# Towards lightweight medical image segmentation deep models

Encadrant : Pierre-Henri Conze

Pierre ROUYER
Nampoina RAVELOMANANA
Ilias SABIR
Rémi CHEN
Killian ASSAMEUR

## Introduction

Les modèles de segmentation d'image médicale demandent des quantités de ressources calculatoires conséquentes. Cependant, le déploiement de modèles lourds peut être freiné par les capacités techniques des hôpitaux ou ne pas répondre aux réels besoins d'utilisation. Par exemple, la segmentation en direct d'une opération assistée par ordinateur doit se faire de manière quasi-immédiate, de gros modèles sont donc un frein à ce genre de cas d'utilisation. On peut aussi exposer la limitation imposée par un système embarqué pour justifier l'importance de créer des modèles plus compacts. Cependant ces modèles compacts doivent impérativement fournir des résultats aussi bons, ou très légèrement inférieurs aux modèles lourds.

Cependant, de manière générale, alléger un modèle a souvent un impact négatif sur les performances et limite donc son utilisation.

L'objectif de ce projet est d'implémenter différentes techniques (Knowledge Distillation, Dilated Multi-Fiber Network) permettant de booster les résultats de modèles léger pour qu'ils soient comparables à ceux d'un modèle lourds et ainsi permettre l'utilisation de ces algorithmes en direct ou sur des machines moins puissantes.

## 1. Implemented methodologies

### A) Implémentation de U-NET 3D

L'implémentation de U-Net 3D pour la segmentation d'images médicales peut être réalisée même avec des ressources limitées en suivant les étapes suivantes :

1. Acquisition et préparation des données : Les images médicales nécessaires pour l'entraînement et la validation du modèle doivent être acquises et préparées. Les images doivent être prétraitées pour réduire le bruit et augmenter le contraste afin d'améliorer la qualité des images.

2. Création de l'architecture du modèle : L'architecture U-Net 3D doit être créée en utilisant un framework de deep learning tel que TensorFlow ou PyTorch. Les couches de convolution, de pooling et de déconvolution sont utilisées pour construire l'architecture.
3. Préparation des données d'entraînement : Les images médicales doivent être divisées en ensembles d'entraînement, de validation et de test. Les images doivent être normalisées grâce au jeu d'entraînement.
4. Entraînement du modèle : Le modèle doit être entraîné en utilisant l'ensemble d'entraînement. La fonction de perte doit être choisie avec soin, en fonction de l'application.
5. Validation du modèle : Le modèle doit être validé en utilisant l'ensemble de validation. Les résultats doivent être analysés pour déterminer si le modèle fonctionne correctement.
6. Test du modèle : Le modèle doit être testé en utilisant l'ensemble de test. Les résultats doivent être comparés à la vérité terrain (ground truth) pour déterminer la précision du modèle.
7. Analyse des résultats : Les résultats doivent être analysés pour déterminer les performances du modèle. Des mesures telles que la précision, la sensibilité et le DICE SCORE peuvent être utilisées pour évaluer le modèle.

En résumé, pour implémenter U-Net 3D pour la segmentation d'images médicales avec des ressources limitées, il faut acquérir et préparer les données, créer l'architecture du modèle, préparer les données d'entraînement, entraîner et valider le modèle, tester le modèle et enfin analyser les résultats.

## B) U-NET 3D  U-NET 2D

Le choix de passer d'UNET3D à UNET2D est justifié par plusieurs raisons. Tout d'abord, UNET3D est conçu pour traiter des images volumétriques 3D, ce qui nécessite une quantité de mémoire importante pour stocker et manipuler ces données. Or, cette quantité de mémoire peut rapidement devenir un problème si nous traitons des images de grande taille. En outre, notre équipe a rencontré des problèmes de type "out of memory" lors de l'exécution de UNET3D sur certaines images. En passant à UNET2D, nous travaillons avec des images planes 2D, ce qui permet de réduire la quantité de données à manipuler et donc d'alléger considérablement le modèle. Cette solution permettra également d'améliorer les temps de traitement et la rapidité d'exécution de notre modèle de segmentation.

## C) Implémentation de la Knowledge Distillation

Afin de permettre à un réseau allégé que l'on nommera par la suite "réseau student" d'avoir d'aussi bonnes performances qu'un réseau lourd que nous appellerons "réseau teacher", nous utiliserons le concept de *knowledge distillation*. Cela consiste en un transfert de connaissance d'un réseau puissant mais encombrant vers un modèle allégé pour améliorer les performances de ce dernier sans nuire à son efficacité.

Il existe plusieurs moyens d'effectuer ce transfert de connaissances. Nous avons implémenté et combiné deux de des processus qui étaient présentés dans l'article "Efficient Medical Image Segmentation Based on Knowledge Distillation" écrit par D.Qin, J.Bu, Z.Liu, X.Shen, S.Zhou, J.Gu, Z.Wang, L.Wu et H.Dai:

- Prediction Maps Distillation
- Importance Map Distillation

**Prediction Maps Distillation**

L'approche la plus basique de la distillation de connaissances consiste à influencer le réseau student, de telle sorte à ce que la sortie du réseau student soit similaire à la sortie du réseau teacher. Pour cela, il suffit d'introduire dans la fonction de coût du réseau student, une fonction qui calcule la différence entre les prédictions des deux réseaux. Dans notre cas, nous avons une tâche de segmentation, et nous avons ainsi à la sortie une distribution de probabilité de chaque pixel d'appartenir à la classe considérée (ici le foie). Nous pouvons ainsi introduire dans la fonction de coût, la fonction de divergence de Kullback-Leiber:

$$L_{PM} = \frac{1}{N} \sum_{i \in \mathbf{N}} KL(p_i^s || p_i^t)$$

où N=W × H est le nombre de pixels de l'image, KL(.) est la fonction de Kullback-Leiber et $p_i^s$ et $p_i^t$ représentent les probabilités du ième pixel d'appartenir à une classe, selon le réseau student et le réseau teacher respectivement.

Pour implémenter cela, il nous a suffi d'importer la fonction *KLDivLoss* du module torch.nn.

**Importance Maps Distillation**

En plus d'utiliser les couches finales des deux modèles pour faire la distillation de connaissances, il est aussi possible d'utiliser les "features maps", c'est-à-dire les représentations intermédiaires extraites de l'image.

Dans notre cas, nous allons nous baser sur l'article "Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. " écrit par S.Zagoruyko et N.Komodakis. Elle se base sur l'intuition que la valeur d'activation d'un neurone d'une certaine couche indique son importance, dans le processus de segmentation. Nous allons ainsi utiliser dans la fonction de coût du réseau student, une fonction de différence entre les différentes sorties des couches d'activation. Ces dernières pouvant être de taille différente entre le réseau student et le réseau parent, nous n'allons prendre que les couches d'activation en sortie de chaque bloc du UNet (un bloc étant constitué d'une couche de convolution, normalisation, activation relu, convolution, normalisation, maxpooling dans la partie "encoder" du UNet, et de convolution transposée, convolution, normalisation, activation relu, convolution, normalisation dans la partie "decoder").

$$\mathcal{L}_{AT} = \mathcal{L}(\mathbf{W}_S, x) + \frac{\beta}{2} \sum_{j \in \mathcal{I}} \| \frac{Q_S^j}{\|Q_S^j\|_2} - \frac{Q_T^j}{\|Q_T^j\|_2} \|_p$$

où L(.,.) est la fonction de coût basique du réseau student

et $Q_S^j = vec(F(A_S^j))$ et $Q_T^j = vec(F(A_T^j))$

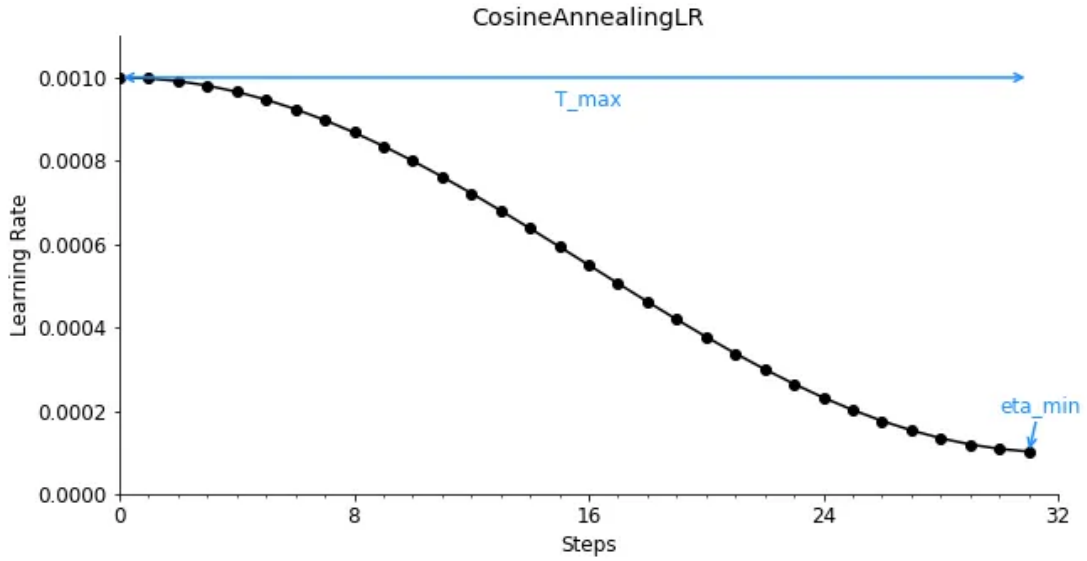
où $\mathcal{F} : R^{C \times H \times W} \rightarrow R^{H \times W}$

avec H,W les dimensions de l'image et C le nombre de canaux et $A_s^j$ et $A_t^j$ les "feature maps" du réseau student et du réseau teacher respectivement.

Nous avons choisi comme fonction $F^p_{\text{sum}}(A) = \sum_{i=1}^{C} |A_i|^p$

## D) Spécificités d'implémentation

Afin d'améliorer nos résultats, nous avons tout d'abord utilisé un taux d'apprentissage dynamique. Pour cela, nous avons utilisé CosineAnnealingLR du module torch.optim.lr_scheduler. Cet outil permet de réduire le taux d'apprentissage selon le nombre d'epochs. L'optimizer choisi est Adam et notre batch size est de 16.




Ensuite, nous avons augmenté dynamiquement notre base de données. C'est-à-dire que durant chaque epoch, pour chaque batch, nous allons appliquer des modifications (flips, crops) à certaines image aléatoirement choisies, et les utiliser comme données d'entraînement. Ensuite, nous passerons au batch suivant.

## E) Implémentation du Dilated Multi-Fiber Network (Explication de l'échec)

Nous avons tenté de convertir un réseau dilaté multi-fibre pour la segmentation d'image 3D, mais cela n'a pas fonctionné comme prévu. Nous avons adapté l'architecture du modèle pour qu'il puisse traiter des images 2D en ajustant les paramètres et en enlevant une dimension.Ensuite, nous avons entraîné le modèle en utilisant les images d'entraînement comme pour le Unet2D. Cependant, nous n'avons pas constaté de changement significatif de la validation loss lors de l'évaluation de performance. Après avoir examiné plusieurs hypothèses, nous avons conclu que le modèle n'était pas adapté à la tâche de segmentation 2D probablement due à la dimension du kernel qui est de un (un

vecteur). Malgré nos efforts pour le modifier, notre réseau dilaté multi-fibre n'était tout simplement pas adapté pour traiter les images de manière optimale dans le contexte d'image en 2 dimensions.

## Performed experiments

Nous avons utilisé comme réseau parent, un modèle UNet avec les nombres de filtres suivants:

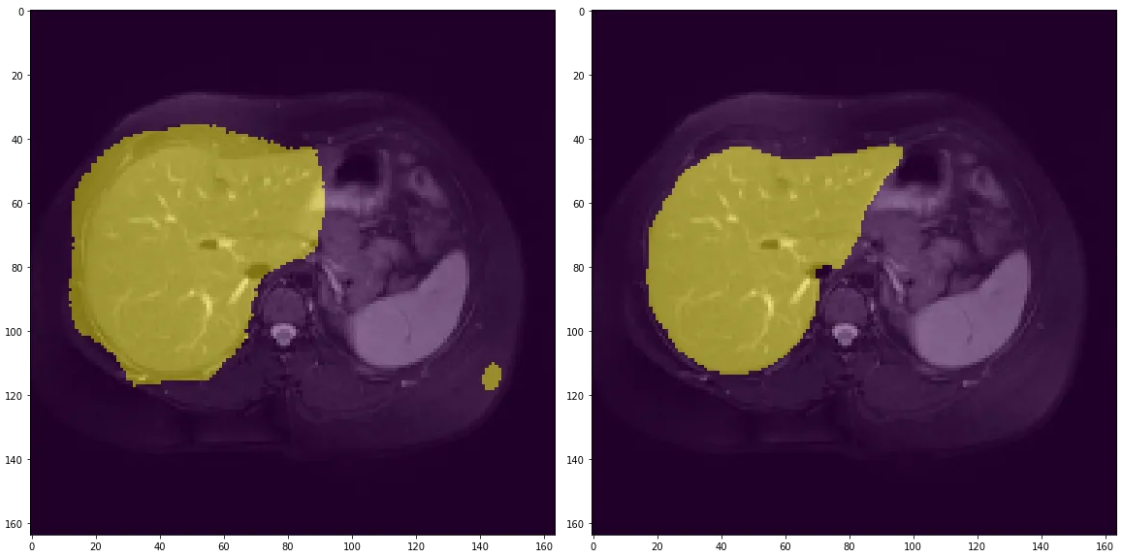
- 64, 128, 256, 512 dans la partie "encoder"
- 256, 128,64,1 dans la partie "decoder"

Le réseau parent a 7,696,193 de paramètres.

Ensuite, nous avons testé les réseaux student suivants en faisant varier leur architecture et en comparant le Dice score du student normal (sans distillation de connaissance) et le Dice score du student distillé (avec distillation de connaissance).

Les réseaux ont été évalués sur une base de données de test de 125 images, sur lesquelles nous calculerons le Dice score moyen.

Avec le réseau parent, nous avons obtenu un Dice score de **0.7408** et les résultats visuels suivants:



(a) Prédiction du réseau teacher (b) Vérité terrain

| Architecture utilisée | Nombre de paramètres | Dice score du student normal | Dice score du student distillé |
|---|---|---|---|
| 32,64,128,256<br>128, 64,32,1 | 1,925,025 | 0.6931 | 0.7345 |
| 32,64,128<br>64,32,1 | 465,953 | 0.5647 | 0.6487 |
| 32,64<br>32,1 | 100,961 | 0.3767 | 0.4708 |

Nous pouvons remarquer une amélioration des performances entre le student non distillé et le student distillé d'au moins 5%. Cela correspond plus ou moins aux résultats énoncés dans l'article.

Il est aussi à préciser qu'ici, nous nous sommes contentés d'entraîner tous les modèles sur 20 epochs (et de sauvegarder le meilleur modèle durant ces 20 epochs, afin d'éviter le surapprentissage) afin de les comparer en utilisant le même nombre d'epochs. Ainsi, certains modèles peuvent souffrir de sous-apprentissage (notamment les réseaux student distillés).

## Conclusion et Perspectives

Pour conclure, ce projet nous a permis de modeler un modèle de segmentation d'image répondant au mieux aux attentes. Ainsi, nous avons eu besoin de nos compétences acquises en Deep Learning pour créer ce modèle mais la plus tâche a d'abord été de nous former à de nouveaux concepts notamment le Knowledge Distillation et le Dilated Multi-Fiber Network. Malgré nos ressources limitées, nous avons réussi à nous adapter et à reformuler notre modèle. Cependant, ces contraintes nous ont aussi parfois mis au pied du mur. En effet, concernant le Dilated Multi-Fiber Network, nous n'avons à ce jour pas obtenu de résultats satisfaisants. En échangeant avec certains intervenants, nous nous sommes rendus compte que cette technologie n'était tout simplement pas adaptable à un modèle U-NET 2D.

Dans une optique d'amélioration de ce projet, nous pensons que la première étape indispensable serait d'accéder à des ressources calculatoires supérieures. Ainsi, il serait possible de créer un modèle U-NET 3D qui nous permettrait d'une part d'obtenir de meilleurs résultats mais aussi d'implémenter le Dilated Multi-Fiber Network. Il pourrait être aussi intéressant d'élargir les champs d'observation et étudier d'autres choses que la segmentation d'organes. Ce modèle allégé pourrait par exemple être utile pour la détection de tumeurs ou même dans beaucoup d'autres domaines pas forcément en lien avec le médical. Si nous poussons l'idée encore plus loin alors chacun pourrait bientôt disposer d'algorithmes de deep learning sur son ordinateur personnel sans passer par des outils comme Google Colab.