PFE - Deep Learning Computed Tomography

Benjamin CASTET - Nicolas DECAM

March 2021

1 Abstract

Ce projet a pour but l'expérimentation d'approche de deep learning pour la reconstruction de volume 3D de obtenue à l'aides de radiographie (tomographie). L'objectif principal est d'atteindre une qualité proche de celle obtenue à l'aide de méthode itérative très coûteuse mais dans un temps bien plus raisonnable afin, afin de pouvoir segmenter les images pour reconstituer des modèles 3D des neurones.

2 Introduction

2.1 Contexte

Les techniques de tomodensitométrie 3D sont beaucoup utilisées dans le domaine de l'imagerie médical et industriel.
La reconstruction 3D des parties intérieures d'un modèle acquis est obtenu à l'aide d'une série de projections mesurées autour du modèle. C'est par exemple le fonctionnement du scanner à rayon-X médical.

Il existe deux algorithmes de reconstruction couramment utilisé en tomodensitométrie 3D : le premier est FBP (filtered back-projection) qui est une implémentation directe de la transformée inverse de Radon. Cette algorithme utilise les projections pour récupérer les volumes 3D du modèle acquis. Cette reconstruction directe est particulièrement utilisée dans les application de routine en raison de son coût de calcul plus faible. Cependant, il n'est pas adapté à l'acquisition de géométrie complexe. Il est en effet très affecté par le bruit et nécessite un grand nombre de projection pour être précis.

L'autre méthode couramment utilisée est la reconstruction itérative qui n'a pas les mêmes limitation lié au bruit et au grand nombre de projection nécessaire par rapport à FBP. En revanche, elle ne peut pas être utilisée dans les applications de routine car son temps de calcul peut être jusqu'à 100 fois supérieur à celui de FBP. Ce désavantage devient de plus en plus problématique en raison de l'explosion de la résolution d'acquisition, qui a pour effet indirect d'augmenter fortement le coût de calcul de cette méthode.

L'émergence récente du Deep learning rends la littérature dans le domaine de la reconstruction tomographique très épars. Ainsi, la variété de recherche faites dans ce domaine montre que le Deep learning appliqué à la tomographie reste très expérimental et qu'aucune méthode ne fait consensus. Enfin, la plupart des travaux réalisé jusqu'à maintenant sont principalement conceptuels ou s'intéressent à un problème très spécifique tel que la correction d'angle manquant. Il n'existe d'après nos recherche aucune recherche aboutie pour ouvrir Deep learning à une application de routine pour faire de la tomographie.

2.2 Problématique

Notre Problématique est donc la suivante: nous devons, à l'aide du Deep learning, trouver une méthode permettant, avec le moins de projection possible, de reconstruire une image d'une qualité suffisante pour pour pouvoir être utilisée par des algorithmes de segmentation par la suite, et ce dans un temps raisonnable permettant de faire plusieurs reconstructions en une journée.

2.3 Sujet

Les objectifs de ce projet sont :

- Réduire au maximum le temps de calcul grâce au Deep Learning pour pouvoir traiter plusieurs jeu de donnés massif par jour tout en ayant une qualité d'image reconstruite équivalente au technique de reconstruction itérative
- Optimiser la reconstruction Deep learning en sachant que le type d'échantillon va rester globalement le même.
- Trouver une ou des méthodes assez générale pour être appliqué à une grande variété de tâches dès qu'un set d'entraînement adapté composé d'acquisition et des reconstruction associé est fourni.

3 État de l'art / existant

Lors de nos recherche nous avons exploré plusieurs types de méthode pour arriver à notre objectif, on en a retenu trois possibilités :

- Les méthodes de pré-traitement qui vont modifier les données en entrée avant la phase de reconstruction, pour ce faire on peut par exemple créer artificiellement des projections supplémentaire (voir : [1])
- Les méthodes de post-traitement qui vont modifier les images reconstruites pour qu'elles correspondent à nos critère, pour ce faire on peut appliquer des méthode de denoising qui vont enlever le bruit de l'image (voir [3])
- Les méthodes end-to-end qui vont prendre en entrée les sinogrammes et ressortir les images reconstruite sans nécessité de pré ou post traitement(voir : [3, 4, 6, 2, 8, 5])

Pour mieux représenter nos recherche et les différentes méthodes illustrées dans les articles, nous avons fait un tableau comparatif visible ci dessous. Il permet de retracer facilement les avantages et les inconvénients des méthodes de reconstructions présentées dans plusieurs articles.

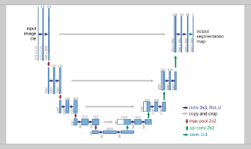
	CNN-Based Projected Gradient Descent for Consistent CT Image Reconstruction	Deep Learning Based Computed Tomography Whys and Wherefores	Deep Learning-Guided Image Reconstruction from Incomplete Data
le contexte / la problématique	diminuer le nombre de projection tout en ajoutant du controle et de la fiabilité sur les résultat obtenu par NN		reconstruire une image à partir de donnée incomplète
la méthodologie / le type de NN utilisé	basé sur U-net		
schéma			
si le contexte / pb n'est pas directement lié à nos pb, est-ce qu'il y a matière - à votre avis - d'exploiter quand même une solution de ce type (et le cas échéant : pourquoi)			
les résultats qu'ils ont obtenus	résultat positif meilleur que les autres solutions (FBP, DL, TV) dans la plupars des cas	dans les cas "low-dose" les solution end-to-end ne semble pas intéressante le image->image semble à privilégier	methode de reconstruction d'image avec un rendu de bonne qualité
les limitations qu'ils ont soulevés sur leur résultats actuels			
ou est-ce qu'on se situe au niveau des data processing : (sinogramme -> sinogramme , image -> image, "end-to-end")	image to image		end to end
la nature / configuration / caractéristiques des training sets	1) 500 clinically realistic, (512×512) CT images from the lower lungs to the lower abdomen of 10 patients. Those were obtained from the Mayo clinic AAPM Low Dose CT Grand Challenge. 2) We use a real (1493 px × 720 view × 377 slice) sinogram from a CT scan of a single rat brain		size 256 × 256, with 7500 training images, 1000 validation images, and 500 testing images.
toute autre information qui vous parrait pertinente	un de nos gros problèmes est le temps et même si la qualité est bonne je ne sais pas combien de temps est nécessaire mais étant donné qu'il le compare à FBP sans précision nous supposons qu'il s'effectue dans le même temps		méthode itérative donc long temps de calcul (??)

Figure 1: tableau comparatif (partie 1)

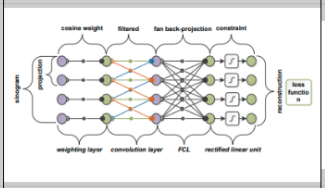
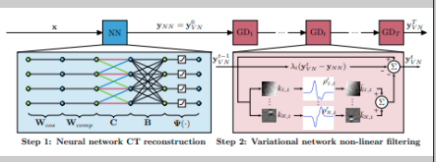
Deep Learning-Guided Image Reconstruction from Incomplete Data	Deep Learning in computed tomography	A Deep Learning Architecture for Limited-Angle Computed Tomography Reconstruction
reconstruire une image à partir de donnée incomplète	Deep learning techniques to replace heuristic compensation steps in CT reconstruction	reconstruire des image malgrés des angles de vue manquant
	Fan-beam neural network architecture	
		
		nous n'avons pas de problème de limited angle
methode de reconstruction d'image avec un rendu de bonne qualité	Evaluations of the reconstruction results show improved image quality compared to FBP while still retaining the same computational demands	outperforms all methods qualitatively and quantitatively. The best results were achieved for a filter kernel size of 13.
end to end	end to end	
size 256 × 256, with 7500 training images, 1000 validation images, and 500 testing images.	size 512 × 512 ten-fold cross validation on 2378 slices of ten different patients	
méthode itérative donc long temps de calcul (??)		

Figure 2: tableau comparatif (partie 2)

Un type d'architecture de réseau de neurone revenait assez fréquemment, le U-net (voir [7]) qui semblait bien correspondre à nos besoin. Cette architecture était notamment utilisé dans [3] dont la problématique peut bien être appliqué à notre problème et dont les résultat présenter dans l'article nous semble très convainquant .

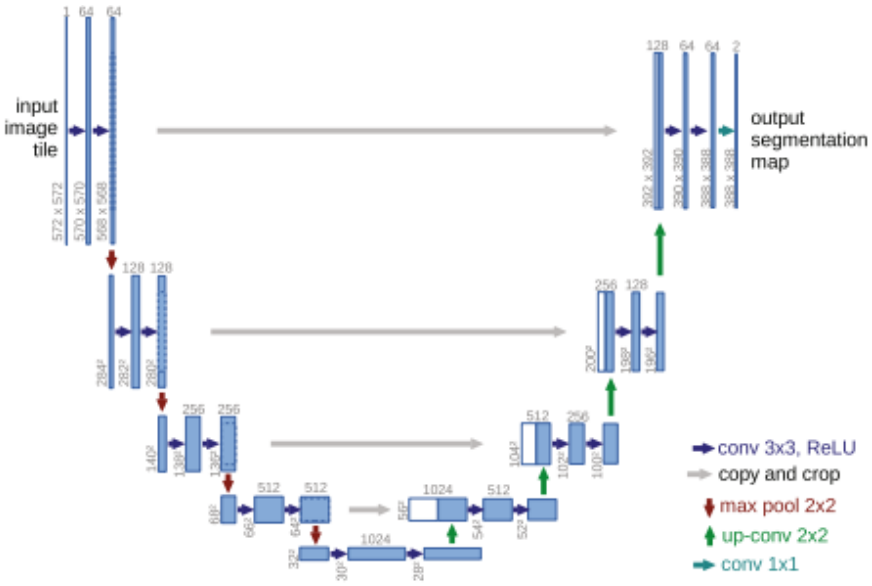


Figure 3: architecture U-net

4 User stories

Les Users stories sont les suivantes :

- Pouvoir traiter un large volume de donnés.
- Avoir une qualité d'image en sortie similaire à l'algorithme de récursion itérative avec 600 projections.
- Plusieurs sets de données doivent pouvoir être traités chaque jour.
- Optimiser en sachant que le types de données en entrée doit rester globalement le même.
- Pouvoir reconstruire à partir du moins de projections possible. En effet, à chaque projection des rayons X sont envoyé sur l'échantillon, ce qui endommage les tissus et peut causer des modifications au cours de l'acquisition si elle est trop longue, ou bien empêcher de réutiliser l'échantillon pour d'autres test.

5 Besoins

5.1 Fonctionnels

Le besoin fonctionnel consistait à mettre au point une méthode pour obtenir des images reconstruite à partir d'une acquisition CT-scan. Les méthodes actuelles n'étant pas satisfaisante, l'objectif était donc d'implémenter un réseau de neurones pour accélérer la reconstruction.

5.2 Non-fonctionnels

Les besoins non-fonctionnel attendus pour le rendu final sont :

- Un temps de calcul permettant de d'effectuer plusieurs reconstruction par jour. En effet le but est que ce système de reconstruction fasse partie d'application de routine. Il faut donc que le temps de reconstruction pour un volume soit de l'ordre de quelques heures au maximum.
- Les images reconstruites doivent pouvoir être utilisées par des algorithmes de segmentation

6 Choix logiciels

6.1 langage

Nous avons choisi de partir sur le langage Python, standard dans l'utilisation d'algorithmes de Deep Learning. Certains articles que nous avons consultés utilisaient également Matlab. Ayant déjà de l'expérience sur Python et PyTorch pour les algorithmes de Deep Learning, et l'existant ainsi que les bibliothèques étant largement plus développé et documenté, nous avons choisi de nous orienter sur ce dernier.

6.2 Bibliothèques

Comme pour la plupart des travaux effectués en Deep Learning aujourd'hui, nous avons décidé d'utiliser PyTorch, développé par Facebook, qui permet facilement de manipuler de nombreux paramètres d'entraînement tout en restant facilement lisible.

Pour visualiser les résultats, nous nous sommes naturellement orientés vers Matplotlib.

7 Architecture

L'architecture est extrêmement simple, on a simplement trois manière de lancer le script en entraînement, en test ou en reconstruction. Chaque méthode lance un script qui effectue la tâche prévue, affiche les résultats et les enregistre. Pour modifier les paramètres, on modifie des fichiers de config.

8 Gantt

Pour nos prévisions avec la réalité, tout s'est déroulé comme prévu jusqu'au moment nous avons dû faire fonctionner le code avec notre dataset. Nous devions utiliser les données de nos clients afin de tester le modèle, malheureusement le temps de conversion et volume des donnés (plus d'1 To) nous à fait prendre du retard. La suite de complications explicitée dans Difficultés nous as fait prendre un retard assez considérable, sans laisser de réelles opportunités de corriger le tir. Il ne nous restait donc qu'une semaine pour tenter une nouvelle solution et une semaine pour écrire le rapport et préparer la soutenance.

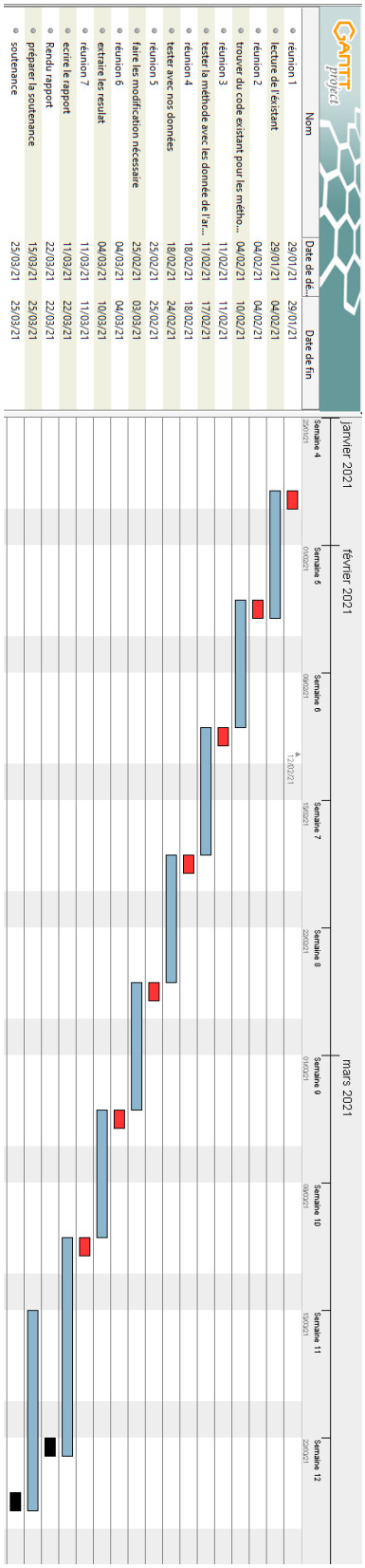


Figure 4: Gantt prévisionnel

9 Réalisation

9.1 Algorithmes/méthodes/données

Après notre état de l'art, il est apparu que la méthode la plus commune était de reconstruire une image en haute qualité à partir d'une image en basse qualité, reconstruite avec peu de projections. Pour cela, la méthode la plus commune est d'utiliser un réseau de neurone type U-net, et différents travaux similaires au notre ont déjà été effectués.

Les données ont étées fournies par le client. Il s'agit de CT-scans de cerveaux, 2159 slices de 2560x2560 pixels. Nous avons subdivisé chaque slice en 64 patchs pour des raisons d'espace mémoire vidéo, puis quelque essais plus tard supprimé les patchs en bordure d'image.

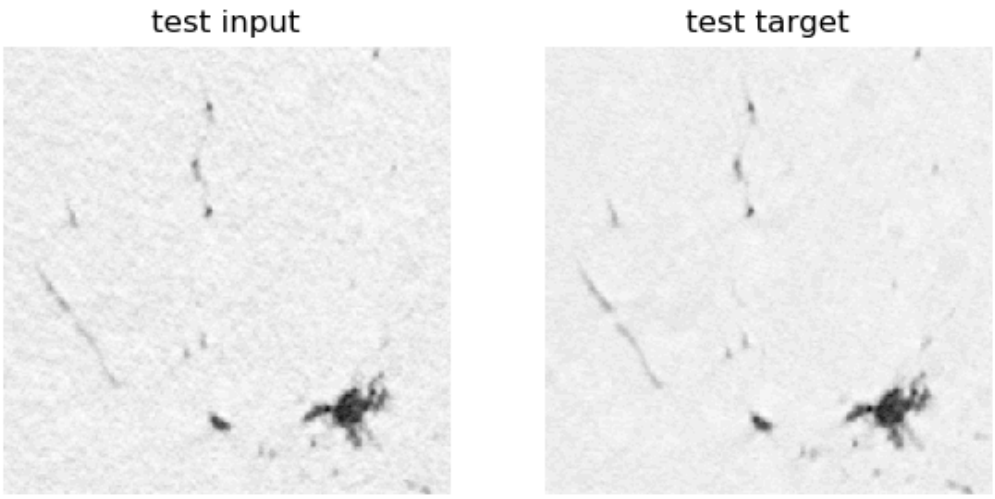


Figure 5: Données en entrée

Nous avons donc repris le code proposé ici : https://github.com/PhanHuyThong/Image-Reconstruction-by-CNN-based-PGD . Il s'agit d'un U-net basique, avec une double convolution en entrée, deux down, deux up et une convolution en sortie. L'entraînement en revanche est un peu plus atypique :

```
train1.cfg          : train the CNN with loss = criterion(output1, target)
train2.cfg          : train the CNN with loss = (criterion(output1, target) + criterion(output2, target))/2
train3.cfg          : train the CNN with loss = (criterion(output1, target) +
                                                 criterion(output2, target) +
                                                 criterion(output3, target))/3
              where output1 = model(inp)
                    output2 = model(output1)
                    output3 = model(target)


train_projector.cfg : train a projector by going through all 3 options above sequentially.
```

Figure 6: Entraînement

Nous avons donc utilisé cette méthode directement implémentée pour modifier les paramètres de notre réseau.

Afin d'avoir une meilleure variété dans les images utilisées en entraînement, nous avons également modifié la classe mydataset proposée sur le github. En effet, utiliser les 500000 images produites était complètement irréaliste et contre productif. Nous avons donc modifié cette classe afin d'équilibrer les échantillons de manière uniforme. Cette modification permettait également d'éviter d'utiliser les images en bordure sans avoir à reconvertir tout le dataset.

9.2 Résultats

Avec cette méthode nous avons pu obtenir les résultats suivants :

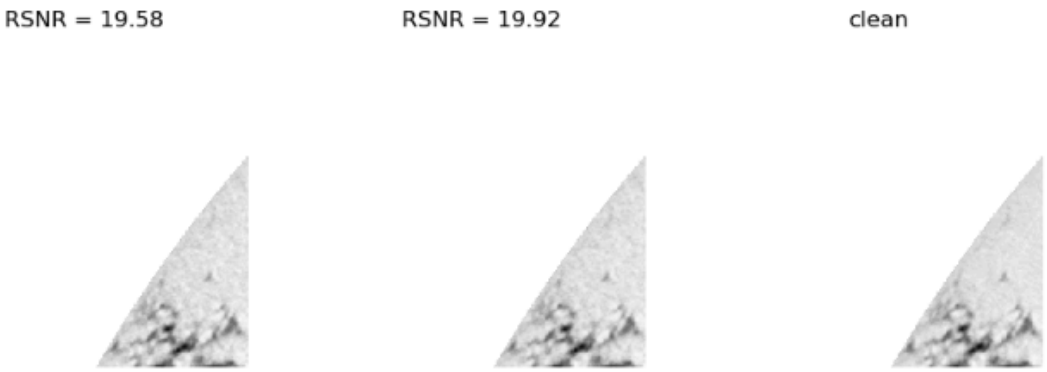


Figure 7: Résultats 1er réseau

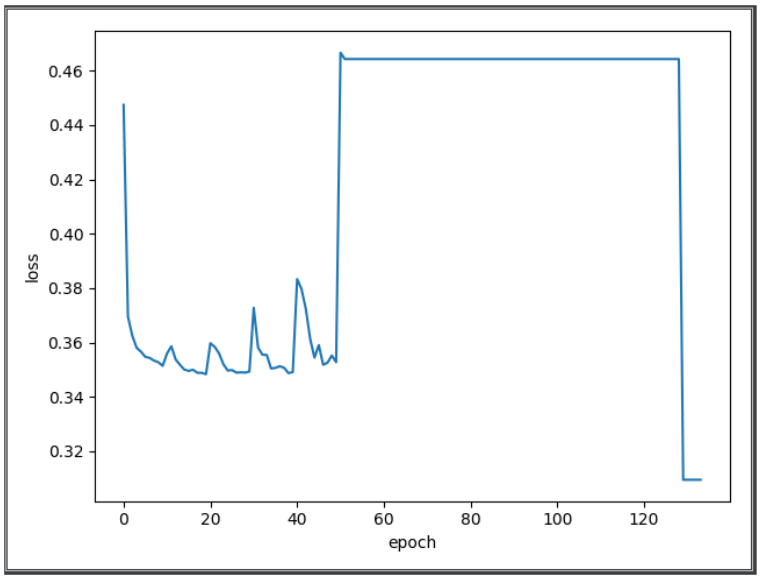


Figure 8: Loss 1er réseau

On remarque assez rapidement que l'évolution de la loss n'est pas normale, et pour cause : le réseau a simplement appris à reproduire l'image d'entrée (ce qui n'est évidemment pas le résultat souhaité). Malheuresement, ce comportement restera similaire quel que soit les paramètres ou le réseau utilisé (nombre d'images considérées, nombre d'epochs...).

Nous avons essayé de changer de réseau utilisé, pour passer sur https://github.com/mateuszbuda /brain-segmentation-pytorch qui est directement importable dans PyTorch. Celui ci calcule une segmentation map, ce qui est cohérent avec nos besoin bien que différent dans le fonctionnement. Cependant, le résultat est assez décevant comme on peut le voir ci-dessous.

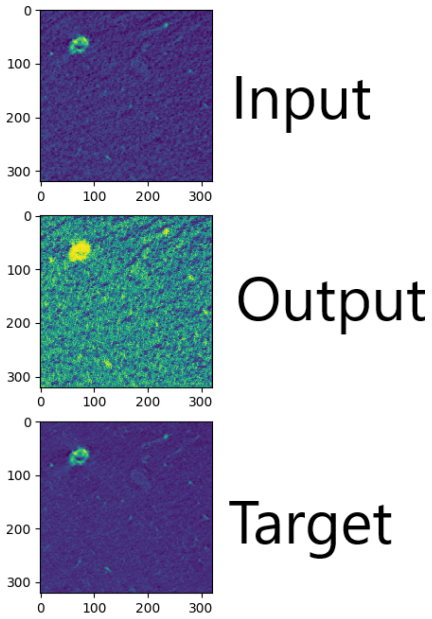


Figure 9: Résultat 2nd réseau

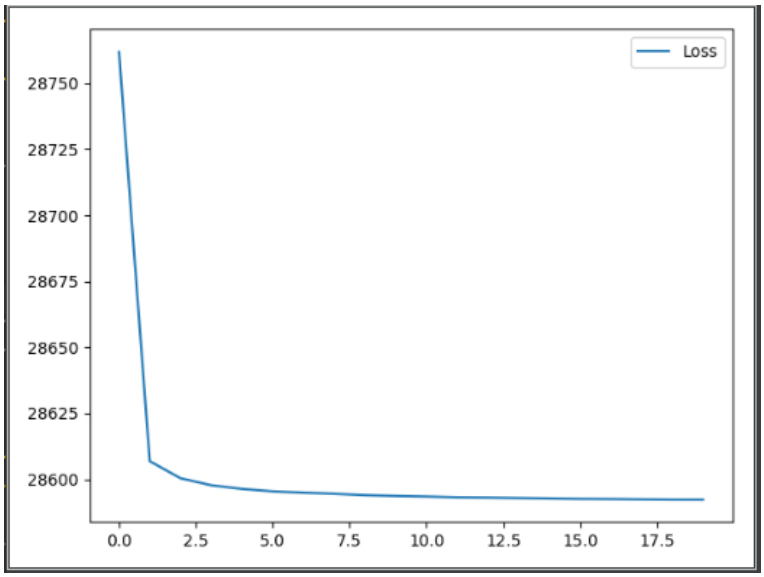


Figure 10: Loss 2nd réseau

10 Tests

Comme métrique de test, nous avions prévu d'utiliser tout d'abord la MSE (utilisée comme loss dans le réseau) ainsi que le PSNR. Cependant, les difficultés rencontrées (détaillées dans la section suivante) n'ont pas permis d'utiliser ces tests efficacement.

11 Difficultés

Après avoir fait notre état de l'art, nous avons trouvé un lien github (https://github.com /PhanHuyThong/Image-Reconstruction-by-CNN-based-PGD) correspondant à un article proche de notre problématique. Nous avons donc décidé d'utiliser ce code et de l'adapter à notre problématique, mais plusieurs problèmes sont apparus.

Premièrement, le format d'entrée des images était un .mat, qui correspond à un format Matlab, et qui dans le code est chargé avec la bibliothèque Scipy. Nous avons donc dû convertir l'ensemble du dataset fourni par notre client dans ce format, ce qui représentait un temps de calcul assez considérable (une demi-journée environ). Cependant, nous avons du revoir cette approche en raison de différents problèmes pratiques survenu pendant le développement, notamment les 8 Go de mémoire vidéo de la carte graphique utilisée insuffisants pour gérer des images de 2560 par 2560. Il a donc fallu reconvertir le dataset plusieurs fois afin d'obtenir quelque chose d'utilisable.

Tout d'abord, nous avons converti les images à l'origine en 3 canaux en images 1 canal, les images étant de toute façon en nuances de gris. Puis, nous nous sommes rendu compte d'un problème dans la gestion du format .tiff dans la librairie PIL et nous avons appliqué un changement d'échelle de valeurs pour que l'image soit correctement prise en compte par Scipy. Ces opérations ont coûté un temps précieux. Toutes ces opérations étaient effectuées par un script Python, qui était supposé ne servir qu'une seule fois, et

n'était donc pas optimisé pour fonctionner rapidement.

Finalement, ces changements n'étaient malgé tout pas suffisants. Nous avons donc dû faire le choix de diviser toutes les images en 64 patchs afin d'utiliser le moins de mémoire possible, ce qui a finalement fonctionné et nous avons pu commencer l'entraînement du réseau.

L'entraînement a également été très compliqué. En effet, le réseau apprenait l'image d'entrée et la reproduisait à l'identique au lieu de la débruiter (à noter que le temps d'entraînement du réseau était également très long). Nous avons donc modifié le dataset de façon à ne pas utiliser les patchs en bordure, afin d'augmenter la variété d'images traitées. Malgré nos efforts le réseau apprenait toujours l'image en entrée.

Nous avons donc utilisé un autre réseau type U-net (https://github.com/mateuszbuda/brain-segmentation-pytorch) importable directement dans pytorch. En utilisant le même dataset, on retrouve le même problème : le réseau apprend l'image d'entrée. Le problème vient donc probablement de notre gestion du dataset.

12 Conclusion (Bilan / Perspectives)

La plupart des problèmes rencontrés viennent d'une mauvaise gestion en amont du projet. Il aura fallu avoir un moyen rapide de convertir le dataset et configurable simplement, ce que nous n'avons pas mis en place et qui a plus tard entraîné des complications. Également, la décision d'entraîner le réseau sur nos machines personnelles n'était pas optimal, l'entraînement étant relativement long même pour des datasets de 5000 images, ne permettait pas de lancer plusieurs entraînements en simultané avec des paramètres différents, ni de le laisser tourner trop longtemps, nos machines étant utilisées pour des activités extérieures.

De manière plus large, le workflow mis en place est à revoir et les difficulté rencontrées aurait dû être anticipées et gérées en amont. Au final, ces erreurs ont fait perdre un temps considérable qui n'a pas permis de déboucher sur quelque chose de fonctionnel.

Étant donné que le stage de Benjamin se porte sur le même sujet (ou sur un sujet similaire), on pourra:

- Mettre en place un environnement pour entraîner le réseau en SSH
- Retravailler la gestion du dataset (optimisation des conversions, format différent...)
- Utiliser un format différent pour les images en entrée.

References

[1] Rushil Anirudh, Hyojin Kim, Jayaraman J. Thiagarajan, K. Aditya Mohan, Kyle Champley, and Timo Bremer. Lose the views: Limited angle CT reconstruction via implicit sinogram completion. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 6343–6352. IEEE Computer Society, 2018.

[2] Shabab Bazrafkan, Vincent Van Nieuwenhove, Joris Soons, Jan De Beenhouwer, and Jan Sijbers. Deep learning based computed tomography whys and wherefores. *CoRR*, abs/1904.03908, 2019.

[3] Harshit Gupta, Kyong Hwan Jin, Ha Q. Nguyen, Michael T. McCann, and Michael Unser. Cnn-based projected gradient descent for consistent CT image reconstruction. *IEEE Trans. Medical Imaging*, 37(6):1440–1453, 2018.

[4] Kerstin Hammernik, Tobias Würfl, Thomas Pock, and Andreas K. Maier. A deep learning architecture for limited-angle computed tomography reconstruction. In Klaus Hermann Maier-Hein, Thomas Martin Deserno, Heinz Handels, and Thomas Tolxdorff, editors, *Bildverarbeitung für die Medizin 2017 - Algorithmen - Systeme - Anwendungen. Proceedings des Workshops vom 12. bis 14. März 2017 in Heidelberg*, Informatik Aktuell, pages 92–97. Springer, 2017.

[5] Kyong Hwan Jin, Michael T. McCann, Emmanuel Froustey, and Michael Unser. Deep convolutional neural network for inverse problems in imaging. *IEEE Trans. Image Process.*, 26(9):4509–4522, 2017.

[6] Brendan Kelly, Thomas P. Matthews, and Mark A. Anastasio. Deep learning-guided image reconstruction from incomplete data. *CoRR*, abs/1709.00584, 2017.

[7] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells III, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015 - 18th International Conference Munich, Germany, October 5 - 9, 2015, Proceedings, Part III*, volume 9351 of *Lecture Notes in Computer Science*, pages 234–241. Springer, 2015.

[8] Hanming Zhang, Liang Li, Kai Qiao, Linyuan Wang, Bin Yan, Lei Li, and Guoen Hu. Image prediction for limited-angle tomography via deep learning with convolutional neural network. *CoRR*, abs/1607.08707, 2016.