UNIVERSITÉ
CAEN
NORMANDIE

M2 Informatique
IDM
Année 2021-2022

Projet FLAP
Segmentation automatique sur imagerie scanner des lambeaux de reconstruction utilisés en chirurgie carcinologique ORL

Stage de M2

Etudiant :
RENIER Valentin

Enseignants :
CHAHIR Youssef
THARIAT Juliette
MODZELEWSKI Romain

Table des matières

Remerciements

Tout d'abord, je remercie mes encadrants, M. Youssef CHAHIR (**GREYC** Caen), le Pr. Juliette THARIAT (**Laboratoire de Physique Corpusculaire**, Caen), et le Dr. Romain MODZELEWSKI (**LITIS** Rouen) pour m'avoir proposé ce stage à la suite de mon projet annuel et de m'y avoir accompagné en proposant leur expertise durant nos réunions hebdomadaires. Ce temps accordé m'a permis de toujours me fixer un cap pour chaque semaine à venir.

Mes remerciements vont aussi aux membres de l'équipe *Image* du **GREYC** pour m'avoir fait participer à leurs séminaires ainsi que de m'avoir porté conseils concernant les points techniques liés à leurs domaines de recherche. Je remercie tout particulièrement M. Alexis LECHERVY pour m'avoir octroyé de son temps pour répondre à mes questions et pour m'avoir fait participer à une des formations qu'il propose, ce qui me fut très utile et fut pour beaucoup dans l'avancée du stage.

Enfin je tenais à remercier **NormaSTIC** pour l'octroi d'une bourse qui a permis de financer ce stage.

Introduction

0.1 Cadre du stage

Ce stage se déroule dans l'équipe *Image* du laboratoire de recherche **GREYC** de Caen dont les travaux se concentrent sur la vision par ordinateur.

Ce stage prend place dans le cadre de la consolidation des interactions **GREYC** (Caen) et **LITIS** (Rouen) pour des applications en intelligence artificielle dans le domaine de l'imagerie oncologique et de radiothérapie. Cette collaboration implique les deux établissements de lutte contre le cancer de Caen (Centre Baclesse) et Rouen (Centre Becquerel).

0.2 Positionnement du problème (médical)

Les progrès médicaux en cancérologie comportent un axe de minimisation de la morbidité de la chirurgie de résection des tumeurs, cette morbidité est d'autant plus élevée que la tumeur est volumineuse ou placée dans une région anatomique fonctionnellement importante. Les tumeurs ORL sont "localement avancées" (volumineuses) dans plus de 60% des cas chez les 16 000 personnes qui ont un cancer ORL et au carrefour des fonctions de déglutition / alimentation, élocution / parole. En plus de la chirurgie, la radiothérapie contribue à la morbidité du traitement des cancers. Pour restaurer les fonctions perdues avec la résection tumorale, les chirurgiens mettent désormais en place des lambeaux de tissu sain prélevé en dehors de la zone tumorale. Cette pratique, permise par la miniaturisation de l'instrumentation et les progrès en optique, est de plus en plus courante dès lors que l'augmentation de l'expertise des équipes s'étend et que ces mêmes équipes inventent des lambeaux de plus en plus versatiles (c'est à dire maniables, conformables pour être le plus fidèles possible à l'anatomie originelle).

Pour autant, la pratique de la radiothérapie s'est peu adaptée aux progrès chirurgicaux et en particulier pas à l'utilisation de lambeaux. Les chirurgiens décrivent une altération des lambeaux (et résultats fonctionnels qui en sont attendus) avec la radiothérapie ; ce constat repose plus sur des impressions que des évaluations formalisées. L'irradiation des lambeaux est effectivement dans l'angle mort des cancérologues radiothérapeutes ; la stratégie prévalente est globalisante et consiste à irradier large (pour éviter tout risque de récidive tumorale), quitte à inclure le lambeau dans la zone opératoire et à "sur-irradier" ce lambeau. Les radiothérapeutes n'identifiant pas les lambeaux lors de leur planification de radiothérapie, ils ne peuvent pas évaluer formellement l'effet de la radiothérapie sur ces lambeaux. Pour ce faire, il faudrait contourer les lambeaux, suivre leur devenir en volume et texture et suivre le devenir fonctionnel des patients.

0.3 Expression du besoin (médical)

La documentation des relations dose-effets (les effets étant des récidives ou des toxicités (difficultés de déglutition, élocution) pouvant être corrélées à une réduction de volume ou une modification de texture). Le pré-requis de cette documentation est la segmentation.

Sur la base d'images patients segmentées et de données cliniques, il faudra à terme étudier la zone de récidive ; l'hypothèse étant qu 'elle débuterait à la jonction entre tissus natifs et lambeau, et les toxicités, l'hypothèse étant que plus la dose est élevée et distribuée sur un grand volume des dits lambeaux plus le risque toxique est élevé. Si ces hypothèses sont à terme vérifiées, il sera possible de proposer une stratégie d'irradiation plus adaptée aux lambeaux, minimisant le volume et la dose au lambeau tout en n'augmentant pas le risque de récidive tumorale.

0.4 Travaux préalables

Le besoin a été confirmé par un travail de consensus international (publication internationale Carsuzaa 2021). Un atlas de segmentation manuelle des lambeaux a été définie avec des tutoriels (publication internationale Le Guevelou 2021). Une étude inter-observateurs en a découlé, qui a montré un index de Dice de 0.7 (publication internationale Beddok 2022).

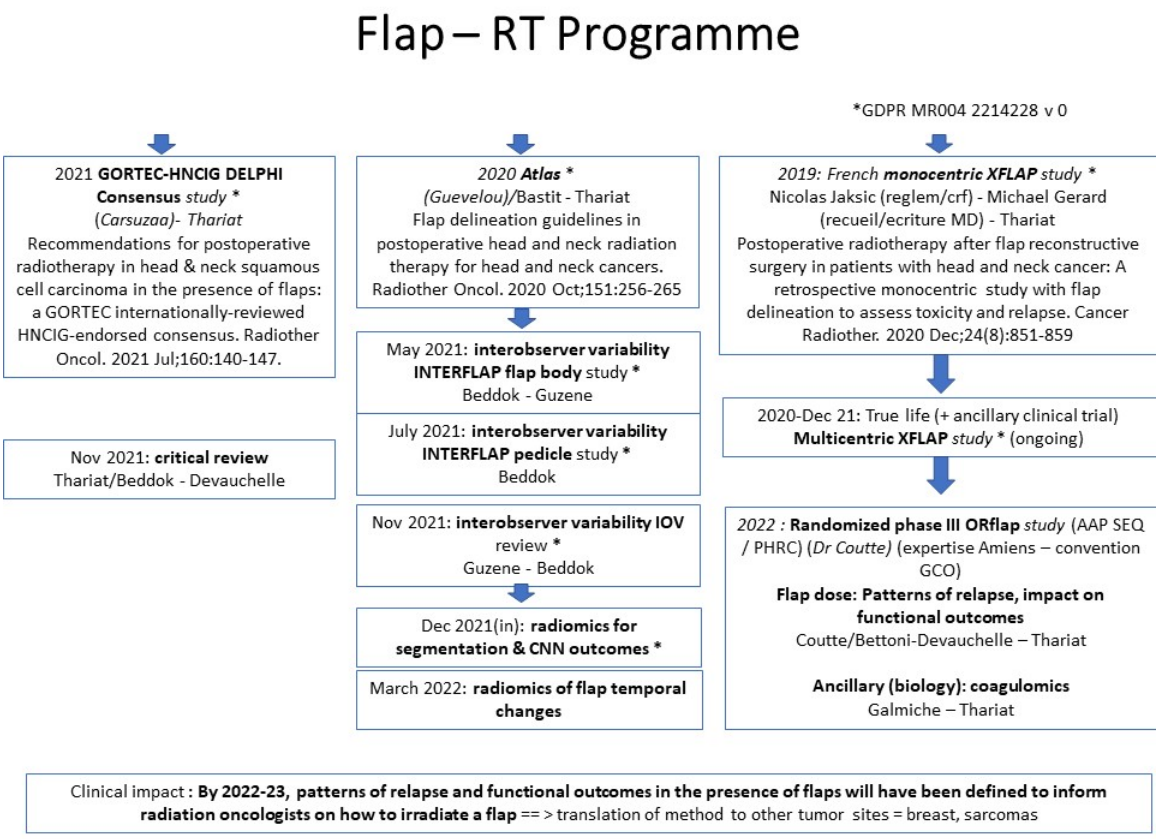


FIGURE 1 – le Programme Flap

0.5 Objectif du présent travail de Master 2

L'objectif du travail présenté ici est de définir une méthode de segmentation basée sur les réseaux de neurones permettant aux radiothérapeutes de générer automatiquement un contour de lambeau dans leur plan de radiothérapie.

1 Matériel et Méthodes

1.1 Matériel Informatique

1.1.1 Machine à disposition

Le travail effectué a été réalisé sur une machine avec un CPU *ex Intel(R) Core(TM) i7-10875H CPU @ 2.30GHz*, une mémoire RAM de *32Go*, un processeur *intel CORE i7 vPRO* ainsi qu'une carte graphique *NVIDIA Quadro T2000*.

1.1.2 Serveur de calcul

Nous avons utilisé un serveur de calcul muni de *2 processeurs Intel Xeon E5-2640 v4 2.40GHz*, 40 coeurs, une RAM de 256Go, 8 cartes graphiques *MSI GeForce GTX 1080 Ti 3584 Cores 11G RAM* et d'un espace disque de *3,5 To*.

1.2 Données

1.2.1 Origine des bases de données

Ces bases sont anonymisées et RGPD-compatibles.

Base Flap La base de données, renommée “flap”, dont nous disposons est composée d’images de planification de traitement par rayons X (radiothérapie) de 300 patients atteints de cancers ORL et traités dans le cadre d’un essai thérapeutique (qui teste une radiothérapie avec chimiothérapie ou une radiothérapie avec chimiothérapie ET immunothérapie). Cette base comporte 150 patients avec lambeau, 150 sans.

On dispose aussi d’un tableur **.csv** comportant des informations concernant les images (par exemple des données catégorielles quantifiant l’impact d’artefacts liés à des matériaux métalliques dentaires) ou la qualité des plans (par exemple, présence et criticité d’une déviation par rapport au protocole). Ce tableur sera utilisé dans un développement du travail ultérieur au stage de Master. D’autres paramètres pourront d’ailleurs être ajoutés comme le site tumoral.

Base PRPO Nous disposons d’une autre base de données de 600 patients atteints de cancers ORL et traités selon les pratiques de soins courants, constituée dans le cadre du projet structurant PRPO du Cancéropôle Nord (Caen Rouen Lille Bruxelles) Ouest 2017.

1.2.2 Acquisition des données

Le scanner de radiothérapie par photons est calibré pour étalonner la dose en fonction des densités électroniques des composants du corps de patients (tissus et matériaux), elles-mêmes corrélées aux unités Hounsfield (UH).

La qualité des images est variable en fonction de divers paramètres non corrigibles par nous. Les paramètres d’acquisition ne sont pas complètement standardisés. L’épaisseur des coupes est communément de 3mm ; l’injection de produit de contraste est inconstante, rare par rapport au diagnostic.

Des algorithmes de minimisation d’artefacts (amalgames, couronnes, ou implants dentaires ou plaques de fixation en titane des segments de lambeaux osseux) sont variablement appliqués selon les centres et les patients avec, comme leur nom l’indique, des corrections partielles.

Les logiciels de segmentation et de calcul de dose pour la radiothérapie (Treatment planning system TPS) sont divers. Certains comme celui de tomothérapie utilisent un format propriétaire aboutissant à une dégradation de la résolution spatiale de 512*512 en 256*256. Ce paramètre affecte la segmentation manuelle (vérité terrain / annotation, quantification non faite).

Tous ces défauts de qualité de l’image sont évalués semi-quantitativement par un indice de sévérité dans le tableur **.csv**, évoqué plus haut, visant à quantifier leur impact sur la segmentation manuelle reposant sur la visualisation des images coupes par coupe et dans les trois plans (transversal, sagittal, coronal). Dans cet objectif de mesure d’impact, plusieurs dossiers ont été segmentés manuellement deux fois par la même personne en aveugle de sa première segmentation manuelle.

1.2.3 Images au format DICOM

Description du format Le format **DICOM** (*Digital Imaging and Communications in Medicine*) est le format standard concernant l'imagerie médicale. Il est "relativement" normé, laissant une latitude importante pour la désignation des images et sur les données des patients.

Un fichier **DICOM** se compose de deux parties : une image unique du patient, mais également un entête, contenant des données liées au patient, mais aussi des données qui décrivent l'image. Un médecin peut annoter des remarques ou des observations sur le patient.

Dans le cadre de la radiothérapie, on utilise la variante **DICOM RT** qui comprend d'une part les fichiers **CT** qui sont les coupes d'images sauvegardées une par une à l'extension **.dcm** et d'autre part les fichiers **RS** ou **RTSTRUCT** qui comportent les structures contourées par l'expert et sont aussi d'extension **.dcm**. Nous appelons communément ces structures *"régions d'intérêt"* ou **"ROI"**. Les métadonnées stockées en entête de ce fichier nous permettent de relier les structures aux coupes correspondantes.

Pour ce projet, il y a deux fichiers de structures, l'un contenant uniquement le contour du lambeau et un second contenant tous les autres contours.

Par ailleurs, il y a aussi systématiquement un fichier **RD** ou **RTDOSE** pour la matrice de dose et parfois un fichier **RP** ou **RTPLAN** comportant les paramètres faisceaux.

Dans le cadre du stage nous ne nous servons pas de ces deux types de fichiers.

Manipulation des fichiers Dicom Dans un premier temps nous avons créé notre propre système de gestion des fichiers **DICOM** à l'aide du module **pydicom**.

Partant d'une région d'intérêt donnée, nous allions la chercher dans la liste des régions d'intérêt du fichier **RS** correspondant afin de récupérer les paires images-contours.

Ces contours ne sont pas stockés sous forme d'images mais sous forme de liste de points dans l'espace réel. Il a donc fallu appliquer une transformation mathématique à ces points pour en faire des pixels correspondant à l'image.

Une fois cela fait nous nous sommes rendus compte que la simple superposition de toutes ces images ne recréent pas le patient car les coupes ne sont pas ordonnées. Nous avons donc dû les remettre dans le bon ordre ainsi que les contours leur correspondant.

L'objectif de la récupération des contours est de créer une vérité terrain pour le réseau de neurones. Il faut donc passer de contour à masque, en "remplissant" le contour comme on le ferait avec un outil *pot de peinture* de logiciel d'imagerie. Pour cela nous avons donc d'abord utilisé la méthode *fill* du module *skimage* mais il suffisait qu'un pixel manque pour fermer le contour et tout se remplissait. Par ailleurs, cette méthode nécessitait un germe pour connaître le point de départ du remplissage. Nous calculions donc les *moments géométriques*[1] du contour afin d'obtenir son *barycentre*. Cette méthode ne fonctionne qu'avec des formes convexes mais par exemple dans le cas des mandibules, le barycentre se trouve en dehors de l'objet.

C'est pour ces deux raisons que nous avons donc plutôt opté pour la méthode *binary_fill_holes* du module **scipy** qui est bien plus robuste.

1. "projections" d'une fonction (image) sur une base polynomiale, les moments du premier ordre d'une image sont les coordonnées de son barycentre

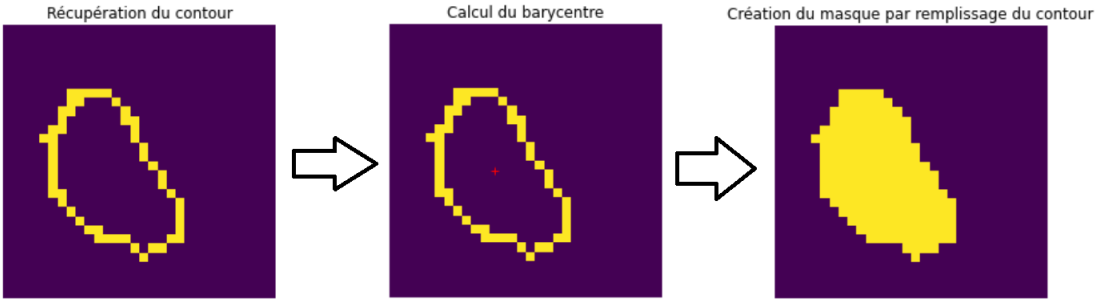


FIGURE 2 – Création du masque depuis le contour

Cependant, notre méthode de manipulation de fichiers n'était pas assez robuste aux irrégularités des fichiers **DICOM** dont les métadonnées renvoyaient parfois vers des fichiers inexistants. Autre défaut, notre méthode se concentrait sur une région d'intérêt de par sa structure et ne nous permettait pas d'en chercher plusieurs efficacement sans parcourir à nouveau toutes les données pour chaque patient.

Utilisation du Dicom Reader Nous avons ensuite découvert le module **DicomRTTool** issu d'un article[2] qui propose plusieurs outils pour manipuler les **DICOM**. Nous utilisons une instance de la classe **DicomReaderWriter** qu'il nous propose. Nous donnons à cet objet le dossier contenant chaque patient du projet puis il se charge de parcourir chaque dossier et fichier **DICOM** de chaque patient.

Grâce à ses diverses méthodes on peut manipuler les patients en fonctions de leurs régions d'intérêt qui peuvent être facilement listées. Pour cela, on fourni au **DicomReader** des *associations* c'est à dire un identifiant normalisé relatif à une **ROI** par exemple "mandibule" couplé avec un ensemble de **ROI** correspondantes, sélectionnées ou non par des *expressions régulières*. Cela permet de regrouper sous un même label des régions équivalentes mais nommées différemment selon les centres médicaux ou les médecins, telles que "mandibule, mandible, Unknown_mandible" et leurs variants en terme de casse.

```
if roiName == "mandible":
    str_match = [x for x in all_rois if re.search('mandi', x)
                 and "bone" not in x and "coupe" not in x
                 and "over" not in x and "pt" not in x
                 and "prv" not in x and "max" not in x]
```

Ces expressions régulières permettent aussi de filtrer les contours avec des suffixes particuliers qui sont seulement là à titre indicatif pour le radiologue par exemple contenant le terme *ptv* pour *radiotherapy planning target volume*.

Le **DicomReader** nous fournis ensuite chaque patient contenant au moins une de ces régions d'intérêt en faisant lui même la transformation mathématique pour passer des coordonnées des points qui sont en millimètres à des coordonnées en pixels. Il s'occupe aussi du passage de contour à masque en utilisant lui aussi la fonction *binary_fill_holes* de **scipy**. Chaque patient est donc représenté comme un couple de deux *Array* (tableaux

2. https ://www.sciencedirect.com/science/article/abs/pii/S1879850021000485

issus du module **numpy**) de dimensions *(nombre de coupes, hauteur, largeur)* : l'un pour les coupes et l'autre pour les masques. C'est à partir de ces deux tableaux que nous créons notre base de données.

1.3 Perte de données

Tout au long du processus partant des données brutes (images au format **dicom**) jusqu'aux données à traiter (images au format **png**), nous perdons des patients du fait de notre incapacité à récupérer toutes les informations nécessaires car il y a fréquemment des erreurs dans les métadonnées contenues en entête des coupes **CT**.

Dans le cadre de la base **FLap**, nous avons pu récupérer quelques données issues d'autres centres ce qui élève le nombre de patients annotés "flap" de *150* à *174*. Malgré cela, une fois le processus de construction de la base terminé, nous n'avons plus que *135* patients.

Concernant la base **PRPO**, notre objectif était de l'utiliser pour pré-entraîner un modèle multiclasse. Malheureusement notre approche de la récupération des données des patients étant exclusives (il faut avoir toutes les régions d'intérêt désirées pour être conservé) et la base ayant encore plus de problèmes de liaisons (contours-coupes) dans ses métadonnées, nous avons décidé de l'écarter du travail final, bien qu'elle nous ait servi tout au long du stage pour tester les modèles que nous trouvions.

1.4 Description de la base

Lors de la manipulation de la base, et avant l'étape de pré-traitement, nous avons 137 patients. Le nombre de coupes moyen d'un patient est de *216*. La nature de l'acquisition de ces images n'est pas la même entre tous les patients issus de différents centres ce qui nous amène à manipuler des images de patients de différentes dimensions :

dimensions	Nombre de Patients
1024 x 1024	1
512 x 512	125
272 x 272	1
268 x 268	2
256 x 256	8

FIGURE 4 – Nombre de patients en fonction des dimensions de ses coupes

Par ailleurs, en moyenne, le lambeau représente *25037* voxels [3] de la matrice représentant un patient par des coupes de dimensions *512 x 512* ce qui représente environ 11.5%

3. Un voxel est à la 3-Dimensions ce qu'est un pixel à la 2-Dimensions

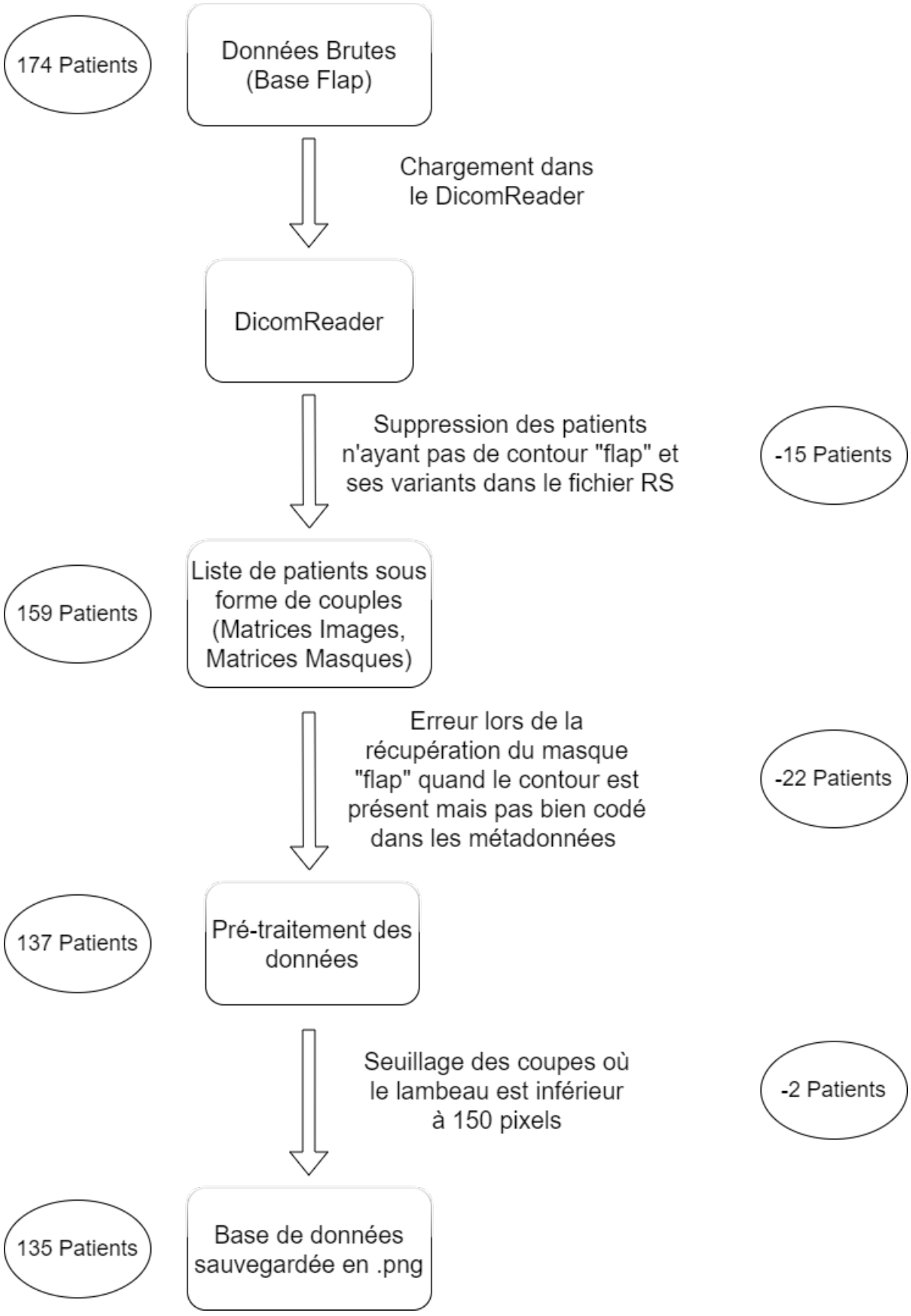


FIGURE 3 – Processus de création de la base et perte des patients

de cette matrice. Par ailleurs, le lambeau est présent dans seulement 10% des coupes des patients.

	Taille du lambeau (en nombre de voxels)
Taille minimale	694
Taille maximale	166570
Taille Moyenne	25037
Taille Médiane	16764
Ecart-Type	25415

FIGURE 5 – Taille du lambeau

	Coupes Contenant le Lambeau	Coupes Totales
Minimum	5	126
Maximum	65	341
Moyenne	27	216
Médiane	23	201
Ecart-Type	14	58

FIGURE 6 – Variabilité du nombre de coupes, les colonnes sont indépendantes

1.5 Annotations

Pour la suite du projet nous avons à notre disposition un tableau **csv** contenant des informations supplémentaires sur le lambeau : s'il est de nature musculaire, graisseuse, osseuse...) ainsi que sur l'image (qualité, présence d'artefacts)... Ces informations ne sont pas utilisées dans le cadre du stage mais pourrons être utiles dans les étapes ultérieures du projet afin de mieux caractériser le lambeau.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	
1	N° d'inclusion	Centre	sitet	Lamb	boneflap	flaptype side		Aanaston	Vanasto	osteo	getOPreport	tracheo	imqual		Artéfacts présents	Les organes à risque sont-ils contourés conforme	Commentaire :
2	10-001	10		Non							1				3 - sévère	Oui	-
3	01-001	1	hl	Oui							1		tomo		3 - sévère	Non	no flap seen direct closure? get op report
4	01-002	1		Non											3 - sévère	Non	plexus missing
5	01-003	1	hl	Oui		jejunum		ace	tlf		0				1 - léger	Oui	constrictors plexus missing
6	01-004	1															
7	01-005	1		Oui							1				3 - sévère	Non	oc plexus constrictors missing

FIGURE 7 – Quelques lignes informatives sur le lambeau

2 Pré-traitement des données

La construction de la base de données se fait itérativement patient par patient (un patient étant représenté par sa matrice de coupes et sa matrice de masques). Pour chaque masque, on vérifie qu'il contient bien un label, en quantité suffisante par rapport à un seuil (fixé à 150 pixels pour une image *256*256*, ce qui équivaut environ à un carré de 12 pixels par 12 pixels) afin d'avoir des images plus facilement abordables pour le réseau de neurones.

Passée cette étape, nous devons prendre des décisions pour le type de données à sauvegarder afin d'être manipulées ce qui altère l'entraînement des données.

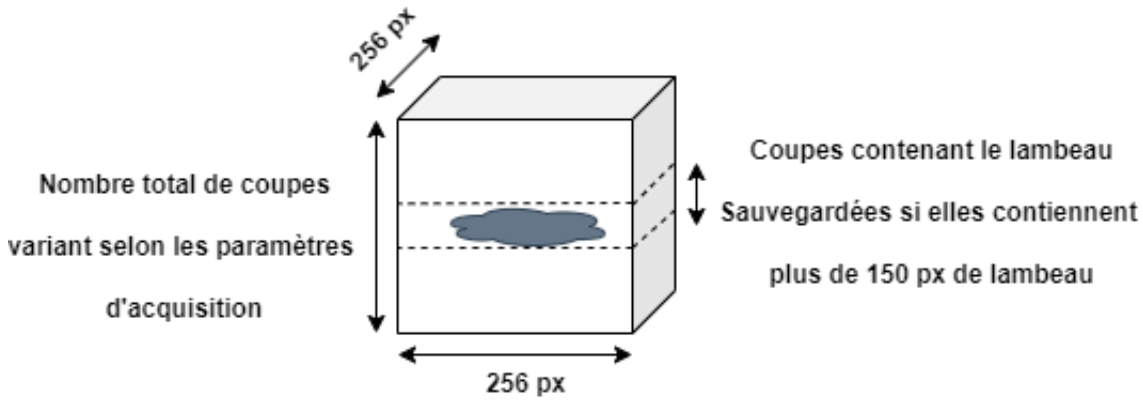


FIGURE 8 – Récupération des coupes contenant le lambeau parmi l'ensemble des coupes d'un patient

2.1 Plans d'une image

À partir de la position anatomique de référence, on décrit trois plans imaginaires en 2 dimensions qui passent par le centre de gravité du corps humain et qui sont perpendiculaires les uns par rapport aux autres. On distingue le plan sagittal, le plan frontal et le plan transversal.

Le plan sagittal C'est un plan vertical qui passe par la ligne médiane du corps et le divise en deux parties symétriques, droite et gauche.

Le plan frontal C'est un plan vertical perpendiculaire au plan sagittal qui divise le corps en deux parties symétriques, antérieure (ventrale) et postérieure(dorsale).

Le plan transversal C'est un plan horizontal, parallèle au sol, qui divise le corps en deux parties symétriques, supérieure (du côté de la tête) et inférieure (du côté des pieds).

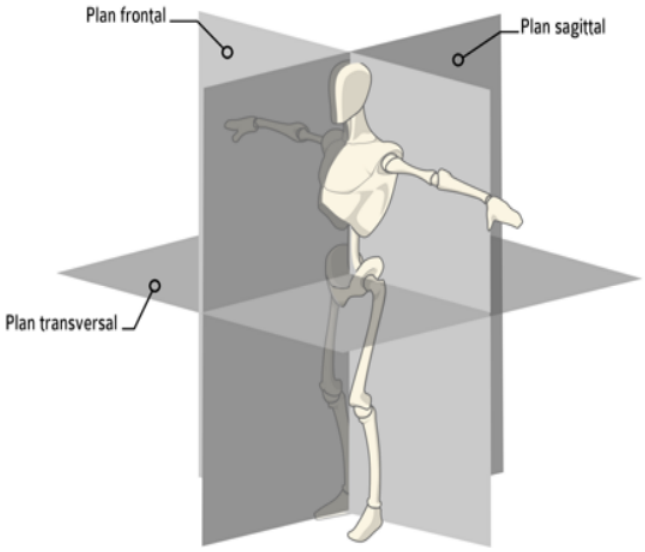


FIGURE 9 – Les plans utilisés en imagerie médicale

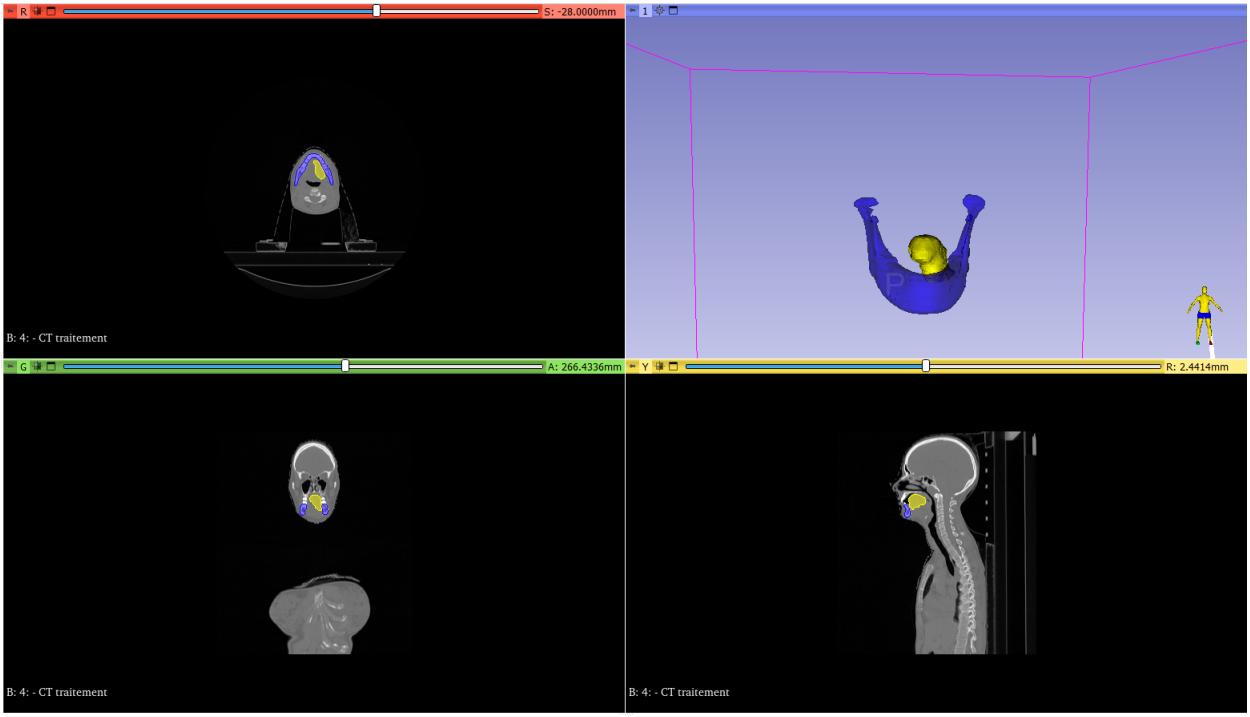


FIGURE 10 – Lambeau et mandibule en fonction des différents plans et en 3-Dimensions

Durant notre projet, nous travaillons sur des images en deux dimensions donc nous avons choisi le plan transversal qui est celui généralement utilisé ainsi que celui que nous fourni le **DicomReader** par défaut.

2.2 Redimensionnement

Selon les patients à traiter et les différents examens médicaux qu'ils ont suivis, les images que l'on récupère peuvent être de dimensions *512 x 512* ou *256 x 256*[4].

Les images en *256 x 256*, principalement celles issues de tomothérapie sont d'une qualité plus médiocre. Il ne nous a donc pas semblé cohérent de les agrandir afin de n'avoir que des images en *512 x 512* et nous avons donc plutôt fait le choix de rétrécir les images en *512 x 512* jusqu'à *256 x 256*.

L'entraînement d'un modèle sur des milliers d'images étant assez lent avec une puissance de calcul limitée et notre but étant de tester différentes implémentations de U-net afin d'en trouver une efficace, nous avons encore réduit la dimension jusqu'à *128 x 128* pour toute la partie de prise en main des modèles durant le stage pour plus de rapidité (les images sont quatre fois plus petites qu'en *256 x 256* et huit fois plus qu'en *512 x 512*).

On travaille en *256 x 256* uniquement pour les résultats du modèle final afin d'avoir un résultat plus précis.

2.3 Format d'images

En début de stage nous avons travaillé sur deux formats différents afin de sauvegarder nos images et nos masques. Dans un premier temps nous sauvegardions les images sous forme de matrices **Numpy** au format **.npy**. Ce format a l'avantage de conserver les véritables valeurs d'intensité lumineuse en type *float* ce qui nous fait gagner en précision.

Du fait de la sauvegarde en type **float**, une image au format **.npy** est environ cent fois plus lourde qu'une image sauvegardée au format **.png**. En contrepartie, les images **PNG** perdent grandement en précision car elles sont codées en entiers entre 0 et 255. Malgré ce biais nous avons décidé de travailler en images **PNG** pour le gain de place dans un espace de stockage limité et donc plus rapide à transférer et à visualiser sachant que par ailleurs les masques (c'est à dire la moitié de la base de données) ne contenant que des labels (donc des petites valeurs entières), il était exagéré de les sauvegarder dans un format aussi lourd que **.npy**.

2.4 Normalisation

Communément, les réseaux de neurones prennent en entrée des images normalisées entre 0 et 1. Cette étape peut être faite lors de la sauvegarde des images si on les sauvegarde sous **.npy** sinon à la création du dataset dans le cas où l'on utilise des formats d'images classiques tels que **PNG**.

L'avantage du format **.npy** est qu'il est directement codé en flottant lors de la sauvegarde donc nos valeurs entre 0 et 1 sont bien plus précises que les 256 valeurs possibles pour un pixel au format **.png** qui sont ensuite normalisées entre 0 et 1 ce qui provoque donc une perte de précision sur l'image, très peu visible à l'oeil humain avec nos outils de visualisation (modules **matplotlib et plotly**) mais qui sont une perte d'information pour le réseau de neurones.

4. Ou exceptionnellement dans d'autres dimensions comme montré dans la description de la base

2.5 Améliorations par approche radiologique

2.5.1 Échelle de Hounsfield

Cependant, normaliser les données nous fait perdre de précieuses informations car les valeurs de pixels entre **-32768 et 32768** correspondent directement à la densité de la matière qu'ils représentent en unité de Hounsfield, notée **UH** et qui correspond à la formule suivante où μ_x est le coefficient moyen d'absorption du pixel et μ_{eau} le coefficient d'absorption linéaire de l'eau :
$UH = 1000 * \frac{\mu_X - \mu_{eau}}{\mu_{eau}}$

On peut donc, à partir de la valeur du pixel, déterminer la nature du tissu, à l'aide de l'échelle de Hounsfield dont voici un tableau qui récapitule quelques valeurs utiles pour notre projet.

Matière	UH
Air	-1000
Graisse	-100 à -50
Eau	0
Sang	+30 à +45
Muscle	+10 à +40
Tissus mous	+100 à +300
Os	+700 (os spongieux) à +3000 (os denses)

FIGURE 11 – échelle de Hounsfield

On remarquera que ces intervalles peuvent se chevaucher puisque par exemple, l'intervalle de valeurs correspondant au sang est en majeure partie contenu dans celui du muscle. Il ne suffit donc pas d'utiliser ces valeurs pour garantir l'exactitude de la nature de la matière mais peut cependant servir à en discriminer certaines, telles que l'air.

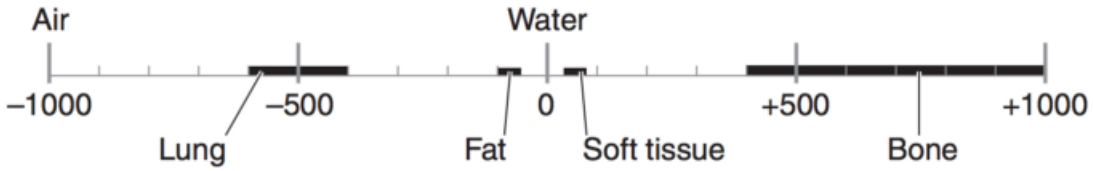
2.5.2 Fenêtrage

On peut aussi utiliser le fenêtrage de Hounsfield, basé sur l'échelle de Hounsfield et défini par son centre **C** et sa largeur **W**. Toutes les valeurs comprises dans cet intervalle vont être étalées sur les niveaux de gris tandis que les valeurs de Hounsfield en dehors des bornes deviendront blanches ou noires (respectivement pour les valeurs au dessus et en dessous des bornes). Cette opération permet d'obtenir des images qui mettent en évidence certaines matières et en suppriment d'autres.

Dans le cas de la recherche de lambeaux, on cherche à mettre en évidence les tissus mous, la graisse et les muscles. Nous avons donc choisi de prendre l'intervalle [-125, +250].

Nous avons donc le choix entre sauvegarder les images **CT** avec leur fenêtrage de base ou avec le fenêtrage mettant en évidence les tissus mous. Avec pour objectif de voir

si concentrer l'information de l'image permettait au réseau de neurones de converger plus facilement vers la prédiction du lambeau ou apportait un meilleur score de prédiction.



Bone	+400 → +1000
Soft tissue	+40 → +80
Water	0
Fat	−60 → −100
Lung	−400 → −600
Air	−1000

FIGURE 12 – Intervalles de fenêtrages pour certains tissus

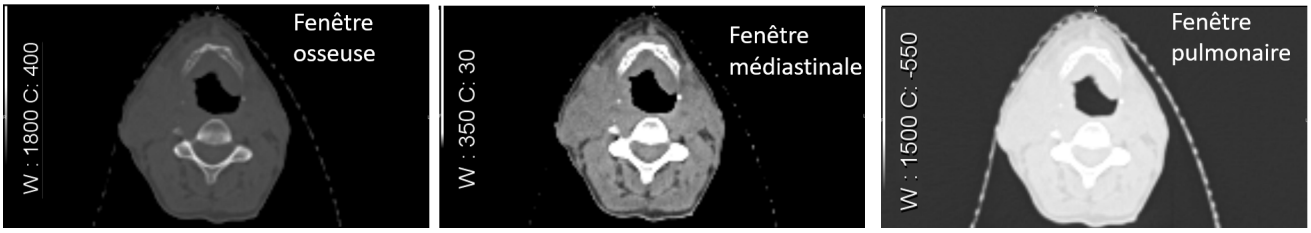


FIGURE 13 – Différents fenêtrages pour mettre en valeur différents tissus

2.6 Multiclasse

2.6.1 Par régions d'intérêt

Plutôt que de seulement segmenter le lambeau dans l'image, il est aussi possible de segmenter plusieurs types de tissus. Nous avons donc choisi comme régions d'intérêt les zones à risque lors des séances de radiothérapie. L'objectif étant de voir si l'ajout de contexte autour du lambeau peut aider le modèle à le trouver. Nos régions d'intérêt sont donc le lambeau, la mandibule, la cavité buccale, les glandes parotides, le larynx, et les tumeurs.

Nous travaillons en multiclasse, c'est à dire qu'un masque est codé sur un seul canal avec les différents labels, contrairement au multilabel où il n'y a sur chaque masque que les valeurs 0 et 1 avec un nombre de canaux équivalent au nombre de labels.

Cependant, n'avoir qu'un seul canal peut causer des problèmes car les vérités terrain de contours tracés par les médecins impliquent fréquemment une superposition entre

deux régions différentes. Ainsi, comment choisir le label à associer à ce pixel parmi plusieurs labels ?

Nous avons donc choisi d'additionner les labels afin d'obtenir un nouveau label désignant la superposition des deux régions d'intérêt. Cette méthode est efficace mais risque de mal annoter certains labels selon les additions : par exemple le label 1 et le label 2 donneront le label 3, il faut donc qu'il n'y ait pas de régions correspondant à ce label. Pour cela, nous choisissons non pas seulement de ranger les labels par ordre croissant de 0 jusqu'au nombre de labels mais de les élever en puissances de deux. Grâce à cela, nous nous assurons qu'en aucun cas la superposition de plusieurs labels ne se confondra avec un label existant.

Nos images étant sauvegardées au format **.png**, codées en entiers allant de 0 à 255, cette méthode se limite à l'utilisation de 7 régions d'intérêt puisque dans le cas où tous les labels seraient présent, la somme des labels ferait 255. Le masque à sauvegarder dans la base de données est donc calculé à partir de cette formule :
$masque = masquelambeau + 2 * masqueMandibule + 4 * masqueCaviteBuccale + 8 * masqueParotide + 16 * masqueLarynx + 32 * masqueTumeur$

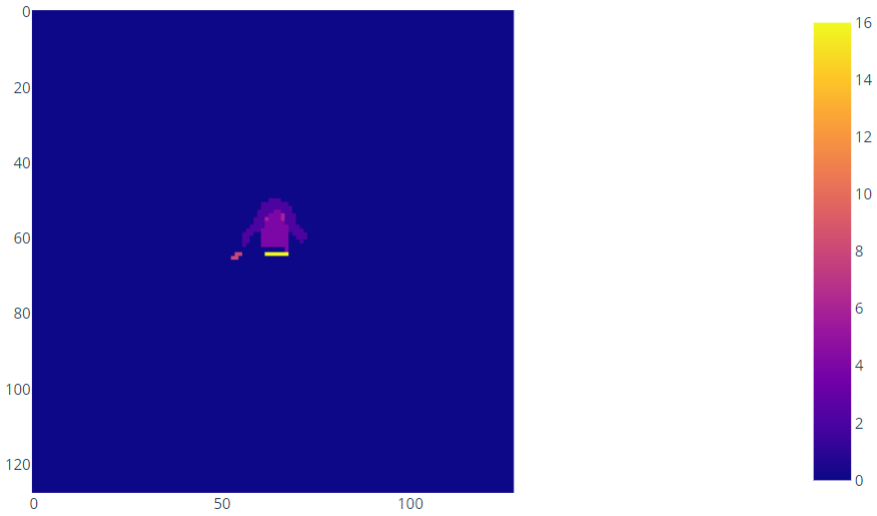


FIGURE 14 – Masque multiclasse

2.6.2 Par composition du lambeau

Les deux composantes principales d'un lambeau sont la graisse et le muscle. Pour empêcher des saignements dans le lambeau, il y a parfois besoin d'utiliser des agrafes métalliques qui forment des artefacts sur l'image. Il arrive parfois aussi qu'une partie osseuse soit reconstruite.

La composante osseuse peut facilement être récupérée par son contour nommé *flap-bone*. Le reste du lambeau étant seulement contouré au label *flap*, c'est à nous de le

décomposer manuellement pour en isoler la graisse et le muscle à l'aide d'une méthode de classification.

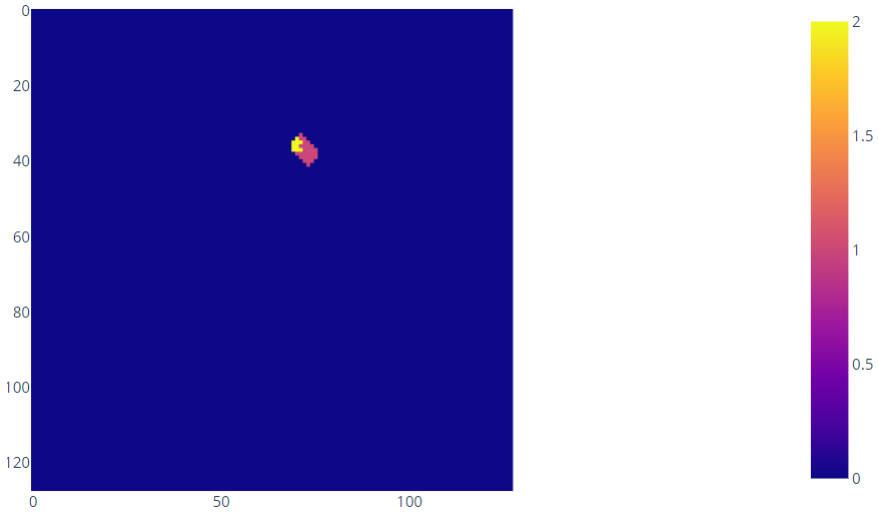


FIGURE 15 – Masque d'un lambeau décomposé en graisse et muscle

La méthode FCM Pour cela, nous utilisons la méthode **FCM** (*Fuzzy C-Means*) basée sur *C-Means* à la différence près que chaque observation peut appartenir à plusieurs groupes. D'abord, on choisi le nombre de groupes et donc de régions à segmenter. Ensuite les coefficients correspondant à chaque pixel sont assignés aléatoirement pour être placés dans les groupe. A chaque itération, on calcule le barycentre de chaque groupe et le coefficient d'appartenance aux différents groupes. On s'arrête quand la variation des coefficients devient inférieure au seuil choisi.

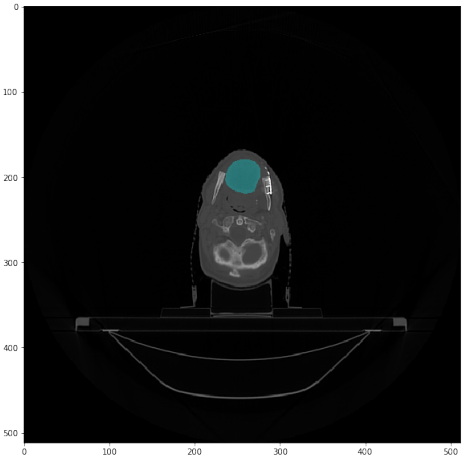


FIGURE 16 – Image et son masque

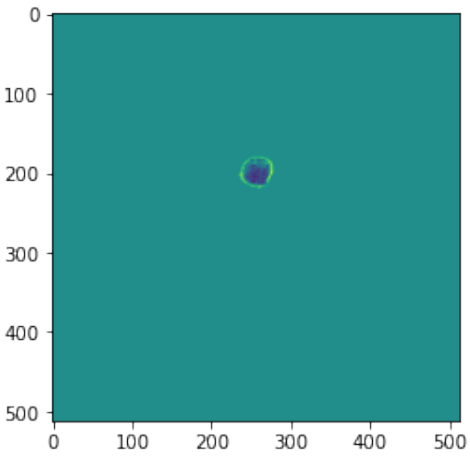


FIGURE 17 – Image à classifier par FCM

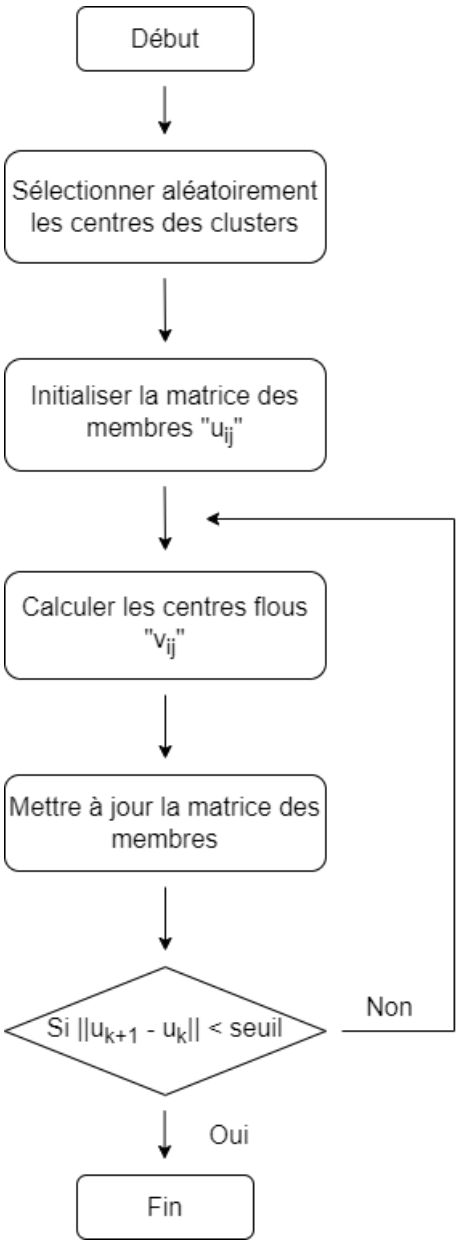


FIGURE 18 – Algorithme FCM

Dans notre cas, on travaille uniquement dans la zone de l'image contenue à l'intérieur du masque et on assigne trois clusters afin d'obtenir la graisse, le muscle et le reste (qui sera supprimé du masque). Cette méthode classifie donc l'intérieur du contour du lambeau en trois classes dont les labels changent à chaque utilisation selon l'initialisation des germes. On ne peut donc pas déduire directement quel label correspond à quelle zone et nous devons donc *caractériser* chaque classe.

Pour cela, nous calculons la médiane de chaque classe, non pas dans l'image mais dans une version fenêtrée de l'image via Hounsfield qui met en évidence la graisse et le muscle afin d'en déduire quel type de tissus correspond à chaque médiane et donc à chaque classe.

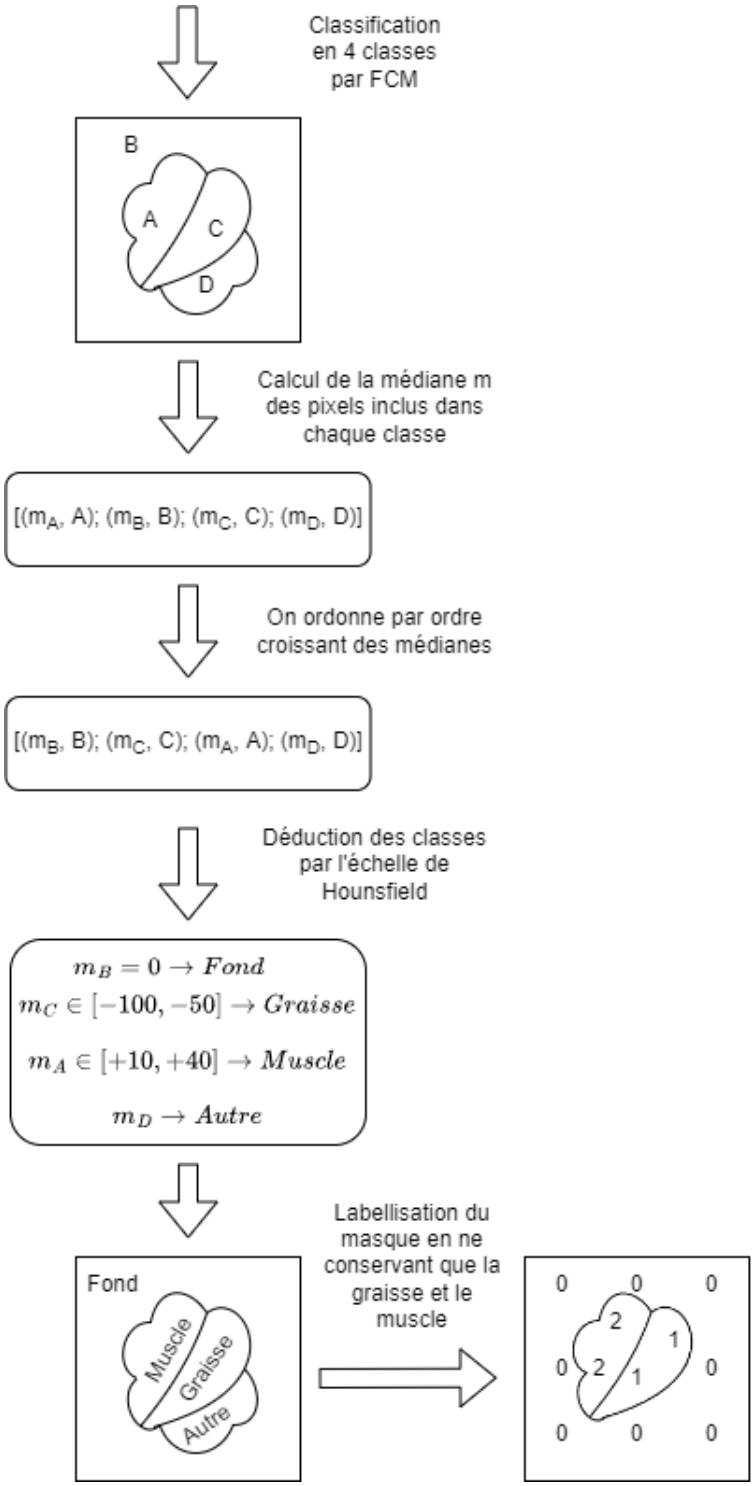


FIGURE 19 – Détermination des tissus correspondant aux classes prédites par FCM

2.7 Jeu de données

Une fois nos données traitées il faut les stocker dans un jeu de données (ou *dataset*) pour l'apprentissage. C'est cet objet qui permet de manipuler les données pour les charger, les séparer, les augmenter...

2.7.1 Chargement des données

Dans un premier temps, quand nous travaillions avec le module **Tensorflow** nous avons essayé de charger une grosse matrice avec toutes les données d'entraînement (ainsi qu'une pour le test et la validation). Avoir une immense matrice avec des milliers d'images était très lourd à charger et faisait excéder la mémoire de la machine.

C'est une erreur qui a pu être corrigée en utilisant une approche bien plus légère, c'est à dire de stocker une liste d'index (le chemin vers le fichier **.png**) puis charger l'image correspondant à l'index.

2.7.2 Séparation de la base

La séparation de la base se fait en termes de patients de sorte à ce qu'un patient ayant servi à l'entraînement ne se retrouve jamais (même en partie) dans la phase de test.

Nous avons fait le choix de séparer la base avec les proportions suivantes :

- Jeu d'entraînement (70% des patients) pour estimer les paramètres du modèle lors de la phase d'apprentissage.
- Jeu de validation (20% des patients) pour évaluer la performance du modèle et son ajustement aux données pendant la phase d'apprentissage.
- Jeu de test (10% des patients) pour mesurer la performance du modèle final, notamment s'il a une bonne capacité de généralisation ou au contraire s'il est en situation de sur-apprentissage.

2.7.3 Augmentation de la base

Les algorithmes d'apprentissage profond nécessitent de traiter énormément de données pour fournir un résultat cohérent. Notre base de données étant relativement limitée (environ 2500 images pour la partie entraînement/validation) nous appliquons des transformations mathématiques aux images afin d'agrandir notre base. Pour cela, nous avons utilisé des rotations et la symétrie verticale ce qui étend notre base à plus de 28000 images pour la partie entraînement/validation.

Dans un premier temps nos rotations allaient d'un angle de 30 à 330 degrés par pas de 30. Nous avons ensuite suivi une démarche de réalisme clinique en faisant des rotations jusqu'à 30 degrés par pas de 5 dans les deux sens en considérant que ces rotations minimes étaient probables cliniquement contrairement à un angle de 90 ou 180 degré qui représenterait une tête inclinée ou à l'envers lors d'une IRM. Sur une même base de patients, l'augmentation de données via réalisme clinique nous a donné une amélioration

de notre score d'environ 10% par rapport aux simples rotations. Ces augmentations permettent de multiplier par onze le nombre d'images de nos bases d'entraînement et de validation.

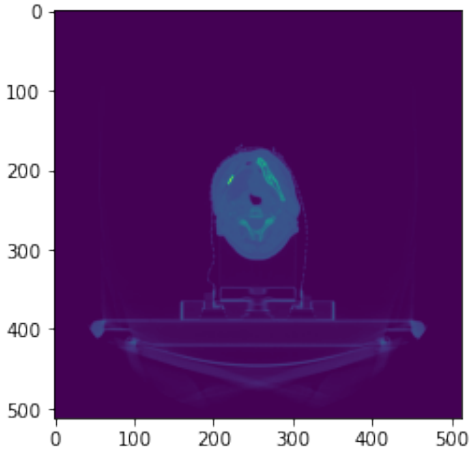


FIGURE 20 – L'image de base

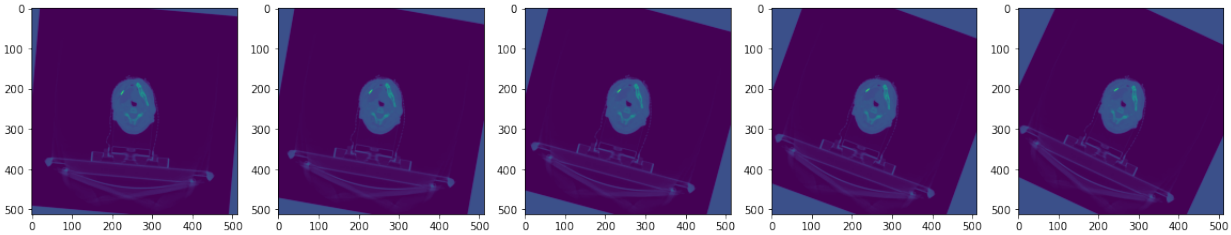


FIGURE 21 – Rotation de 5 à 30° par pas de 5 dans le sens horaire

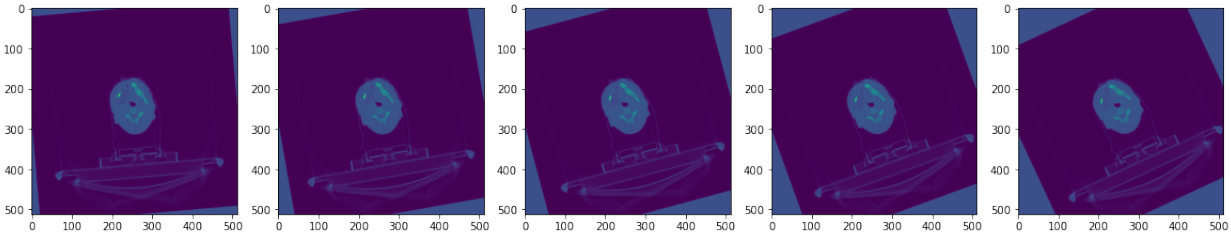


FIGURE 22 – Rotation de 5 à 30° par pas de 5 dans le sens anti-horaire

3 Analyse des données

3.1 Choix du réseau U-Net

Le réseau **U-Net** proposé en 2015 par Olaf Ronneberger, Philipp Fischer et Thomas Brox [5] est un réseau de neurones à convolution composé de trois blocs connectés entre eux : *l'encodeur*, *le goulot d'étranglement* et le *décodeur* d'où la forme en U. Nous avons

5. https ://arxiv.org/abs/1505.04597

choisi ce modèle car il est réputé pour son efficacité dans le domaine médical[6]. Il s'agit d'un modèle supervisé, dans notre cas l'apprentissage se fait sur des images avec comme vérité terrain des masques issus de contours réalisés par un médecin.

La longueur des blocs est très variable puisqu'on ajoute le nombre de couches de convolution que l'on veut. Dans notre code en **Keras** on utilise quatre couches successives de convolution alors qu'en **Lightning-Flash** l'encodeur utilisé est un **ResNet152**, c'est à dire qu'il y a cent-cinquante-deux couches de convolution successives, complétées par des *connexions de saut*[7] qui connectent les activations des couches en sautant certaines couches intermédiaires.

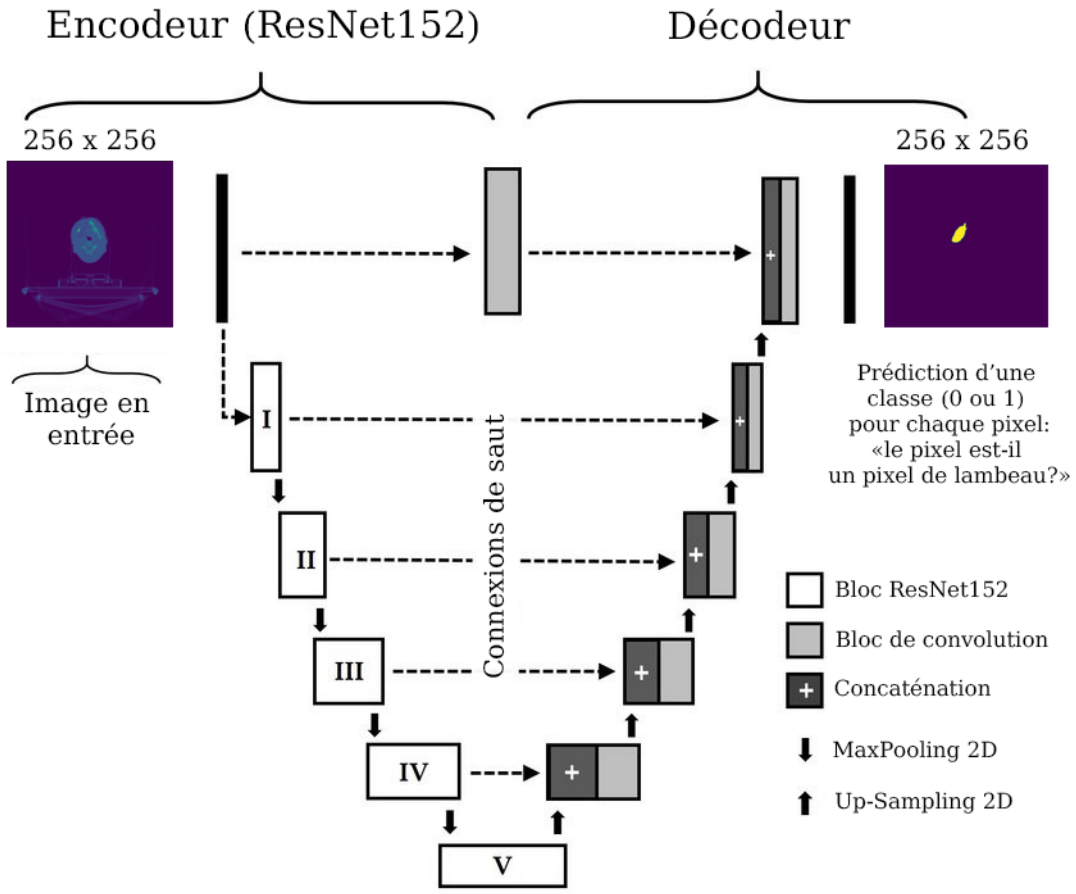


FIGURE 23 – Architecture du U-net que nous utilisons

6. vainqueur du Grand Challenge for Computer-Automated Detection of Caries in Bitewing Radiography at ISBI 2015, et du Cell Tracking Challenge at ISBI 2015

7. connexion qui saute par dessus une ou plusieurs couches de neurones

3.1.1 Encodeur ou bloc contractant

L'encodeur consiste en une succession d'enchaînements de convolutions suivies de fonctions d'activation de type **ReLu** [8](unité linéaire rectifiée) puis de **MaxPooling** [9] ce qui induit une réduction des dimensions mais permet la récupération de caractéristiques de l'image.

Il s'agit d'un bloc convolutionnel classique donc il permet un grand choix entre de nombreuses architectures (Variants de ResNet, VGG, EfficientNet...).

3.1.2 Pont ou goulot d'étranglement

Dans la littérature, le U-net est composé d'un bloc central, le goulot d'étranglement qui relie l'encodeur et le décodeur tout en effectuant des convolutions supplémentaires pour améliorer les informations transmises. Ce bloc n'est pas toujours implémenté dans les codes en ligne pour simplifier le code.

3.1.3 Décodeur ou bloc d'expansion

Ce bloc est composé de convolutions de transposition (Transposée de la convolution appliquée en phase d'encodage), Chaque bloc remontant étant relié au bloc descendant équivalent. Cela permet de remonter jusqu'aux dimensions initiales de l'image tout en bénéficiant des informations caractéristiques obtenues lors de l'encodage.

3.2 Principaux hyperparamètres d'un réseau

Une fois l'architecture fixée, le réseau doit être paramétré pour atteindre le meilleur résultat de segmentation tout en évitant le sur-apprentissage [10]. Les paramètres que nous manipulons pour régler l'entraînement sont nommés hyperparamètres par opposition aux paramètres qui sont calculés automatiquement par le modèle comme le biais d'un neurone.

3.2.1 Époque

Une époque correspond à une itération de l'étape d'entraînement et de mise à jour des poids des neurones sur tout l'échantillon d'images d'entraînement.

8. Fonction d'activation qui permet de remplacer les résultats négatifs par zéro
9. Couche du réseau qui permet de réduire la taille des images sans modifier leurs caractéristiques importantes
10. Situation où le modèle s'adapte trop à la base d'entraînement et n'arrive donc plus qu'à prédire de bons résultats sur des images identiques à celles d'entraînement mais prédit de mauvais résultats sur la phase de test.

3.2.2 Fonction objectif

La fonction objectif (ou *Loss function*) est la fonction qui permet de mesurer la distance entre la prédiction et la vérité terrain afin de donner au modèle la direction vers laquelle converger, communément en minimisant cette fonction.

Dans le cadre de notre projet, nous utilisons une fonction objectif basée sur le *score de Dice*. Ce score consiste à promouvoir l'intersection tout en sanctionnant les faux positifs et faux négatifs. Un score parfait de *Dice* vaut *un* et un score nul *zéro*. Les fonctions objectif étant généralement à minimiser, nous définissons comme DiceLoss la fonction qui associe à un couple prédiction-vérité terrain 1 moins son score de Dice.

$scoreDice = \frac{2*TP}{(TP+FP)+(TP+FN)}$

Nous avons fait le choix de ce score car il est très utilisé dans le contexte médical [11].

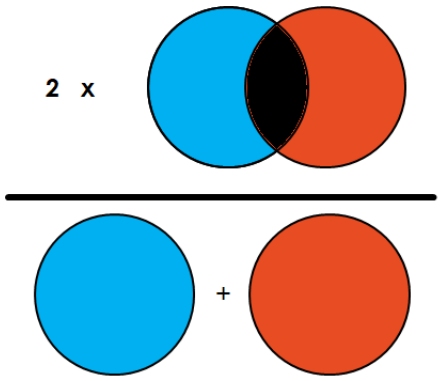


FIGURE 24 – le score de Dice

Pour tester la qualité de nos prédictions, nous utilisons directement le score de Dice à la différence près que l'on ne calcule le score que sur la prédiction du lambeau mais pas sur celle du fond. En effet, dans le cas contraire un modèle qui prédirait une image vide aurait systématiquement un bon score de Dice puisque le lambeau représente en moyenne 10% de l'image.

11. Optimizing the Dice Score and Jaccard Index for Medical Image Segmentation : Theory & Practice, Jeroen Bertels and Tom Eelbode and Maxim Berman and Dirk Vandermeulen and Frederik Maes and Raf Bisschops and Matthew B. Blaschko, 2019

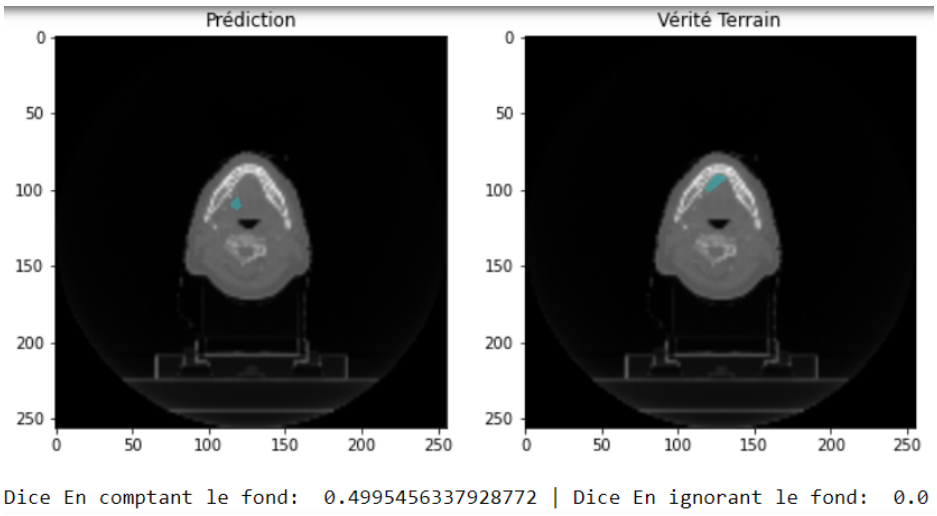


FIGURE 25 – Même en ne prédisant aucun bon pixel de lambeau, on obtient un bon score si on n'ignore pas le fond dans le score de Dice

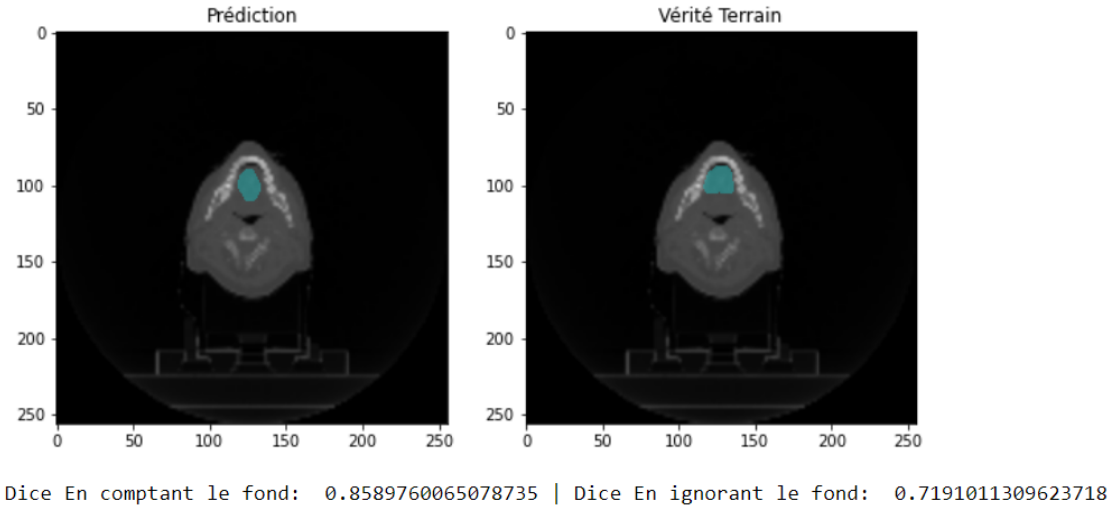


FIGURE 26 – Différence de score de Dice en fonction de la prise en compte du fond pour une bonne prédiction de lambeau

3.2.3 Taux d'apprentissage

Le taux d'apprentissage ou *learning rate* est le coefficient qui est appliqué au gradient lors de la descente de gradient. Un coefficient élevé fera converger vers une solution plus rapidement mais un coefficient trop élevé peu rater la solution. Il sert donc à mettre à jour les poids de chaque neurone.

Ce coefficient peut être fixé ou adaptatif selon la méthode d'optimisation qu'on utilise.

3.2.4 Optimiseur

Les optimiseurs sont les méthodes qui permettent d'avancer sur la courbe de la fonction objectif afin d'en trouver l'optimum. Nous avons fait le choix d'utiliser la méthode *Adam*

car c'est la plus réputée de par son efficacité et sa stabilité. Elle consiste à augmenter continuellement le taux d'apprentissage tant que le gradient calculé va dans la même direction que les précédents.

Durant nos apprentissages, nous laissons le taux d'apprentissage par défaut d'Adam, c'est à dire 0.001 afin de s'assurer de garder en précision car même en partant d'un faible taux d'apprentissage, on sait qu'il va finir par accélérer donc on a quand même un temps d'exécution viable.

3.2.5 Taille des batchs

Les réseaux de neurones manipulant énormément de données, on gagne beaucoup de temps à agglomérer les données par paquets (ou *batchs*) avec pour limite de taille ce que la carte graphique peut contenir en mémoire.

Cependant d'après cet article [12] les méthodes avec des batchs larges tendent à converger vers des *sharp minimizers* (pointus) des fonctions d'entraînement et des fonctions de test – et qu'un *sharp minima* produit de mauvaises généralisations. Au contraire, les méthodes avec des batchs petits convergent vers des flat minimizers (plats).

En effet pour un nombre d'époques fixé, un modèle réglé avec une grande taille de batch poursuivra son apprentissage jusqu'au bout quand bien même il arrivera en sur-apprentissage alors qu'un nombre de batch plus restreint arrivera à terme plus lentement mais avec un taux d'erreur à la phase de test bien moindre.

Le compromis entre grande taille de batchs et sur-apprentissage peut être contrôlé en utilisant une méthode d'*early stopping*, c'est à dire en arrêtant l'entraînement si la fonction objectif de validation atteint un plateau pendant un nombre défini d'époques même si la fonction objectif d'entraînement continue à baisser pour mieux (voire trop) se conformer aux données d'entraînement ce qui induit du sur-apprentissage et donc des scores de test plus faibles que si le modèle avait été arrêté avant.

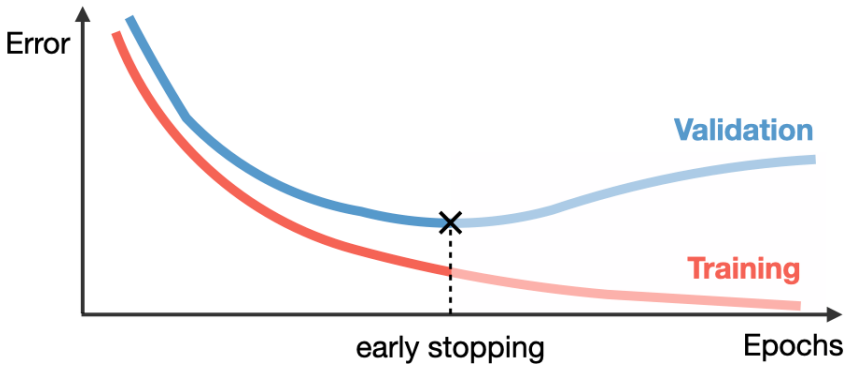


FIGURE 27 – Arrêt de l'entraînement quand la *fonction objectif de Validation* stagne sur un plateau

12. https ://wandb.ai/wandb_fc/french/reports/Quelle-est-la-Batch-Size-Optimale-pour-Entra-ner-un-Neural-Network—Vmlldzo1NzkyMzc

3.3 Implémentation

3.3.1 Choix des notebooks

Dans le cadre de l'apprentissage profond, il n'y a pas de consensus sur la manière de programmer en python brut ou par l'utilisation de *Jupyter-Notebook* car on trouve autant de code implémenté de chaque manière avec cependant une légère tendance Pytorch-Fichiers Python et Tensorflow-Notebook.

Nous avons instantanément choisi d'utiliser Jupyter-Notebook car il offre une grande lisibilité et est très pratique pour l'affichage d'images et de graphiques. De plus, le projet étant multi-disciplinaire, alliant informaticiens et radiothérapeutes, Le principe des cellules nous permet de montrer la démarche du code, étape par étape plus facilement ce qui était particulièrement utile pour les rendez-vous hebdomadaires avec les encadrants.

Nous utilisons toutefois quelques fichiers Python pour les boîtes à outils réalisées pour manipuler les fichiers dicom, gérer l'affichage et autres divers outils ainsi que pour les objets dont l'implémentation n'a plus à être modifiée comme notre fonction objectif afin d'alléger le notebook qui reste le corps de l'exécution.

La limite des Notebooks s'est située, pour nous, au niveau de l'utilisation des serveurs de calculs car nous ne pouvions pas y lancer le programme puis le détacher du serveur via les commandes telles que *tmux* qui permettent de se déconnecter de l'ordinateur tout en laissant tourner le programme sur le serveur. De même, le module *Lightning-Flash* offre la possibilité de travailler facilement en multi-GPU mais cela n'est pas possible si l'on utilise des Notebooks.

3.3.2 Choix des frameworks

Durant le stage nous avons beaucoup navigué entre les deux principaux frameworks de machine learning : *Pytorch* et *Tensorflow*. *Pytorch* nous était presque unanimement conseillé par les professeurs et les doctorants ce qui nous a poussé à nous tourner vers lui au départ afin d'avoir des personnes à qui demander conseil plus facilement en cas de besoin tandis que sur internet, la grande majorité des travaux présentés les plus récents étaient implémentés en *Tensorflow* ce qui fait que nous n'avons pas fait l'impasse dessus.

Pytorch comme Tensorflow fournissent aussi leur module de plus haut-niveau permettant de manipuler des objets complexes déjà implémentés et manipulables via des versions assouplies des méthodes de Pytorch et de Tensorflow. Ces frameworks de plus haut niveau se nomment *Pytorch-Lightning* et *Tensorflow Keras*.

Au départ, nous avons utilisé Pytorch car il était bien adapté pour travailler sur les serveurs de calculs du GREYC alors que je n'ai réussi à utiliser Tensorflow avec les cartes graphiques qu'au bout de plusieurs jours d'installation : Les erreurs concernant les cartes graphiques sont peu claires et la version de cuda (technologie permettant d'effectuer les

calculs sur les GPUs) annoncée par la commande shell *nvidia-smi* n'est pas la version réelle de cuda mais la version des plugins les plus récents qu'elle peut faire tourner. Il faut utiliser la commande *cat /usr/local/cuda/version.txt* pour connaître la bonne version de cuda. La version de cuda étant plus vieille ce n'était pas directement Tensorflow qu'il fallait installer mais *Tensorflow-GPU*, ancienne version adaptée au GPU alors qu'avec les versions récentes cela est directement implémenté dans Tensorflow. Il a ensuite fallu s'assurer de bien installer le module dans l'environnement anaconda sur le bon noyau Python.

Nous n'avons jamais travaillé en Tensorflow pur mais seulement en Tensorflow Keras car tous les codes récents en Tensorflow que nous avons rencontrés étaient sous ce framework. Cela se justifie d'une part par sa plus simple manipulation mais aussi parce qu'il propose directement les implémentations des principales métriques, fonctions objectif et optimiseurs ainsi que des couches qui permettent de créer un modèle (convolutions, Max-Pooling...).

3.3.3 Sélection de Lightning-Flash

Les créateurs de Pytorch et Pytorch-Lightning développent aussi actuellement un module encore plus haut niveau que Pytorch-Lightning nommé Lightning-Flash. Ce module permet de créer un modèle et l'entraîner de manière très abstraite, en très peu de lignes de codes. Ce module étant très récent, il n'y a que très peu de code fourni par la communauté et la documentation n'est pas encore très complète mais sa facilité de manipulation nous a fait l'adopter.

Ce module propose la plupart des réseaux disponibles dans l'état de l'art pour la segmentation sémantique avec la possibilité de les utiliser pré-entraînés. On peut aussi choisir les nombreuses métriques et fonctions objectif proposées par *torchmetrics* ou alors utiliser les nôtres à condition de les faire hériter des classes manipulées par Pytorch.

Flash propose une nouvelle approche de la manipulation des données : les *datamodules* qui permettent à l'utilisateur d'ignorer la partie chargement des données via *dataloader* voire même des datasets selon les applications. Dans notre cas, comme nous avons déjà fait l'augmentation de données lors de la création de la base, nous n'avons pas besoin de créer un dataset car on peut utiliser la méthode *fromFolders* du *datamodule* pour simplement fournir les dossiers d'entraînement et de test, la séparation entre l'entraînement et la validation se faisant aussi dans le *datamodule*. Il offre la liberté d'utiliser la plupart des extensions de fichiers d'images et de matrices et effectue lui-même la normalisation des images.

Nous avons longtemps travaillé sur de la classification binaire, principalement avec Keras alors que Flash nous a ouvert la porte à la segmentation multiclasse bien qu'il soit aussi possible de segmenter de manière binaire puisque le nombre de classes désiré est totalement paramétrable, se répercutant sur le nombre de canaux de l'image prédite, chaque canal correspondant à un label.

Le dernier point qui nous a poussé à conserver Flash pour les expérimentations est que celui-ci nous permet d'utiliser des réseaux pré-entraînés sur *Imagenet*, une base de données composée de plus d'un million d'images annotées.

Ainsi au lieu de lancer notre modèle avec des poids complètement aléatoires, nous chargeons les poids ayant été appris sur *Imagenet* ce qui aide notre modèle à converger plus vite puisqu'il a déjà appris des textures et des caractéristiques.

Cependant, *Imagenet* est constituée d'images de la vie courante et non spécifiquement d'images médicales alors en quoi cela peut-il nous aider à prédire un lambeau ?

En réalité, on va geler les couches basses du réseau, c'est à dire en fixer les poids pour les conserver sur notre modèle. En effet, ces couches basses sont responsables de la détection des textures les plus simples de la géométrie comme les traits horizontaux et verticaux. Ces formes sont communes à toutes les images et peuvent donc être directement utilisées par nôtre réseau plutôt que de passer du temps à les ré-apprendre.

3.3.4 Outil pour interpréter les résultats

Pour voir le processus d'entraînement en temps réel et pouvoir en garder les traces, nous utilisons un *logger* nommé *Wandb* (Weights & Biases) qui permet d'afficher toutes les courbes des métriques, de la fonction objectif et même les images prédites au cours de l'entraînement.

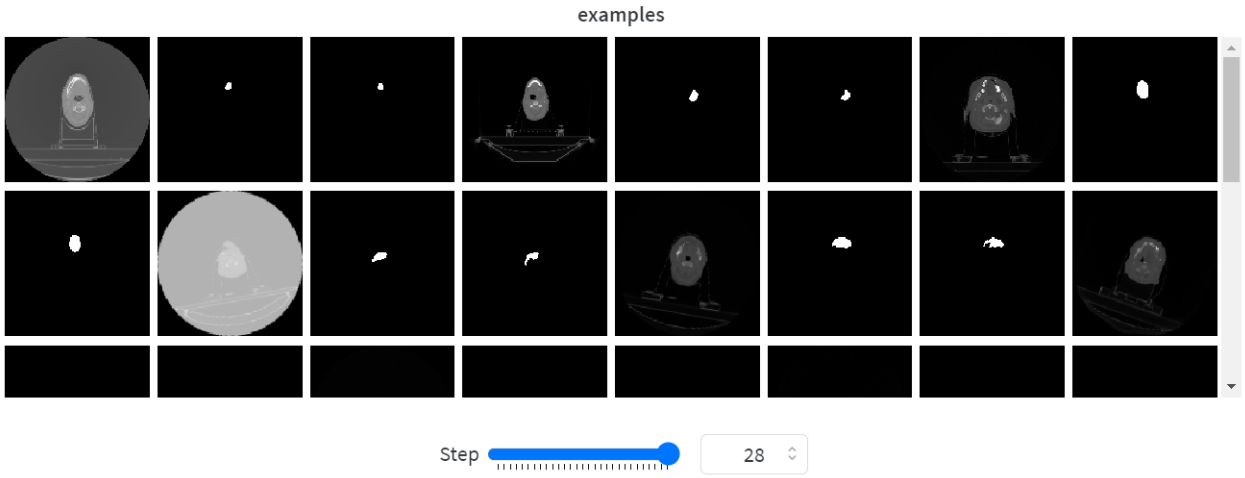


FIGURE 28 – les images prédites au cours de l'entraînement

Wandb hiérarchise les apprentissages par projets ce qui nous permet d'afficher les courbes d'un même projet sur le même graphique afin de comparer l'évolution des divers apprentissages et les résultats de leur phase de test.

FIGURE 29 – les métriques et la fonction objectif d'un apprentissage

4 Résultats

Nous avons constitué notre base de données tout au long du stage ce qui fait que nous avons travaillé longtemps sur une base relativement réduite mais cela ne nous a pas empêché de tester plusieurs hypothèses de travail. L'objectif n'était alors pas forcément d'atteindre de bons résultats mais surtout d'illustrer des tendances pour savoir sur quelle piste nous concentrer une fois la base de données finale acquise.

4.1 Tensorflow

Le code que nous utilisons en Keras fonctionne uniquement pour la classification binaire. Ce U-net est basé sur la succession de quatre couches de convolutions. Avec ce modèle nous avons principalement utilisé les métriques **IOU** (Intersection over Union), la **distance de Hausdorff** et le **Dice**. Ces expérimentations nous ont confirmé qu'utiliser Dice comme fonction objectif était le plus pertinent.

Cependant, ce modèle a vite eu tendance à faire du sur-apprentissage donc nous l'avons écarté au profit de celui codé en *lightning-flash*.

En arrêtant l'entraînement autour de la trentième époque, avant le sur-apprentissage grâce à la méthode d'*Early-Stopping*, nous faisons converger notre modèle vers un score de dice moyen de **0.378** et un score médian de **0.400**.

4.2 Fenêtrage de Hounsfield

A l'oeil nu, utiliser un fenêtrage met en valeur le lambeau et supprime les matières qui ne sont pas incluses dans le fenêtrage. Nous avons donc regardé si filtrer les images dans

min dice :	0.004366812227074236
max dice :	0.9415204678362573
moyenne dice :	0.37830418691985257
médiane dice :	0.40026446953514394

FIGURE 30 – Résultats du modèle programmé en Tensorflow

l'intervalle [-125, +250], afin de prendre en compte la graisse, le muscle et les tissus mous apportait un plus dans le score de Dice.

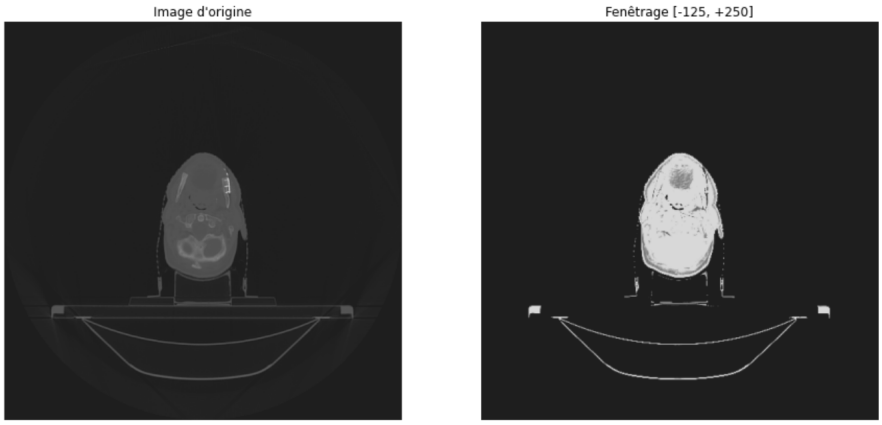


FIGURE 31 – Mise en valeur visuelle du lambeau par filtrage [-125, +250]

Nous avons utilisé un U-net composé d'un *ResNet152* pré-entraîné avec pour fonction objectif notre *DiceLoss* durant 80 époques.

Le modèle a été entraîné sur une base de 30 patients et testé sur 6 patients.

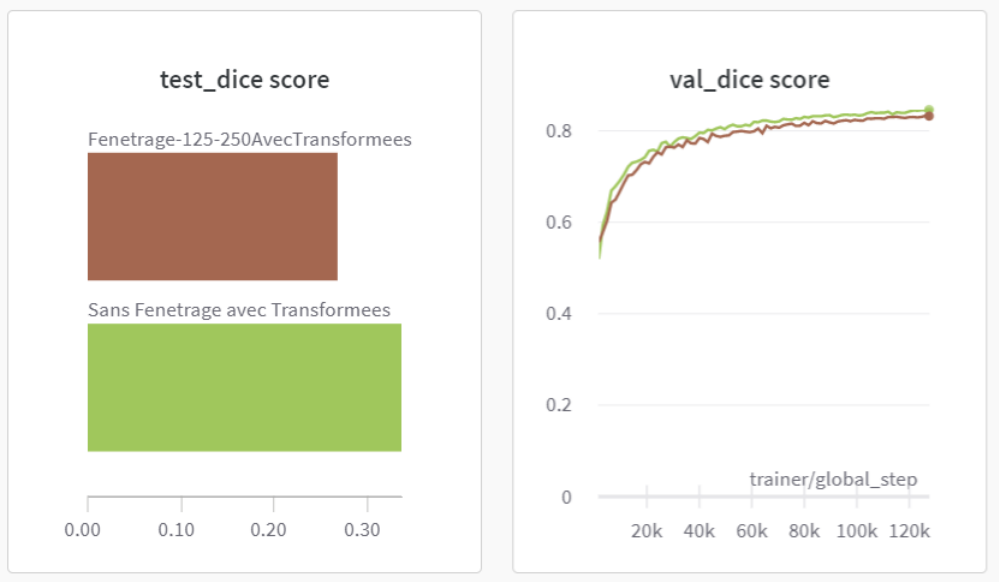


FIGURE 32 – Score de dice en fonction de l'utilisation ou non du filtrage

Notre modèle nous fourni un score de Dice de 0.3369 sans utilisation de filtrage contre un score de 0.2687 avec l'utilisation du fenêtrage [-125, +250]. Par ailleurs, à aucun moment de l'entraînement la courbe du fenêtrage ne dépasse celle de l'absence de fenêtrage.

Nous en déduisons donc que l'utilisation du filtrage détériore trop les images et donc qu'il vaut mieux utiliser les images brutes pour prédire un lambeau.

4.3 Graisse et Muscle

Le lambeau étant composé de graisse et de muscle, nous avons voulu voir l'influence de ces deux composantes sur la détection du lambeau dans l'image. Au lieu d'avoir un masque binaire (lambeau = 1, le reste = 0), on travaille avec un masque composé de trois classes : 1 pour la graisse, 2 pour le muscle et 0 pour le reste.

Pour expérimenter cela, nous avons utilisé deux U-net composés d'un *ResNet152* pré-entraîné avec pour fonction objectif notre DiceLoss durant 30 époques avec notre base de données finale. Avec les mêmes paramètres et le même découpage des données, l'un des modèles est entraîné sur les lambeaux binaires et l'autre sur les lambeaux séparés en graisse et muscle.

Pour les scores de test issus de Flash-Lightning, la qualité de la prédiction est jugée sur l'efficacité à prédire les composantes graisseuses et musculaires. Pour nos propres calculs, puisque l'objectif final est simplement de prédire le lambeau, on calcule le score de Dice sur l'union des composantes graisseuses et musculaires binarisées pour reconstruire le lambeau.

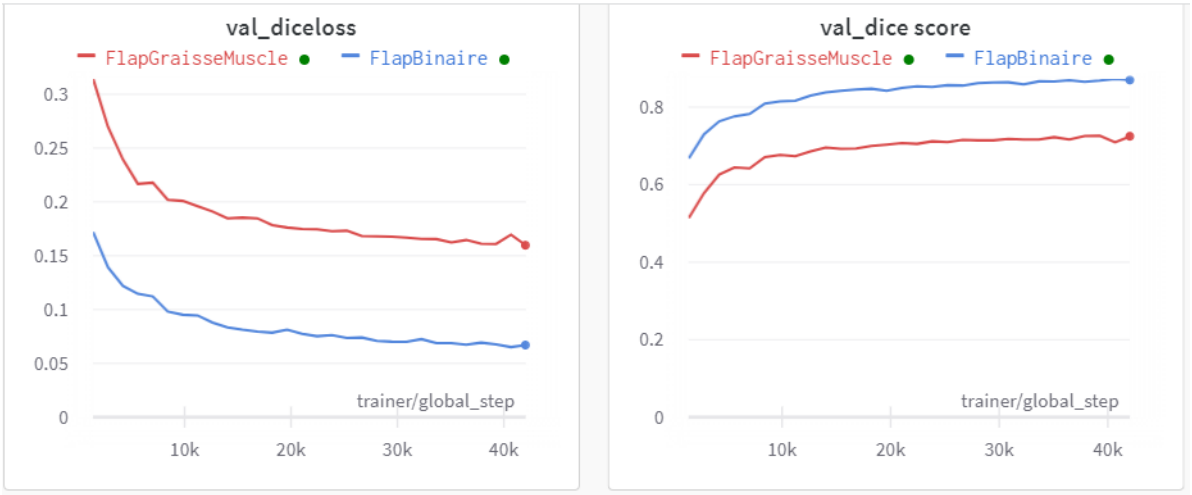


FIGURE 33 – Courbes de validation en fonction de la décomposition ou non en composantes graisseuses et musculaires

L'allure des deux courbes est similaire mais le modèle du lambeau décomposé n'arrive pas à atteindre un aussi bon score que le modèle binaire fait en seulement quelques époques.

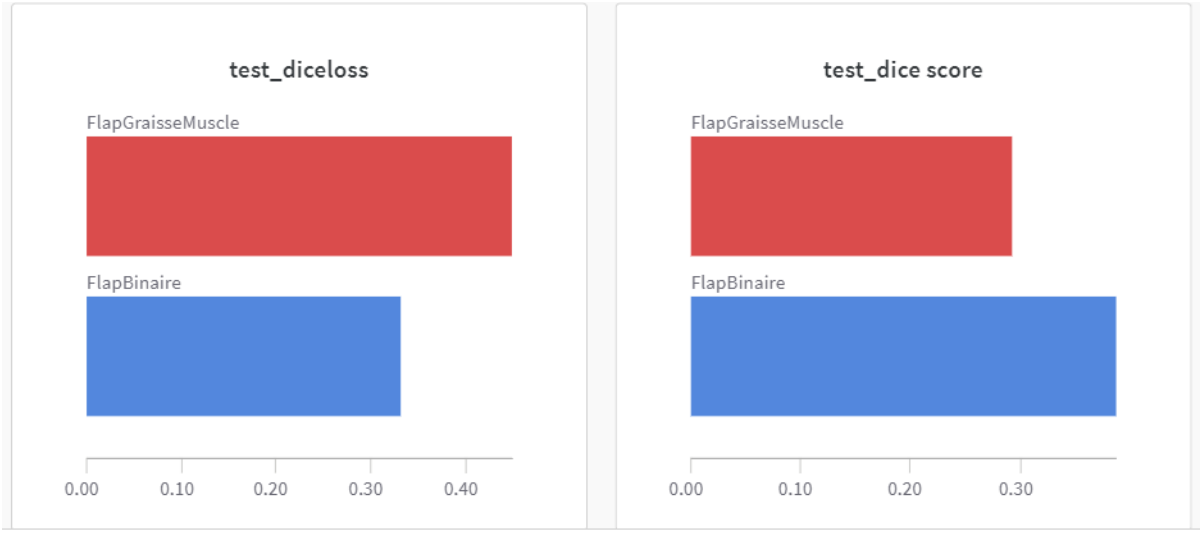


FIGURE 34 – Score de dice en fonction de la décomposition ou non en composantes graisseuses et musculaires

On obtient donc un score de dice de 0.2927 Pour la décomposition graisse/muscle contre 0.3874 pour le lambeau. Le calcul via la *test_diceLoss* (1 moins diceLoss pour obtenir le score) donne un meilleur résultat : 0.5515 Pour la décomposition graisse/muscle contre 0.6681 pour le lambeau car la bonne prédiction du fond augmente le score de la diceLoss mais est ignorée dans le cas du *test_dice score*.

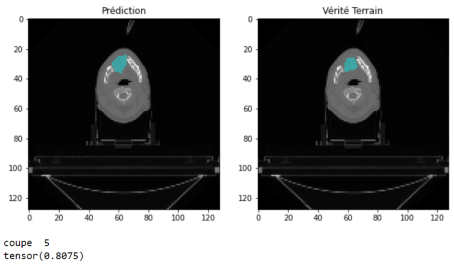


FIGURE 35 – Exemple de prédiction du modèle binaire

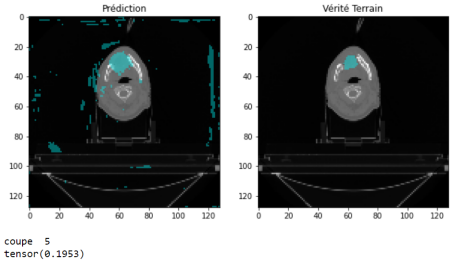


FIGURE 36 – Exemple de prédiction du modèle par décomposition graisse et muscle

Pour calculer nos scores de Dice, il est plus pertinent de les rapporter en scores par patient car l'important est de prédire un lambeau plus qu'une coupe de lambeau. L'écart entre les deux modèles est encore plus mis en avant par les histogrammes des moyennes par patient ci-après.

En conclusion, la décomposition en composantes graisseuses et musculaires du lambeau n'est pas un moyen pertinent d'aider à la discrimination du lambeau durant la segmentation.

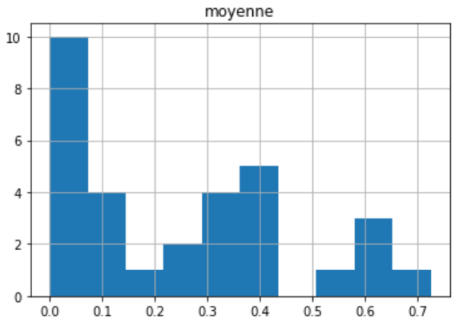


FIGURE 37 – Histogramme du score moyen par patient avec le modèle binaire

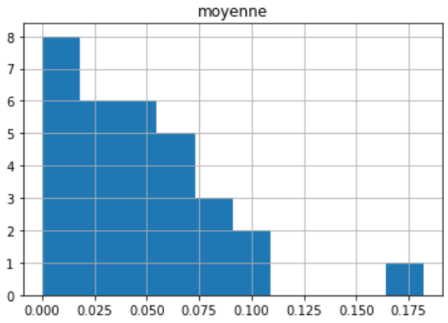


FIGURE 38 – Histogramme du score moyen par patient par décomposition graisse et muscle

4.4 Multiclasse

Nous avons aussi émit l'hypothèse qu'ajouter des informations sur le contexte du lambeau à l'aide d'autres organes à risques voisins pouvait aider l'apprentissage du modèle. Nos régions d'intérêt sont donc le lambeau, la mandibule, la cavité buccale, les glandes parotides, le larynx, et les tumeurs.

Pour expérimenter cela, nous avons utilisé deux U-net composés d'un ResNet152 pré-entraîné avec pour fonction objectif notre DiceLoss durant 30 époques avec notre base de données finale. Avec les mêmes paramètres et le même découpage des données, l'un des modèles est entraîné sur les lambeaux binaires et l'autre sur les images contenant le lambeau et les organes à risque voisins. Afin d'assurer que notre fonction objectif se concentre bien sur le lambeau, ce dernier se trouve sur strictement toutes les images contrairement aux autres régions.

Dans l'apprentissage multiclasse, le score de Dice de la partie test est calculé toutes régions d'intérêt confondues. Autrement dit, on peut imaginer avoir un très bon score de Dice si toutes les régions sont prédites correctement mais pas le lambeau. Pour connaître la vraie qualité du modèle multiclasse à prédire le lambeau, nous ne calculerons le score de Dice avec nos propres méthodes de calcul uniquement sur celui-ci. Comme expliqué précédemment, dans le cadre du multiclasse, les intersections entre plusieurs régions d'intérêts sont traduites par la création d'une nouvelle classe. Ainsi, le lambeau étant dans la grande majorité des cas inclus dans la cavité buccale, on ne se contente pas de mesurer la prédiction de la classe *lambeau* mais on utilise aussi la classe *lambeau + cavité buccale*.

L'allure des deux courbes est similaire mais le modèle multiclasse n'arrive pas à atteindre un aussi bon score que le modèle binaire fait en seulement quelques époques.

Dans la phase de test, on obtient donc un score de Dice de 0.3507 pour la prédiction multiclasse toutes classes confondues contre 0.3735 pour le lambeau uniquement via le modèle binaire.

FIGURE 39 – Courbes de validation des modèles multiclasse et binaires

FIGURE 40 – Score de Dice sur toutes les régions d'intérêt pour le modèle multiclasse et sur le lambeau pour le modèle binaire

L'objectif du projet étant la segmentation automatique de lambeau et non pas des autres régions d'intérêt, nous calculons plutôt le score de Dice par patient sur l'intersection *lambeau - Cavité Orale*.

La moyenne du score de Dice pondérée par le nombre de coupes sur l'ensemble des patients de test nous donne un score de 0.1782 pour le modèle multiclasse et un score de 0.2874 pour le modèle binaire. Ces deux scores sont minimisés par l'influence de patients avec des scores nuls (trois pour le modèle multiclasse, deux pour le modèle binaire).

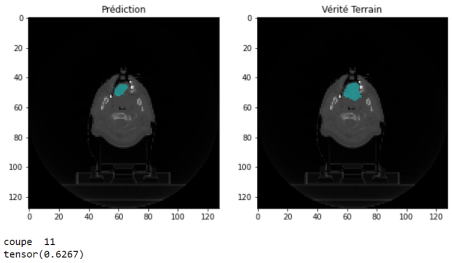


FIGURE 41 – Exemple de bonne prédiction du lambeau avec le modèle binaire

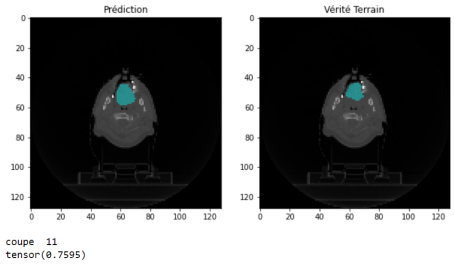


FIGURE 42 – Exemple de bonne prédiction du lambeau inclus dans la cavité buccale avec le modèle multiclasse

En conclusion, l'utilisation d'un modèle multiclasse, qui prédit les régions d'intérêt voisines (ou superposées en 2D) et le lambeau fourni un résultat moins bon que l'utilisation d'un modèle de classification binaire qui ne se concentre que sur le lambeau.

4.5 Modèle Final

Le modèle que nous avons choisi, programmé à l'aide de **Lightning-Flash** utilise un encodeur de type ResNet152. Nous avons fait ce choix par comparaison avec quelques autres modèles proposés par **Lightning-Flash** (variants d'EfficientNet, DenseNet, VGG,...) car pour un résultat équivalent, **ResNet152** avait, par exemple, moité moins de paramètres que **VGG**. [13] Ce plus faible nombre de paramètres nous permet de gagner beaucoup de temps sur la phase d'apprentissage.

Ce modèle a été entraîné sur notre base finale durant 40 époques avec comme fonction objectif notre DiceLoss.

13. 60 Millions de paramètres pour ResNet152 contre 138 Millions pour VGGNet d'après https ://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

FIGURE 43 – Courbes du modèle final sur 40 époques

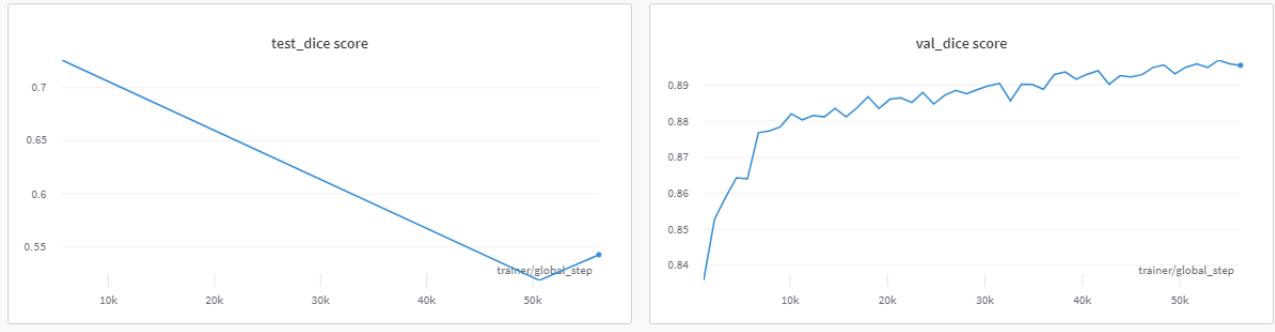


FIGURE 44 – La poursuite de l'apprentissage sur 50 époques supplémentaires induit du sur-apprentissage : le score de Dice diminue drastiquement

Ce modèle nous donne un score de Dice moyen pour une coupe de 0.73 en phase de test. L'objectif restant la prédiction du lambeau, toutes les statistiques à suivre concernent plutôt des scores de Dice calculés par patient (score de Dice moyen pondéré par le nombre de coupes).

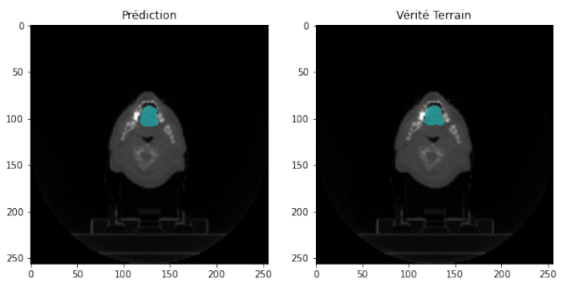


FIGURE 45 – Bonne prédiction du lambeau. Dice = 0.9117

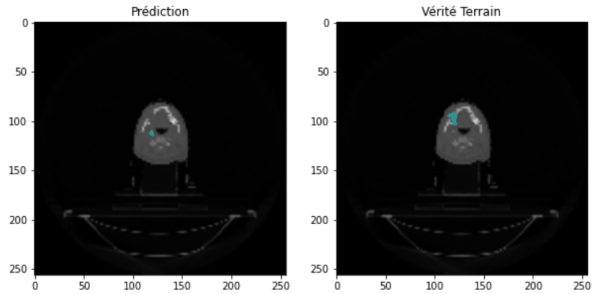


FIGURE 46 – Mauvaise prédiction du lambeau. Dice = 0

	patient	min	max	moyenne	mediane	coupes	PixelsFlap/Patient
0	Patient100	0.33766234	0.94009215	0.8068461	0.8717949	17	5543
1	Patient101	0.0	0.81784385	0.44968316	0.66715777	14	1450
2	Patient102	0.0	0.9050773	0.57885414	0.7183888	14	2402
3	Patient103	0.0	0.81948423	0.45788416	0.6039216	9	1131
4	Patient104	0.5449102	0.8764045	0.7728943	0.8001815	28	10838
5	Patient105	0.0	0.9225589	0.58650094	0.7692308	31	2589
6	Patient106	0.0	0.8845209	0.64898336	0.77078086	17	3866
7	Patient107	0.051282052	0.8765432	0.75059927	0.7987421	29	8883
8	Patient109	0.0	0.88212925	0.77914006	0.81287724	61	12679
9	Patient110	0.0	0.86577183	0.48432705	0.5942029	49	5542
10	Patient111	0.0	0.75384617	0.12422529	0.0	13	573
11	Patient112	0.0	0.85809314	0.5760982	0.73728454	32	8723
12	Patient114	0.0	0.0	0.0	0.0	13	628
13	Patient116	0.4262295	0.8779715	0.7644914	0.8030014	16	3558
14	Patient117	0.0	0.8344827	0.5988808	0.75524473	27	3910
15	Patient118	0.0	0.24920128	0.08719408	0.0111524165	8	946
16	Patient121	0.0	0.91442543	0.67839533	0.85209006	15	2681
17	Patient122	0.0	0.8773389	0.6771601	0.79591835	19	3377
18	Patient126	0.0	0.0	0.0	0.0	10	560
19	Patient127	0.14117648	0.8888889	0.6234967	0.713253	13	2207
20	Patient128	0.0	0.6320988	0.36497784	0.45943847	16	1475
21	Patient130	0.0	0.78752434	0.45496738	0.5041322	11	1453
22	Patient131	0.0	0.6857143	0.07898582	0.0	22	13080
23	Patient132	0.0	0.6666667	0.18620592	0.018691588	27	2842
24	Patient133	0.115869015	0.69447577	0.57244086	0.6099244	24	10113
25	Patient134	0.0	0.88125	0.6135784	0.72789633	24	9949
26	Patient95	0.0	0.9	0.7697821	0.82258064	47	10514
27	Patient96	0.0	0.9430693	0.78159994	0.8783977	17	4697
28	Patient97	0.0	0.8506944	0.49551275	0.6967298	18	2608
29	Patient98	0.0	0.8770764	0.6592147	0.72139305	17	2851
30	Patient99	0.0	0.91168094	0.7482259	0.8422753	22	70721

FIGURE 48 – Scores de Dices calculés par patient

	min	max	moyenne	mediane	coupes	PixelsFlap/Patient
count	31.000000	31.000000	31.000000	31.000000	31.000000	31.000000
mean	0.052165	0.770159	0.521650	0.592151	21.935484	6851.258065
std	0.134690	0.244962	0.249269	0.308332	12.173566	12459.758318
min	0.000000	0.000000	0.000000	0.000000	8.000000	560.000000
25%	0.000000	0.770685	0.452325	0.549168	14.000000	1841.000000
50%	0.000000	0.876405	0.586501	0.721393	17.000000	3377.000000
75%	0.000000	0.886705	0.713311	0.799462	27.000000	8803.000000
max	0.544910	0.943069	0.806846	0.878398	61.000000	70721.000000

FIGURE 47 – Statistiques mesurées sur les 31 Patients de Test

Sur les statistiques mesurées par la base, pour connaître le score moyen d'un patient, on peut utiliser la moyenne des moyennes qui est de 0.521650 mais ce score est biaisé par le fait que les moyennes ne sont plus pondérées par le nombre de coupes contenues par le patient : le score moyen d'un patient de 6 coupes compte autant qu'un de 60 coupes. Pour cela on a aussi calculé la moyenne pondérée par le nombre de coupes ce qui nous donne un score de Dice de 0.5714 Par ailleurs, cette valeur est tirées vers le bas par l'influence de deux patients dont toutes les prédictions sont vides de part la mauvaise qualité des images.

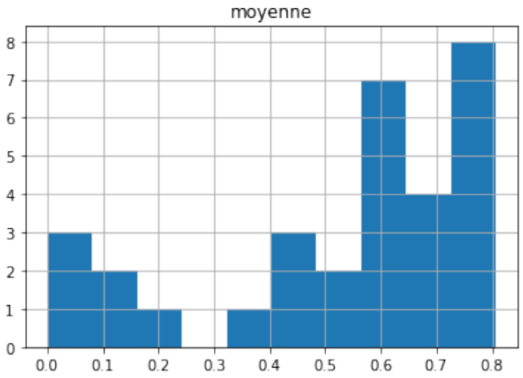


FIGURE 49 – Histogramme de la moyenne par patient

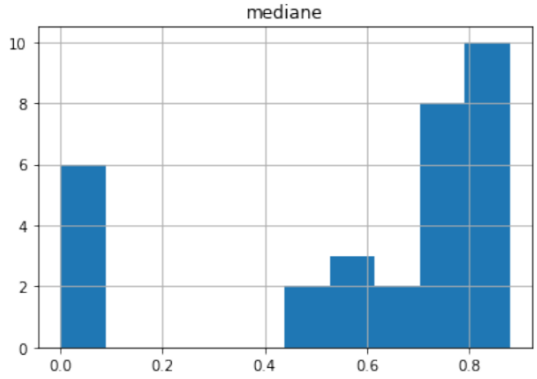


FIGURE 50 – Histogramme de la médiane par patient

Ce modèle nous donne un score de Dice moyen sur une coupe de 0.73 mais plus précisément pour un patient, son score moyen pondéré par son nombre de coupes est de 0.5214 tandis que son score médian est de 0.72. Ce modèle n'est donc pas encore opérationnel pour un usage clinique.

4.6 Analyse des données

A partir de cette base de test nous avons ensuite étudié différentes pistes qui semblaient biaiser les données.

Nous avons tout d'abord suivi une stratégie qui consistait à n'utiliser que les coupes ayant du lambeau. Si cette stratégie risque de trouver du lambeau là où il n'y en a pas (faux positifs), il faut cependant savoir que le radiothérapeute qui trace le contour sur un scanner de planification de radiothérapie postopératoire a toujours à disposition le compte rendu opératoire. Ce compte rendu opératoire comprend un temps de résection tumorale, et le cas échéant de prélèvement du lambeau puis de transplantation du lambeau à l'endroit de la résection tumorale.

Ainsi notre stratégie était robuste à ce domaine d'application correspondant au besoin médical.

Cependant, nous avons noté que sur les 31 patients de test (nombre de coupe moyen : 21, minimum : 8, maximum : 61), le modèle était mis en défaut notamment dans deux situations majeures :

4.6.1 Influence du nombre de coupes

D'une part, il nous a semblé que les patients avec moins de coupes sauvegardées (<10) avaient un score bien plus mauvais. Nous avons donc calculé le coefficient de Pearson du score de Dice en fonction du nombre de coupes ce qui nous a donné un résultat de *0.37*. On en déduit donc que le nombre de coupes et le score du patient ne sont que faiblement corrélés.

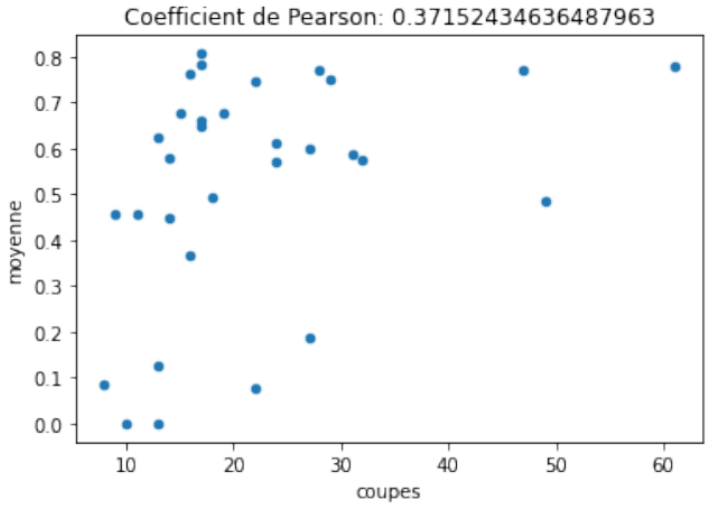


FIGURE 51 – Score de Dice du patient en fonction du nombre de coupes

4.6.2 Prédiction des coupes aux extrémités

D'autre part, on constate aussi que pour certains patients le modèle a du mal à prédire les coupes extrêmes en bas (caudal) ou haut (cranial). On suppose que cela est dû au nombre de coupes des extrémités du lambeau qui est moindre du fait du seuillage des coupes avec peu de lambeau (<150px).

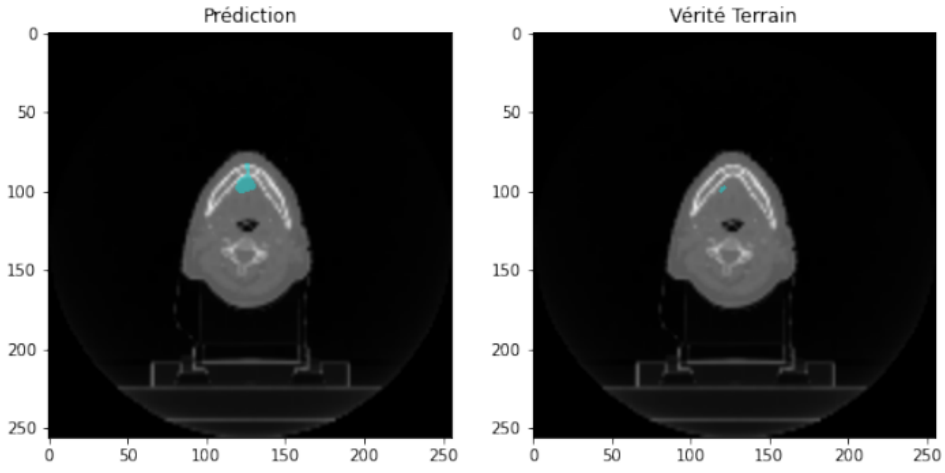


FIGURE 52 – Première coupe. Dice = 0.1304

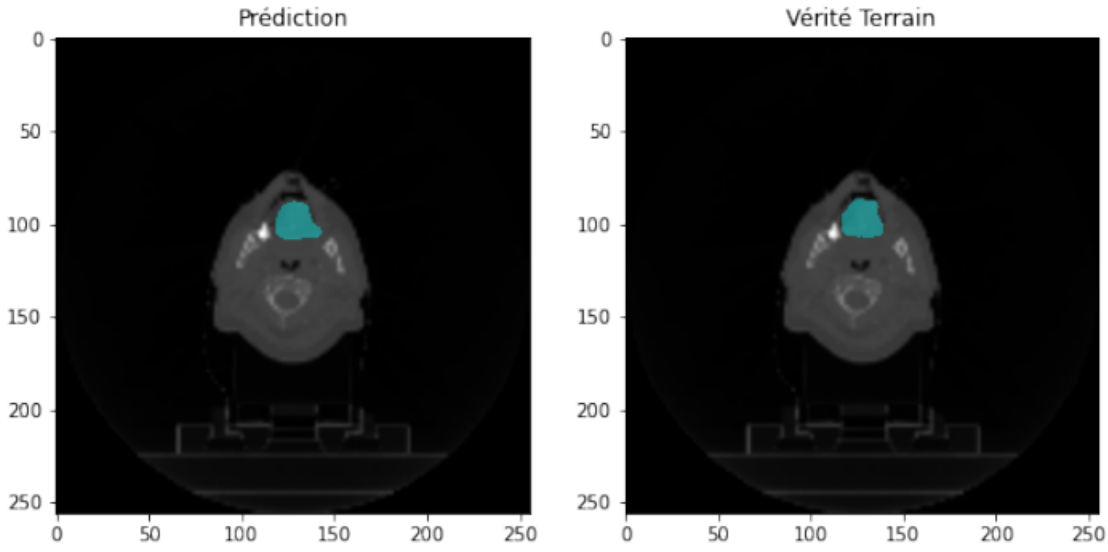


FIGURE 53 – Coupe médiane. Dice = 0.8507

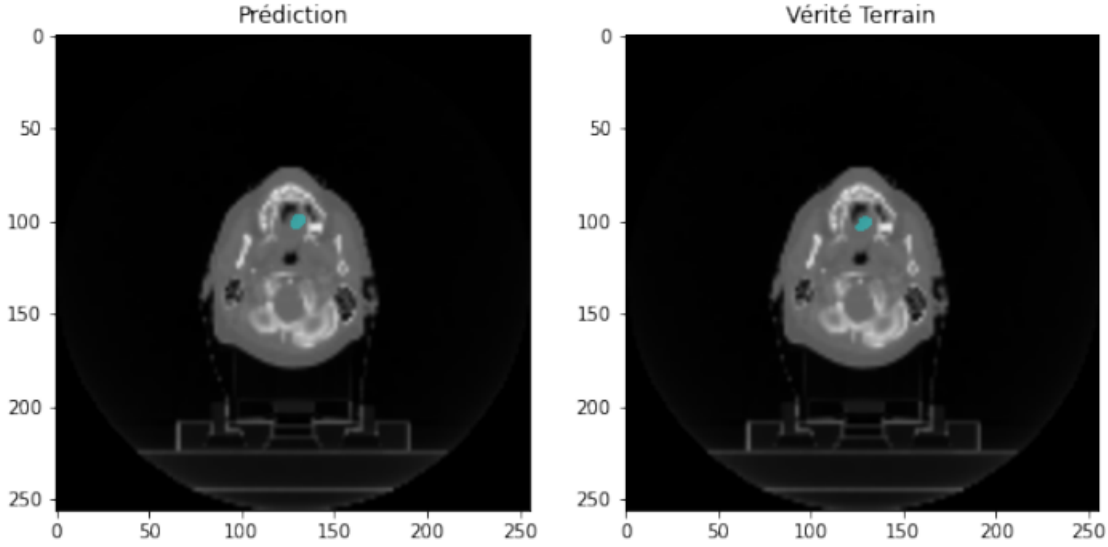


FIGURE 54 – Dernière coupe. Dice = 0.6602


```
{'patient': 'Patient99',
 'min': 0.0,
 'max': 0.91168094,
 'moyenne': 0.7482259,
 'mediane': 0.8422753,
 'coupes': 22,
 'PixelsFlap/Patient': 70721}
```


FIGURE 55 – Informations statistiques du patient 99

Nous avons alors utilisé toutes les coupes de lambeau pour la phase de Test. Ce modèle nous permettait aussi de tester si les performances du premier modèle, mis en défaut aux extrémités, pouvaient être améliorées.

4.6.3 Taille du lambeau

Enfin, il nous a semblé que le modèle produisait un meilleur score de Dice chez les patients munis d'un plus grand lambeau. Nous avons donc calculé le coefficient de Pearson du score de Dice en fonction du nombre de pixels de lambeau ce qui nous a donné un résultat de *0.3940*. On en déduit donc que la taille du lambeau et le score du patient ne sont que faiblement corrélés et que les facteurs influant sur la recherche du lambeau sont à chercher du côté de la qualité de l'image ou du type de lambeau qui varie fortement selon les patients.

On remarquera toutefois qu'en dessous de 2000 pixels on a toujours un score de Dice inférieur à *0.5* mais qu'à une exception près, le score de Dice des lambeaux de plus de 3000 pixels est toujours supérieur à 0.5.

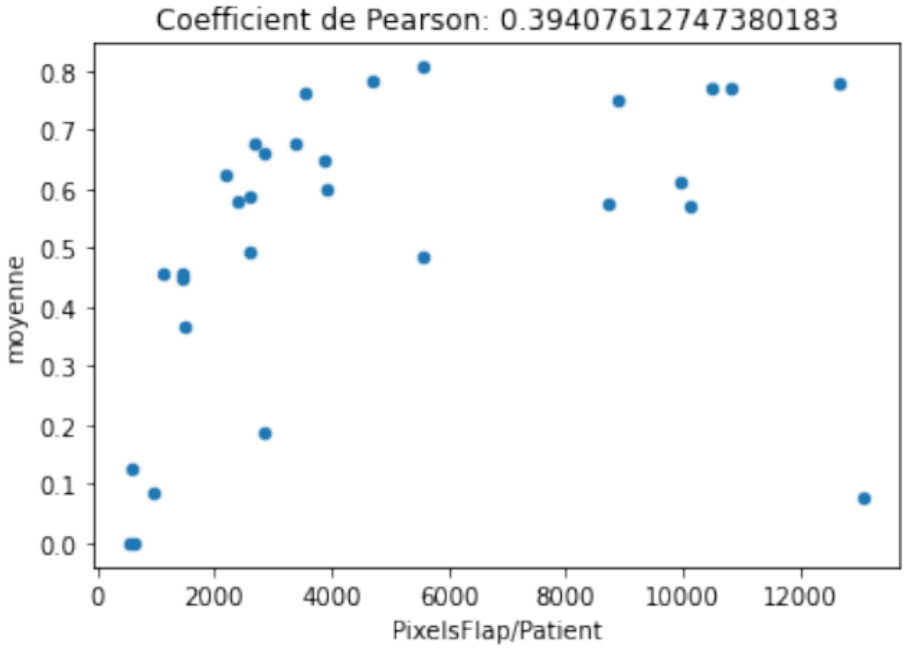


FIGURE 56 – Score moyen de Dice en fonction de la taille du lambeau (en pixels)

5 Conclusion et Perspectives

5.1 Conclusion

Les premiers mois du stage ont été consacrés à la prise en main et la manipulation de la base de données clinique. Il nous a donc fallut nous familiariser avec l'imagerie médicale d'une part mais aussi à sa terminologie.

Le fil rouge du projet a réellement été d'allier les technologies de deep learning tout en bénéficiant des informations propres au domaine médical et radiothérapeutique que ce soit pour augmenter la base en suivant un réalisme clinique, utiliser les valeurs de Hounsfield des pixels puis utiliser le score de Dice comme garant de la qualité de nos prédictions.

Tout cela a été possible grâce à l'interdisciplinarité entre informaticiens et radiothérapeutes qui a été très enrichissante pour tous grâce à nos réunions hebdomadaires qui nous permettaient de faire le point d'une part sur l'avancement du travail tout en l'expliquant didactiquement à des non-initiés au deep learning et d'autre part en conservant les enjeux liés à la radiothérapie et en comprenant mieux ses termes et les conséquences qu'elle pouvait avoir sur les différentes structures anatomiques pour mieux les manipuler.

Par ailleurs, le fait que ce stage apporte une pierre à l'édifice d'un véritable programme de recherche et qui permettra à d'autres de poursuivre ces travaux fut très motivant.

En conclusion, ce mémoire relate mes travaux des six derniers mois au sein du laboratoire GREYC de Caen qui s'inscrivent dans le travail pionnier du **Flap RT - Programme**. Durant ce stage, nous avons développé un outil qui, à terme, aidera les radiothérapeutes pour le suivi de leurs patients à l'aide d'un outil de segmentation automatique des lambeaux. Pour cela, nous avons vu différentes approches basées sur la nature médicale du lambeau afin de le segmenter. Nous en avons déduit que la meilleure approche était la segmentation binaire (le lambeau contre le reste). Nous gardons comme modèle final un **U-Net** en 2-Dimensions conçu avec *Lightning-Flash* et utilisant un **ResNet152** comme encodeur.

Le score de Dice moyen produit par ce modèle en segmentation automatique est donc équivalent à celui du Dice interobservateurs obtenu en segmentation manuelle (Dice 0.7, beddok 2022).

Même si ce modèle n'est pas encore prêt pour un usage clinique, ce travail fait toutefois l'objet d'un article en cours de rédaction dans la revue *International Journal of radiation Biology and Physics* (Rang A, Impact Factor 7).

5.2 Perspectives

Pour la suite du projet, il nous reste encore plusieurs approches à étudier qui n'ont pas encore été vues faute de temps.

5.2.1 2D Augmentées

Comme vu en sous-section *Plans d'une image*, le lambeau peut être appréhendé sous trois angles différents. Parfois le radiothérapeute a du mal à distinguer le lambeau sur une coupe et peut donc se servir des deux autres pour le trouver. Nous voulons donc mettre en place une nouvelle approche similaire où le lambeau serait préservé sous les trois plans.

Le lambeau étant un volume, une autre approche serait de considérer non pas une seule coupe de lambeau mais un paquet de coupes contenant quelques coupes précédentes et successives.

5.2.2 3D

Le meilleur moyen de considérer le volume du lambeau reste d'utiliser directement un U-Net en 3-Dimensions dont le principe reste le même à part que les convolutions se font elles aussi en 3-Dimensions. Au lieu d'utiliser les coupes, on pourrait donc directement utiliser les matrices représentant les patients.

5.2.3 Utilisation du volume d'intérêt

Dans un deuxième temps, nous aurions pu tester un modèle modifié, élargissant la zone d'intérêt des coupes pour chaque patient en restreignant la recherche de lambeau aux images contenant le volume d'intérêt **PTV HR**, dont le nommage, comme pour le lambeau "flap", est également normalisé.

Sachant que le lambeau compense la perte de substance tumorale, le volume cible tumoral appelé **PTV BR** correspond au volume cible irradié à faible dose pour contrôler une faible probabilité de rechute [14]. Le plus large doit donc inclure le tissu de comblement, c'est à dire le lambeau. Ceci sera testé pour l'article en cours d'écriture.

5.2.4 Utilisation d'un outil d'optimisations des hyperparamètres

Dans le cadre du stage nous avons fixé des hyperparamètres pour qu'ils répondent à notre besoin mais il est possible de bien les optimiser à l'aide d'outils tels qu'*Optuna* [15].

5.2.5 Choix d'une meilleur fonction objectif

Dans le cadre de la segmentation en 2-Dimensions, nous sommes impérativement confrontés à des coupes avec peu de lambeau. Dans ce cadre, les métriques utilisées pour mesurer la qualité de la prédiction sont biaisées par la taille des structures pour une même forme à différentes échelles [16]. Il pourrait donc être intéressant de trouver une meilleure métrique ou d'améliorer le score de Dice afin qu'il soit plus robuste aux coupes contenant peu de pixels de lambeau.

14. maladie microscopique uniquement sans masse tumorale détectable clinique ou en imagerie
15. Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta,and Masanori Koyama. 2019. Optuna : A Next-generation Hyperparameter Optimization Framework. In KDD.
16. Annika Reinke.. (2021) Common Limitations of Image Processing Metrics : A Picture Story

5.2.6 Amélioration de la vérité terrain

Le radiothérapeute a de nombreux contours à tracer sur deux à trois cents coupes. Pour gagner du temps il les fait parfois grossièrement et peut aussi n'en faire que sur certaines coupes, son logiciel étant capable d'intrapoler les contours des coupes intermédiaires. On a donc une vérité terrain contenant la région d'intérêt et un peu du tissu l'entourant. Ainsi notre fonction objectif visant à prédire une région d'intérêt tendrait à reproduire ce contour grossier en convergeant vers le score optimal. Paradoxalement, en utilisant le score de Dice par exemple, un score légèrement inférieur à un pourrait être de meilleure qualité qu'une prédiction avec un score égal à un. Plutôt que de se compliquer à améliorer notre métrique pour prendre en compte ce biais, il pourrait être plus simple d'améliorer la vérité terrain du radiothérapeute.

Puisque l'on part d'un contour, les méthodes les plus pertinentes pour améliorer la vérité terrain sont les méthodes par contour actif. A modérer toutefois car dans le cas des lambeau la frontière n'est pas toujours visible entre tissus d'origine et tissus greffé, il est donc possible qu'un algorithme de contour actif englobe du tissu de lambeau dans le tissu d'origine.

Cette piste d'amélioration est corroborée dans l'état de l'art [17] qui confirme que les contours actifs améliorent la segmentation initiale.

17. Sheela, C. & Suganthi, G.. (2020). Brain tumor segmentation with radius contraction and expansion based initial contour detection for active contour model. Multimedia Tools and Applications. 79. 10.1007/s11042-020-09006-1.

Références

[1] https ://towardsdatascience.com/understanding-dicoms-835cd2e57d0b

[2] https ://fr.wikipedia.org/wiki/%C3%89chelle_de_Hounsfield

[3] https ://www.kaggle.com/dcstang/see-like-a-radiologist-with-systematic-windowing

[4] https ://pydicom.github.io/pydicom/stable/old/pydicom_user_guide.html

[5] https ://www.sciencedirect.com/science/article/abs/pii/S1879850021000485

[6] https ://www.phys4med.be/construction/fenetrage

[7] https ://www.researchgate.net/publication/306033192_Automatic _detection_of_ the_pulmonary_nodules_from_CT_images

[8] https ://www.unil.ch/issul/files/live/sites/issul/files/shared/HMB-EM.2019.pdf

[9] https ://arxiv.org/abs/1505.04597

[10] https ://blent.ai/unet-computer-vision/

[11] https ://ichi.pro/fr/creation-d-un-modele-u-net-tres-simple -avec-pytorch -pour-la-segmentation-semantique-des-images-satellites-37635222857477

[12] https ://www.researchgate.net/figure/U-net-architecture-with-ResNet34-blocks-in-the-down-sampling-path_fig2_338723610

[13] https ://stanford.edu/ shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks

[14] Optimizing the Dice Score and Jaccard Index for Medical Image Segmentation : Theory & Practice, Jeroen Bertels and Tom Eelbode and Maxim Berman and Dirk Vandermeulen and Frederik Maes and Raf Bisschops and Matthew B. Blaschko, 2019

[15] https ://larevueia.fr/3-methodes-pour-optimiser-les-hyperparametres -de-vos-modeles-de-machine-learning/ bibitem https ://wandb.ai/wandb_fc/french/reports/Quelle-est-la-Batch-Size-Optimale-pour-Entra-ner-un-Neural-Network—Vmlldzo1NzkyMzc

[16] https ://stanford.edu/ shervine/teaching/cs-230/cheatsheet-deep-learning-tips-and-tricks

[17] https ://deeplylearning.fr/cours-theoriques-deep-learning/transfer-learning/

[18] https ://towardsdatascience.com/the-w3h-of-alexnet-vggnet-resnet-and-inception-7baaaecccc96

[19] Annika Reinke.. (2021) Common Limitations of Image Processing Metrics : A Picture Story

[20] An Active Contour Model without Edges, Tony Chan and Luminita Vese, Scale-Space Theories in Computer Vision, 1999

Annexe

Processus d'annotation

Trois radiothérapeutes (>10 ans de radiothérapie ORL) ont été sollicités pour assurance qualité de la radiothérapie et détection de présence (ou absence) de lambeau puis un expert a segmenté manuellement ce lambeau s'il était présent.Etapes de la démarche de segmentation manuelle :

1. Fenêtrage tissus mous : 45-250 UH
2. Recherche de graisse et rupture de symétrie droite gauche dans le plan transversal (ou si non vu, sagittal ou si non vu dans le plan coronal)
3. Recherche de transitions tissulaires (flap, gris de la périphérie du lambeau a centre noir / transition/jonction de 1-5mm noire comme graisse, tissus natifs gris ou le cas échéant, blanc si contact os ou métal)
4. Doute sur jonction, jonction confirmée par présence de clips (mais attention, peuvent être dans la lambeau ou en dehors pour coaguler des saignements // attention clips différents selon centres et parfois plusieurs types par patient)
5. Contourage en " brosse"
6. Non contourage sur coupes très artéfactées en axial, recherche en sagittal ou si non vu dans le plan coronal)
7. Si suffisamment de coupes de part et d'autre de la zone très artéfactée, interpolation (logiciel de contourage constant sur cette BD) dans le plan de contourage
8. Pas de lissage automatique mais écrêtage des pics de contours aberrants (l'écrêtage peut induire des pics dans les autres plans ; impact de l'épaisseur des coupes)
9. Recherche d'asymétrie de forme de la mandibule ou d'os du massif facial ou d'artefacts en regard des os ; si absente, stop
10. Si lambeau avec composantes osseuses, fenêtrage -120 a -30 UH
11. 1-6 ostéotomies (segments osseux) géométriques segmentées en lignes (spline), plaques de fixation exclues