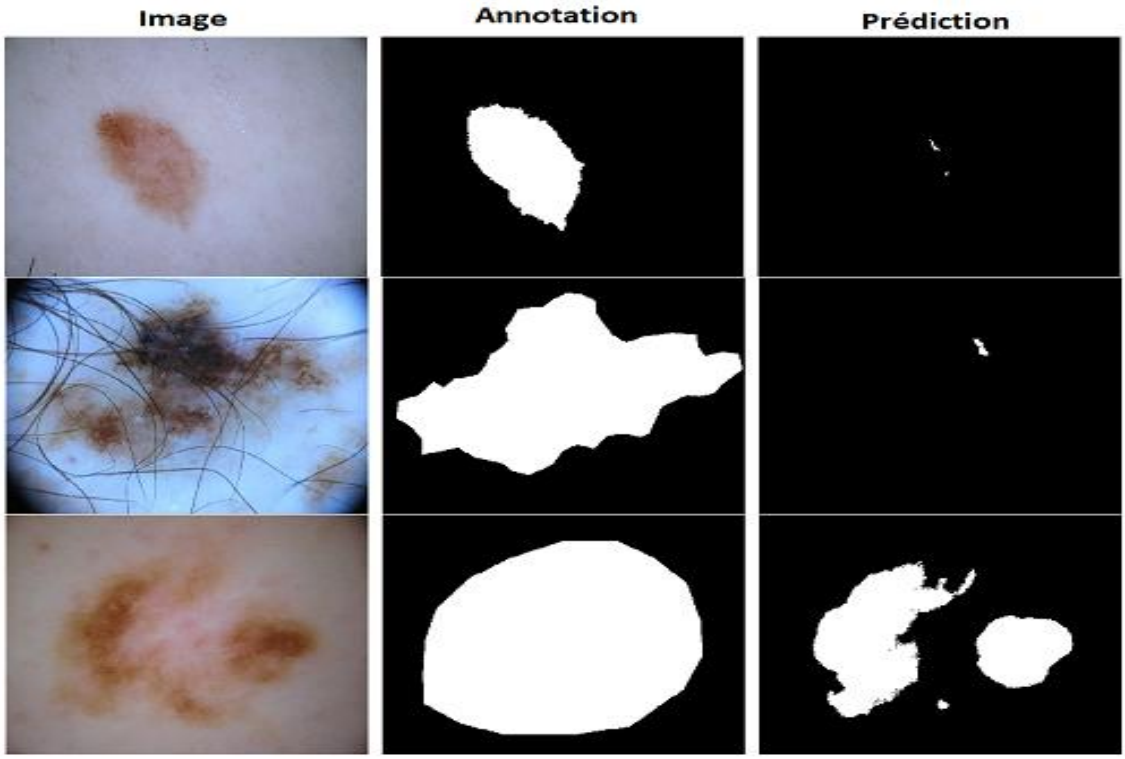


Figure 29Exemple de mal segmentation des images de lésions dermoscopiques.

Un entrainement avec une base de données plus grande et un prétraitement et un poste traitement augmentera la précision de la segmentation.

3.7 Comparaison aux modèles de l'état de l'art

Pour une meilleure lisibilité et une éventuelle comparaison, les performances rapportées par le modèle est comparé aux méthodes de l'état de l'art, y compris CE-Net (50), DeepLab V3+ et (51), SLSDeep (52). Notre méthode et les autres méthodes sont implémentées et appliqué aux différents ensembles de données. De plus l'algorithme que nous avons programmé (U-Net) n'utilise pas la technique d'augmentation de données et même l'environnement de développement n'est pas le même.

Tableau 4 :Comparaison entre les différentes méthodes de segmentation

Méthode	F1 Score (%)	Precision (%)	Accuracy (%)	Jaccard	Recall
U-Net	83.54	86.75	93.74	74.60	86.28
CE-Net (50)	-	-	95.42	80.61	87.59
DeepLab V3+ (51)	-	-	93.24	78.05	86.64
SLSDeep (52)	-	-	93.60	78.20	85.75

Il est remarquable à travers le tableau 3 que le modèle proposé atteint des performances qu'on peut les comparer avec des méthodes qui utilisent une combinaison de récentes techniques d'apprentissage. Notez que la performance du modèle peut être améliorer considérablement en utilisant des différentes stratégies notamment l'augmentation des données, l'utilisation de modules résiduels récurrents et l'utilisation des portes d'attention pour aider le réseau de segmentation à différencier la cible pixel de l'arrière-plan.

3.8 Conclusion

Dans ce chapitre nous avons montré les étapes de programmation de la méthode U-Net que nous avons utilisé pour la segmentation des lésions cutanées. Notre principale contribution a été la combinaison de cette architecture avec un certain nombre de stratégies d'entrainement des données et comparer les résultats obtenus avec d'autres architectures. Ces stratégies nous ont permis d'utiliser un réseau de bout en bout relativement simple pour générer des résultats de segmentation finement détaillés.

Conclusion générale

Dans ce travail, une approche d'apprentissage en profondeur basée sur l'architecture U-Net est présentée pour segmenter les lésions cutanées à partir d'images dermoscopiques.

Nous avons en premier lieu portée notre réflexion sur les notions générales de segmentation d'images, inclure la définition de la segmentation d'images, les types et les méthodes de segmentation d'images. Et d'un autre part nous avons présenté les lésions cutanées, inclure une description sur le mélanome et les différentes tumeurs malignes et bénignes et nous avons décrit ensuite les méthodes de diagnostic de mélanome.

En deuxième lieu, nous avons discutée des notions fondamentales des réseaux de neurones en générale et des réseaux de neurone convolutionnels en particulier. Nous avons introduit ces réseaux de neurones convolutionnels en présentant leurs couches : la couche de convolution, la couche de pooling et la couche entièrement connecté (fully connected). Et nous avons décrit certaines architectures utilisées dans les problématiques de segmentation et nous avons met l'accent en particulier sur l'architecture U-Net qu'on a utilisé dans l'application.

Ensuite, nous avons entamé la partie pratique qui consiste à implémenter l'architecture U-Net pour segmenter les images dermoscopiques et comparer ses résultats avec d'autres architectures de CNN. Durant l'élaboration nous avons essayé de respecter chaque de développement du modèle, de la préparation des images, de la conception du modèle et l'ajustement des hyper-paramètres et de l'entrainement et l'évaluation de la performance du modèle. Les résultats obtenus lors de la phase de test sont prometteurs avec une Accurancy de 93%, un taux de précision de 86% et un taux de similarité de 74% et peuvent être comparés avec des méthodes qui utilisent une combinaison de récentes techniques d'apprentissage.

Ses résultats confirment l`efficacité de l'architecture U-Net sur les images médicales. Néanmoins, avec quelques images mal segmentées. Cette imprécision peut être contournée en utilisant des différentes stratégies notamment l'augmentation des données, un prétraitement et un poste traitement peuvent améliorer les résultats de la segmentation.

Ce mémoire de master nous a permet approfondir nos connaissances théoriques et construire une solide base de savoir-faire dans le domaine de la vision par ordinateur particulièrement, la segmentation des objets. Ce travail est une expérience enrichissante dont il faut tenir compte pour progresser à l'avenir.

Références

1. **Siegel RL, Miller KD, Fuchs HE, Jemal A.** Cancer statistics, 2022. *CA Cancer J Clin.* 72(1):7-33, 2022 , Vol. doi: 10.3322/caac.21708, Epub 2022 Jan 12. PMID: 35020204.

2. **Andre Esteva, Brett Kuprel,Roberto A.Novoa,Justin Ko,Sunsan M.Swetter,Helen M.Blau et Sebastian Thrun.** *Dermatologist-level classification of skin cancer wuth deep neural networks.* s.l. : Nature, 2017. 542 (7639) :115-118.

3. **S.Philipp, J.-P.Cocquerez and.** *Analyse d'Images:Filtrage et segmentation.* s.l. : Masson,ISBN, 1995.

4. **P.Swoboda, A.Abbas and.** *Combinatorial optimization for panoptic segmentation.* 2021. 2106.03188.

5. **M. Sonka, V. Hlavac, R. Boyle,.** *«Image processing, analysis, and machine vision, 2 ed., PWS, 1998 ».* 1998.

6. **M. Kass, A. Witkin, D. Terzopoulos,.** *Snakes: Active contour model,International journal of compute.* 1988.

7. **R Kasmi, K Mokrani, RK Rader, JG Cole,.** *WV Stoecker Biologically inspired skin lesion segmentation using a geodesic active contour technique, - Skin Research and Technology.* 2016.

8. **Erkol, Bulent, et al.** *"Automatic lesion boundary detection in dermoscopy images using gradient vector flow snakes."* . s.l. : Skin Research and Technology , 2005. 11.1: 17-26.

9. **C. Xu, J.L. Prince, .** *«Snakes, shapes, and gradient vector flow», «IEEE Transactions Image Processing, 1998 ».* 1988.

10. **H. Zhou, G. Schaefer, M.E. Celebi, F. Lin, T. Liu,.** *«Gradient vector flow with mean shift for skin lesion segmentation», «Computerized Medical Imaging and Graphics, 2011».* 2011.

11. **N. Otsu.** *A threshold selection method from gray-level histograms.* s.l. : IEEE Transactions on Systems, Man and Cybernetics, 1979.

12. **A. Wong, J. Scharcanski, P. Fieguth,.** *Automatic skin lesion segmentation via iterative stochastic region merging, IEEE Trans. Inf. Technol. Biomed.* 2011.

13. **Chen, C., Belavy, D., Yu,W., Chu, C., Armbrecht, G., Bansmann, M., Felsenberg, D. & Zheng,.** *Localization and Segmentation of 3D Intervertebral Discs in MR Images by data drive estimation.* 2015.

14. **Didier Bessis.** *La Pathologie dermatologique en médecine interne.* s.l. : Arnette, 1999.

15. **Glaister J (2013).** *«Automatic segmentation of skin lesions from dermatological photographs ». « Department of Systems Design Engineering », «University of Waterloo, Waterloo ».* 2013.

16. **M. Hossein Jafari. Ebrahim Nasr-Esfahani, Nader Karimi,S. Reza Soroushmehr,.** *« Extraction of skin lesions from non-dermoscopic images for surgical excision of melanoma»,.*

17. **Mandi Gobindgarh, Fatehgarh Sahib, Pun.** *«Segmentation of Skin Lesions from Digital Images using an Optimized Approach: Genetic Algorithm ».* s.l. : Research Scholar, Department of Computer Science and Engineering.

18. **Glaister J.** *«Automatic segmentation of skin lesions from dermatological photographs ». « Department of Systems Design Engineering ».* s.l. : «University of Waterloo, Waterloo »., 2013.

19. **https://www.fondation-arc.org/cancer/facteurs-risque-cancer.** facteurs-risque-cancer . *https://www.fondation-arc.org/cancer/facteurs-risque-cancer .* [En ligne]

20. **Jing,Y.,Bian,Y.,Hu,Z.** *Deep Learning for Drug Design.* AAPS J 20,58 : An Artificial Itelligence Paradigm for Drug Discovery, 2018.

21. **Battiti.R.,Brunato.M,.** *Reactive search Optimization:learning while optimizing.* US : In Handbook of Metaheuristics, 2010. 543-571.

22. **Amri C,Paauw T,Aly R,Lavric M.** *Indentifying child abuse through text mning and machine learning.* s.l. : Expert Syst Appl, 2017. 88:402-18.

23. **Crawford M, Khoshgoftaar TM,Prussa JD,Richter AN,Al Najada H.** *Survey of review spam detection using machinelearning techniques.* s.l. : I Big Data, 2015. 2(1):23.

24. **Djeldoo Y, Elahi M,Cremonesi P,Garzotto F,Piazozolla P,Quadrana M.** *Content-based video recommendation system based on stylistic visual features.* s.l. : J Data Semant, 2016. 5(2):99-113.

25. **Intisar Rizwan I Haque, Jeremiah Neubert,.** Deep learning approaches to biomedical image segmentation. *Informatics in Medicine Unlocked.* 2020, Vol. 18.

26. **LeCun Y, Bengio Y,Hinton G.** *Deep learning.* 521(7553) : Nature, 2015. 436-44.

27. **Zhang Z, Cui P,Zhu W.** *Deep learning on graphs:a survey.* 2020 : IEEE Trabs Knowl Data Eng. https://doi.org/10.1109/TKDE.2020.2981333.

28. **Schretha A, Mahmood A.** *review of deep learning algorithms and architectures.* 7 : IEEE Access, 2019. 53040-65.

29. **Suzuki, K.** *Overview of deep learning in medical imaging.* 2017. pp. 257-273. Vol. 10.

30. **Vijay Patil Priyanka, N. C. Patil.** *Gray Scale Image Segmentation using OTSU Thresholding Optimal Approach.* s.l. : Journal for Research, 2016. Vol. 2.

31. **Jena, Manaswini & Mishra, Smita & Mishra, Debahuti.** *A survey on applications of machine learning techniques for medical image segmentation.* s.l. : International Journal of Engineering and Technology, 2018.

32. **Mnih, V., Kavukcuoglu, K., Silver, D. et al.** Human-level control through deep reinforcement learning. *Nature.* 2015, Vol. 518.

33. **Li, Yuxi.** Deep Reinforcement Learning: An Overview. 2017.

34. **Hewamalage H, Bergmeir C,Bandara K.** *Reccurrent neural networks for times series forecasting:current status and future directions.* s.l. : Int J Forecast, 2020. 37(1):388-427.

35. **Krizhevsky A, Sutskever I,Hinton GE.** *Imagenet classification with deep convolutional neural networks .* 60(6) : Commun ACM, 2017. 84-90.

36. **Gu J, Wang Z,Kuen J,Ma L,Shahroudy A,Shuai B,Liu T,Wang X,Wang G,Cai J.** *Recent advances in convoltional neural networks.* s.l. : Pattern recogn, 2018. 77:354-77.

37. **Fang W, Love PE,Luo H,Ding L.** *Computer vision for behaviour-based safety in construction:a review and future directions.* s.l. : Adv Eng Inform, 2020. 43:100980.

38. **PalaZ D, Magimai-Doss M,Collobert R.** *End-to-end acoustic modeling using convolutional neural networks for hmm-based automatic speech recognition.* s.l. : Speech Commun, 2019. 108:15-32.

39. **Loffe S, Szegedy C.** *Batch normalization:accelerating deep network training by reducing internal covariate shift.* s.l. : arXiv prepint arXiv, 2015. 1502.03167.

40. **Couchot, Jean-François & Couturier, Raphaël & Guyeux, Christophe & Salomon, Michel.** Steganalysis via a Convolutional Neural Network using Large Convolution Filters for Embedding Process with Same Stego Key. *arXiv.* 2016.

41. **Simonyan, K., & Zisserman, A.** Very deep convolutional networks for large-scale image recognition. *arXiv.* 1409.1556, 2014.

42. **C.Szegedy, V.Vanhoucke,S.Loffe,J.Shlens and Z.Wojna.** *Rethinking the Inception Architecture for Commputer Recognition (CVPR).* 2016. 2818-2826.

43. **Christian Szegedy, Wei Liu,Yangqing Jia,Pierre Sermanet,Scott Reed,Dragomir Anguelov,Dumitru Erhan,Vincent Vanhoucke,Andrew Rabinovich.** *Going Deeper with Concolutions .* 2014.

44. **Olaf Ronneberg, Philipp Fischer,and Thomas Brox.** *U-Net convolution networls for biomedical image segmentation .* s.l. : International conference on medical image computing , 2015. 1505.0459.

45. **Noel Codella, Veronica Rotemberg,Philipp Tschandl,M.Emre Celebi,Stephen Dusza,David Gutman,Brian Helba,Aadi Kalloo,Konstantinos Liopyris,Micheal Marchetti,Harald Kittler,Allan Halpen,.** *"Skin Lesion Analysis Toward Melanom Detection".* s.l. : A Challenge Hosted By the International Skin Imaging Collaboration (ISIC), 2018.

46. **Ketkar, N., & Santana, E.** *Deep learning with Python.* s.l. : Berkeley: Apress, 2017.

47. **E.Flores, J.Scharcanski,.** *"Segmentation of pignmented melanocytic skin lesion based on learned dictinaries ans normalized graph cuts".* s.l. : using feature learning and dictinaries, 2016. APPL.56.

48. **Saponara, S., Elhanashi, A.** *Impact of Image Resizing on Deep Learning Detectors for Training Time and Model Performance.* s.l. : Springer, 2022. Vol. 866.

49. **L.Yu, H.Chen,Q.Dou,J.QIN,P.A.Heng,.** *"Automated melanoma recognition in dermoscopy images via very deep residual network".* s.l. : "IEEE trans.Med.Imaging 36 (4) (2018), 2018.

50. **Zaiwang Gu, Jun Cheng, Huazhu Fu, Kang Zhou, Huaying Hao, Yitian Zhao, Tianyang Zhang, Shenghua Gao, and Jiang Liu,.** *Ce-net: Context encoder network for 2d medical image segmentation,.* s.l. : IEEE transactions on medical imaging 38 (2019), 2019. no. 10, 2281–2292..

51. **Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam,.** *Encoder-decoder with atrous separable convolution for semantic image segmentation, Proceedings of the European conference on computer vision (ECCV), .* 2018. pp. 801–818.

52. **Md Mostafa Kamal Sarker, Hatem A Rashwan, Farhan Akram, Syeda Furruka Banu, Adel Saleh, Vivek Kumar Singh, Forhad UH Chowdhury, Saddam Abdulwahab, Santiago Romani, Petia Radeva, et al.,.** *Slsdeep: Skin lesion segmentation based on dilated residual and pyram pooling networks.* s.l. : International Conference on Medical Image Computer-Assisted Intervention,Springer, 2019. pp.21-29.

53. **Albawi, Saad & Abed Mohammed, Tareq & ALZAWI, Saad.** Understanding of a Convolutional Neural Network. *10.1109/ICEngTechnol.* 2017.

Résumé

Les cancers de la peau sont la forme de cancer la plus fréquente chez l'être humain, un médecin trouve nombreuses difficultés pour un diagnostic précis de la lésion à travers de ses caractéristiques et à l'œil nu. Pour cela, il a été nécessaire de développer des méthodes automatiques pour la détection et l'identification anticipée de mélanome afin d'augmenter la précision du diagnostic en se basant sur des techniques de traitement d'image. La segmentation est une étape critique dans les systèmes automatiques de diagnostic de mélanome qui consiste à définir la lésion comme une région d'intérêt. Plusieurs méthodes de segmentation ont été développées pour la détection des lésions cutanées. Dans ce travail nous nous s'intéressant à l'une des approches de la segmentation basée sur les réseaux de neurones convolutifs pour segmenter les lésions cutanées. Nous avons choisi un réseau convolutionnel U-Net. Les résultats obtenus sont satisfaisants et montre une bonne détection de la lésion cutanées dans l'image numérique.

Mots-clés : mélanome, segmentations, réseau de neurone convolutif, Deep learning, U-Net, Lésion, images dermosopiques.

Abstract

Skin cancers are the most common form of cancer in humans; dermatologists face difficulties to diagnosis accurately malignant lesions manually through its characteristics. For this, it was necessary to develop automatic methods for the detection and early identification of melanoma in order to increase the accuracy of diagnosis based on image processing techniques. Segmentation is a critical step in automatic melanoma diagnostic; it consists on defining the lesion as a region of interest. Several segmentation methods have been developed for skin lesion detection. In this work we are interested in one of the approaches of the automated segmentation of skin lesions based on convolutif neural networks. We chose the U-Net architecture. The results obtained are satisfactory and show good detection of the skin lesion in the digital image.

Keywords: melanoma, segmentations, convolutional neural network, Deep learning, U-Net, Lesion, dermosopic images.