



---

# Projet de Fin d'Année

Spécialité : Informatique Industrielle et Automatique

---

## Détection des nodules pulmonaires en utilisant UNET (à partir des images CT) et reconstruction 3D

---

Présenté par  
**BOUABID RYM**  
**JEDDOU MONTASSAR**  
**TRIKI GHAYLEN**  
**BEN SALAH MEHDI ISMAIL**

Soutenu le 18 juin 2021 devant le jury composé de :

Mr Raouf KTATA	Jury
Mr Mustapha HAMDI	Encadrant

**Année Universitaire : 2020 / 2021**

# *Remerciements*

Le plus dur n'est pas de rédiger le rapport mais de remercier toutes les personnes qui nous ont soutenu pour réaliser ce projet.

Nous tenons à exprimer notre sincère gratitude à notre encadrant Mr **Mustapha HAMDI** pour tous les efforts qu'il a fournis pour diriger le travail et pour le temps qu'il nous a consacré. Nous le remercions vivement pour son courage de travailler pendant cette période de pandémie, et en dépit de ses responsabilités il était toujours présent par ses directives fructueuses. Nous profitons de l'occasion pour exprimer notre reconnaissance à tous les professeurs qui nous ont enseigné durant notre cursus d'études primaires, secondaires et supérieurs et qui nous ont aidé à trouver notre voie et à nous épanouir.

Nous remercions également l'honorable jury Mr **Raouf KTATA**, Maître de Conférences à l'INSAT d'avoir accepté d'évaluer notre travail. Finalement, un grand merci à mes collègues stagiaires pour leur bonne humeur tout au long de ce stage.

# Résumé

Le cancer du poumon est la forme de cancer la plus couramment diagnostiquée, c'est la principale cause de décès par cancer tant chez l'homme que chez la femme avec 1.6 million de décès chaque année d'après la société canadienne du cancer. Sa détection précoce est très importante puisqu'elle permet d'accélérer la procédure de traitement et par la suite diminuer le taux de mortalité et sauver des vies. Cependant, ce processus n'est pas évident en raison du nombre considérable d'images médicales à analyser, les expertises nécessaires pour se faire et l'inexistence d'une formule claire pour décrire une tumeur. Les techniques d'intelligence artificielle peuvent être utiles dans ces cas pour offrir un système capable de détecter automatiquement les nodules et accélérer le processus de diagnostic tout en gardant un faible taux d'erreur.

Dans cette étude, nous avons mis en œuvre un algorithme qui utilise la base de données LUNA 16 pour entraîner notre modèle de détection des nodules ainsi que sa reconstruction en 3D.

# Abstract

Lung cancer is considered the deadliest cancer worldwide with around 1.6 millions deaths every year, according to the CCS (Canadian Cancer Society). Early detection of lung cancer is very important in healthcare as it can help save lives and decrease the cancer mortality rate. Due to the huge number of CT scans that need to be analyzed annually, the detection of lung cancer at early stages become a tedious task for radiologists.

Artificial intelligence techniques can be useful in developing a solution for this problem by offering a system that can automatically detect nodules and speed-up the diagnosis process while making the result more repeatable and accurate.

In this study, we implemented a deep learning algorithm that uses a database, provided by the Lung Nodule Analysis 2016 challenge (LUNA 16), to train our model for detection and 3D reconstruction of nodules.

# Table des matières

<b>INTRODUCTION GENERALE.....</b>	<b>9</b>
<b><i>Chapitre 1 : Contexte du projet.....</i></b>	<b>11</b>
1.1 Introduction .....	11
1.2 Nodules pulmonaires .....	11
1.2.1 Définition d'un nodule pulmonaire .....	11
1.2.2 Technique d'imagerie médicale.....	11
1.2.3 Caractéristique d'un nodule.....	13
1.2.4 Problématique .....	14
1.3 Deep Learning .....	15
1.3.1 Concepts clés de l'Intelligence Artificielle .....	15
1.3.1.1 Réseaux neuronaux convolutifs.....	15
1.3.1.2 Classification des images : .....	16
1.3.1.3 Segmentation : .....	16
1.3.1.4 Evaluation des performance .....	18
1.3.1.4.1 Overfitting & Underfitting .....	18
1.3.1.4.2 Métriques et hyperparameters .....	19
1.3.2 Algorithmes de détection.....	19
1.3.2.1 Les algorithmes les plus utilisés .....	20
1.3.2.1.1 RCNN : Region-based CNN .....	20
1.3.2.1.2 Fast RCNN .....	21
1.3.2.1.3 Faster RCNN .....	23
1.3.2.1.4 Mask RCNN .....	24
1.3.2.1.5 YOLO - You Only Look Once.....	25
1.3.2.1.6 U-NET .....	26
1.3.2.2 Comparaison des algorithmes.....	27
1.3.2.3 Algorithme choisis.....	28
1.4 Reconstruction 3D .....	28
1.4.1 Égalisation de l'histogramme .....	28
1.4.2 Assemblage d'images médicale dans un espace 3d .....	29
1.4.3 Interpolation trilinéaire .....	29
1.4.4 Redimensionnement de l'image 3D .....	30
1.5 Conclusion .....	31
<b><i>Chapitre 2 : Ressources et environnements .....</i></b>	<b>32</b>
2.1 Introduction .....	32
2.2 Environnements techniques .....	32
2.2.1 Les différentes approches d'environnements .....	32
2.2.1.1 Limitations des ressources matérielles .....	32
Google Colab : Gratuit mais limité.....	32

.....	32
2.2.1.232	
2.2.1.3Google Colab PRO .....	33
2.2.2 Outils et modules .....	33
2.2.2.1 Python .....	33
2.2.2.2Keras & TensorFlow .....	33
2.2.2.3SimpleITK .....	33
2.2.2.4Git & GitHub .....	34
2.2.2.5Python packages .....	34
2.3 Base de données .....	35
2.3.1 Comparaison des bases de données .....	35
2.3.2 Présentation du dataset choisis : LUNA16 .....	35
2.4 Conclusion .....	36
<b>Chapitre 3 : Détection des nodules .....</b>	<b>37</b>
3.1 Introduction .....	38
3.2 Réalisation du modèle.....	38
3.2.1 Préparation des données.....	38
3.2.1.1Changement de base et rééchantillonnage.....	38
3.2.1.2Segmentation des poumons/nodules.....	38
3.2.1.3Normalisation .....	39
3.2.1.4Transformation en PNG.....	40
3.2.1.5Augmentation des données .....	40
3.2.2 Architecture du réseau (UNET).....	40
3.2.3 Initialisation des paramètres .....	44
3.2.4 La fonction de perte (binary_crossentropy).....	44
3.2.5 Optimisation du calcul.....	44
3.2.6 Optimisation du traitement .....	44
3.3 Tests et résultats.....	45
3.3.1 Données en entrées .....	45
3.3.2 Résultats Finals.....	46
<b>Chapitre 4 : Reconstruction 3D .....</b>	<b>48</b>
4.1 Introduction .....	49
4.2 Reconstruction des poumons .....	49
4.2.1 Positionner le scan .....	49
4.2.2 Segmentation des poumons .....	49
4.2.3 Algorithme de création de maillage approximatif : Marching Cubes .....	49
4.2.4 Création du plot 3D par Poly3DCollection .....	49
Résultats obtenus .....	50
4.2.5	50
4.3 Reconstruction 3D des nodules pulmonaires.....	50
<b>CONCLUSION GENERALE ET PERSPECTIVES .....</b>	<b>52</b>
<b>BIBLIOGRAPHIE .....</b>	<b>54</b>

# Liste des figures

Figure 1 : Schéma de fonctionnement d'un TDM .....	12
Figure 2 : Exemples d'un scanner thoracique sain (a) et scanner thoracique avec nodule (b) .....	12
Figure 3: UH des matières courantes .....	13
Figure 4: Différentes tailles de nodules .....	13
Figure 5: Exemples de nodules solide (a), semi-solide (b) et en verre dépoli pur (c).3 .....	14
Figure 6: Exemple de nodule à bordure spiculée (a) versus à bordure lisse (b).3 .....	14
Figure 7: Entrée du CNN .....	15
Figure 8 : Images d'entrées après division .....	15
Figure 9: Image de sortie .....	16
Figure 10: Architecture de CNN .....	<b>Error! Bookmark not defined.</b>
Figure 11: Applications de segmentation .....	17
Figure 12: (a) Segmentation sémantique de Mask (b)Instance Segmentation Mask.....	18
Figure 13: Démonstration de Data fitting .....	18
Figure 14: Réglage du Learning rate .....	19
Figure 15: Détection par RCNN .....	20
Figure 16: Détection par FAST RCNN .....	22
Figure 17: comparaison entre RCNN et Fast RCNN en temps d'exécution .....	22
Figure 18: architecture de Faster RCNN .....	23
Figure 19: Comparaison entre les algorithmes précédents et Faster RCNN en temps d'exécution.....	24
Figure 20: Segmentation par Mask R-CNN .....	25
Figure 21: Principe de détection par YOLO .....	26
Figure 22: Principe de fonctionnement de U-Net .....	26
Figure 23: principe de reconstruction 3D .....	28
Figure 24: Assemblage des images CT pour les représenter en 3D .....	29
Figure 25: Visualisation du Voxel .....	30
Figure 26: Voxel avec interpolation .....	31
Figure 27: Cube 3D contenant un voxel .....	31
Figure 28: les packages de python utilisés.....	34
Figure 29 : Fichier d'annotation .....	36
Figure 30: Fichier des candidats .....	36
Figure 31: Poumons après segmentation .....	39
Figure 32: Poumons Avant segmentation.....	39
Figure 33 Image d'un patient avec nodule      Figure 34 Masque respective (Après segmentation).....	39
Figure 35 Image avant normalisation      Figure 36 Image après normalisation .....	40
Figure 37 Entrée et sortie de modèle U-Net .....	40
Figure 38 Architecture du modèle U-Net .....	41
Figure 39 Les deux chemins de U-Net .....	42
Figure 40 Sortie de chemin expansif .....	42
Figure 41 Tableau récapitulatif du modèle Custom Unet.....	44
Figure 42 Division de Dataset en :Train , validation et Test .....	45
Figure 43 Contenu de chacun des dossiers .....	45
Figure 44 Exemple de contenus des dossiers images/class et labels/class .....	46
Figure 45 Les entrées de notre modèle .....	46

Figure 46	Resultat satisfaisante de modèle après training.....	47
Figure 47	résultat un peu erroné .....	47
Figure 48	Les 15 cas possibles de configuration des polygones .....	49
Figure 49	Reconstruction 3D sans segmentation	
Figure 50	Reconstruction 3D avec segmentation.	50
Figure 51	Recadrage d'une image médicale d'un patient ayant un nodule de diamètre 19mm .....	50
Figure 52	Visualisation d'un nodule de diamètre 19mm.....	51



# INTRODUCTION GENERALE

L'intelligence artificielle est un domaine qui a été développé après la seconde guerre mondiale dont le but est de créer des machines capables d'apprendre et de s'améliorer d'une manière autonome. Cette technologie se révèle être utile dans toutes les tâches intellectuelles. Ce concept s'est ensuite étendu pour englober plusieurs sous domaines dont la segmentation des images qui a connu un progrès considérable au cours de ces dernières années. Ce progrès est dû aux nombreux travaux dans ce domaine et à la disponibilité des bases d'images internationales qui ont permis aux chercheurs de signaler l'exécution de leurs approches.

En 2011 et 2012 trois événements ont changé la situation. Tout d'abord, les GPU (Graphical Processing Unit) capables de calculer plus que mille milliards d'opérations par seconde sont devenus disponibles pour un prix moins cher. Ces processeurs spécialisés, initialement conçus pour le rendu graphique des jeux vidéo, se sont avérés être très performants pour les calculs des réseaux neuronaux. Deuxièmement, des expériences menées simultanément à Microsoft Google et IBM avec l'aide du laboratoire de Geoff Hinton ont montré que les réseaux profonds pouvaient diminuer de moitié les taux d'erreurs des systèmes de reconnaissance vocale. Troisièmement, plusieurs records en reconnaissance d'image ont été battus par des réseaux de neurones convolutifs. L'événement le plus marquant a été la victoire éclatante de l'équipe de Toronto dans la compétition de reconnaissance d'objets «ImageNet ». La diminution des taux d'erreurs était telle qu'une véritable révolution.

Du jour au lendemain, la majorité des équipes de recherche en parole et en vision ont abandonné leurs méthodes préférées ensuite ils sont passés aux réseaux de neurones convolutifs ainsi qu'autres réseaux neuronaux. Dans ce contexte s'inscrit ce projet de fin d'année intitulé « Lung nodules detection and 3D reconstruction » qui ambitionne de développer un algorithme d'apprentissage profond basé sur les réseaux de neurones convolutifs. Ceci permettra la détection des nodules pulmonaires à partir des images 3D.

Dans ce travail, nous allons utiliser des algorithmes de Machine Learning qui sont déjà appliqués au problème de détection des images en comparant les résultats de plusieurs modèles allant du plus simple au plus sophistiqué que les ressources disponibles nous permettent. Nous essayerons par la suite de faire plusieurs optimisations en faisant varier les hyper paramètres qui constituent le modèle

afin de comprendre l'impact de chacun d'eux sur l'apprentissage et de choisir la valeur qui nous mène à améliorer la métrique.

Le présent rapport s'étale sur 4 chapitres. Dans le premier chapitre, nous présentons le cadre du projet. Le deuxième chapitre présente les ressources et environnement utilisés pour la conception de ce dernier. Dans le troisième chapitre, nous entamons le modèle de détection des nodules ainsi que ses résultats obtenus et le dernier chapitre est consacré à la reconstruction 3D.

---

## **Chapitre 1 : Contexte du projet**

---

### **1.1 Introduction**

Dans ce chapitre, nous présentons les nodules pulmonaires en termes médicaux, suivis par la comparaison des algorithmes de détection d'objets afin de choisir le plus efficace. Enfin, nous allons voir les notions nécessaires pour réaliser une reconstruction 3D.

### **1.2 Nodules pulmonaires**

#### **1.2.1 Définition d'un nodule pulmonaire**

Un nodule pulmonaire est un amas cellulaire plus ou moins arrondie, de moins de 3 cm de diamètre entouré de tissu pulmonaire sain. Au-delà de 3 cm, on ne parle plus de nodule mais de masse lorsque le diamètre est inférieur à 3 mm, on parle de micronodule.

On distingue les nodules malins (cancer pulmonaire primaire, métastase ou tumeur carcinoïde) des nodules bénins (infectieux, hamartome ou autre tumeur bénigne, malformation vasculaire ou nodule inflammatoire associé à une pathologie auto-immune, par exemple).

#### **1.2.2 Technique d'imagerie médicale**

Le terme tomodensitométrie, ou CT-scan, fait référence à une procédure d'imagerie aux rayons X informatisée dans laquelle un faisceau étroit de rayons X est dirigé vers un patient et rapidement tourné autour du corps, produisant des signaux qui sont traités par l'ordinateur de la machine pour générer des images en coupe - ou "tranches" - du corps. Ces coupes sont appelées images

tomographiques et contiennent des informations plus détaillées que les radiographies conventionnelles.

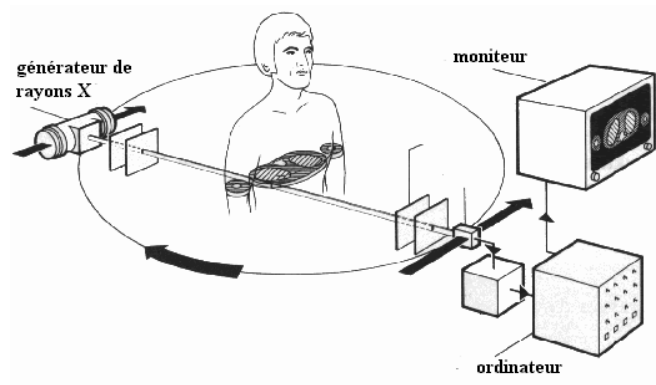


Figure 1 : Schéma de fonctionnement d'un TDM

Une fois qu'un certain nombre de coupes successives sont collectées par l'ordinateur de la machine, elles peuvent être “empilées” numériquement pour former une image tridimensionnelle du patient qui permet une identification et une localisation plus faciles des structures de base ainsi que d'éventuelles tumeurs ou anomalies.

Le nodule pulmonaire est défini au scanner comme une opacité focale entourée de parenchyme pulmonaire (tissu fonctionnel des poumons)

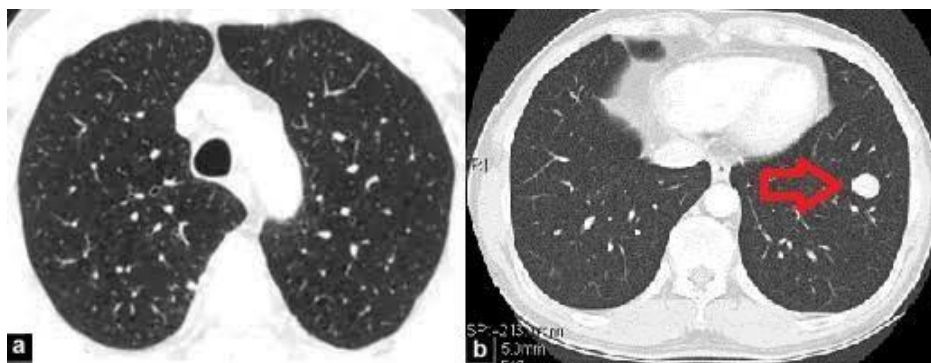


Figure 2 : Exemples d'un scanner thoracique sain (a) et scanner thoracique avec nodule (b)

Les images CT comportent des niveaux de gris qui traduisent les coefficients Hounsfield.

L'échelle des unités de Hounsfield (UH) est une échelle quantitative qui décrit l'incapacité relative de ces types de rayonnement électromagnétique de passer à travers un matériau particulier.

Compte tenu de la dynamique propre des performances de l'œil humain, il est nécessaire de se limiter à l'étude d'une fraction des densités qui peuvent s'étaler sur une large échelle de -1000 à +1000.

Matière	UH
Air	-1 000
Poumon	-500
Graisse	-100 à -50
Eau	0
Liquide cérébro-spinal	15
Rein	30
Sang	+30 à +45
Muscle	+10 à +40
Matière grise	+37 à +45
Matière blanche	+20 à +30
Foie	+40 à +60
Tissus mous	+100 à +300
Os	+700 (os spongieux) à +3 000 (os denses)

Figure 3: UH des matières courantes

### 1.2.3 Caractéristique d'un nodule

#### - La taille du nodule :

De façon générale, plus le nodule est grand plus il est suspect mais ceci n'est pas toujours vrai.

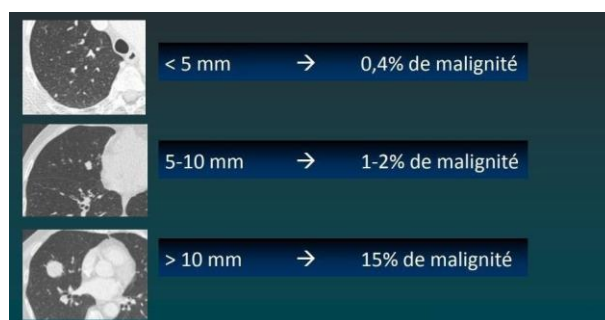


Figure 4: Différentes tailles de nodules

#### - La densité du nodule :

Concernant la densité, on distingue :

- Nodules solides (les plus fréquents et caractérisés par une densité tissulaire qui efface les contours des vaisseaux).
- Nodules mixtes (qui combinent les deux composantes).

- Nodules non solides (aspect en verre dépoli focal).

Les nodules solides sont les plus nombreux, mais les nodules mixtes sont associés au plus haut risque de malignité.

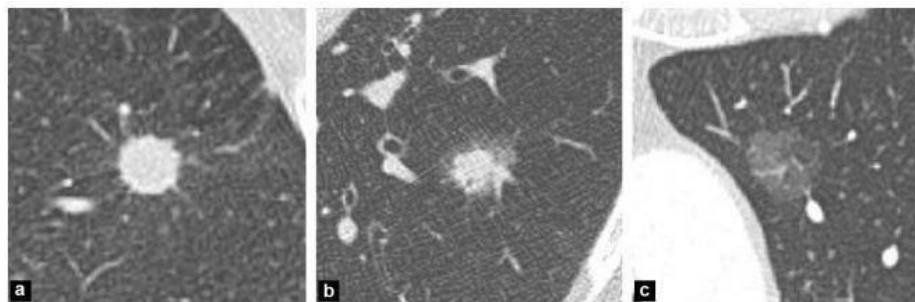


Figure 5: Exemples de nodules solide (a), semi-solide (b) et en verre dépoli pur (c).<sup>3</sup>

### La croissance du nodule :

Concernant la croissance, un nodule dont la taille augmente est associé à une haute probabilité d'être un cancer. A l'inverse, un nodule solide stable durant 2 ans ou un nodule mixte stable durant 5 ans ont une faible probabilité d'être un cancer.

### L'aspect de la bordure du nodule (irrégulière, voire spiculée versus lisse) :

Des contours arrondis bien limités « lisses » orientent plus vers une maladie bénigne, que vers un cancer primitif du poumon. Des contours irréguliers, « spiculés » c'est-à-dire présentant des prolongements orientent plutôt vers un cancer primitif.

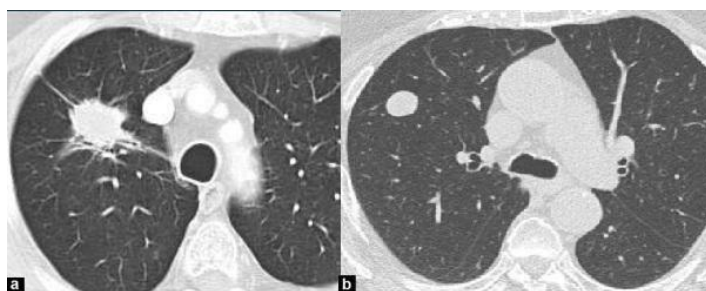


Figure 6: Exemple de nodule à bordure spiculée (a) versus à bordure lisse (b).<sup>3</sup>

## 1.2.4 Problématique

Annuellement, les radiologues doivent examiner des milliers d'images. Mais l'interprétation d'un scanner n'est pas toujours facile : « un nodule peut avoir 30 à 40 diagnostics différents »,

fait valoir le Dr Alain Livartowski, directeur des data de l'ensemble hospitalier de l'Institut

Curie (Paris). Nous constatons que le format, la taille et la localisation de la tumeur ne nous donnent pas une interprétation évidente sur sa nature. En plus, une erreur peut conduire à passer à côté d'un cancer ou à infliger une biopsie inutile, ce qui peut s'avérer décisif pour le patient. Ainsi, les tumeurs détectées sont en général dans un stade avancé ce qui diminue les chances de réussite du traitement.

En effet, les radiologues se trouvent souvent dans une situation où ils doivent prendre une décision le

plus tôt possible tout en conservant une faible probabilité d'erreur. C'est là que les logiciels d'intelligence artificielle entrent en jeu.

### 1.3 Deep Learning

#### 1.3.1 Concepts clés de l'Intelligence Artificielle

##### 1.3.1.1 Réseaux neuronaux convolutifs

Un réseau neuronal convolutif est un réseau neuronal puissant qui utilise des filtres pour extraire des caractéristiques des images. Il le fait également de manière à conserver les informations de position des pixels.

Une convolution est une opération mathématique appliquée sur une matrice. Cette matrice est généralement l'image représentée sous la forme de pixels/chiffres. L'opération de convolution permet d'extraire les caractéristiques de l'image.

Les réseaux neuronaux ont permis d'énormes avancées dans l'apprentissage automatique et sont la raison fondamentale du boom de l'apprentissage profond. Les réseaux neuronaux comme les CNN se sont révélés particulièrement efficaces pour travailler avec des données d'image.

Nous transmettons une image au réseau, qui la fait passer par diverses convolutions et couches de mise en commun. Enfin, nous obtenons la sortie sous la forme de la classe de l'objet.

Voyons comment nous pouvons résoudre un problème général de détection d'objets à l'aide d'un CNN.

1ere etape : Tout d'abord, nous prenons une image en entrée :



Figure 7: Entrée du CNN

2 eme etapes : Ensuite, nous divisons l'image en plusieurs régions :

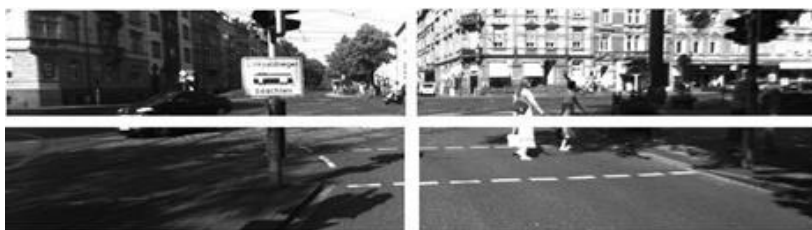


Figure 8 : Images d'entrées après division

3 eme etapes : Nous considérerons ensuite chaque région comme une image distincte.

4 eme etapes : Passer toutes ces régions (images) au CNN et les classer en différentes classes

5 eme etapes : Une fois que nous avons divisé chaque région dans sa classe correspondante, nous pouvons combiner toutes ces régions pour obtenir l'image originale avec les objets détectés :



Figure 9: Image de sortie

Une simple architecture de CNN peut etre représentée comme suit:

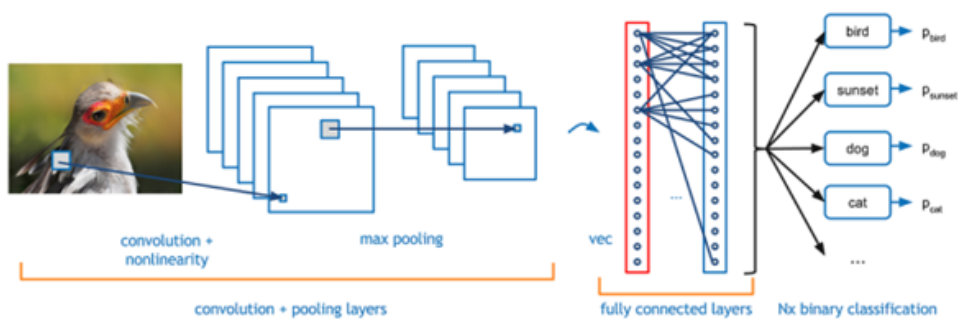


Figure 10: Architecture de CNN

### 1.3.1.2 Classification des images :

La classification d'images consiste à prédire la classe d'une image d'entrée en calculant la probabilité d'appartenance à une classe particulière.

probabilité d'appartenance de l'image à une classe particulière. En d'autres termes, il s'agit du processus d'attribution d'une étiquette à une image d'entrée parmi un ensemble de catégories

Il s'agit de l'une des questions clés de Computer vision, qui a un large éventail d'applications pratiques malgré sa simplicité.

### 1.3.1.3 Segmentation :

La segmentation a de nombreuses applications dans l'imagerie médicale (localisation de tumeurs, mesure de volumes de tissus, étude de l'anatomie, planification de la chirurgie, etc.), les voitures à



conduite autonome (localisation de piétons, d'autres véhicules, de feux de freinage, etc.), l'interprétation d'images satellites (bâtiments, routes, forêts, cultures), etc.

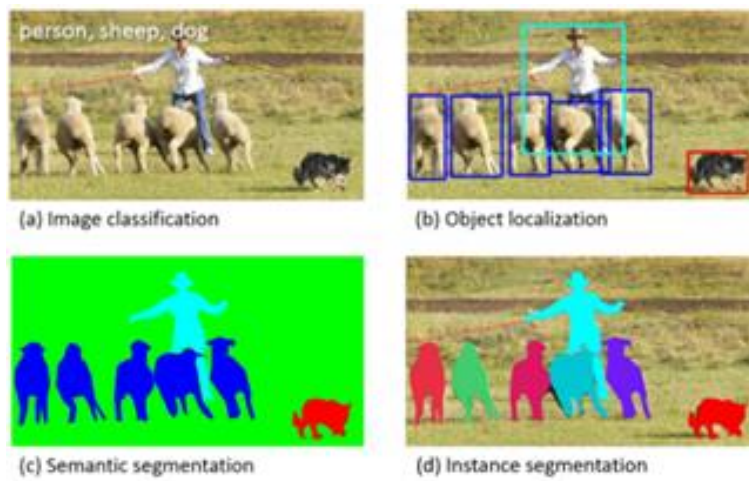


Figure 11: Applications de segmentation

La figure ci-dessus illustre quatre tâches courantes liées aux images :

- a. La classification, dans laquelle un modèle produit les noms des classes dans l'image (dans ce cas, personne, mouton et chien) ;
- b. La localisation d'objets, plus communément appelée "détection d'objets", dans laquelle un modèle fournit les coordonnées de la boîte englobante pour chaque objet de l'image
- c. La segmentation sémantique, dans laquelle le modèle attribue une étiquette de catégorie d'objet à chaque pixel de l'image. Dans cet exemple, les pixels des moutons sont colorés en bleu, ceux des chiens en rouge, ceux des humains en saumon et ceux de l'arrière-plan en vert. Remarquez que, bien qu'il y ait plusieurs moutons dans l'image, ils partagent tous la même étiquette.
- d. La segmentation d'instance, dans laquelle le modèle attribue une étiquette "objet individuel" à chaque pixel de l'image. Dans cet exemple, les pixels de chaque mouton individuel sont étiquetés

séparément. Au lieu d'avoir une classe de pixels générique " mouton ", nous avons maintenant cinq classes pour les cinq moutons représentés : mouton1, mouton2, mouton3, mouton4 et mouton5.

Voici une application de la segmentation sémantique et de la segmentation par instance à des images d'algues :

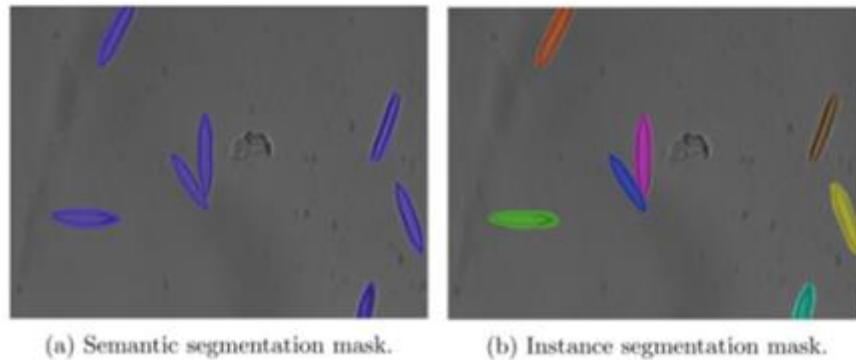


Figure 12: (a) Segmentation sémantique de Mask (b) Instance Segmentation Mask

### 1.3.1.4 Evaluation des performance

#### 1.3.1.4.1 Overfitting & Underfitting

Pour comprendre la cause d'une mauvaise performance du modèle, il est important de comprendre l'ajustement du modèle pour nous guider dans la prise de mesures correctives.

En examinant les erreurs de prédiction sur les données de et d'évaluation, nous pouvons déterminer si un modèle est sous-ajusté ou surajusté.

Underfitting : On parle de l'Underfitting lorsque le modèle ne capture pas correctement les caractéristiques des données et ne s'adapte pas bien aux données.

Overfitting : On parle de l'Overfitting lorsqu'un modèle apprend trop bien les détails, y compris le bruit, des données d'apprentissage, au point de nuire à la qualité des données et avoir un impact négatif sur les performances du modèle sur de nouvelles données (données d'évaluation) par conséquent, la capacité de généralisation du modèle est très faible.

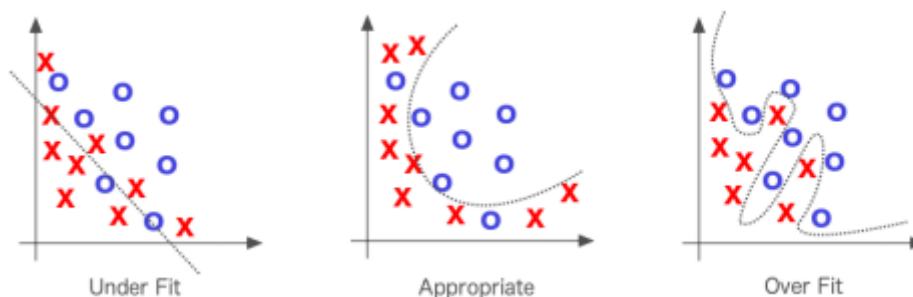


Figure 13: Démonstration de Data fitting

### 1.3.1.4.2 Métriques et hyperparameters

**Loss :** Une fonction de perte est une mesure de la précision avec laquelle le résultat attendu (la vérité de base) peut être prédit par un modèle. En entrée, la fonction de perte prend deux valeurs : la valeur de sortie du modèle et la valeur attendue de la vérité terrain. La sortie de la fonction de perte est une mesure de la qualité de la prédiction du résultat par le modèle. si la prédiction du modèle est parfaite, la perte sera nulle ; sinon, elle sera plus grande. L'objectif est de trouver un ensemble de poids et de biais sur tous les exemples qui minimisent la perte.

**Accuracy :** La précision est le rapport entre le nombre de valeurs correctement prédites et l'ensemble des prédictions effectuées. Il s'agit de la mesure la plus courante utilisée pour évaluer les problèmes de classification.

**Learning rate :** Le taux d'apprentissage est un hyper-paramètre qui contrôle à quel point nous modifions les poids de notre réseau par rapport au gradient de la perte. Plus la valeur est faible, plus nous descendons lentement la pente (voir figure II.5). La relation entre la vitesse d'apprentissage et les poids du modèle est la suivante :

$$w_{i,j} = w_{i,j} - \alpha \times \frac{dL}{dw_{i,j}}$$

Où  $\alpha$  est le taux d'apprentissage,  $w_{i,j}$  est le poids et  $L$  est la fonction de perte du modèle. De plus amples informations sur l'apprentissage et les hyperparamètres sont fournies dans l'annexe.

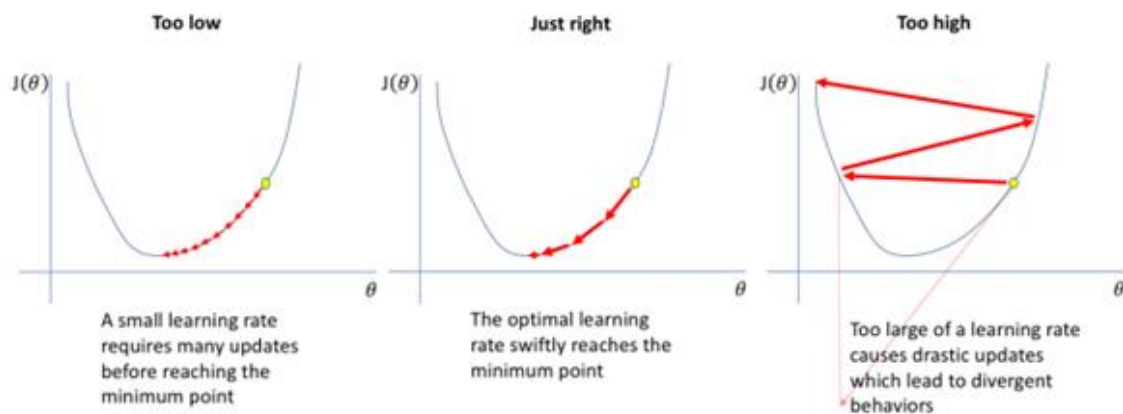


Figure 14: Réglage du Learning rate

## 1.3.2 Algorithmes de détection

Récemment, le domaine de l'intelligence artificielle a connu de nombreuses avancées grâce au deep learning et au traitement des images. Il est maintenant possible de reconnaître des images ou même de trouver des objets à l'intérieur d'une image.

### 1.3.2.1 Les algorithmes les plus utilisés

#### 1.3.2.1.1 RCNN : Region-based CNN

Pour résoudre le problème de la sélection d'un grand nombre de régions, RCNN a proposé une méthode où nous utilisons la recherche sélective pour extraire seulement 2000 régions de l'image et il les a appelées propositions de régions.

Donc, maintenant, au lieu d'essayer de classifier un grand nombre de régions, vous pouvez juste travailler avec 2000 régions. Ces 2000 propositions de régions sont générées en utilisant l'algorithme de recherche sélective qui est écrit ci-dessous.

L'algorithme RCNN propose un groupe de boîtes dans l'image et vérifie si l'une de ces boîtes contient un objet de ces boîtes contient un objet.

RCNN utilise la recherche sélective pour extraire ces boîtes d'une image (ces boîtes sont appelées régions).

Recherche sélective :

1. Générer une sous-segmentation initiale, nous générons de nombreuses régions candidates.
2. Utilisation d'un algorithme gourmand pour combiner récursivement les régions similaires en régions plus grandes.
3. Utiliser les régions générées pour produire les propositions finales de régions candidates.

Comment identifie-t-on les différentes régions ?

Il y a fondamentalement quatre régions qui forment un objet : des échelles, des couleurs, les textures, et l'enfermement. La recherche sélective identifie ces motifs dans l'image et, sur cette base, propose différentes régions et, sur cette base, propose différentes régions.

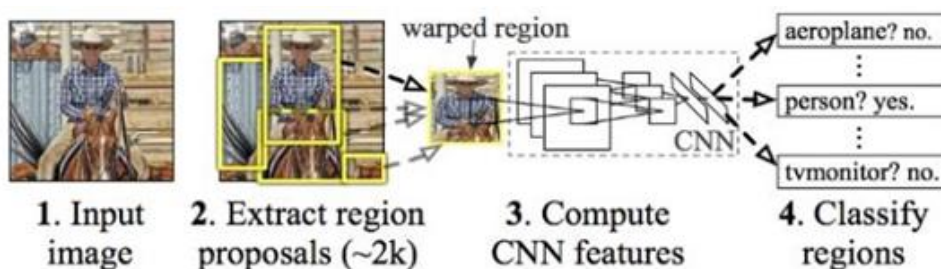


Figure 15: Détection par RCNN

Problèmes avec RCNN :

La formation d'un modèle RCNN est coûteuse et lente grâce aux étapes suivantes :

- ♦ Extraction de 2000 régions pour chaque image basée sur une recherche sélective.
- ♦ Extraction de caractéristiques à l'aide du RCNN pour chaque région de l'image. Supposons que nous ayons  $N$  images, alors le nombre de caractéristiques CNN sera de  $N \times 2000$  processus de détection d'objets à l'aide de RCNN comporte trois modèles :

- a. CNN pour l'extraction de caractéristiques
- b. Classificateur SVM linéaire pour l'identification des objets
- c. Modèle de régression pour resserrer les boîtes de délimitation.

Tous ces processus se combinent pour rendre le RCNN très lent. Il faut environ 40-50 secondes pour faire des prédictions pour chaque nouvelle image, ce qui rend essentiellement le modèle encombrant et pratiquement impossible à construire lorsqu'on est confronté à une situation de crise.

Solution : Fast RCNN

#### **1.3.2.1.2 Fast RCNN**

Le même auteur de l'article précédent (R-CNN) a résolu certains des inconvénients du R-CNN pour construire un algorithme de détection d'objets plus rapide, appelé Fast R-CNN. L'approche est similaire à l'algorithme R-CNN mais, au lieu d'alimenter le CNN avec les propositions de régions, nous alimentons le CNN avec l'image d'entrée pour générer des propositions de régions.

l'image d'entrée au CNN pour générer une carte de caractéristiques convolutives. A partir de la carte de caractéristiques convolutives, nous identifions la région des propositions et les déformons en carrés

et, en utilisant une couche de mise en commun des RoI, nous les remodelons en une taille fixe afin qu'ils puissent être introduits dans une couche entièrement connectée.

A partir du vecteur de caractéristiques RoI, nous utilisons une couche softmax pour prédire la classe de la région proposée ainsi que le décalage entre les deux de la région proposée ainsi que les valeurs de décalage pour la boîte englobante.

Ainsi, au lieu d'utiliser trois modèles différents (comme dans le RCNN), le RCNN rapide utilise un seul modèle qui extrait les caractéristiques de la région proposée, les divise en différentes classes, et renvoie simultanément les boîtes de délimitation pour les classes identifiées simultanément.

La raison pour laquelle "Fast R-CNN" est plus rapide que R-CNN est qu'il n'est pas nécessaire d'introduire 2000 propositions de régions dans le modèle parce qu'il n'est pas nécessaire d'alimenter le réseau de neurones convolutifs avec 2000 propositions de régions à chaque fois.

Au lieu de cela, l'opération de convolution n'est effectuée qu'une seule fois par image et une carte de caractéristiques est générée à partir de celle-ci.

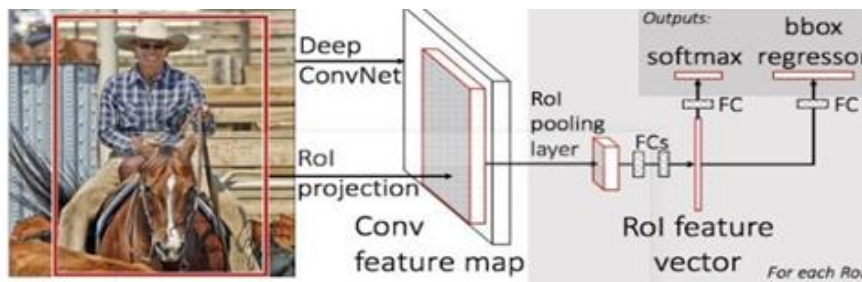


Figure 16: Détection par FAST RCNN

C'est ainsi que le RCNN rapide résout deux problèmes majeurs du RCNN, à savoir, passer une au lieu de 2 000 régions par image au ConvNet, et l'utilisation d'un seul modèle au lieu de trois pour l'extraction de caractéristiques, la classification et la génération de boîtes englobantes.

Les propositions de régions prédites sont ensuite remodelées en utilisant une image dans la région proposée et prédire les valeurs de décalage pour les boîtes.

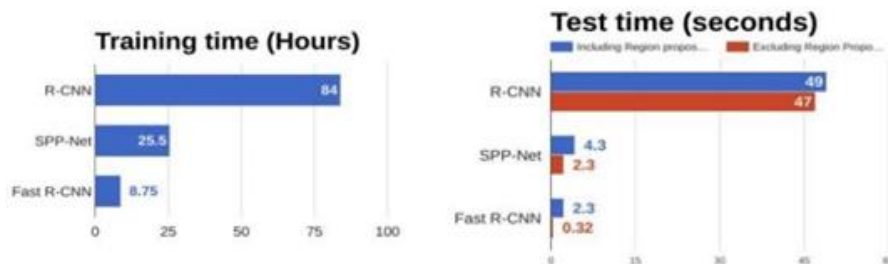


Figure 17: comparaison entre RCNN et Fast RCNN en temps d'exécution

## Problèmes avec Fast RCNN

Le RCNN rapide présente certains problèmes. Il utilise également la recherche sélective comme méthode de proposition pour trouver les régions d'intérêt, ce qui est un processus lent et processus lent et chronophage. Il faut environ 2 secondes par image pour détecter des objets, ce qui est bien mieux que le RCNN mais lorsque nous considérons de grands ensembles de données de la vie réelle de données réelles, même le fast RCNN ne semble plus aussi rapide.

Solution : Faster RCNN

### 1.3.2.1.3 Faster RCNN

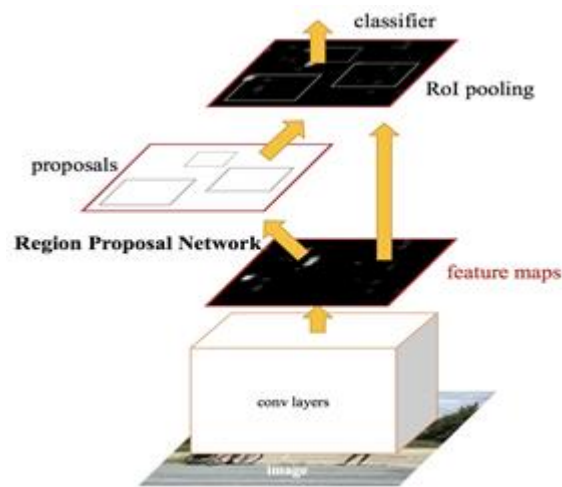


Figure 18: architecture de Faster RCNN

Les deux algorithmes ci-dessus (R-CNN et R-CNN rapide) utilisent la recherche sélective pour trouver les propositions de régions pour trouver les propositions de régions

La recherche sélective est un processus lent et processus lent et fastidieux qui affecte les performances du réseau.

Le RCNN rapide a mis au point un algorithme de détection d'objets qui élimine l'algorithme de recherche sélective qui élimine l'algorithme de recherche sélective et permet au réseau d'apprendre les propositions de régions comme pour le Fast R-CNN , l'image est fournie en entrée d'un réseau

convolutif qui fournit une image convolutive réseau convolutif qui fournit une carte de caractéristiques convolutives.

Au lieu d'utiliser un algorithme de recherche sélective sur la carte de caractéristiques pour identifier les propositions de régions, un réseau séparé est utilisé pour prédire les propositions de régions.

Les propositions de régions prédites sont ensuite remodelées à l'aide d'une couche de mise en commun de l'information sur les origines.

Les propositions de régions prédites sont ensuite remodelées en utilisant une image dans la région proposée et prédire les valeurs de décalage pour les boîtes.

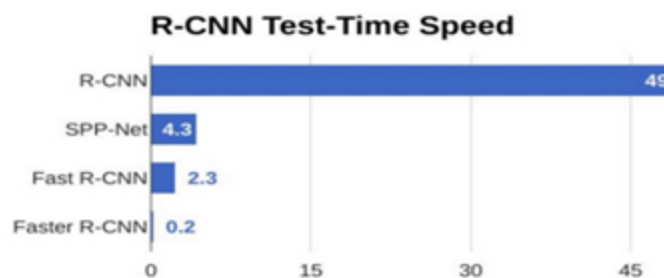


Figure 19: Comparaison entre les algorithmes précédents et Faster RCNN en temps d'exécution

A partir du graphique ci-dessus, vous pouvez voir que le R-CNN plus rapide est beaucoup plus rapide que ses prédécesseurs. Par conséquent, il peut même être utilisé pour la détection d'objets en temps réel.

### Problèmes avec Faster RCNN

Tous les algorithmes de détection d'objets dont nous avons parlé jusqu'à présent utilisent des régions pour identifier les objets. Le réseau ne regarde pas l'image complète en une seule fois, mais se concentre sur des parties de l'image de manière séquentielle mais se concentre sur des parties de l'image de manière séquentielle. Cela crée deux complications :

- ♦ L'algorithme nécessite de nombreux passages à travers une seule image pour extraire tous les objets
- ♦ Comme il y a différents systèmes qui travaillent les uns après les autres, la performances des systèmes suivants dépendent des performances des systèmes précédents.

#### 1.3.2.1.4 Mask RCNN

Mask R-CNN (Mask Regions with CNN) est une extension du modèle populaire de détection d'objets Faster R-CNN.

Dans la segmentation d'instance, au lieu de simplement créer une boîte de délimitation autour de l'objet détecté, les pixels appartenant au même objet sont regroupés. Dans la première partie du masque R-CNN, les régions d'intérêt (RoI) sont sélectionnées. Une RoI est une parcelle de l'image d'entrée qui



contient un objet avec une forte probabilité. Plusieurs régions d'intérêt sont identifiées pour chaque image d'entrée.

Dans la deuxième partie du masque R-CNN, illustrée dans la figure ci-dessous, chaque RdI est utilisée pour obtenir trois sorties de modèle :

la classe finale prédite pour ce RoI (la catégorie de l'objet, par exemple "personne")

la boîte englobante prédite finale obtenue à partir de ce RoI (coordonnées des coins de la boîte englobante, ce qui fournit une localisation de base de l'objet)

la segmentation finale prédite (par exemple, la silhouette de la personne, qui fournit une localisation très détaillée de l'objet).

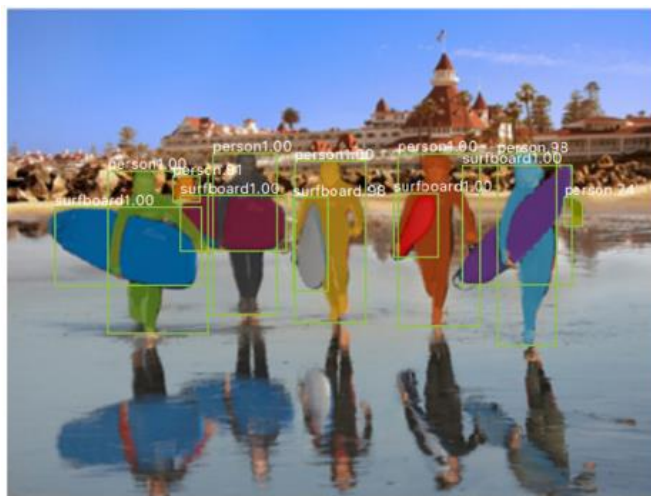


Figure 20: Segmentation par Mask R-CNN

### 1.3.2.1.5 YOLO - You Only Look Once

Tous les algorithmes de détection d'objets précédents utilisent des régions pour localiser l'objet dans l'image. Le réseau ne regarde pas l'image complète. Il examine plutôt les parties de l'image qui ont de fortes probabilités de contenir l'objet. YOLO ou You Only Look Once est un algorithme de détection d'objet très différent des algorithmes basés sur les régions vues précédemment. Dans YOLO, un seul réseau convolutif prédit les boîtes englobantes et les probabilités de classe pour ces boîtes.

Le fonctionnement de YOLO est le suivant : nous prenons une image et la divisons en une grille  $S \times S$ , dans chacune des grilles nous prenons  $m$  boîtes de délimitation. Pour chaque boîte de délimitation, le réseau produit une probabilité de classe et des valeurs de décalage pour la boîte de délimitation. Les

boîtes de délimitation dont la probabilité de classe est supérieure à une valeur seuil sont sélectionnées et utilisées pour localiser l'objet dans l'image

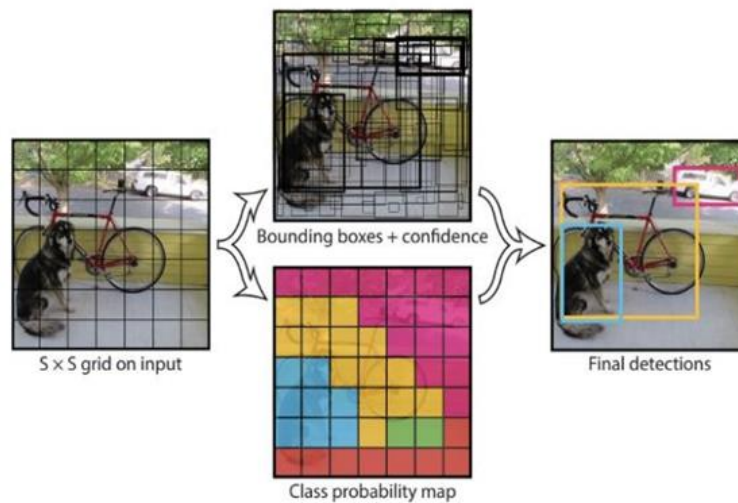


Figure 21: Principe de détection par YOLO

Le principe de YOLO est le suivant : on prend une image et on la divise en une grille  $S \times S$ , dans chacune des grilles, nous prenons  $m$  boîtes de délimitation. Pour chacune des boîtes de délimitation, le réseau produit une probabilité de classe et des valeurs de décalage pour la boîte de délimitation. Les boîtes englobantes dont la probabilité de classe supérieure à une valeur seuil sont sélectionnées et utilisées pour localiser l'objet dans l'image.

### 1.3.2.1.6 U-NET

L'U-Net est un type de réseau neuronal convolutif (CNN). L'idée de base de l'U-Net est de réaliser la tâche suivante :



Figure 22: Principe de fonctionnement de U-Net

Étant donné une image d'entrée, dans ce cas une image microscopique de cellules en niveaux de gris, le modèle U-Net produit un masque binaire de 1 et 0, où 1 indique une cellule et 0 indique le fond (y compris les frontières entre les cellules). Notez qu'il s'agit d'une tâche de segmentation sémantique car toutes les cellules reçoivent la même étiquette de "cellule" (c'est-à-dire que nous n'avons pas

d'étiquettes différentes pour distinguer les différentes cellules individuelles, comme nous le ferions pour la segmentation par instance).

L'idée principale est de compléter un contracting network par couches successives, les opérations de pooling sont remplacées par des opérateurs de suréchantillonnage. Par conséquent, ces couches augmentent la résolution de la sortie. De plus, une couche convolutionnelle successive peut alors apprendre à assembler une sortie précise à partir de cette information.

Une modification importante dans U-Net est qu'il existe un grand nombre de canaux de fonctions dans la partie de rééchantillonnage, ce qui permet au réseau de propager des informations de contexte vers des couches de résolution supérieure. En conséquence, le chemin d'expansion est plus ou moins symétrique à la partie contractante et donne une architecture en forme de «U». Le réseau utilise uniquement la partie valide de chaque convolution sans aucune couche entièrement connectée. Pour prédire les pixels dans la zone de bordure de l'image, le contexte manquant est extrapolé en reflétant l'image d'entrée. Cette stratégie de mosaïque est importante pour appliquer le réseau aux images volumineuses, car sinon la résolution serait limitée par la mémoire du processeur graphique

### 1.3.2.2 Comparaison des algorithmes

- comparaison entre les algorithmes de famille RCNN:

Algorithm	Features	Prediction time / image	Limitations
CNN	Divides the image into multiple regions and then classify each region into various classes.	–	Needs a lot of regions to predict accurately and hence high computation time.
RCNN	Uses selective search to generate regions. Extracts around 2000 regions from each image.	40-50 seconds	High computation time as each region is passed to the CNN separately also it uses three different model for making predictions.
Fast RCNN	Each image is passed only once to the CNN and feature maps are extracted. Selective search is used on these maps to generate predictions. Combines all the three models used in RCNN together.	2 seconds	Selective search is slow and hence computation time is still high.
Faster RCNN	Replaces the selective search method with region proposal network which made the algorithm much faster.	0.2 seconds	Object proposal takes time and as there are different systems working one after the other, the performance of systems depends on how the previous system has performed.

-YOLO est plusieurs fois plus rapide (45 images par seconde) que les autres algorithmes de détection d'objets algorithmes de détection d'objets. Les limites de l'algorithme YOLO sont les suivantes :

Il a du mal à détecter les petits objets dans l'image par exemple, il peut avoir des difficultés à détecter une volée d'oiseaux. Ceci est dû aux contraintes spatiales de l'algorithme.

- U-NET est utilisé pour la segmentation sémantique alors que Mask RCNN est utilisé pour La segmentation d'instance

### 1.3.2.3 Algorithme choisis

Dans notre projet , on a décidé d'implémenter le modèle UNET pour effectuer la segmentation sémantique des nodules pulmonaires .

## 1.4 Reconstruction 3D

Dans cette section, nous détaillons la méthode proposée et les données qui comprennent des fichiers CT avec diverses métadonnées pour cette étude. La figure 1 montre le schéma de la méthode proposée. Tout d'abord, nous améliorons chaque tranche d'image CT en appliquant une égalisation d'histogramme. Ensuite, les tranches d'images CT améliorées sont assemblées en trois dimensions en empilant les images. Pour reconstituer les informations entre les tranches et lisser l'image, nous appliquons une interpolation trilineaire. Comme les paramètres de l'image peuvent varier pendant l'acquisition, nous redimensionnons l'image 3D reconstruite pour générer des images de même taille.

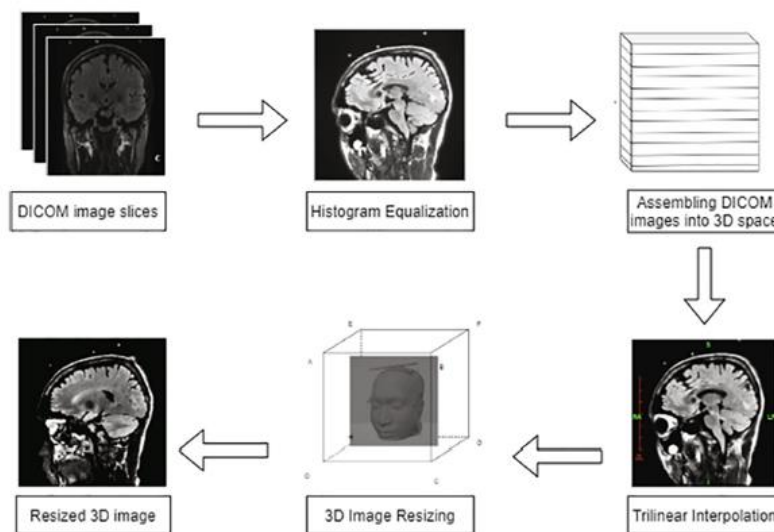


Figure 23: principe de reconstruction 3D

### 1.4.1 Égalisation de l'histogramme

Les images médicales CT (dicom et TDM ) doivent souvent être prétraitées avant d'être utilisées dans des méthodes telles que les algorithmes d'apprentissage automatique et d'apprentissage profond. Pour faciliter le prétraitement, les images CT peuvent être améliorées. À cette fin, nous utilisons l'égalisation d'histogramme, qui est généralement utilisée pour l'amélioration des images en niveaux de gris. Comme les voxels des images CT sont codés par intensité, ils sont similaires à des images en niveaux de gris avec un canal pour la couleur.

L'égalisation d'histogramme utilise la fonction de distribution cumulative (CDF) de l'intensité. La CDF reflète la probabilité d'occurrence du niveau de pixel  $i$  dans l'image  $x$  ( $px(i)$ ) comme suit :

$$p_x(i) = p(x = i) = \frac{n_i}{n}, 0 \leq i < L \quad (1)$$

où L est la valeur maximale du pixel.

Comme une image IRM CT ressemble à une image en niveaux de gris dont l'intensité (v) est comprise entre 0 et 255 pour chaque voxel, L représente le niveau maximal de 256. Les résultats de probabilité de l'équation (1) sont ensuite utilisés pour calculer la FCD par valeur d'intensité :

$$cdf_x(v) = \sum_{j=0}^v p_x(x = j) \quad (2)$$

L'équation (2) fournit la CDF par valeur d'intensité. L'égalisation de l'histogramme de la valeur d'intensité, h(v), est donnée par

$$h(v) = \text{round}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}}\right) \quad (3)$$

Où cdf min est la valeur CDF minimale qui est supérieure à 0.

La valeur d'intensité égalisée est ensuite utilisée pour remplacer la valeur originale et obtenir l'image améliorée, qui présente un contraste plus élevé que l'image originale.

### 1.4.2 Assemblage d'images médicale dans un espace 3d

Les tranches d'images CT sont empilées pour construire une représentation 3D de l'image IRM du cerveau. Par conséquent, la résolution de l'image 3D inclut le nombre de tranches (H) en plus de L et W lors de l'acquisition des scans IRM.

L'image CT IRM cérébrale 3D obtenue a une taille de L W H, et donc la résolution des images 3D reconstruites peut différer selon les images CT acquises. Dans cette étude, les images DICOM ont été obtenues de différents hôpitaux et sont donc susceptibles d'avoir un nombre différent de coupes par patient. Par conséquent, nous redimensionnons les images 3D reconstruites à la même résolution pour des raisons d'uniformité.

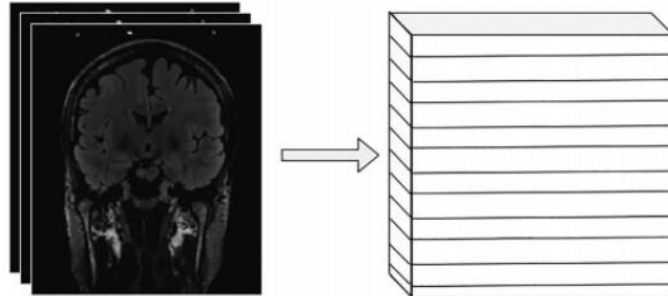


Figure 24: Assemblage des images CT pour les représenter en 3D

### 1.4.3 Interpolation trlinéaire

La figure 4 montre un voxel comme un cube contenant une valeur d'intensité (v) en trois dimensions. La valeur d'intensité v du voxel est contenue dans les sommets ABCDEFGH, et la valeur n'est pas homogène. Dans les images CT, les valeurs aux sommets représentent l'intensité de la coordonnée donnée.

Ainsi, pour obtenir des valeurs intermédiaires, on applique la formule d'interpolation suivante :

$$L_{tx} = (1 - t_{tx}) \cdot v_{x1} + t_{tx} \cdot v_{x2} \quad (4)$$

Où :

Vx = intensité au point X

L'interpolation trilinéaire utilise l'équation (4) de manière récursive au moins trois fois. A partir de l'Eq. (4), on obtient la valeur d'interpolation (L) de deux coordonnées le long de l'axe X (tx) du même côté.

La figure 5 montre la cible d'interpolation txyz et l'étape nécessaire pour obtenir sa valeur. Après avoir calculé tx à l'aide de l'équation (4), chaque tx est utilisé pour calculer txz. Enfin, chaque txz est utilisé pour calculer txyz en utilisant également l'équation (4).

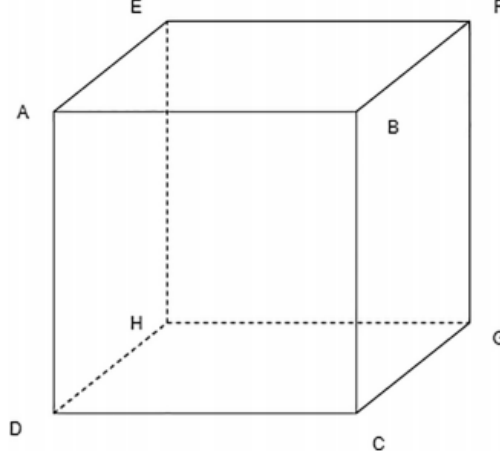


Figure 25: Visualisation du Voxel

#### 1.4.4 Redimensionnement de l'image 3D

L'assemblage des tranches d'images CT en une représentation 3D illustrée à la Fig. 3 nécessite un redimensionnement pour que toutes les images 3D reconstruites aient la même résolution. À cette fin, nous proposons la méthode de redimensionnement illustrée à la Fig. 6.

La figure 6 montre une image 3D contenant un voxel T. Comme nous empilons les images comme indiqué sur la figure 3, l'image 3D résultante a  $S_x = L$ ,  $S_y = W$  et  $S_z = H$ . Par conséquent, le nombre de voxels de l'image 3D est  $S_x S_y S_z$ . Soit T un voxel d'intensité  $v$  en coordonnées  $(x, y, z)$ .

Pour redimensionner l'image résultante en un cube de taille souhaitée,  $(D_x; D_y; D_z)$ , nous rééchantillonnons l'intensité de chaque voxel comme suit :

$$\sum_{j=(0,0,0)}^i v_j = \frac{((x_j, y_j, z_j) - \frac{(D_x, D_y, D_z)}{2})}{K} + \frac{S_x, S_y, S_z}{2} \quad (5)$$

$$\text{Avec: } K = \min \left( \frac{D_x}{S_x}, \frac{D_y}{S_y}, \frac{D_z}{S_z} \right) \quad (6)$$

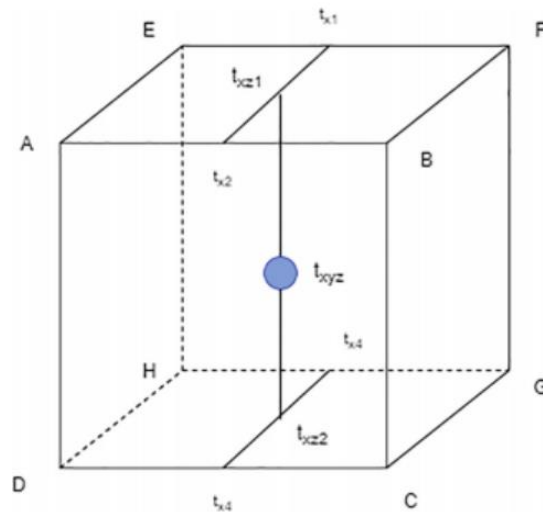


Fig. 5. Voxel with interpolated values.

Figure 26: Voxel avec interpolation

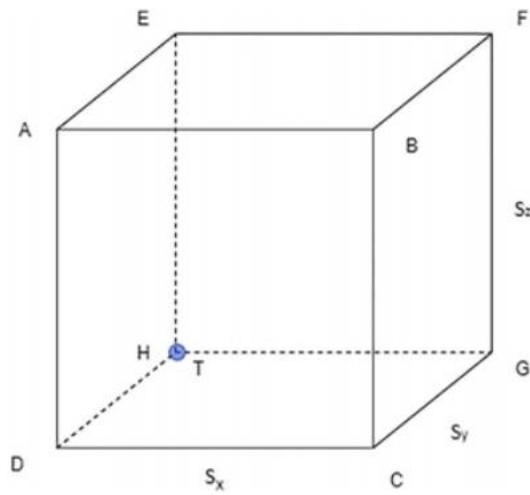


Fig. 6. 3D cube containing voxel T.

Figure 27: Cube 3D contenant un voxel

## 1.5 Conclusion

Dans ce chapitre, nous avons présenté le cadre général de ce projet tout en définissant les notions principales qui vont nous servir comme des prérequis pour les chapitres suivants.

---

## **Chapitre 2 : Ressources et environnements**

---

### **2.1 Introduction**

Dans ce deuxième chapitre, nous décrivons diverses sources de données contenant des scans médicaux pulmonaires et nous les comparons afin de faire le meilleur choix pour notre projet.

Dans la deuxième partie, nous décrivons les environnements d'apprentissage profond disponible et expliquons notre choix de configuration.

### **2.2 Environnements techniques**

#### **2.2.1 Les différentes approches d'environnements**

##### **2.2.1.1 Limitations des ressources matérielles**

La première approche la plus couramment utilisée pour former des modèles d'apprentissage automatique est l'apprentissage local. Il s'agit d'entraîner l'algorithme en utilisant le matériel d'une machine physiquement disponible disposant de ressources de calcul importantes (CPU, GPU, RAM). Cela implique également de configurer l'environnement logiciel afin que les paquets et modules requis soient installés avant le début de l'apprentissage.

Malheureusement, dans notre cas, nous n'avons pas eu accès à des ressources matérielles suffisantes pour former nos modèles localement : PC Intel (R) Core (TM) i7 8th gen, 8Go de RAM, système d'exploitation windows 10

##### **2.2.1.2 Google Colab : Gratuit mais limité**

Colab permet à n'importe qui d'écrire et d'exécuter le code Python de son choix par le biais du navigateur. C'est un environnement particulièrement adapté au machine learning, à l'analyse de données et à l'éducation. En termes plus techniques, Colab est un service hébergé de notebooks Jupyter qui ne nécessite aucune configuration et permet d'accéder gratuitement à des ressources informatiques, dont des GPU.



Cependant, bien que cet outil en ligne soit idéal pour prototyper ou tester des projets d'apprentissage automatique à petite échelle, il n'est pas le mieux adapté aux projets à grande échelle comme le nôtre en raison des limitations de ressources suivantes :

- Mémoire limitée : Colab fournit 12 Go de mémoire vive, ce qui est souvent insuffisant pour calculer les paramètres d'un grand modèle, en particulier pour les réseaux neuronaux.



- Stockage limité : Cet outil ne fournit que 50 Go de disque dur ce qui, dans notre cas, ne nous permet pas de stocker notre jeu de données LUNA 16 et d'autres fichiers du projet.
- GPU limité : L'accès aux GPU les plus rapides est très restreint, et les limites d'utilisation sont bien inférieures à celles appliquées par Colab Pro

### 2.2.1.3 Google Colab PRO

	Prix	GPU	Temps de session	Mémoire RAM
Colab	gratuit	K80	Jusqu'à 12 heures	12GB
Colab Pro	\$9.99/mois (avant taxe)	T4 & P100	Jusqu'à 24 heures	25GB

Les ressources de Colab Pro nous suffisent pour réaliser le projet

## 2.2.2 Outils et modules

### 2.2.2.1 Python

Python est un langage de programmation interprété, de haut niveau, dynamique et polyvalent, largement adopté par la communauté de la science des données.



En raison de sa courbe d'apprentissage plus facile et de ses bibliothèques utiles, il s'est imposé comme l'un des choix les plus populaires en science des données. En fait, il dispose de bibliothèques dédiées aux différentes étapes des projets comme Pandas, Matplotlib, scikit-learn, ainsi que certaines bibliothèques avancées comme TensorFlow et Keras pour l'apprentissage profond.

### 2.2.2.2 Keras & TensorFlow

**TensorFlow** est une bibliothèque logicielle libre pour le flux de données et la programmation différentiable dans une variété de tâches. Elle est utilisée pour les applications d'apprentissage automatique, comme les réseaux neuronaux. Google l'utilise également pour la recherche et la production.



**Keras** est une API de haut niveau utilisée pour la construction et l'entraînement des modèles de Deep Learning, le prototypage, la recherche avancée et la production, et présente trois avantages principaux : Conviviale, modulaire, facile à étendre.

### 2.2.2.3 SimpleITK

ITK est un système multiplateforme à code source ouvert qui fournit aux développeurs une large gamme de logiciels d'analyse d'images. SimpleITK fait partie de ces outils. Il s'agit d'une couche simplifiée construite au-dessus d'ITK pour faciliter son utilisation dans les domaines du prototypage rapide, de l'éducation et des langages interprétés.



### 2.2.2.4 Git & GitHub

**Git** : Git est de loin le système de contrôle de version le plus largement utilisé aujourd'hui. Git est un projet open source avancé, qui est activement maintenu. De plus en plus de projets logiciels reposent sur Git pour le contrôle de version, y compris des projets commerciaux et en open source. Les développeurs qui travaillent avec Git sont bien représentés dans le pool de talents disponible, et la solution fonctionne bien sur une vaste gamme de systèmes d'exploitation et d'environnements de développement intégrés (IDE).

**Github** : est un fournisseur d'hébergement Internet pour le développement de logiciels et le contrôle de versions utilisant Git. Il offre les fonctionnalités de contrôle de version distribué et de gestion du code source (SCM) de Git, plus ses propres caractéristiques.

### 2.2.2.5 Python packages



Figure 28: les packages de python utilisés

- Pandas : Pandas est une bibliothèque open source qui fournit des structures de données de langage de programmation Python hautes performances et faciles à utiliser et des outils d'analyse de données.
- NumPy : NumPy est le package de base avec Python pour le calcul scientifique. Il peut également être utilisé comme un conteneur de données générique multidimensionnel efficace.
- Matplotlib : Matplotlib est une bibliothèque de traçage 2D en Python qui produit des figures de qualité pour les publications sur toutes les plateformes, dans une variété de formats papier et d'environnements interactifs.
- Scikit-learn : Scikit-learn est une bibliothèque gratuite d'apprentissage automatique pour Python. Elle propose divers algorithmes de classification, de régression et de regroupement. Elle est conçue pour interagir avec les bibliothèques numériques et scientifiques de Python : NumPy et SciPy.
- Opencv : est une bibliothèque graphique. Elle est spécialisée dans le traitement d'images, que ce soit pour de la photo ou de la vidéo.

## 2.3 Base de données

Au cours de notre recherche d'un ensemble de données de scanners pulmonaires approprié pour notre projet, nous avons trouvé de nombreuses sources disponibles publiquement. Ci-dessous, nous énumérons et comparons les trois bases de données les plus utilisées, principalement en fonction de leur taille, du nombre de scans et du processus d'annotation.

### 2.3.1 Comparaison des bases de données

Data source	NIH Chest X-ray	LIDC-IDRI	LUNA 16
Dataset size	45.7 GB	124 GB	66.7 GB
Number of CT series	NaN	1,018	888
Number of images	100,000+	244,527	226,805
Number of patients	30,000+	1,010	888
Labels	14 (multiple lung diseases)	4 (nodule<3mm, nodule>3mm, non-nodule>3mm)	2 (non-nodule, nodule>3mm)
Annotation process	The image labels are mined from radiology reports using NLP techniques	4 expert radiologists (XML format)	4 expert radiologists (CSV format)
Image file format	png	dicom	mhd/raw
Pros and cons	(+) straightforward manipulation of files  (-) image format and annotations not reliable (-) X-ray more difficult to interpret than CT	(+) reliable medical image format and annotations (+) large number of images  (-) time-consuming to download and process due to size	(+) reliable medical image format and annotations (+) still large enough

Comme le résume le tableau ci-dessus, nous avons constaté que la base de données de LUNA 16 est la plus adaptée aux exigences de notre projet et aux ressources disponibles. Il a une taille raisonnable qui reste assez facile à télécharger et à traiter, sans sacrifier le nombre total de coupes CT incluses.

Il a également été annoté par 4 radiologues experts. Cela, ainsi que le format d'image médicale Méta Image garantit la fiabilité des informations fournies.

### 2.3.2 Présentation du dataset choisis : LUNA16

LUNA16 est un nouveau défi de détection utilisant l'ensemble de données LIDC-IDRI susmentionné.

Le défi LUNA 16 est entièrement ouvert. Il utilise la base de données LIDC-IDRI qui est accessible au public, en excluant les scans dont l'épaisseur de coupe dépasse 2,5 mm. Au total, 888 scanners sont inclus. La base de données LIDC-IDRI contient également des annotations qui ont été recueillies par 4 radiologues expérimentés au cours d'un processus d'annotation en deux phases.

Chaque radiologue a marqué les lésions qu'il a identifiées comme étant non nodulaires, nodulaires < 3 mm, et nodulaires >= 3 mm. La norme de référence du défi LUNA 16 est constituée de tous les nodules >= 3 mm acceptés par au moins 3 des 4 radiologues. Les annotations qui ne sont pas incluses dans le standard de référence sont les non-nodules, les nodules < 3 mm et les nodules annotés par seulement 1 ou 2 radiologues.

La base de données complète est divisée en 10 sous-ensembles qui sont disponibles sous forme de fichiers zip compressés.

Les images CT sont stockées en format Méta Image (mhd/raw) 1 dans chaque sous-ensemble. Un fichier binaire ".raw" distinct pour les données pixel est stocké pour chaque fichier ".mhd".

Le fichier d'annotation est un fichier ".csv" (valeurs séparées par des virgules) qui contient une constatation par ligne. Chaque ligne contient le Series Instance ID du scan, la position x, y et z de chaque découverte en coordonnées mondiales et le diamètre correspondant en mm. Le fichier d'annotation contient 1186 nodules.

	A	B	C	D	E	F	G	H	I	
1	seriesuid,coordX,coordY,coordZ,diameter_mm									
2	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860,-128.6994211,-175.3192718,-298.3875064,5.651470635									
3	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860,103.7836509,-211.9251487,-227.12125,4.224708481									
4	1.3.6.1.4.1.14519.5.2.1.6279.6001.100398138793540579077826395208,69.63901724,-140.9445859,876.3744957,5.786347814									
5	1.3.6.1.4.1.14519.5.2.1.6279.6001.100621383016233746780170740405,-24.0138242,192.1024053,-391.0812764,8.143261683									

Figure 29 : Fichier d'annotation

Le fichier des candidats (candidates.csv) est un fichier csv qui contient une lésion (candidat) par ligne. Chaque ligne contient le nom du scan, la position x, y, et z de chaque candidat en coordonnées mondiales, et la classe correspondante (0 : non nodule et 1 : nodule). Le fichier des candidats contient 754975 candidats

	A	B	C	D	E	F	G	H	
1	seriesuid,coordX,coordY,coordZ,class								
2	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860,68.42,-74.48,-288.7,0								
3	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860,-95.20936148,-91.80940617,-377.4263503,0								
4	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860,-24.76675476,-120.3792939,-273.3615387,0								
5	1.3.6.1.4.1.14519.5.2.1.6279.6001.100225287222365663678666836860,-63.08,-65.74,-344.24,0								

Figure 30: Fichier des candidats

## 2.4 Conclusion

Les outils choisis pour développer notre projet sont : Google colab Pro comme environnement, Python comme langage de programmation ainsi que ses packages cités dans ce chapitre, Tensor Flow/Keras comme bibliothèque pour concevoir notre modèle de deep learning et SimpleITK pour manipuler les images d'extension Metalmage (.mhd et .raw). Sans oublier Git et Github pour la gestion du projet

Nous pouvons maintenant commencer à réaliser le modèle de détection des nodules ainsi que la reconstruction 3D.

---

## **Chapitre 3 : Détection des nodules**

---

### **3.1 Introduction**

Ce troisième chapitre comporte les différentes étapes pour la conception d'un modèle de détection des nodules ainsi que les résultats obtenus

### **3.2 Réalisation du modèle**

#### **3.2.1 Préparation des données**

De manière générale, en data science, les données n'arrivent jamais sous une forme directement exploitable par le data scientist. Le Deep Learning ne fait pas exception. En réalité, le plus gros travail consiste à appliquer le prétraitement convenable sur les données : forme, composition etc... En effet, lors de la phase d'entraînement le réseau ne saura pas vers quelles valeurs se tourner pour apprendre des caractéristiques de cette image, et lors de la phase de prédiction, nous n'aurons aucun moyen de vérifier rigoureusement ce que fait le réseau dessus. Ceci implique donc de faire un filtrage des données.

##### **3.2.1.1 Changement de base et rééchantillonnage**

Tout d'abord, nous devons comprendre les systèmes de coordonnées en imagerie médicale. En fait, il existe trois systèmes de coordonnées fréquemment utilisés pour les applications d'imagerie : les systèmes de coordonnées du monde, anatomiques et d'image. En bref : le système de coordonnées du monde est un système cartésien dans lequel un modèle (un scanner ou un patient) est positionné ; le système de coordonnées de l'image décrit comment une image a été acquise par rapport au système de coordonnées du monde .

Par conséquent, nous devons convertir les coordonnées de l'emplacement des nodules des coordonnées du monde en coordonnées image ou voxel en appliquant la transformation suivante :

$$voxelCoord = \frac{(worldCoord - Origin)}{Spacing}$$

où Origin représente la position du premier voxel (0, 0, 0) dans le système de coordonnées anatomiques, et Spacing est la distance entre les voxels le long de chaque axe.

Puis, nous avons définis une méthode de rééchantillonnage pour rendre les images isomorphes, l'espacement (spacing) par défaut est [1, 1, 1]mm

##### **3.2.1.2 Segmentation des poumons/nodules**

Après avoir lu le CT Scan, la première étape du prétraitement est la segmentation des structures pulmonaires car il est évident que les régions d'intérêt se trouvent à l'intérieur des poumons. Il est visible que les poumons sont les régions les plus sombres dans les scans CT. Les régions claires à l'intérieur des poumons sont les vaisseaux sanguins ou l'air. Un seuil -400 HU est utilisé à tous les endroits car les expériences ont montré qu'il fonctionne parfaitement. Nous augmentons les structures pulmonaires à partir de chaque tranche de l'image CT Scan et essayons de ne pas perdre les éventuelles régions d'intérêt attachées à la paroi pulmonaire. Certains nodules peuvent être attachés à la paroi pulmonaire.

Résultats obtenus :

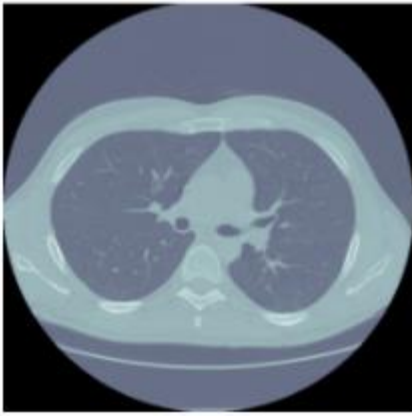


Figure 32: Poumons Avant segmentation



Figure 31: Poumons après segmentation

Après avoir segmenté les structures pulmonaires à partir des images de tomodensitométrie, notre tâche consiste à trouver les régions candidates aux nodules, car l'espace de recherche est très vaste. Les expériences ont montré que toutes les régions d'intérêt ont une intensité  $> -400$  HU. Nous avons donc utilisé ce seuil pour filtrer les régions les plus sombres. Cela réduit le nombre de candidats par un grand nombre et préserve toutes les régions importantes avec un rappel élevé.

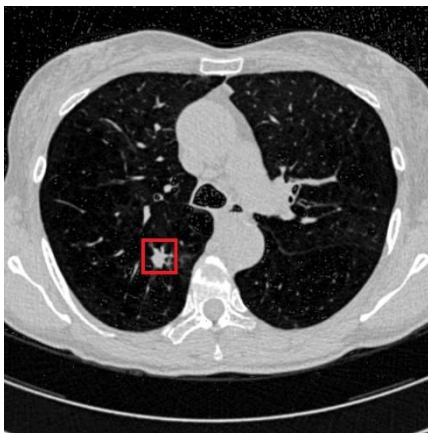


Figure 33: Image d'un patient avec nodule

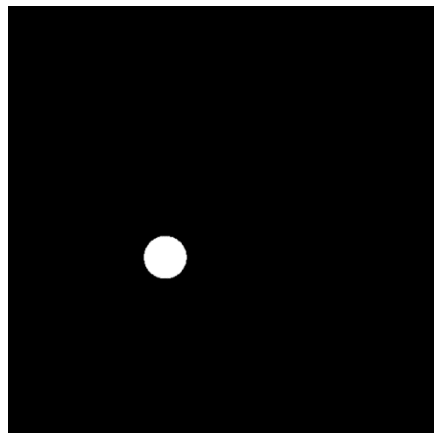


Figure 34: Masque respective (Après segmentation)

#### **3.2.1.3 Normalisation**

La normalisation est un processus qui modifie la plage des valeurs d'intensité des pixels.

Cette étape est nécessaire car les valeurs des pixels des images doivent être mises à l'échelle entre 0 et 1 avant de fournir les images en entrée d'un modèle de réseau neuronal d'apprentissage profond pendant l'apprentissage ou l'évaluation du modèle.

Elle est aussi utilisée pour la transformation en image PNG.

Nos valeurs vont actuellement de -1024 à environ 2000. Tout ce qui est supérieur à 400 n'est pas intéressant pour nous, car il s'agit simplement d'os ayant une radiodensité différente. Un ensemble de seuils couramment utilisés dans la compétition LUNA16 pour normaliser sont entre -1000 et 400.

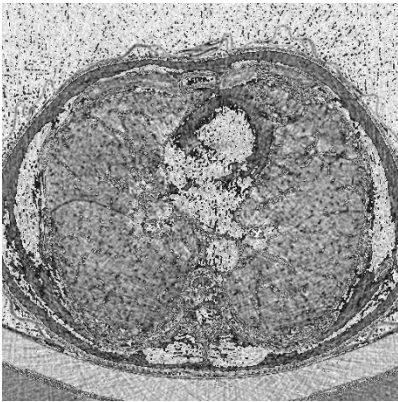


Figure 35 Image avant normalisation



Figure 36 Image après normalisation

#### 3.2.1.4 Transformation en PNG

Nous avons transformé les images médicales et leurs images segmentées respectives en plusieurs images .PNG car c'est l'extension la plus simple à utiliser pour l'apprentissage du modèle

Nous avons normalisé les images entre 0 et 1 puis nous avons multiplié les pixels par 255 pour pouvoir les enregistrer sous format .PNG

#### 3.2.1.5 Augmentation des données

Il existe plusieurs techniques qui permettent d'augmenter le nombre des exemples. Parmi eux, nous citons la translation de l'image, le cropping qui permet de zoomer l'image sur des endroits, la rotation, le changement d'intensité, l'effet miroir (Flipping) etc. Notez bien que ces méthodes permettent non seulement le balancement des exemples d'entraînement, mais aussi elles présentent une sorte de régularisation, permettant la réduction de l'effet de surapprentissage.

#### 3.2.2 Architecture du réseau (UNET)

Dans cette partie l'accent sera mis sur l'architecture de l'UNET.  
Ce schéma montre la configuration de la formation d'un modèle U-Net .

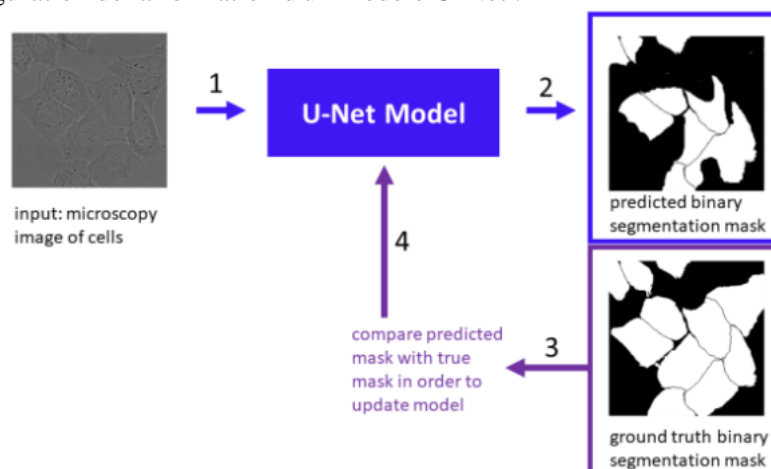


Figure 37 Entrée et sortie de modèle U-Net



Avant que le modèle ne soit complètement entraîné, pour une image d'entrée donnée, il produira un masque de segmentation binaire qui présente des problèmes, par exemple le "masque de segmentation binaire prédit" illustré dans la figure ci-dessus, où certaines cellules sont manquantes ou ont des bords incorrects. La fonction de perte de l'U-Net compare le masque prédit avec le masque de référence, afin de permettre la mise à jour des paramètres qui permettront au modèle d'effectuer une meilleure segmentation sur le prochain exemple d'entraînement.

Voici l'architecture de U-Net :

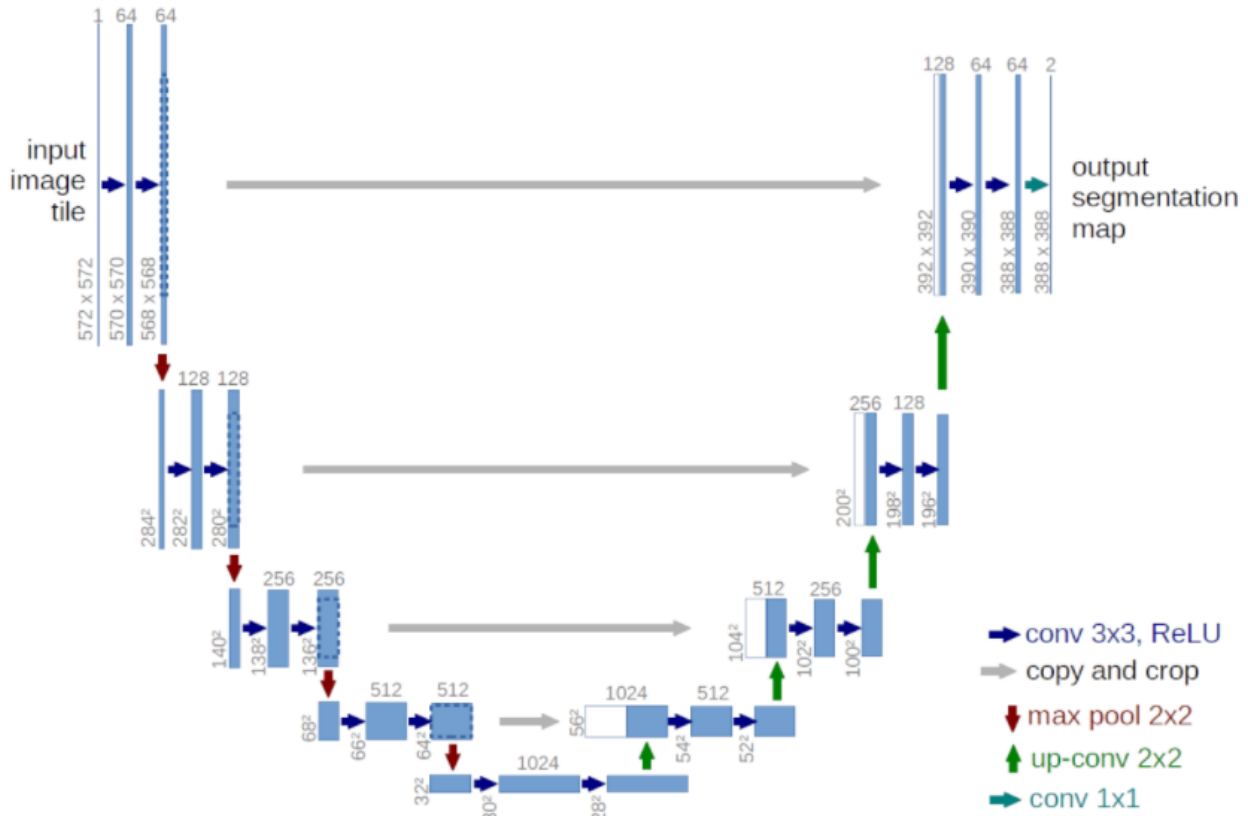


Figure 38 Architecture du modèle U-Net

Sur cette figure, le nom "U-Net" est évident, car le diagramme d'architecture montre une forme en U. L'idée de base du U-Net est d'obtenir d'abord une représentation en basse dimension de l'image par le biais d'un réseau neuronal convolutif traditionnel, puis de sur-échantillonner cette représentation en basse dimension pour produire la carte de segmentation finale en sortie.

Il est utile d'utiliser la clé du diagramme d'architecture fournie dans le coin inférieur droit, qui explique à quelles opérations correspond chaque type de flèche.

Le U-Net se compose d'un "chemin contractuel" et d'un "chemin expansif" :

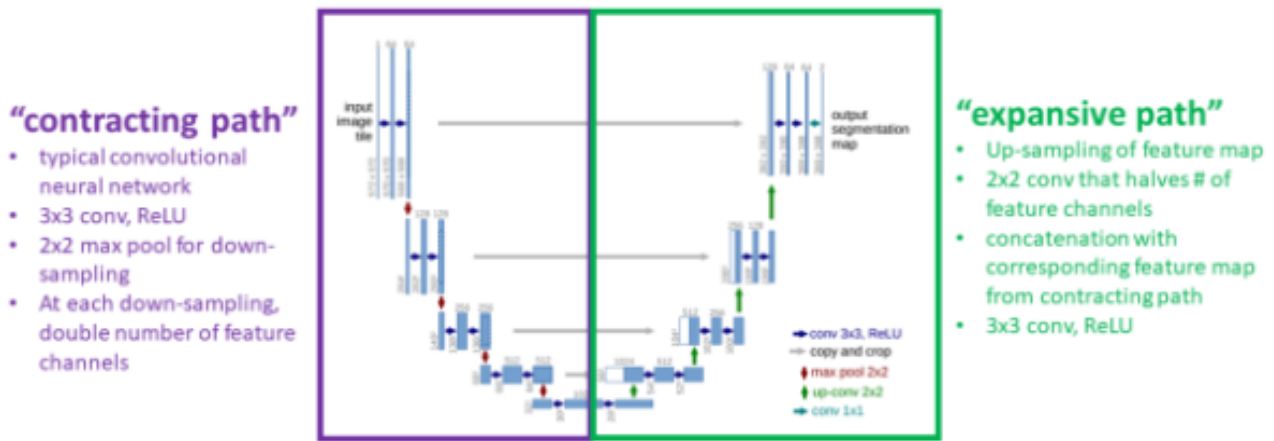


Figure 39 Les deux chemins de U-Net

Le "chemin contractant" est un CNN traditionnel, qui produit la représentation à faible dimension. Le "chemin expansif" échantillonne la représentation pour produire la carte de segmentation finale. Les flèches grises représentent des opérations de copie, dans lesquelles les cartes de caractéristiques à haute résolution du "chemin contractant" sont copiées et concaténées aux cartes de caractéristiques du "chemin expansif" pour faciliter l'apprentissage d'une segmentation à haute résolution par le réseau.

Le U-Net ne comporte aucune couche entièrement connectée, ce qui signifie qu'il s'agit d'un réseau entièrement convolutif.

Production de la carte de segmentation prédite : Convolution 1 x 1 et Softmax Pixel-Wise

Au niveau de la dernière couche du réseau U-Net, une convolution 1 x 1 est appliquée pour mettre en correspondance chaque vecteur de caractéristiques à 64 canaux avec le nombre de classes souhaité, qui, dans cet article, est considéré comme étant deux classes (cellule/arrière-plan). Si vous souhaitez utiliser le U-Net pour la segmentation sémantique avec plusieurs classes, par exemple 6 classes (chien, chat, oiseau, tortue, vache, arrière-plan), le vecteur de caractéristiques de 64 canaux peut être mappé sur 6 classes (6 canaux).

Voici un gros plan de la figure 1 montrant la toute dernière partie du "chemin expansif" où un vecteur caractéristique de 64 canaux est produit par une opération [conv 3x3, ReLU], et est finalement mappé à un vecteur caractéristique de 2 canaux (cellule vs. arrière-plan) en utilisant une flèche de couleur sarcelle représentant une convolution 1 x 1 :

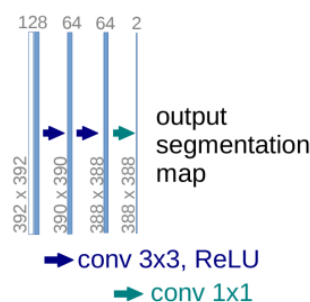


Figure 40 Sortie de chemin expansif

On a choisi le modèle custom-UNET du package `keras_unet.models` en choisissant comme entrée `input_shape(512,512,1)` tq la taille de l'image d'entrée est de (512,512) et `z=1` représente la couleur de l'image qui est le gris pour notre cas.

On a une seule classe car on est en train d'implémenter une segmentation sémantique. Concernant les filtres qui représentent les poids appris des convolutions on a choisi 64 Le pourcentage de dropout est de 0.2. Enfin notre `output_activation` est la fonction 'sigmoid' il s'agit d'une fonction mathématique qui présente une courbe caractéristique en forme de S. Il existe un certain nombre de fonctions sigmoïdes courantes, telles que la fonction logistique, la tangente hyperbolique et l'arctangente.

Soit ce tableau récapitulatif du modèle qui indique la force de la relation entre le modèle et la variable dépendante obtenu suite à la fonction `model.summary()`:

```
keras-unet init: TF version is >= 2.0.0 - using 'tf.keras' instead of 'Keras'
```

---

Model: "model"

Layer (type)	Output Shape	Param #	Connected to
<hr/>			
input_1 (InputLayer)	[(None, 512, 512, 1)]	0	
conv2d (Conv2D)	(None, 512, 512, 64)	640	input_1[0][0]
spatial_dropout2d (SpatialDropo	(None, 512, 512, 64)	0	conv2d[0][0]
conv2d_1 (Conv2D)	(None, 512, 512, 64)	36928	spatial_dropout2d[0][0]
max_pooling2d (MaxPooling2D)	(None, 256, 256, 64)	0	conv2d_1[0][0]
conv2d_2 (Conv2D)	(None, 256, 256, 128)	73856	max_pooling2d[0][0]
spatial_dropout2d_1 (SpatialDro	(None, 256, 256, 128)	0	conv2d_2[0][0]
conv2d_3 (Conv2D)	(None, 256, 256, 128)	147584	spatial_dropout2d_1[0][0]
max_pooling2d_1 (MaxPooling2D)	(None, 128, 128, 128)	0	conv2d_3[0][0]
conv2d_4 (Conv2D)	(None, 128, 128, 256)	295168	max_pooling2d_1[0][0]
spatial_dropout2d_2 (SpatialDro	(None, 128, 128, 256)	0	conv2d_4[0][0]
conv2d_5 (Conv2D)	(None, 128, 128, 256)	590880	spatial_dropout2d_2[0][0]
max_pooling2d_2 (MaxPooling2D)	(None, 64, 64, 256)	0	conv2d_5[0][0]
conv2d_6 (Conv2D)	(None, 64, 64, 512)	1180160	max_pooling2d_2[0][0]
spatial_dropout2d_3 (SpatialDro	(None, 64, 64, 512)	0	conv2d_6[0][0]
conv2d_7 (Conv2D)	(None, 64, 64, 512)	2359808	spatial_dropout2d_3[0][0]
max_pooling2d_3 (MaxPooling2D)	(None, 32, 32, 512)	0	conv2d_7[0][0]
conv2d_8 (Conv2D)	(None, 32, 32, 1024)	4719616	max_pooling2d_3[0][0]
spatial_dropout2d_4 (SpatialDro	(None, 32, 32, 1024)	0	conv2d_8[0][0]
conv2d_9 (Conv2D)	(None, 32, 32, 1024)	9438208	spatial_dropout2d_4[0][0]
conv2d_transpose (Conv2DTranspo	(None, 64, 64, 512)	2897664	conv2d_9[0][0]
concatenate (Concatenate)	(None, 64, 64, 1024)	0	conv2d_transpose[0][0] conv2d_7[0][0]

conv2d_10 (Conv2D)	(None, 64, 64, 512)	4719104	concatenate[0][0]
conv2d_11 (Conv2D)	(None, 64, 64, 512)	2359808	conv2d_10[0][0]
conv2d_transpose_1 (Conv2DTrans	(None, 128, 128, 256	524544	conv2d_11[0][0]
concatenate_1 (Concatenate)	(None, 128, 128, 512	0	conv2d_transpose_1[0][0] conv2d_5[0][0]
conv2d_12 (Conv2D)	(None, 128, 128, 256	1179904	concatenate_1[0][0]
conv2d_13 (Conv2D)	(None, 128, 128, 256	590808	conv2d_12[0][0]
conv2d_transpose_2 (Conv2DTrans	(None, 256, 256, 128	131200	conv2d_13[0][0]
concatenate_2 (Concatenate)	(None, 256, 256, 256	0	conv2d_transpose_2[0][0] conv2d_3[0][0]
conv2d_14 (Conv2D)	(None, 256, 256, 128	295040	concatenate_2[0][0]
conv2d_15 (Conv2D)	(None, 256, 256, 128	147584	conv2d_14[0][0]
conv2d_transpose_3 (Conv2DTrans	(None, 512, 512, 64)	32832	conv2d_15[0][0]
concatenate_3 (Concatenate)	(None, 512, 512, 128	0	conv2d_transpose_3[0][0] conv2d_1[0][0]

Figure 41 Tableau récapitulatif du modèle Custom Unet

### 3.2.3 Initialisation des paramètres

Dans le domaine de deep learning, l'initialisation des poids joue un rôle important et peut créer une différence en assurant une convergence du modèle dans un temps raisonnable. En effet, assigner des poids au réseau avant de commencer l'apprentissage semble être un processus aléatoire puisque nous ne savons pas le produit du modèle avec nos données. Une bonne façon est d'assigner les poids à partir d'une distribution gaussienne. Évidemment, cette distribution aurait une moyenne nulle et une variance finie. Cela nous aide à empêcher le signal d'exploser à une valeur élevée ou à disparaître à zéro. En d'autres termes, nous devons initialiser les poids de telle sorte que la variance reste constante tout au long du réseau. Ce processus est noté l'initialisateur Xavien (connu aussi sous le nom « Glorot initialization »).

### 3.2.4 La fonction de perte (binary\_crossentropy)

afin d'optimiser la descente de gradient on a opté pour l'algorithme

Adam :

ADAM (Adaptive Moments) est l'un des algorithmes les plus récents et les plus efficaces pour l'optimisation de descente de gradient. Il permet d'adapter automatiquement le taux d'apprentissage pour chaque paramètre. Pour la mise à jour des paramètres, il combine les deux algorithmes cités précédemment « Momentum » et « RMSprop »

### 3.2.5 Optimisation du calcul

Contrairement à TensorFlow, Keras n'offre pas la fonctionnalité de paralléliser l'entraînement sur plusieurs machines. Cependant, l'implémentation de data-generator invoque l'utilisation de la fonction `fit_generator`. Cette fonction nous permet d'introduire la fonctionnalité de parallélisme des GPUs sur une même machine en fixant le paramètre `use_multiprocessing` à « True ». Dans ce cas, le nombre de cœurs utilisés est fixé par le paramètre « workers ».

Ainsi, nous avons contribué à améliorer le temps d'exécution des GPUs eux-mêmes grâce à une approche de lecture des données par batch. En d'autres termes, au lieu d'accéder de façon répétitive à la mémoire pour lire une seule donnée, nous avons augmenté la quantité des données (i.e le nombre des batches) chargées à la fois par le GPU.

### 3.2.6 Optimisation du traitement

Après avoir créé le modèle, le compiler et l'entraîner, il faut trouver les bonnes valeurs des hyper paramètres qui nous permettent d'avoir un modèle qui maximise la métrique AUC. Les tests seront faits sur l'architecture illustrée auparavant.

Le nombre d'itérations sera fixé selon le test et la convergence du modèle. La taille de batch est fixée à 8. La taille du cache RLU est mise à 1. La fonction de perte est la «binary\_crossentropy».

### 3.3 Tests et résultats

#### 3.3.1 Données en entrées

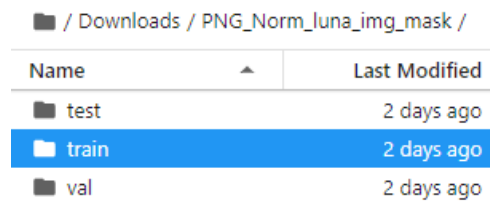
L'ensemble de données LUNA 16 a été fourni dans 10 fichiers compressés. Chacun de ces fichiers .zip contenait des sous-ensembles d'images CT de taille à peu près égale.

Nous avons au début créé 10 nouveaux fichiers dans lesquels on a seulement gardé des images PNG contenant des nodules obtenues à partir des images CT et le fichier annotation.csv .

Nous avons utilisé leur structure de sous-ensembles pour diviser l'ensemble de données en :

- 70% dédiés à l'entraînement (subset 0 1 2 3 4 5 6 )
- 20% dédiés à la validation (subset 7,8)
- 10% dédiés au test (subset 9 )

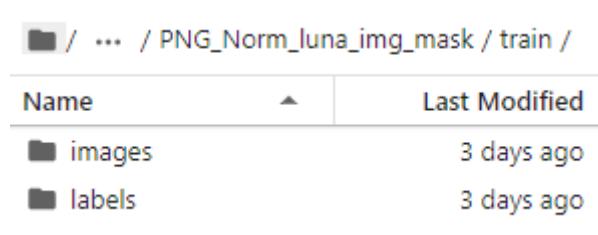
	Nombres des images PNG
TRAIN SET	837
VALIDATION SET	229
TEST SET	100



Name	Last Modified
test	2 days ago
train	2 days ago
val	2 days ago

Figure 42 Division de Dataset en :Train , validation et Test

Dans chacun des dossiers train, test et val on a créé 2 dossiers images et labels



Name	Last Modified
images	3 days ago
labels	3 days ago

Figure 43Contenu de chacun des dossiers

Dans ces dossiers on trouve 1 dossier qui représente la classe des nodules appelé "class" comme présenté ci-dessous :

PNG\_Norm\_luna\_img\_mask > train > images > class

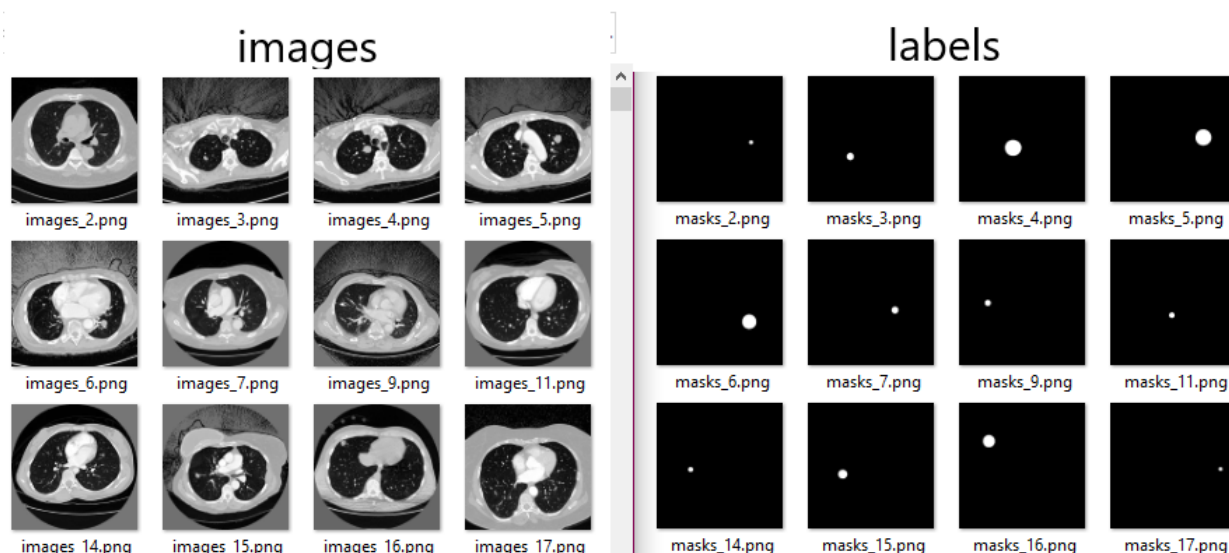


Figure 44 Exemple de contenus des dossiers images/class et labels/class

Entrées du modèle UNET : les 2 entrées qui sont les images et les étiquettes ( labels ) étant les masques des nodules générés, présentés précédemment dans le chapitre 2

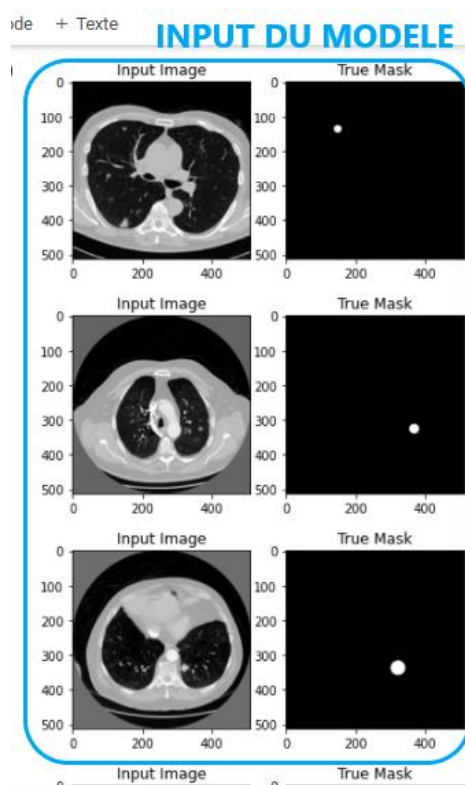


Figure 45 Les entrées de notre modèle

### 3.3.2 Résultats Finals

Pour NUM\_OF\_EPOCHS = 50 on eu des résultats acceptables. Soit un des meilleurs résultats obtenus tel que le Predicted mask présenté dans la figure est obtenu par la fonction : `pred_mask=model_i.predict (image):`

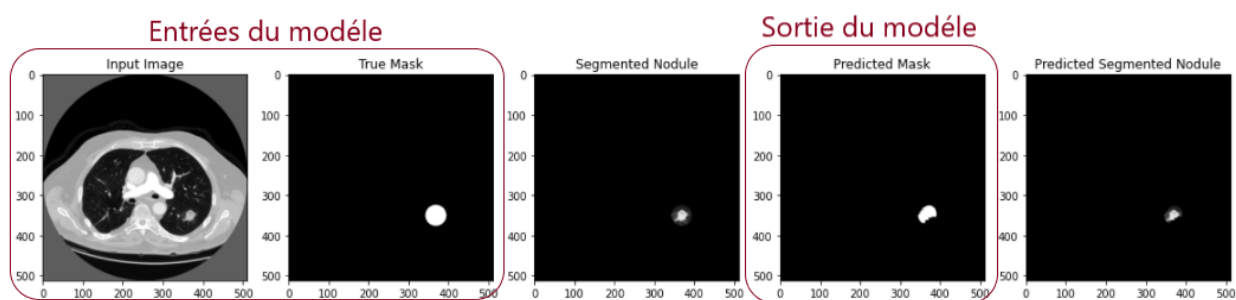


Figure 46 Resultat satisfaisante de modèle après training

On peut aussi avoir des résultats un peu erronés comme la figure suivante :

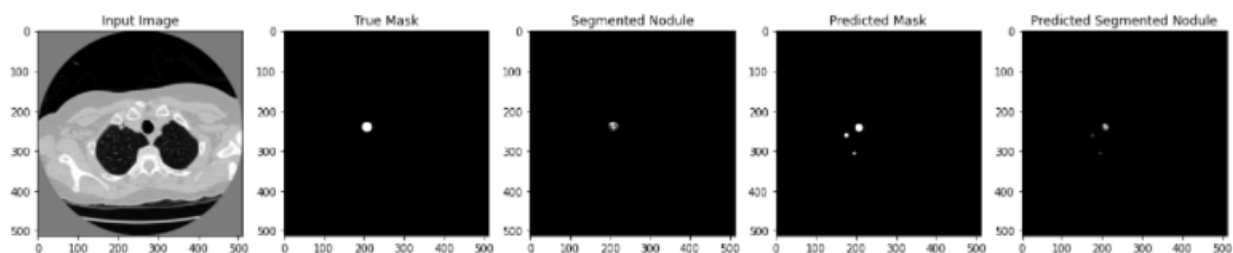


Figure 47 résultat un peu erroné

---

## **Chapitre 4 : Reconstruction 3D**

---



## 4.1 Introduction

Ce dernier chapitre est consacré à la reconstruction 3D ainsi que les étapes pour sa réalisation. Il admet deux parties ; la reconstruction 3D des poumons et celle des nodules.

## 4.2 Reconstruction des poumons

### 4.2.1 Positionner le scan

Avant de commencer la reconstruction 3D il vaut mieux placer le scanner en position verticale, de sorte que la tête du patient soit en haut, face à la caméra. D'où l'utilisation de la fonction transpose (2,1,0).

### 4.2.2 Segmentation des poumons

Afin d'assurer une bonne visualisation des poumons on a eu recours à la segmentation présentée précédemment dans le chapitre 3. C'est-à-dire on a laissé seulement l'intérieur des poumons (threshold=-500 HU ) pour clarifier la vue.

### 4.2.3 Algorithme de création de maillage approximatif : Marching Cubes

Pour visualiser les poumons en 3D on a effectué un Tracé 3D du scan ( Plot)

Malheureusement, les paquets disponibles dans ce type d'image skimage sont très limités dans ce sens, en effet il' y a seulement le module measure donc nous allons utiliser la fonction `skimage.measure.marching_cubes_classic(volume)` Les applications principales de cet algorithme sont du domaine de la visualisation médicale, comme la reconstruction de surfaces à partir des images issues de scanners ou d'IRM Ce qui est notre cas.

Principe de fonctionnement :

Marching cubes est un algorithme permettant d'extraire un maillage approximatif de surface 2D d'un volume 3D. Cela peut être conceptualisé comme une généralisation 3D des isolignes sur les cartes topographiques ou météorologiques. Il fonctionne en itérant à travers le volume, à la recherche de régions qui traversent le niveau d'intérêt. Si de telles régions sont trouvées, des triangulations sont générées et ajoutées à un maillage de sortie. Le résultat final est un ensemble de sommets et un ensemble de faces triangulaires.

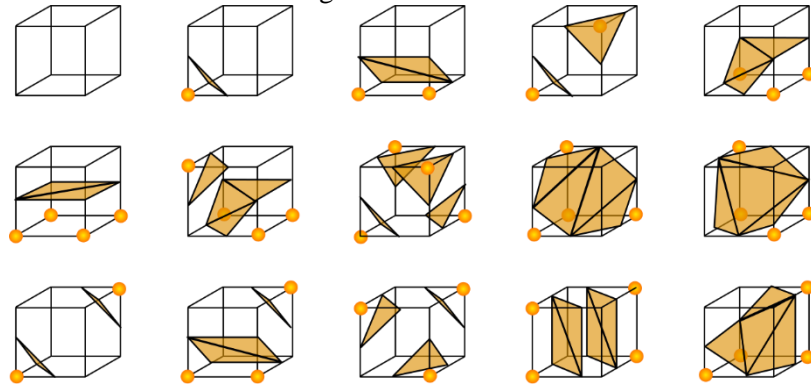


Figure 48 Les 15 cas possibles de configuration des polygones

### 4.2.4 Création du plot 3D par Poly3DCollection

Après pour créer les polygones 3D qui représentent le volume mesh (surface maillé) on a utilisé la fonction `matplotlib.collections.Poly3DCollection()` de la bibliothèque `mpl_toolkits.mplot3d.art3d`

Enfin suite à la fonction `matplotlib.pyplot` ayant comme entrée le CT scan on obtient notre plot3D.

### 4.2.5 Résultats obtenus

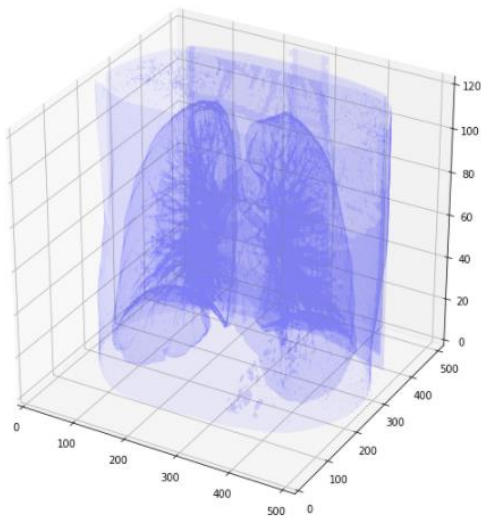


Figure 49 Reconstruction 3D sans segmentation

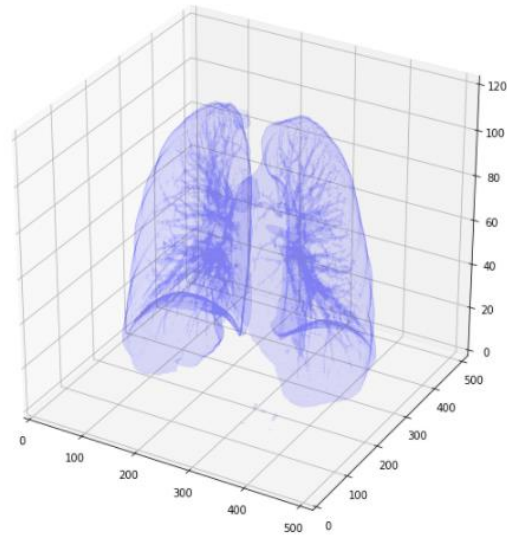


Figure 50 Reconstruction 3D avec segmentation

### 4.3 Reconstruction 3D des nodules pulmonaires

Cette partie consiste à extraire seulement les tranches d'image où le nodule est présent.

Il faut commencer par appliquer un rééchantillonnage (resampling) comme présenté précédemment dans le chapitre 3 pour visualiser les images avec des dimensions correctes.

Après, on parcourt le fichier annotation.csv où on trouve les coordonnées de chaque nodule et son diamètre..

Et on encadre les tranches d'images par rapport au diamètre du nodule comme le montre la figure suivante

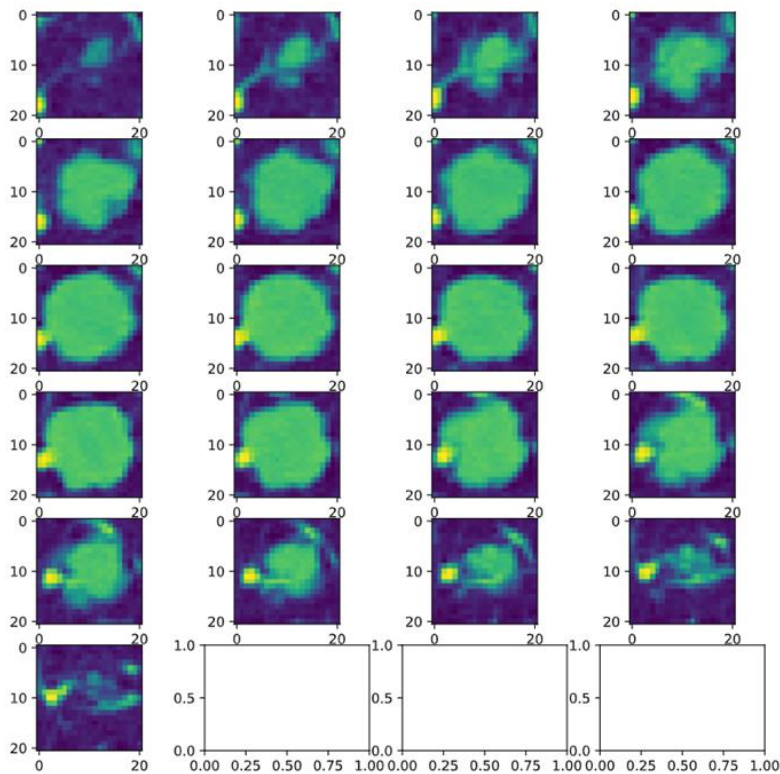


Figure 51 Recadrage d'une image médicale d'un patient ayant un nodule de diamètre 19mm

La dernière étape est d'enregistrer ces images CT recadrés en fichier '.mhd' '.raw' et '.npy' et d'utiliser un logiciel de visualisation 3D : "Image J"

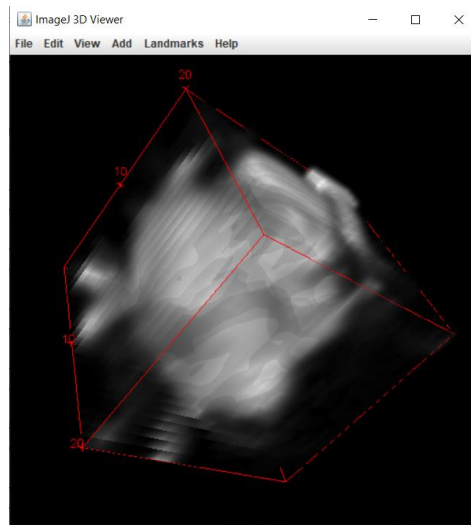


Figure 52 Visualisation d'un nodule de diamètre 19mm

# CONCLUSION GENERALE ET PERSPECTIVES

L'intelligence artificielle dans le domaine des soins de santé a suscité beaucoup d'attention ces dernières années en raison de l'énorme potentiel qu'elle offre pour résoudre certains problèmes de santé les plus urgents dans le monde.

Notre projet est au cœur de cette vague d'innovation, s'attaquant à l'une des priorités les plus urgentes du domaine : le cancer du poumon. En veillant à ce que davantage de cas de cancer soient détectés à un stade précoce, les chances de survie augmentent considérablement. La clé pour résoudre ce problème est un meilleur diagnostic, et c'est à cela que notre projet y contribue.

Notre projet de fin d'année est un système d'aide au diagnostic médical (CAD : computer aided diagnosis) qui vise à aider les radiologues et les professionnels de la santé à réduire le temps perdu pour la détection d'un nodule et à mieux interpréter les scanners ce qui permet de réduire les erreurs de diagnostic des cancers de poumons

En effet, notre solution est une architecture complète en deux phases qui comprend un module pour la segmentation et l'extraction des nodules à partir des tranches de tomodensitométrie, et un second module pour la reconstruction 3D

Nous avons commencé par une compréhension générale de notre problème. En effet, nous avons passé beaucoup de temps à étudier et analyser les articles liés au sujet des nodules pulmonaires, des algorithmes de détection, de segmentation et les techniques de reconstruction 3D

Puis une recherche approfondie sur les ensembles de données de tomodensitométrie disponibles, les outils et les environnements accessibles.

Ensuite, nous sommes passés à la phase de modélisation, où nous avons implémenté chacun des deux modules, les avons entraînés et avons évalué leurs résultats.

Ce projet a été réalisé durant le deuxième semestre de la 4ème année à l'INSAT.

En prenant en considération les ressources limitées, la complexité et le temps donné pour réaliser un tel projet, nous sommes fiers du résultat obtenu. Ils sont considérés comme convenables et ils ont répondu à certaines de nos attentes.

Il est toujours possible d'améliorer notre projet si de meilleures ressources (GPU) deviennent disponibles.

Des travaux futurs et des perspectives peuvent être envisagés, y compris mais non

non limité à :

Réaliser une classification sur la malignité et la b nignit  du nodule

Am liorer l'apprentissage de notre mod le de d tection en augmentant le nombre d'images, d'epochs et en ajoutant une autre classe (classe 0 : poumon normal)

# BIBLIOGRAPHIE

- [1] **Jérôme Plojoux**, *Rev Med Suisse*; volume 14. 227-228 2018
- [2] **Océane Imagerie**, <https://www.ocean-imagerie.fr/fiche-conseil/nodule-pulmonaire/>
- [3] **M. Lederlin**, *CAT devant la découverte fortuite d'un nodule pulmonaire* 2015
- [4] **NIBIB**, *Computed Tomography (CT)* 2019
- [5] **Institut national du cancer**, *Cancer du poumon*
- [6] **Mickaël Ohana**, *Nodules pulmonaires : comment les mesurer ?*
- [7] **P. Grenier et C. Beigelman-Aubry**, *Conduite à tenir devant un nodule pulmonaire de découverte fortuite*
- [8] **D. Jeanbourquin, J. Bensalah and K. Duong**, *Nodule pulmonaire solitaire*
- [9] Deep Learning for Vision Systems This book was written by Mohamed Elgendy and published in 2020.
- [10] Computer Vision : Algorithms and Applications This book was written by Richard Szeliski and published in 2010.
- [11] Computer Vision : Models, Learning, and Inference This book was written by Simon Prince and published in 2012.
- [12] Rafeek Mamdouh, Hazem M El-Bakry and Ala el-din mohamed Riad, "Converting 2D-Medical Image Files DICOM into 3D- Models, Based on Image Processing, and Analysing Their Results with Python Programming", February 2020
- [13] Aziz Fajer, Riyanarto Sarno, Chastine Fatichah and Achmad Fahmi, "Reconstructing and resizing 3D images from DICOM files", Journal of King Saud University – Computer and Information Sciences, December 2020
- [14] <https://colab.research.google.com/signup>
- [15] <https://buomsoo-kim.github.io/colab/2020/03/15/Google-newly-launches-colab-pro.md/>
- [16] <https://luna16.grand-challenge.org/>
- [17] <https://wiki.cancerimagingarchive.net/display/Public/LIDC-IDRI>
- [18] <https://www.kaggle.com/nih-chest-xrays/data>
- [19] <https://www.python.org/>
- [20] <https://keras.io/>
- [21] <https://www.tensorflow.org/?hl=fr>
- [22] <https://simpleitk.readthedocs.io/en/v2.0.0/>
- [23] [https://github.com/madsendennis/notebooks/tree/master/volume\\_segmentation\\_with\\_unet](https://github.com/madsendennis/notebooks/tree/master/volume_segmentation_with_unet)
- [24] [https://keras.io/examples/vision/oxford\\_pets\\_image\\_segmentation/](https://keras.io/examples/vision/oxford_pets_image_segmentation/)
- [25] <https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47>
- [26] <https://keras.io/api/losses/>
- [27] <https://www.kaggle.com/qzuidhof/full-preprocessing-tutorial>
- [28] <https://www.kaggle.com/rodenluo/crop-save-and-view-nodules-in-3d>
- [29] <https://www.kaggle.com/arnavkj95/candidate-generation-and-luna16-preprocessing/notebook>

- [30]<https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>
- [31]<https://glassboxmedicine.com/2020/01/21/segmentation-u-net-mask-r-cnn-and-medical-applications/>
- [32]<https://www.programmersought.com/article/95774764546/>

## **Titre : Détection des nodules pulmonaires en utilisant UNET (à partir des images CT) et reconstruction 3D**

---

**Résumé :** Le cancer du poumon est la forme de cancer la plus couramment diagnostiquée, c'est la principale cause de décès par cancer tant chez l'homme que chez la femme avec 1.6 millions de décès chaque année d'après la société canadienne du cancer. Sa détection précoce est très importante puisqu'elle permet d'accélérer la procédure de traitement et par la suite diminuer le taux de mortalité et sauver des vies. Cependant, ce processus n'est pas évident en raison du nombre considérable d'images médicales à analyser, les expertises nécessaires pour se faire et l'inexistence d'une formule claire pour décrire une tumeur. Les techniques d'intelligence artificielle peuvent être utiles dans ces cas pour offrir un système capable de détecter automatiquement les nodules et accélérer le processus de diagnostic tout en gardant un faible taux d'erreur.

Dans cette étude, nous avons mis en œuvre un algorithme qui utilise la base de données LUNA 16 pour entraîner notre modèle de détection des nodules ainsi que sa reconstruction en 3D.

## **Title: Lung nodule detection using UNET (from CT images) and 3D reconstruction**

---

**Abstract:** Lung cancer is considered the deadliest cancer worldwide with around 1.6 million deaths every year, according to the CCS (Canadian Cancer Society). Early detection of lung cancer is very important in healthcare as it can help save lives and decrease the cancer mortality rate. Due to the huge number of CT scans that need to be analyzed annually, the detection of lung cancer at early stages become a tedious task for radiologists.

Artificial intelligence techniques can be useful in developing a solution for this problem by offering a system that can automatically detect nodules and speed-up the diagnosis process while making the result more repeatable and accurate.

In this study, we implemented a deep learning algorithm that uses a database, provided by the Lung Nodule Analysis 2016 challenge (LUNA 16), to train our model for detection and 3D reconstruction of nodules.